CONTROL DESIGN AND ANALYSIS OF AN ADVANCED INDUCTION MOTOR ELECTRIC VEHICLE DRIVE

Marc A. Herwald

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

Dr. Dusan Borojevic, Chair

Dr. Jason Lai

Dr. Douglas K. Lindner

April 29, 1999

Blacksburg, Virginia

Keywords: Electric Vehicle, Induction Motor Drives, Field Oriented Control, Space Vector Modulation, Drive Cycle Simulation, Comparison of Induction Motor Losses, Drive Resonance Compensation, Stator Flux Linkage Measurement

CONTROL DESIGN AND ANALYSIS OF AN ADVANCED INDUCTION MOTOR ELECTRIC VEHICLE DRIVE

Marc A. Herwald

Abstract

This thesis is about the development and performance enhancement of an induction motor electric vehicle drive system. The fundamental operation of the induction motor drive hardware and control software are introduced, and the different modulation techniques tested are described. A software simulation package is developed to assist in the control design and analysis of the drive system. Next, to establish the efficiency gains obtained by using space vector modulation in the improved drive system, an inverter with hysteresis current control is compared to the same inverter with space vector modulation in steady state and on separate driving profiles. A method for determining induction motor harmonic losses is introduced and is based on obtaining the phase current harmonics from sampled induction motor stator phase currents obtained. Using a semi-empirical loss model, the induction motor losses are compared between different pulse width modulation control strategies throughout the torque versus speed operating region. Next, several issues related to the robustness of the control design are addressed. To obtain good performance in the actual vehicle, a new method for driveline resonance compensation is developed and proven to work well through simulation and experiment. Lastly, this thesis discusses the development of a new method to compensate for the gain and phase error obtained in the feedback of the d-axis and q-axis stator flux linkages. Improved accuracy of the measured stator flux linkages will be shown to improve the field oriented controller by obtaining a more accurate measurement of the feedback electromagnetic torque.

ACKNOWLEDGMENTS

I would like to thank the many people who have contributed to this thesis. I have benefited greatly by being a student at the Center for Power Electronics Systems (CPES), formerly the Virginia Power Electronics Center (VPEC), at Virginia Tech. I received help from many past and present students, faculty, and staff from CPES and the university. I would like to thank my committee members who have been supportive, and in particular, I would like to thank my advisor, Dr. Borojevic, for sharing his knowledge with me and for providing me with much guidance, support, and encouragement. Special thanks to R.W. Boothe who taught me a lot about many of the engineering aspects of induction motor electric vehicle drives that are discussed in this thesis. Most of all, I would like to thank everyone in my family, especially my parents, who have helped to make me into what I am.

TABLE OF CONTENTS

FIGURES

TABLES

LIST OF SYMBOLS

I_a	Stator phase a current	(Amps)
I_c	Stator phase c current	(Amps)
I_{sd}	d-axis component of the stator phase current	(Amps)
I_{sq}	q-axis component of the stator phase current	(Amps)
I_s	Stator phase current = $(I_{sd}^2 + I_{sq}^2)^{1/2}$	(Amps)
λ_{sd}	d-axis component of the stator flux linkage	(Wb)
λ_{sq}	q-axis component of the stator flux linkage	(Wb)
λ_s	Stator flux linkage = $(\lambda_{sd}^2 + \lambda_{sq}^2)^{1/2}$	(Wb)
θ_r	Angular rotor position	(rad)
θ_{slip}	Slip angle	(rad)
θ_s	Stator field angle	(rad)
α	Power factor angle	(rad)
ω_r	Angular rotor speed	(rad/sec)
ω_{slip}	Slip angular frequency	(rad/sec)
ω_e	Applied stator angular frequency = $\omega_r + \omega_{slip}$	(rad/sec)
V_{dc}	DC link voltage supplied to the inverter from the vehicle batteries	(Volts)
T_e	Electromagnetic torque applied to the rotor	(N-m)
T_L	Load torque applied to the rotor	(N-m)
T_{corr}	Correction torque that is output from the shudder compensator	(N-m)
T_{meas}	Electromagnetic torque measured from the stator current and flux	(N-m)
B	Rotor + load damping factor	(N-m-s)
J	Rotor inertia factor	(N-m-s^2)
P	Number of pole pairs	
τ_r	Rotor break frequency	(s^{-1})
τ_{rl}	Leakage break frequency	(s^{-1})
K^*	Field oriented controller gain	
σ	Leakage factor	
R_s	Stator resistance per phase	(Ω)
R_r	Rotor resistance per phase	(Ω)
L_{ls}	Stator leakage inductance	(H)
L_{lr}	Rotor leakage inductance	(H)
L_h	Main inductance	(H)
L_s	Stator self-inductance per phase = $L_h + L_{ls}$	(H)
L_r	Rotor self-inductance per phase = $L_h + L_{lr}$	(H)
d_a, d_b, d_c	Inverter phase a, b, and c duty cycle commands, respectively	
d_d, d_q	Reference vector duty cycles in the synchronous reference frame	
d_α, d_β	Reference vector duty cycles in the stationary reference frame	
d	Reference vector duty cycle = $(d_\alpha^2 + d_\beta^2)^{1/2}$	
d2	Duty cycle of reference vector projected onto the leading switching vector	

d1	Duty cycle of reference vector projected onto the lagging switching vector	
d0	Zero vector duty cycle = 1 - d1 – d2	
f_s	Inverter switching frequency	(Hz)
t_s	Inverter switching period = 1 / f_s	(s)
τ^*_{rotor}	Dynamometer bench shaft torque command	(N-m)
rpm^*_{shaft}	Dynamometer bench shaft speed reference command	(rpm)
r	Radius of the vehicle wheel	(m)
GR	Step up ratio of the speed of rotation of the wheel to the rotor shaft	
GR_{diff}	Step up ratio of the speed of rotation of the wheel to the drive shaft	
m	Total vehicle test mass	(kg)
v	Vehicle velocity	(m/s)
C_{R0}	Constant rolling resistance force constant	
C_{R1}	Speed-dependent rolling resistance force constant	
C_A	Aerodynamic drag force constant	
A	Frontal area of the vehicle	(m^2)
C_D	Drag coefficient constant	
N	Number of samples of the time waveform	
L	FFT length after zero padding of the time waveform	
F_s	Sampling frequency	(Hz)
T_s	Sampling period = 1 / F_s	(s)
F	Time waveform fundamental frequency	(Hz)
T	Time waveform fundamental period = 1 / F	(s)
n	Time sample	
m	Spectral sample number	
k	Harmonic frequency sample number	
η_{tran}	Transient Efficiency = output energy divided by input energy	
E	Energy	(kW-hr)

[1] If a symbol is shown with a star "*" superscript, it represents either a commanded value or parameter.

[2] A capitalized variable is represented in the synchronous reference frame

[3] A lower case variable is normally in the stationary reference frame, but it can be in the synchronous reference frame depending on the situation

Chapter 1. Introduction

1.1. Description of this Work

The induction motor (IM) drive is a presently a leading candidate for use in fully electric (EV) and alternative fuel vehicles (HEV), including electric cars and transit buses. The main reasons for acceptance of the IM are its ruggedness, reliability, and inexpensiveness, which are all desired qualities for EVs and HEVs. However, control of the induction motor is complicated because to obtain decoupled control of the torque and flux producing components of the stator phase current, both the magnitude and phase of the stator quantities need to be controlled [42]. In addition, there is no direct access to the rotor quantities, such as the rotor fluxes and currents [42]. To overcome these difficulties, high performance vector control algorithms have been developed that can decouple the stator phase currents by using only the measured stator current and flux, as well as the rotor speed [9], [10]. Vector controllers that operate in the stator flux reference frame have become a popular choice for the control of IM EV drives [8], [11], [14].

The current commands generated by the vector controller are regulated by the use of a Current Regulated Pulse Width Modulated (CRPWM) inverter [45]. The IM EV drive system studied in this thesis uses Space Vector Modulation (SVM). SVM can be implemented digitally, has a high DC link utilization, and can reduce both the inverter and induction motor losses [2], [15], [16], [17]. Figure 1.1 shows a block diagram of the system to be studied.

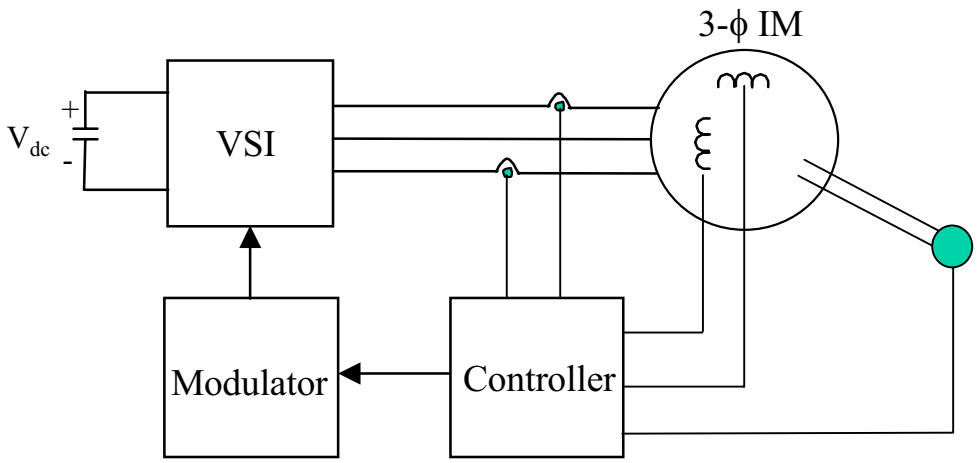


Fig. 1.1. Block Diagram of the VSI fed IM drive.

The benefits of using SVM in a vector controlled IM EV drive have not been fully addressed. Therefore, the first problem this thesis addresses is the need to perform a comprehensive analysis of the benefits of using SVM as part of the control software for a stator flux oriented 100 hp IM EV drive. Specifically, techniques are developed to compare induction motor drive systems with different modulation algorithms by performing steady state and driving profile efficiency comparisons [32]. To solve the problem of how to simulate drive profiles on a dynamometer bench in order to determine the driving profile efficiency, software programs have been developed to control the dynamometer bench in a repeatable manner [23], [32]. The efficiency tests will focus on the IM.

To analyze the drive system losses, the induction motor phase currents with a SVM inverter versus a hysteresis controlled inverter, which was used in a previous generation drive system, will be compared through both simulation and experiment. A drive system simulation software package is developed in C++ to produce simulated stator phase current currents. Separate IM harmonic losses, which are attributed to the phase current harmonics that are found, are modeled in detail, and the effects of the pulse width modulation control strategy on the IM operating efficiency are clearly shown. Large efficiency increases for both the inverter and IM will be found.

This thesis shows that using SVM can largely increase the efficiency of the IM in an EV [32]. However, an analysis of the causes for the IM efficiency increase, specifically the transient efficiency increases on drive cycles, still needs to be determined. Therefore, software programs

have been developed in this thesis that will offer explanations for the reasons for the improved IM efficiency when SVM is used. The development and usage of this software will be described fully. Using the measured energy consumed (in units of kW-hr) on different drive profiles, comparisons of vehicle average driving efficiency and range will be made between the different modulation algorithms tested.

The last part of this thesis discusses several control problems that have been addressed. In the development of the IM EV drive controller with SVM and stator flux oriented vector control, the control problems identified are as follows:

(1) In a vehicle with a long driving shaft, significant vibrations occur between the IM shaft and the vehicle due to drive-line resonance near zero speed (0-5 mph) [33].
(2) Open loop stator flux estimators produce significant error near zero speed (0–30 rad/sec of applied stator angular frequency) due to integrator drift [11], [42].

Problem (1) is solved in this thesis by developing a compensator to correct for drive resonance, also termed shudder, by feeding back the rotor shaft speed [33]. The compensator is added to the digital signal processor (DSP) control software as part of the torque command generating function. Many of the practical implementation issues are considered, such as the type of structure for the digital implementation of the compensator, accurate rotor rotational speed measurement, compensator output saturation reset, and rolloff of the compensator output with increasing vehicle speed.

Problem (2), as listed above, is solved in this thesis by the development of a DSP based stator flux linkage measurement compensator that can improve the performance of the drive system at low speeds -- less than 30 rad/sec of commanded stator electrical frequency. The gain and phase error in the measurement of the stator flux linkage at low speeds degrades the performance of the torque regulator, which is used as part of the field oriented controller, due to an incorrect computation of the feedback electromagnetic torque. First, the effects that the inaccurate measurement of electromagnetic torque has on the drive system are examined. Then, a DSP based compensator is developed to correct for the inaccuracies in the measurement of the

electromagnetic torque in steady state, which will be shown to improve the performance of the torque regulator at low stator angular frequencies.

To summarize, simulation, experimentation, and analysis will be used to meet all of the goals of this thesis:

- Study of modulation algorithms for induction motor control in an EV (Chapter 2)
- Simulating driving cycles on a dynamometer (Chapter 3)
- Simulation of induction motor phase currents (Chapter 4)
- Comparison of modulation algorithms (Chapter 5)
- Control design of a shudder compensator (Chapter 6)
- Control design of a stator flux feedback compensator (Chapter 7)

1.2. System Overview

This section gives an overview of the induction motor drive system that was tested. The system powers a 100 hp induction motor designed by General Electric specially for EV drives. The induction motor used, shown in Fig. 1.2, is oil cooled and has a planetary gear that steps down the rotor speed to shaft output speed by the factor 4.29:1. Stepping down the rotor speed allows less electromagnetic torque to be developed by the rotor and hence smaller phase currents in the stator windings. To propel the vehicle, the induction motor output shaft is required to have a large torque, which is why rotor speed is stepped down 4.29 times.

Fig. 1.2. GE Induction Motor. (Photo Courtesy of VPT, Inc.)

The rotor of the IM shown in Fig. 1.2 rides on precision ball bearings, which allows for a very small airgap (22 mil) between the stator and rotor. The squirrel cage rotor is cast iron with NEMA type D slots for high torque production. Special flux coil windings are wound on the stator for measurement of the d and q axis stator flux linkages, where the "d" coil axis is colinear with the "a" phase belt axis. The flux coils produce a linearly increasing voltage up to the base speed of the induction motor. Beyond the base speed, the flux is reduced inversely with speed producing a constant stator voltage and hence flux coil voltage, since the flux coils reflect the stator voltage. Equation (1.1) shows that the stator flux linkage is found as the integral of the measured coil voltage times a scaling constant.

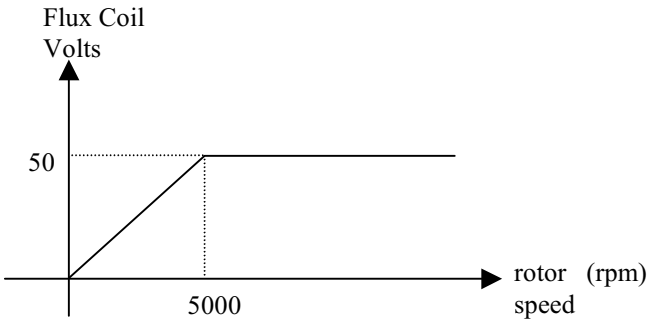


Fig. 1.3. Amplitude of flux coil voltage vs. rotor speed.

$$\lambda_s = K \cdot \int V_{coil} dt \tag{1.1}$$

The flux measurement windings are concentric and distributed on the stator with a 90 electrical degrees separation angle.

The GE induction motor parameters are known from measurement and are summarized in Table 1.1.

Table 1.1: Measured GE Induction Motor Parameters @ 25℃

R_s Stator Resistance	0.015 Ω
R_r Rotor Resistance	0.017 Ω
R_c Core Loss resistance	32.322 Ω
L_{ls} Stator Leakage Inductance	165.8 μH
L_{lr} Rotor Leakage Inductance	165.8 μH
L_h Main Inductance	6.16 mH

To interface the vehicle dc batteries and the induction motor, an inverter power stage and its associated control serves as an interface. The inverter is a DC to AC power converter that delivers ac phase currents to the induction motor from the electric vehicle's batteries. The inverter power stage uses 400A/600V IGBTs as the switching devices. As shown in Fig. 1.4, the arrangement of the IGBTs is for the standard 2-level voltage source inverter (VSI), where one of the three basic power electronic building blocks is shown highlighted. The topology allows motoring and regenerative power flow. A 15000 uF electrolytic capacitor produces a stiff dc link input voltage from the dc batteries of the EV.

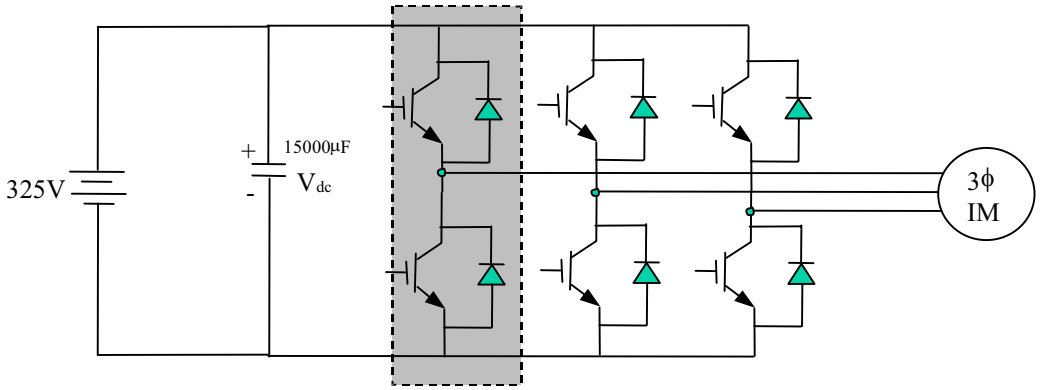


Fig. 1.4. DC/AC Induction Motor VSI used.

The analog scaling circuitry and digital control processors required for the operation of the inverter are contained on a circuit board mounted on top of the power stage as shown in Fig. 1.5.

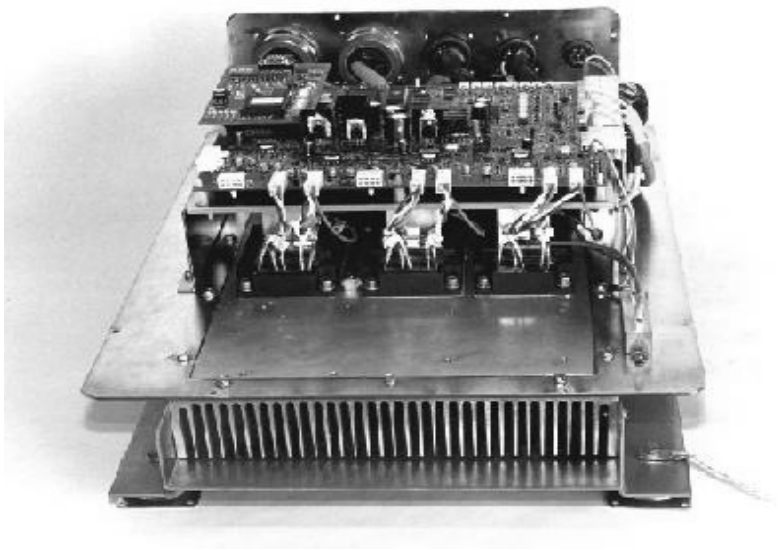


Fig. 1.5. Inverter power stage and controller boards. (Photo courtesy of VPT, Inc.)

Figure 1.6 shows the inverter circuit board block diagram. The basic inverter shown contains enough gate drivers for the standard hard-switched inverter, but additional gate drivers needed for soft-switching components can also be implemented. As shown in Fig. 1.6, the 68HC11 [43] microprocessor contained on the inverter control board communicates with the DSP using a full-duplex, high speed, synchronous serial communication link (SPI). The 68HC11

is used to process status information concerning operation of the drive, and the 68HC11 also implements a primitive peek/poke system that is used to display and modify DSP data memory variables as commanded remotely over the RS-232 serial data link. The 68HC11 receives discrete inputs from the vehicle operator including the setting of the PRNDL (park, reverse, nuetral, drive selection) and the keystart position, and the 68HC11 communicates this information to the DSP. Additionally, the driver indicators, which include the speedometer, fault lights, and alarm lights, are controlled by the 68HC11.

The primary function of the DSP is to process the real time control algorithms required by the field oriented controller and the PWM routines, as well as the additional routines added as a part of this thesis. The DSP is a special purpose motor control chip manufactured by Analog Devices called the ADMC300 [34].

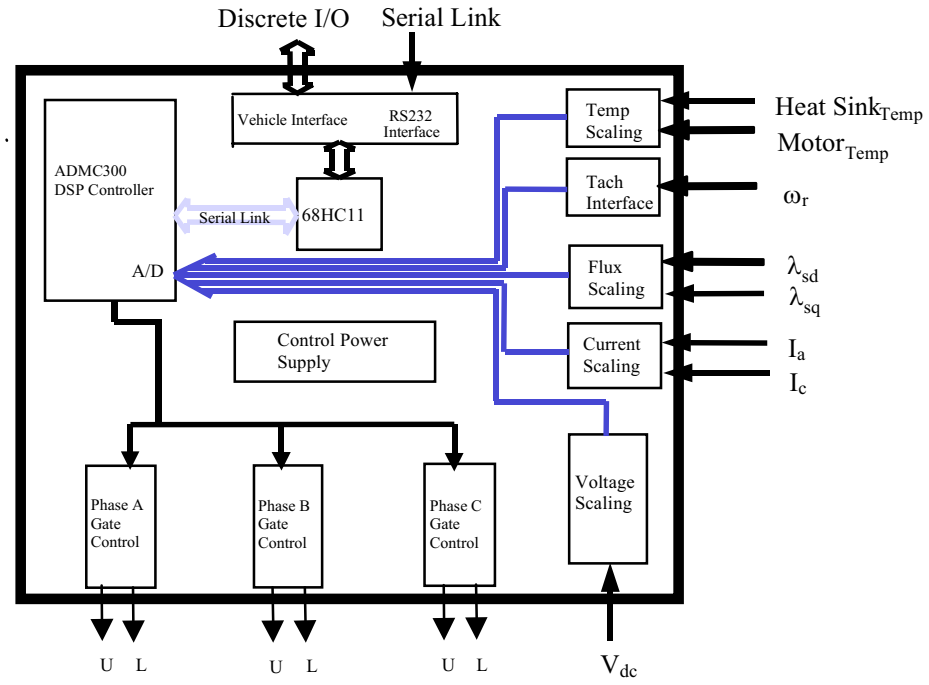


Fig. 1.6. Inverter control board.

For control purposes, the phase currents are measured using Hall-effect current sensors with a 2000:1 conversion ratio. The phase a and c currents are directly measured, and the phase b current is found as the negative of the phase a plus phase c currents as shown in Fig. 1.7.

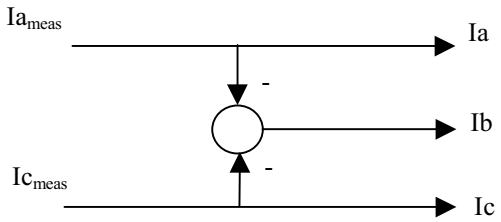


Fig. 1.7. Technique used for measurement of the phase currents.

Also sensed for indirect torque control mode is the rotational speed of the rotor, ω_r, which is done by an attached incremental encoder as explained in section 6.3.2.1. Figure 1.8 summarizes the signals measured from the induction motor that are used for field oriented control.

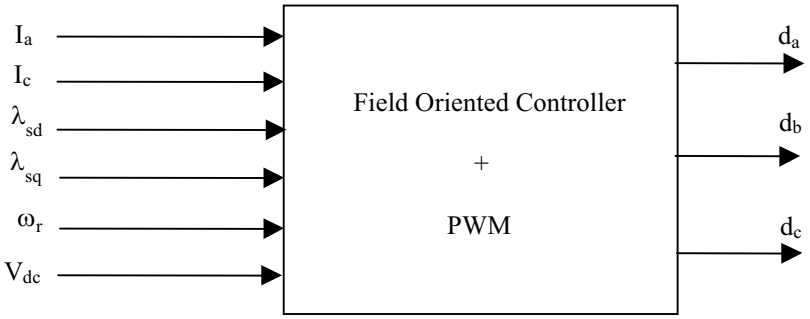


Fig. 1.8. Field oriented controller inputs and PWM outputs.

1.3. Control Algorithms

The DSP control software used in the EV control consists mainly of the field oriented controller, the dq current regulators, and the space vector modulator. These software elements are introduced in this section.

The field oriented controller attempts to separate the torque and flux producing components of the stator phase currents of an induction machine like in a separately excited dc machine, where the separate field and torque producing current components are naturally in quadrature due to the commutator. The field oriented controller uses an induction motor model and feedback of the rotor speed to produce decoupled phase current commands for the flux producing (d-channel) and torque producing (q-channel) components of the stator phase current. This computation of the current commands is done as a part of a software program that is implemented in the DSP chip.

The field oriented controller part of the DSP program uses indirect field oriented control in the stator flux reference frame [7], [9]. Stator flux oriented drives are desirable due to a minimized parameter sensitivity, direct field weakening operation, and the possibility of operation without speed feedback [8], [10], [11]. A torque regulator has also been added to the field oriented controller to help compensate for parameter variations of the induction motor. The torque regulator helps to ensure that parameter variations do not cause the steady-state value of the actual rotor torque to be different from the commanded rotor torque [33]. The complete field oriented controller is shown schematically in Fig. 1.9, as given in [6], with the added torque regulator shown in the dashed box. The torque command shown in Fig. 1.9 is related to the q-axis stator current command as shown in Equation (1.2) [9].

$$T_e^* = \frac{3}{2} I_{sq}^* \lambda_s^* \tag{1.2}$$

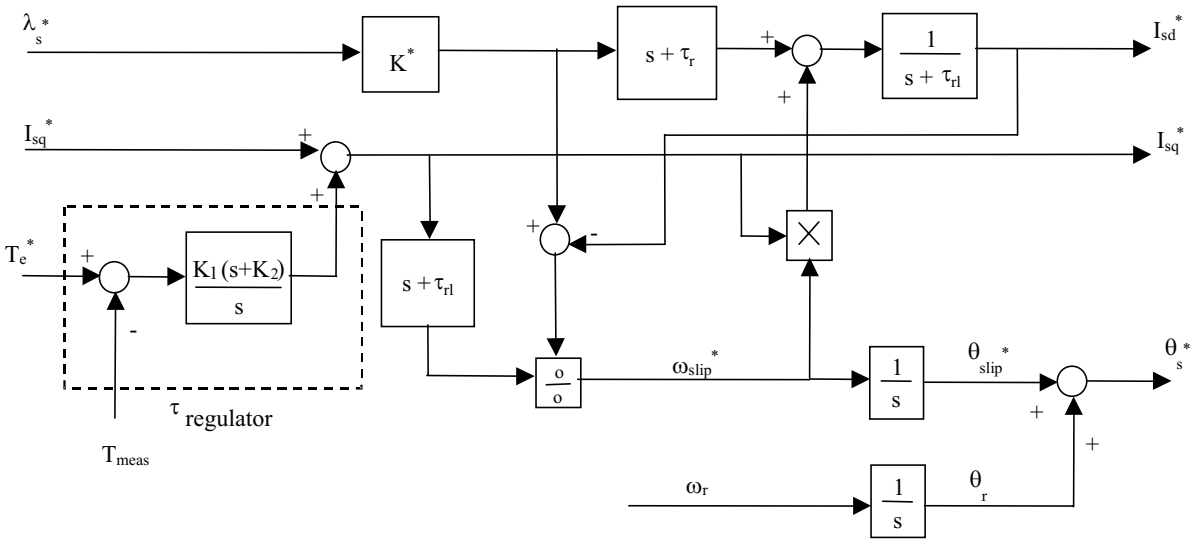


Fig. 1.9. Field oriented controller in the stator flux reference frame (Limiters not shown).

The field oriented controller pole locations, zero locations, and constants are determined as shown in (1.3) - (1.5) for the stator flux reference frame [6], [9].

$$\tau_r = \frac{R_r}{L_r} \qquad \tau_{rl} = \frac{R_r}{\sigma L_r} \qquad K^* = \frac{1}{\sigma L_s}$$

$$\sigma = 1 - \frac{L_h{}^2}{L_s L_r} \quad \text{“total leakage factor”} \tag{1.3}$$

where,

$$L_s = L_h + L_{ls} \tag{1.4}$$

$$L_r = L_h + L_{lr} \tag{1.5}$$

Note: All symbols have been defined in the List of Symbols.

The field oriented controller can be simulated in software as its discrete time equivalent. Figure 1.10 shows the field oriented controller outputs for a step input of the maximum torque

command of 250 N-m and the rated flux linkage command of 0.304 Wb. As shown in the Figure, for the maximum torque command, the maximum slip speed is about 30 rad/sec in steady state. Also, the steady state d current command is about 200 A, and the steady state q current command is about 500 A. These current commands give a power factor of 0.92 as shown in (1.6), which is the expected value from the motor specifications.

$$pf = \cos\left(90 - \tan^{-1}\frac{I_{sq}^{*}}{I_{sd}^{*}}\right) \tag{1.6}$$

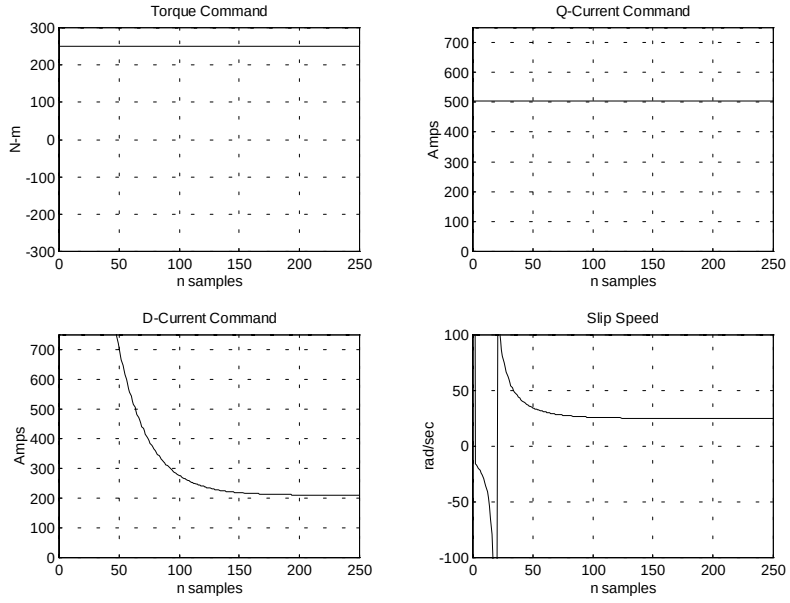


Fig. 1.10. Example waveforms of the field oriented controller with maximum torque command.

The field oriented controller should show decoupling between the d-axis and q-axis current commands for a constant stator flux linkage command input and a varying torque command. Figure 1.11 shows that the d channel current has very little perturbation for a ±100 N-m oscillation of the torque command and a constant stator flux reference.

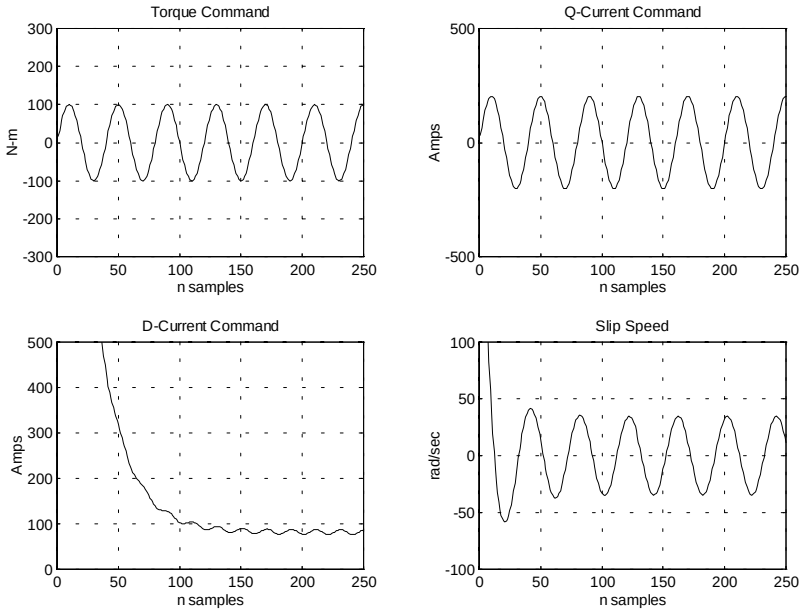


Fig. 1.11. Example waveforms of the field oriented controller with a sinusoidal torque command.

The d and q channel current commands output from the field oriented controller are input to the dq current regulator. The dq current regulator operates in the rotating reference frame with constant error inputs, which avoids the gain and phase errors and cross coupling associated with the stationary frame regulator [13], [14]. Feedfoward of the stator resistance voltage drop is included to improve the regulator transient response. Also, the q current controller allows for feedforward of the airgap emf, E, which is estimated as the stator flux times the stator angular frequency [8]. The dq current regulator is shown schematically in Fig. 1.12.

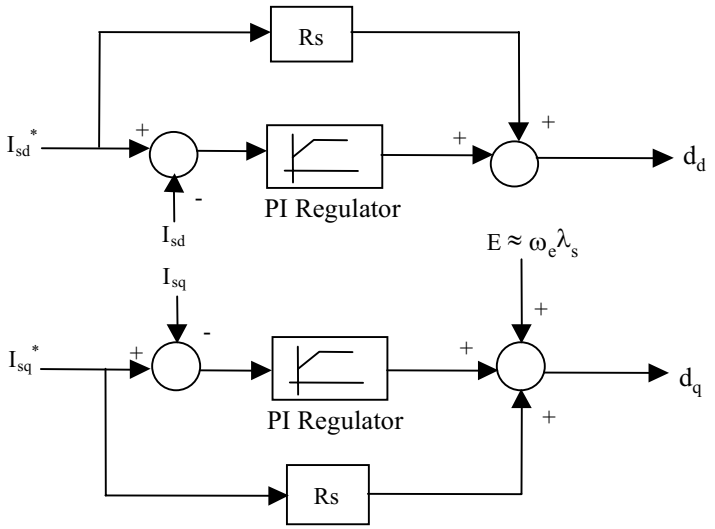


Fig. 1.12. dq current regulators in the synchronous reference frame.

The outputs of the current regulators form the inputs to the space vector modulator, shown in Fig. 1.13.. The outputs of the current regulators, the d_d and d_q duty cycles, are given in the rotating reference frame. The d_d and d_q duty cycles are returned to the stationary reference frame by using a rotating to stationary reference frame transformation, as shown in (1.7). The outputs of the reference frame transformation, d_α and d_β, are used as the inputs to the space vector module.

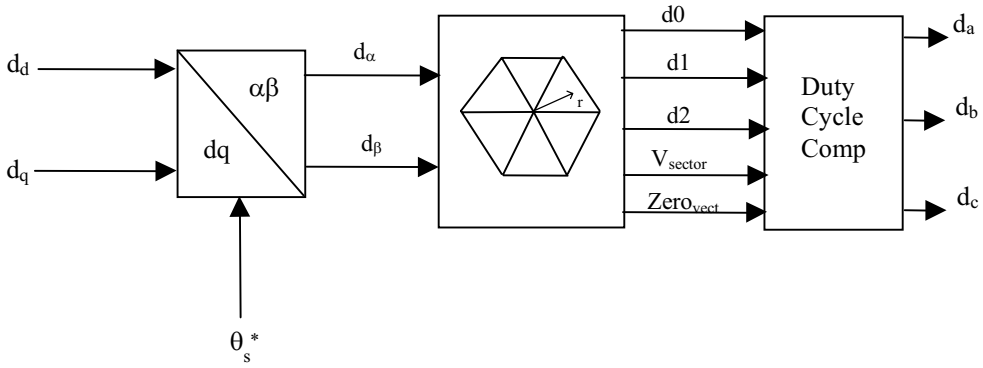


Fig. 1.13. Space Vector Modulation controller.

$$\begin{bmatrix} d_\alpha \\ d_\beta \end{bmatrix} = \begin{bmatrix} \cos\theta_s & -\sin\theta_s \\ \sin\theta_s & \cos\theta_s \end{bmatrix} \begin{bmatrix} d_d \\ d_q \end{bmatrix} \tag{1.7}$$

The space vector module computes the inverter switching states based on the location of the reference vector within the inverter output space [15], [16], [17]. In this case, an optimal zero vector selection routine is used that allows the phase leg with the largest instantaneous current to remain unswitched for the majority of the load operating points [2], [18], while still maintaining sinusoidal phase currents. The final outputs are the "on" duty cycle times for the top IGBT of each inverter phase leg or da, db, and dc. Together, the dq current regulators and the space vector modulator form the PWM control block. The basic controller before the added improvements of shudder compensation and flux feedback compensation is shown in Fig. 1.14.

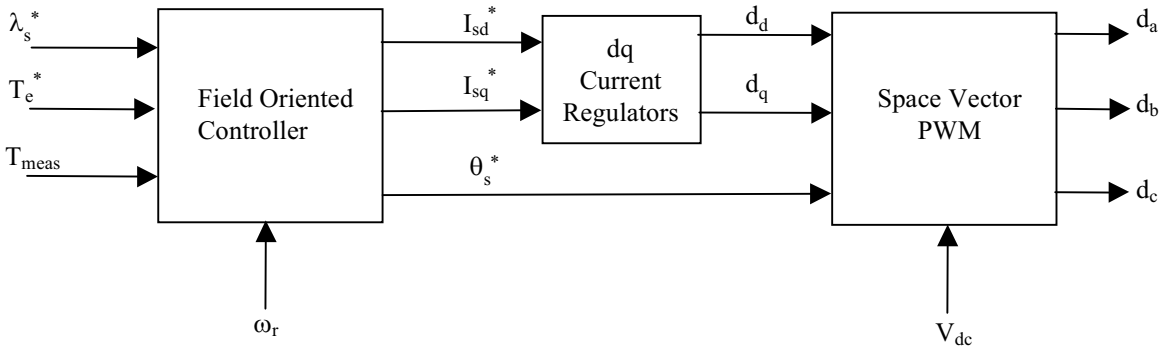


Fig. 1.14. Schematic of basic control software for induction motor drive control.

1.4. Thesis Overview

The following Chapter, Chapter 2, reviews the principles of SVM as the PWM method in an EV. First, the hysteresis controller used as the testing baseline is described. Then, the principles of SVM are shown, and a method for determining the duty cycles from a change of basis matrix is proposed. The manner in which the zero vector is selected is reviewed, and the possible limitations of the optimal zero vector selection technique in an EV drive are discussed.

In Chapter 3, the development of drive profile software programs that are required for dynamic drive performance testing on a dynamometer bench are described. First, the program that generates the required dynamometer bench torque and speed commands is described. Next, the design and operation of the program that implements the actual drive profiles is discussed. The first program, the Drive Profile Generation Program, is written in the Qbasic programming language. The second program, the Drive Profile Implementation Program, is written in the HP-VEE graphical programming language [46].

Chapter 4 introduces the drive system simulation software developed. First, the benefits of the C++ programming language for drive system simulations will be shown, including the use of an object oriented programming approach. Object definitions will be developed for induction motors in the synchronous or stationary reference frames, space vector modulators, hysteresis current controllers, and dq current regulators. The programming approach followed will be shown to produce simulation programs that execute quickly and are very reusable. Different drive system components are shown to be easily connected together through controlled interfaces. Chapter 4 concludes by demonstrating the use of the software developed through a specific case study. The case study compares the induction motor phase currents produced by using SVM as opposed to hysteresis current control. The results obtained will yield valuable insights about the results found in Chapter 5.

In chapter 5, the benefits of using SVM as opposed to hysteresis control, which was used in the previous generation drive system, are determined. The steady state efficiency maps of the inverter and induction motor using SVM control as opposed to hysteresis current control are introduced and compared. To compare the induction motor harmonic losses, an algorithm for the determination of the induction motor phase current harmonics is developed in MATLAB. The measured harmonic currents are then used in a semi-empirical motor loss model that predicts

large harmonic loss decreases with SVM, explaining the experimental results obtained. Additionally, the transient efficiency, or average driving efficiency, of the inverter with SVM control is compared to the inverter with hysteresis control on two simulated driving profiles. One profile is for city driving (FUDS), and the other profile is for highway driving (HWFET) [32]. The largest range increase when using SVM will be shown to be for driving in the city.

Chapter 6 explains the needed addition of a drive shaft resonance (shudder) compensator in addition to the field oriented controller and PWM algorithms to obtain good driveability in the vehicle. The shudder compensator developed uses rotor speed feedback to add a correction torque to the torque command input of the field oriented controller. The plant model used for the shudder compensator is designed in part by making in the vehicle measurements to determine the drive shaft spring constant. The resulting plant model developed is shown to accurately predict the real vehicle drive resonance behavior. Simulation results show that with added shudder compensation the rotor shaft oscillations are largely damped. Lastly, the shudder compensation is added to the DSP software of the inverter and shown to perform well in the vehicle as predicted by the simulation results.

The last Chapter, Chapter 7, discusses the development of a DSP based integrator drift compensator. The complete system is designed beginning from the problem description and ending with successful experimental test results. The system developed is used to compensate for the gain and phase error obtained in the measurement of the d and q stator flux linkages at low rotational speeds of the rotor and is the final control structure added to the present controller.

Chapter 8 concludes the thesis by summarizing the key obtainments.

Chapter 2. Description of EV PWM Algorithms Tested

This chapter describes the different modulation methods that were tested for use in an EV induction motor drive. The first modulation method discussed is hysteresis current control, which will form the performance baseline. Next, the use of space vector modulation (SVM) in an EV is discussed.

2.1 Hysteresis Current Control

Hysteresis current control forms the performance baseline for the modulation methods tested. The hysteresis inverter tested is named the EV2000 and was developed by GE for use in transit bus applications [33].

Hysteresis current control selects the inverter switching states based on maintaining the actual motor phase current within an allowed tolerance of the commanded phase current [3], [22]. The resulting controller structure, shown in Fig. 2.1, is simpler to implement than SVM control and has the added advantages of instantaneous current control and automatic transition into overmodulation (six-step) operation.

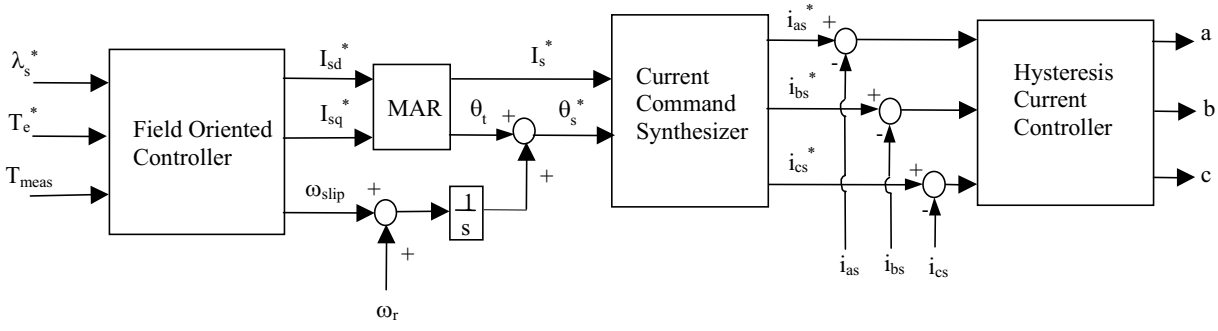


Fig.2.1. Basic control structure for induction motor control using a hysteresis current controller.

The three stator current commands are synthesized in the Current Command Synthesizer from the commanded stator current phasor according to the formulas in (2.1).

$$
\begin{aligned}
i_{as}^{*} &= I_s^{*} \sin\theta_s \\
i_{bs}^{*} &= I_s^{*} \sin(\theta_s - \frac{2\pi}{3}) \\
i_{cs}^{*} &= I_s^{*} \sin(\theta_s + \frac{2\pi}{3})
\end{aligned}
\tag{2.1}
$$

The actual stator phase currents are subtracted from the commanded stator phase currents to determine the current error. With hysteresis control, the inverter switching state depends on these computed instantaneous current errors. The current errors form the inputs into three comparators with hysteresis. Each of the three individual comparators has an input to output characteristic described by Fig. 2.2.

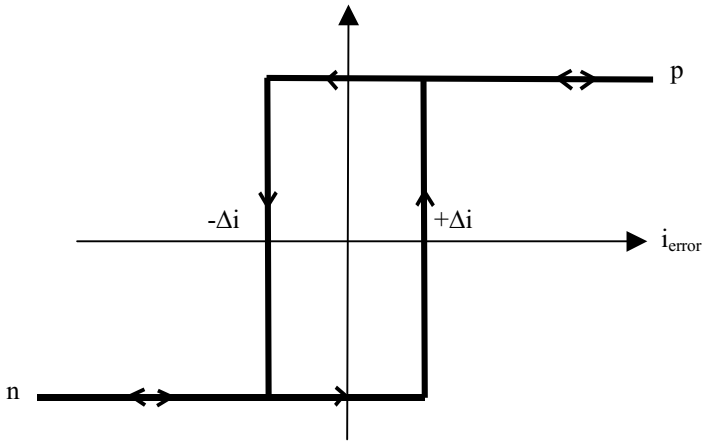


Fig. 2.2. Illustration of hysteresis band setting for the current comparators (Note: Δi depends on the commanded torque).

The EV2000 inverter uses a fixed-band type controller which is the simplest to implement but does not necessarily have the best spectral characteristics [22]. However, the fixed-band

controller used in the EV2000 varies the current bandwidth depending on the commanded torque. The current bandwidth, ±Δi, is always greater than ±20 A but cannot exceed ±100 A [33]. The current bandwidth is adjusted as shown in Fig. 2.3 to help reduce the switching frequency at higher currents.

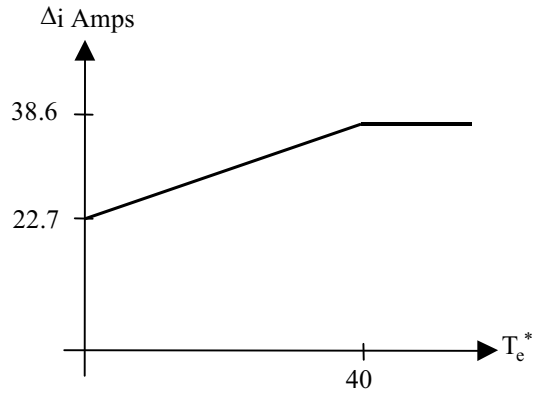


Fig. 2.3. Plot of the Δi vs. torque command for the EV2000.

2.2 Space Vector PWM

The two widely used carrier-based PWM implementation techniques are the direct digital gate pulse programming technique and the triangle-intersection technique [19]. Space vector modulation (SVM) is normally implemented using the direct method and is used in the induction motor EV drive developed. SVM is the preferred strategy for high performance ac drives due to increased utilization of the dc link and the capability to reduce the switching losses as compared to the traditional sine triangle-intersection technique [16]. For the space vector PWM implemented, the direct digital programming method is used and the gate signal on-state times are determined in the DSP, as introduced in Chapter 1.

Based on the topology of the inverter power stage, shown in Fig. 1.4, there are eight possible switching states for the output line voltage space vector, which is based on the constraint that the input lines can never be shorted and the phase currents must be continuous [2].

The eight possible switching states are determined from the different possible values of the induction motor line to line voltages. In addition, the inverter output line to line voltage space vector is designated as $[V_{ab}\ V_{bc}\ V_{ca}]^T$. To help determine the possible inverter output line to line voltages, a simplified representation of the VSI is shown in Fig 2.4. Each phase leg of the induction motor must be either connected to the positive side of the dc link, or p, or to the negative side of the dc link, or n.

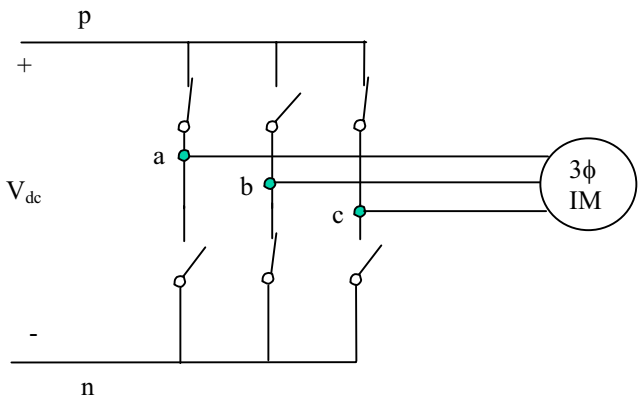


Fig. 2.4. VSI in switching state pnp.

The equation relating the stator abc line-to-line voltages to the stator dq complex space vector is expressed by (2.2) [14].

$$\vec{v}_{qds_{line-line}} = \frac{2}{3}(V_{ab} + V_{bc}e^{j120^\circ} + V_{ca}e^{j240^\circ}) \tag{2.2}$$

The possible switching states, resulting line-to-line voltages, and complex voltage space vectors are given in Table 2.1.

Table 2.1. Switching states in a two-level VSI and the corresponding output line-to-line voltages and stator dq complex voltage space vector.

	V_{ab}	V_{bc}	V_{ca}	$\vec{v}_{qds_{line\text{-}line}}$
nnn	0	0	0	0
ppp	0	0	0	0
pnn	V_{dc}	0	$-V_{dc}$	$V_{dc} + j\frac{V_{dc}}{\sqrt{3}} = \frac{2}{\sqrt{3}} V_{dc} e^{j30^\circ}$
ppn	0	V_{dc}	$-V_{dc}$	$j\frac{2V_{dc}}{\sqrt{3}} = \frac{2}{\sqrt{3}} V_{dc} e^{j90^\circ}$
npn	$-V_{dc}$	V_{dc}	0	$-V_{dc} + j\frac{V_{dc}}{\sqrt{3}} = \frac{2}{\sqrt{3}} V_{dc} e^{j150^\circ}$
npp	$-V_{dc}$	0	V_{dc}	$-V_{dc} - j\frac{V_{dc}}{\sqrt{3}} = \frac{2}{\sqrt{3}} V_{dc} e^{-j150^\circ}$
nnp	0	$-V_{dc}$	V_{dc}	$-j\frac{2V_{dc}}{\sqrt{3}} = \frac{2}{\sqrt{3}} V_{dc} e^{-j90^\circ}$
pnp	V_{dc}	$-V_{dc}$	0	$V_{dc} - j\frac{V_{dc}}{\sqrt{3}} = \frac{2}{\sqrt{3}} V_{dc} e^{-j30^\circ}$

The stator line-to-midpoint induction motor dq complex space vectors are related to the stator line-to-line dq complex space vectors by the expression given in (2.3), which relates the line-to-line voltages of a three phase Y-connected load to the line-to-midpoint voltages.

$$\vec{v}_{qds_{line\text{-}midpt.}} = \frac{\vec{v}_{qds_{line\text{-}line}}}{\sqrt{3}} e^{-j30^\circ} \tag{2.3}$$

The space vector diagram with all of the available inverter output voltage vectors is shown in Fig. 2.5, where the six non-zero voltage vectors have been identified.

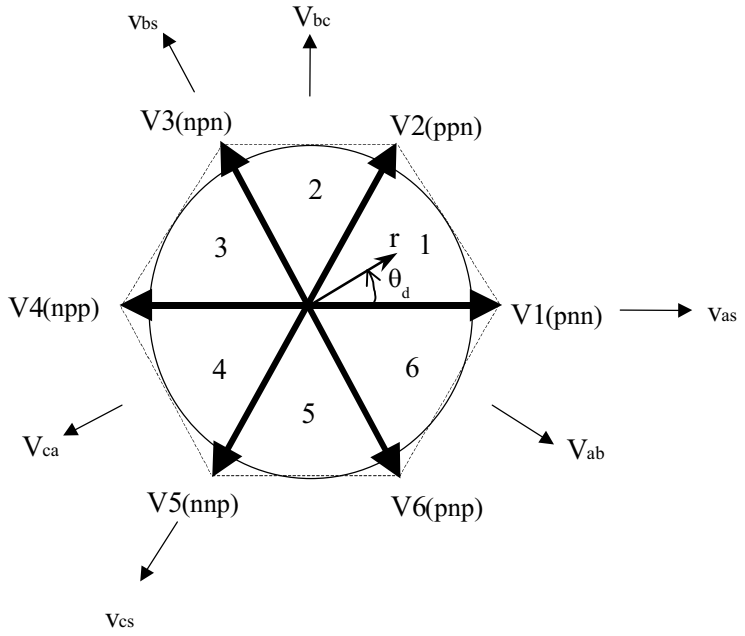


Fig. 2.5. Inverter output voltage vectors in the α-β stationary frame.

Two additional switching vectors, which are not shown in Fig. 2.5, are the zero vectors V7(nnn) and V8(ppp).

The reference vector, $\vec{r}$, shown in Fig. 2.5, has a length of $(^2/_3)\cdot d\cdot V_{dc}$ and an angle of θ_d. The quantities d and θ_d are defined in (2.4) and (2.5), respectively, where d_α and d_β are input as shown in Fig. 1.13.

$$d=\sqrt{d_\alpha{}^2+d_\beta{}^2} \tag{2.4}$$

$$\theta_d = \tan^{-1}\left(\frac{d_\beta}{d_\alpha}\right) \tag{2.5}$$

$$\vec{r} = \frac{2}{3}\bullet V_{dc}\bullet d\ \angle\theta_d \tag{2.6}$$

The tangent inverse used for determining θ_d in Equation (2.5) must be four quadrant to have correct rotation of the reference vector. Positive sequence voltage excitation of the induction motor, abc, corresponds to rotation of the reference vector in Fig. 2.5 in the CCW direction. Negative sequence voltage excitation of the induction motor corresponds to rotation of the reference vector in Fig. 2.5 in the CW direction.

The circle inscribed in the hexagon in Fig. 2.5 is the maximum magnitude of $\vec{r}$ for non-distorted phase voltages. The maximum reference vector magnitude defines the maximum radius of the inscribed circle and is given by (2.7) for normalized duty cycle timings.

$$|r|_{max} = \frac{2}{3} \bullet d_{max} \bullet V_{dc} = \frac{2}{3} \bullet \frac{\sqrt{3}}{2} \bullet V_{dc} = \frac{V_{dc}}{\sqrt{3}} \tag{2.7}$$

Once the reference vector, r, is known, it is synthesized during each switching period by finding its projection onto the nearest two adjacent switching vectors. Using adjacent switching vectors produces the least amount of harmonic distortion and switching losses [2], [20]. In the case of the EV drive system developed, the switching period, T_s, is 100 μs, which corresponds to an inverter switching frequency, F_s, of 10 kHz. The synthesis of r in voltage sector 1 is shown in Fig. 2.6.

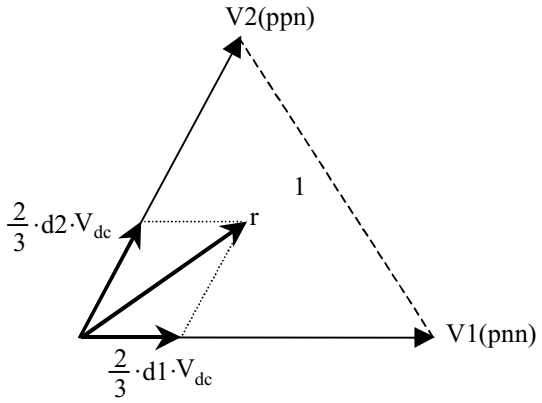


Fig. 2.6. Synthesis of the reference vector in sector 1.

Since the duty cycle timing information is normalized to one, the zero vector must be used to account for the difference between d1 plus d2 and unity, as shown in (2.8).

$$d0 = 1 - d1 - d2 \tag{2.8}$$

If the reference vector magnitude becomes larger than given in (2.7), the zero duty cycle will began to be negative whenever the reference vector leaves the dotted hexagon shown in Fig. 2.5, which is one way to detect the entrance into overmodulation [21].

In a 2-dimensional linear vector space, any set of 2 linearly independent vectors qualifies as a basis. Therefore, the d1 and d2 duty cycles can be determined from a change of basis of $\vec{r}$ from the orthogonal αβ basis to the basis defined by the two adjacent switching vectors, which for sector 1 are V1 and V2 , as shown in Fig. 2.7.

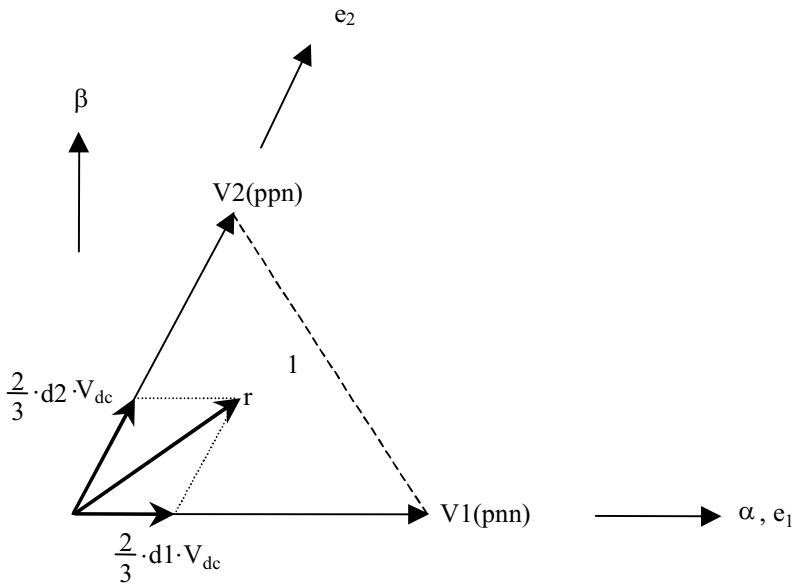


Fig. 2.7. Illustration of basis vector coordinates.

The standard basis is given as shown in (2.9), and the required basis is given by (2.10). The basis vectors have been normalized to unit length in their respective directions.

$$[\alpha, \beta] = \left[\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right] \tag{2.9}$$

$$[e1, e2] = \left[\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix} \right] \tag{2.10}$$

The inverse change of basis matrix is determined by expanding the new basis vectors, $[e_1, e_2]$, in terms of the standard basis vectors, $[\alpha, \beta]$, which is shown in (2.11).

$$e_1 = 1\alpha + 0\beta = [\alpha \ \beta] \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$e_2 = \frac{1}{2}\alpha + \frac{\sqrt{3}}{2}\beta = [\alpha \ \beta] \begin{bmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix} \tag{2.11}$$

Therefore, for sector 1, the inverse change of basis matrix is given as shown in (2.12), and the change of basis matrix is given by (2.13).

$$B^{-1} = \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} \end{bmatrix} \tag{2.12}$$

$$B = \begin{bmatrix} 1 & \frac{-1}{\sqrt{3}} \\ 0 & \frac{2}{\sqrt{3}} \end{bmatrix} \tag{2.13}$$

Given the change of basis matrix, B, the d1 and d2 duty cycles are determined as shown in (2.14) for sector 1.

$$\begin{bmatrix} d1 \\ d2 \end{bmatrix} = B \begin{bmatrix} d_\alpha \\ d_\beta \end{bmatrix}$$

$$\therefore d1 = 1d_\alpha + d_\beta \frac{-1}{\sqrt{3}}$$

$$\therefore d2 = 0d_\alpha + d_\beta \frac{2}{\sqrt{3}}$$

(2.14)

A similar change of basis procedure can be used in the remaining five sectors. The change of basis matrices for each sector are summarized in Table 2.2.

Table 2.2: Change of basis matrix for each of the six sectors.

Sector:	**1**	**2**	**3**
	$B = \begin{bmatrix} 1 & \frac{-1}{\sqrt{3}} \\ 0 & \frac{2}{\sqrt{3}} \end{bmatrix}$	$B = \begin{bmatrix} 1 & \frac{1}{\sqrt{3}} \\ -1 & \frac{1}{\sqrt{3}} \end{bmatrix}$	$B = \begin{bmatrix} 0 & \frac{2}{\sqrt{3}} \\ -1 & \frac{-1}{\sqrt{3}} \end{bmatrix}$
Sector:	**4**	**5**	**6**
	$B = \begin{bmatrix} -1 & \frac{1}{\sqrt{3}} \\ 0 & \frac{-2}{\sqrt{3}} \end{bmatrix}$	$B = \begin{bmatrix} -1 & \frac{-1}{\sqrt{3}} \\ 1 & \frac{-1}{\sqrt{3}} \end{bmatrix}$	$B = \begin{bmatrix} 0 & \frac{-2}{\sqrt{3}} \\ 1 & \frac{1}{\sqrt{3}} \end{bmatrix}$

Once d1, d2, and d0 have been determined, the next step is to compute the d_a, d_b, and d_c duty cycles as shown in Fig. 1.13. The normalized d_a, d_b, and d_c duty cycles correspond to the time the top switch of the corresponding phase a, b, or c leg is turned on during the switching period. When a phase leg is on, the state is 'p' and the top switch of Fig 2.4 is closed and the bottom switch is open. The phase leg duty cycles are determined based on the sector the reference vector is in and the switching state of the adjacent switching vectors. Since there are

two possible zero vector states used for the d0 duty cycle time, V7(nnn) or V8(ppp), there are two separate sets of equations for the d_a, d_b, and d_c duty cycles for each sector. For example, as shown in Fig. 2.6, the d1 duty cycle in sector 1 is determined from V1(pnn) and the d2 duty cycle in sector 1 is determined from V2(ppn). Therefore, for the reference vector in sector 1, the d_a, d_b, and d_c duty cycles are given as shown in (2.15) when the zero vector V7(nnn) is used. The d_a, d_b, and d_c duty cycles are given as shown in (2.16) when the zero vector V8(ppp) is used.

$$\begin{aligned} d_a &= d1 + d2 \\ d_b &= d2 \\ d_c &= 0 \end{aligned} \tag{2.15}$$

$$\begin{aligned} d_a &= 1 \\ d_b &= d2 + d0 \\ d_c &= d0 \end{aligned} \tag{2.16}$$

Once the duty cycle times d_a, d_b, and d_c are determined, they are translated into center based IGBT phase gating signals via the PWM generation unit that is built into the DSP [34], as shown in Fig 2.8 for sector 1 for the case when the zero vector is V8(ppp).

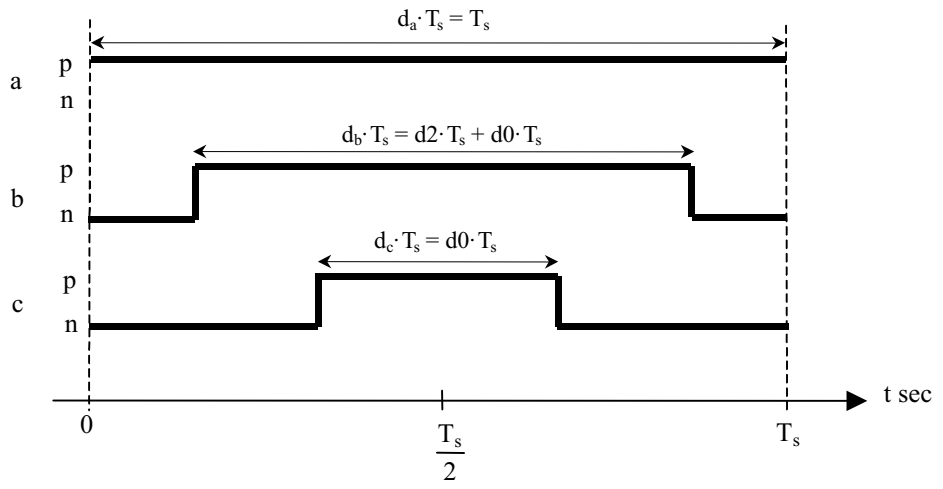


Fig. 2.8. Center based phase gating signals in sector 1 and the zero vector is V8(ppp).

2.3 Selection of Zero Vector with SVM

The space vector PWM methods vary depending on how and when the inverter zero vector is applied [2], [16]. As shown in Fig. 2.8, it is possible to leave one of the phase legs unswitched for the entire switching interval, T_s. Since the switching loss is proportional to the amount of current being switched, the choice of the zero vector should always attempt to leave the inverter phase leg carrying the largest current unswitched. In fact, this is possible as long as the power factor is greater than or equal to 0.866, which means that the switching losses can be reduced by as much as 50% [2], [18]. This reduction of switching losses can help increase the system efficiency and hence the EV driving range as will be shown in Chapter 5.

For a balanced set of sinusoidal currents, every one-sixth of the fundamental period defines a new maximum magnitude of the three phase currents as shown in Fig. 2.9.

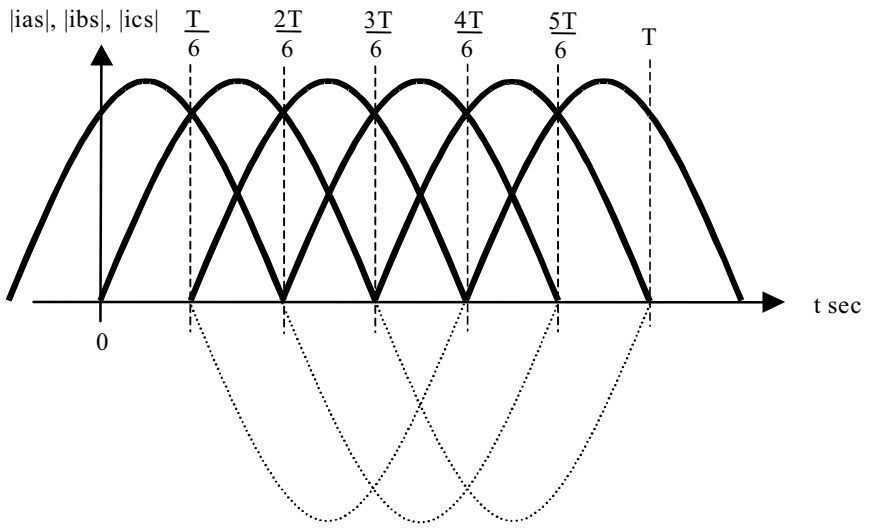


Fig. 2.9. Absolute value of the induction motor phase currents indicating the sectioning into six sectors of the maximum current in one fundamental period.

The one sixth of a period intervals define six current sectors rotated away from the voltage reference axis by the power factor angle, α. This rotation of the current reference axis is shown in Fig. 2.10 for forward motoring, which has an abc positive phase sequence. The current sectors for the case shown in Fig. 2.10 are given Fig. 2.11, where the current sectors that identify the maximum phase current areas have been color coded.

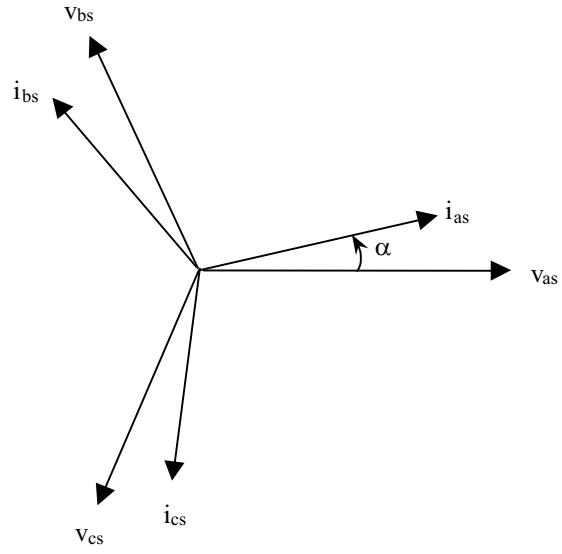


Fig. 2.10. Illustration of current reference axis phase rotation for forward motoring. The current sectors for the case shown in Fig. 2.10 are given in Fig. 2.11.

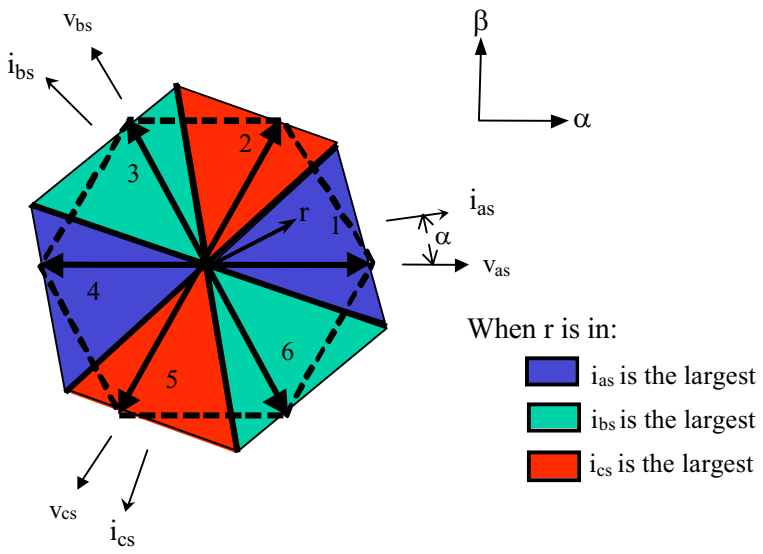


Fig. 2.11. Current sectors for forward motoring. The current sectors have been pattern coded and numbered 1 – 6.

As an example, when the reference vector is in sector 1 for forward motoring, the largest current is either i_{as} or i_{cs} as long as α does not exceed 30°. So, when i_{as} is the largest current, the zero vector V8(ppp) is chosen because the state of the switches for phase a are both p for V1(**p**nn) and V2(**p**pn). Therefore, the phase a current is not switched when i_{as} is the largest current. When i_{cs} is the largest current in sector 1, the zero vector V7(nnn) is chosen because the state of the switches for phase c are both n for V1(pn**n**) and V2(pp**n**). Therefore, the phase c current is not switched when i_{cs} is the largest current. If α is greater than 30°, i_{bs} can be the largest current in sector 1 and so the larger of either i_{as} or i_{cs} is used to determine which zero vector is used. The zero vector selection logic for sector 1 can be represented by Fig. 2.12.

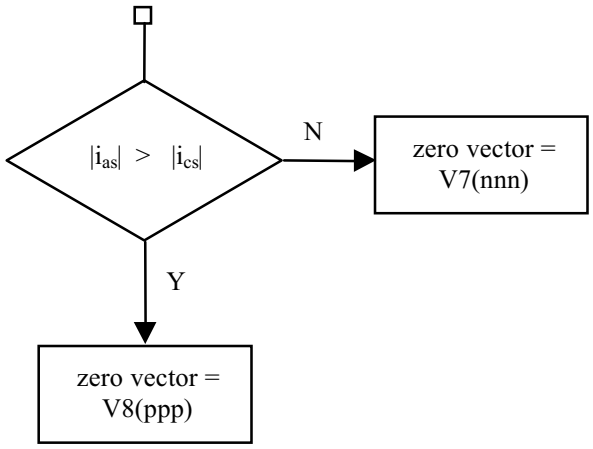


Fig. 2.12. Zero vector selection logic for sector 1.

Similar selection logic is used for the additional sectors. This form of zero vector selection logic that is based on attempting to not switch the largest phase current has been called SVM4 in the literature [2]. It is also known as optimal zero vector selection SVM.

The power factor of the GE Induction is known to be high for most of the torque vs. speed operating points, because it is a high torque, high efficiency motor. A plot of the power factor on the torque vs. speed plane is shown in Fig. 2.13. The power factor is shown to be the largest at locked rotor, which is evident from the measured stator phase a current and phase a voltage at locked rotor shown in Fig. 2.14. As shown in Fig. 2.14, the phase a voltage is unchanged for 60° of phase rotation when the reference vector is in either of the two phase a current sectors shown in Fig. 2.11. If the phase a current is near maximum, the reference vector is in current sector 1 and phase leg a does not switch. The top IGBT carries the phase current continuously because the state is V8(ppp). If the phase a current is near minimum, the reference vector is in current sector 4 and phase leg a does not switch. The bottom IGBT carries the phase current continuously because the state is V7(nnn). The locked rotor condition is the worst possible case for the IGBT carrying a large phase current for a long time. The conduction time of the IGBT can be estimated as shown in (2.17).

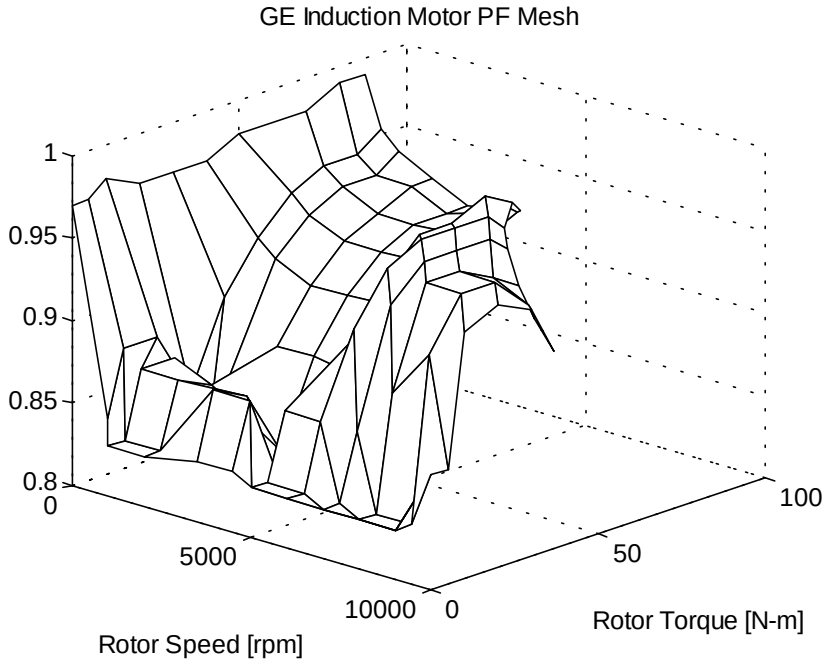


Fig. 2.13. GE induction motor fundamental power factor characteristic.

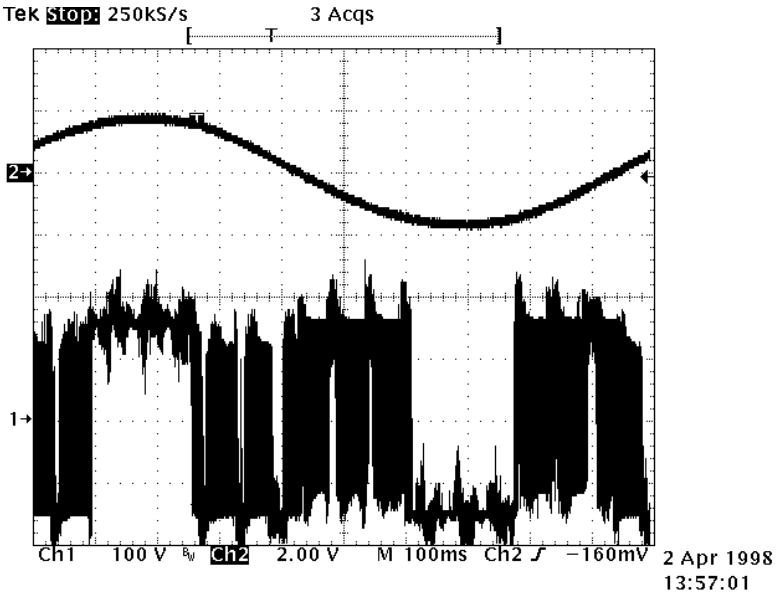


Fig. 2.14. Measured phase a current and phase voltage with optimal zero vector selection SVM.

$$t_{cond} = \frac{2\pi}{6\omega_e} \tag{2.17}$$

$\cong$ 166 ms for the waveform in Fig. 2.14

If optimal zero vector selection SVM is used down to zero rotor speed, then the IGBT main switch must be able to handle the maximum phase current for about 166 ms as computed by (2.17).

The SVM introduced in this section is implemented in Chapter 4 in software simulations, and in Chapter 5, SVM is tested in the actual drive system, which was introduced in Chapter 1.

Chapter 3. Development of Drive Profile Testing Software for EV Drives

In this chapter, the software programs developed to implement drive profile testing are discussed. The program discussed in section 3.1 is used to generate the torque and speed commands for the dynamometer bench shaft torque reference and speed reference induction motor drives, respectively. The way that vehicle dynamics are modeled on the dynamometer is discussed, and the operation of the Qbasic program written to actually generate commands is explained. The output of the Qbasic Drive Profile Generation Program consists of coded torque, speed, and motor direction commands that are stored on disk as ASCII text files.

The commands stored to disk by the Drive Profile Generation Program are used by the Drive Profile Implementation Program, whose development is discussed in section 3.2 of this Chapter. The Drive Profile Implementation Program is written using the HP-VEE graphical programming language, which is a software programming package developed by Hewlett Packard for real time equipment control and data acquisition [46]. The HP-VEE Drive Profile Implementation Program parses the coded commands created by the Qbasic Drive Profile Generation Program. When a command is read and decoded, the HP-VEE program then sends the appropriate messages to the load and test induction motor controllers using the available COM1 and COM3 serial ports of the IBM host PC development station (COM2 is used by the mouse). The internal clock of the IBM host PC is used to obtain the precise one second timing intervals required for the update of the torque and speed references. A graphical user interface was developed for the HP-VEE Drive Profile Implementation Program that allows full control of the implementation of the drive profiles.

3.1. Development of a Program for Generation of Dyno Bench Torque and Speed Commands

This section discusses the Qbasic Drive Profile Generation Program that is used to create commands for the dynamometer bench torque and load induction motors. The dynamometer test bench, whose diagram is shown in Fig. 3.1, is programmed to model the operation of a vehicle on a driving profile.

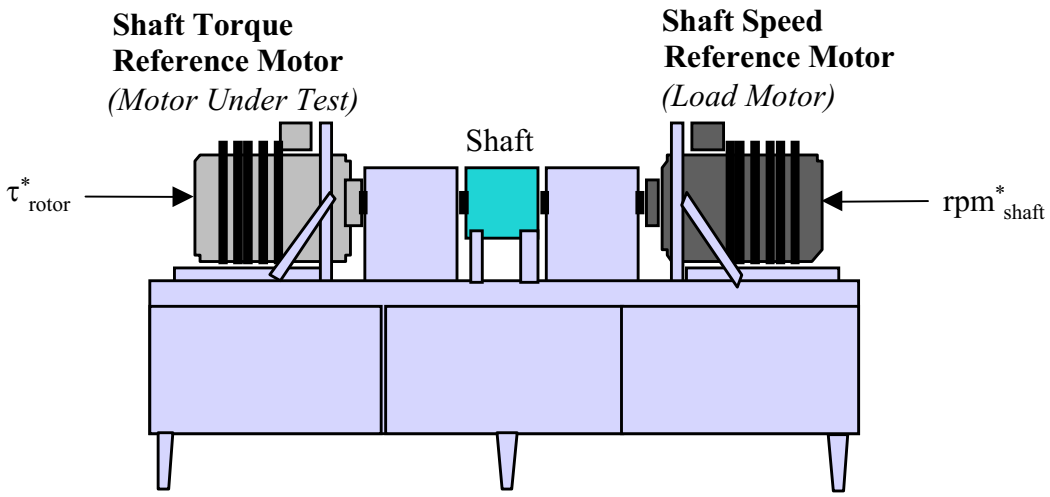


Fig. 3.1. CPES lab electric motor dynamometer.

3.1.1. Modeling Vehicle Dynamics on the Dynamometer

The purpose of the Qbasic Drive Profile Generation Program is to determine the required dynamometer shaft speed and torque that would occur if the EV drive was operated in an actual vehicle [32]. A model of vehicle dynamics is used to determine the correct shaft speed and torque. The model inputs are standard drive profile speed vs. time data points that have been sampled at 1Hz. These profiles are generally plotted as km/h vs. time and are distributed to automobile manufacturers to compare the fuel efficiency of their new vehicle designs.

The drive profiles modeled in this thesis are the Federal Urban Driving Schedule (FUDS) and the Highway Fuel Economy Test (HWFET) [23]. The FUDS profile is considered to represent city type of driving, and the HWFET profile is considered to represent highway type of driving. The plots of these profiles are shown in Fig. 3.2. The particular FUDS and HWFET profiles that are used were obtained from a commercially available software package used for simulation of the operation of EV and HEV vehicles [32].

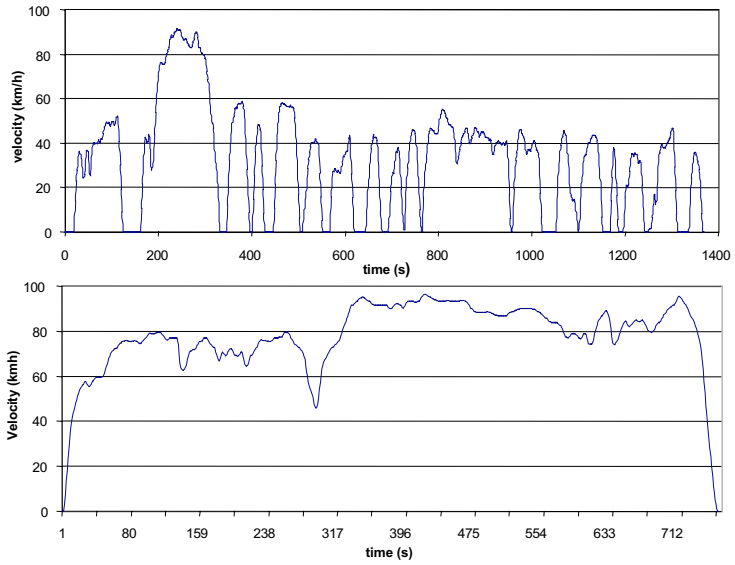


Fig. 3.2. Plot of FUDS (top) and HWFET (bottom) driving profiles.

Given the particular FUDS and HWFET speed vs. time profiles sampled at 1Hz, a discrete time algorithm was devised that computes commanded shaft torque and speed commands based on a simplified differential equation representation of the vehicle dynamics. The algorithm used is shown in block diagram form in Fig. 3.3.

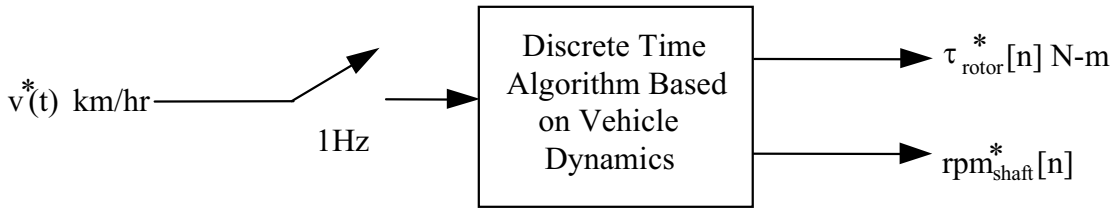


Fig. 3.3. Block Diagram of the Generation of Shaft Speed and Torque Commands

The differential equation used to represent vehicle dynamics is given in (3.1).

$$\tau^*_{rotor} = \frac{r}{GR} \cdot [m\frac{dv^*}{dt} + mg(C_{R0} + \frac{C_{R1}}{v_{ref}} v^* + \frac{C_A}{v_{ref}^2} v^{*2})] \tag{3.1}$$

$$\text{Where,} \qquad C_A = \frac{1}{2}\frac{C_D A \rho_{air} v_{ref}^2}{mg} \tag{3.2}$$

The sum of the terms in brackets in Equation (3.1) represents the force required to propel the vehicle. The definitions of all the variables and constants given in (3.1) and (3.2) are given below:

r = Radius of the vehicle wheel in meters
GR = Step up ratio of the speed of rotation of the wheel to the rotor shaft
m = Total vehicle test mass in kilograms
v* = Commanded vehicle velocity in m/s
g = Acceleration due to gravity at sea-level in m/s^2 = 9.81 m/s^2
C_{R0} = Constant rolling resistance force constant
C_{R1} = Speed-dependent rolling resistance force constant
C_A = Aerodynamic drag force constant
v_{ref} = Vehicle reference velocity in m/s = 27.8 m/s
A = frontal area of the vehicle expressed in square meters
ρ_{air} = density of air in kilograms per cubic meter = 1.23 kg/m^2
C_D = Drag coefficient constant

The speed dependent force and the aerodynamic drag force are both dependent on the size of the commanded vehicle velocity relative to the vehicle reference velocity, as shown by the C_{R1} and C_A force terms. The discrete version of (3.1) is shown in (3.3), where a difference approximation is made for the evaluation of the derivative term.

$$\tau^*_{\text{rotor}}[n] = \frac{r}{GR} \cdot [m(v^*[n] - v^*[n-1]) + mg(C_{R0} + \frac{C_{R1}}{v_{\text{ref}}} v^*[n] + \frac{C_A}{v_{\text{ref}}^2} v^*[n]^2)] \tag{3.3}$$

For (3.3) to be a valid approximation of the derivative, aliasing of the frequencies in the sampled vehicle velocity command must be avoided. Therefore, the maximum frequency component of the FUDS and HWFET profiles must be less than 0.5 Hz, since the sampling frequency is 1 Hz. Figure 3.4 shows that the spectral components of the FUDS and HWFET profiles are mostly less than 0.1 Hz. Also shown in Fig. 3.4 is that the HWFET profile has the largest average speed, while the FUDS profile has more high frequency components.

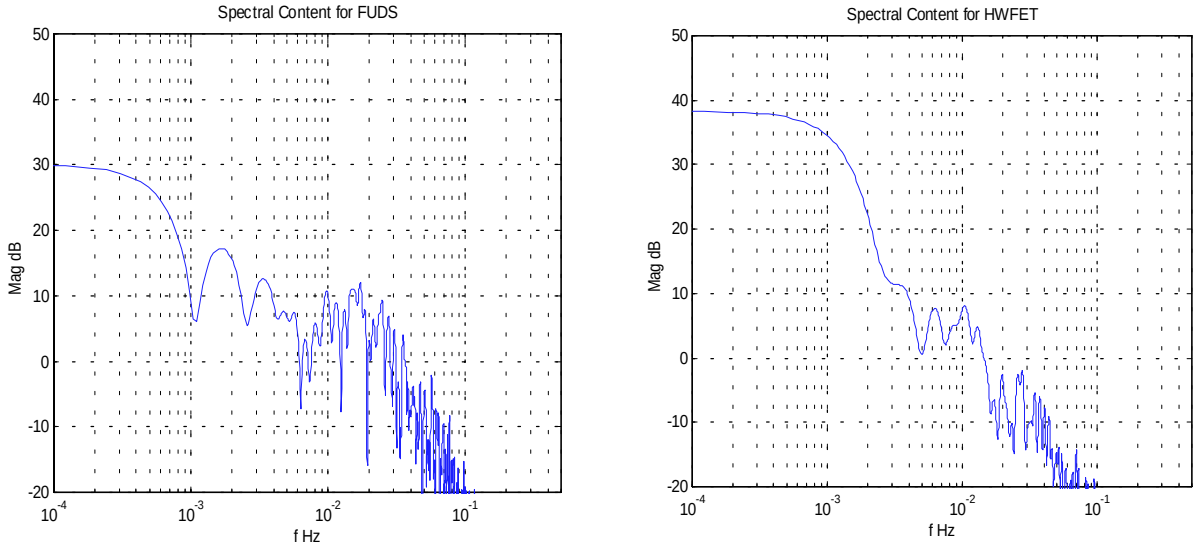


Fig. 3.4. Spectral content of the FUDS and HWFET profiles.

The commanded value of the shaft rpm, which is the speed of the shaft connecting the two induction motors in Fig. 3.1, is determined by the radius of the vehicle wheel and the gearing

ratio of the vehicle differential. Assuming a differential gear ratio of GR_{diff}:1, for every 1 complete revolution of the vehicle drive wheel, there are GR_{diff} revolutions of the drive shaft connecting the induction motors. To determine the number of revolutions of the drive shaft per revolution of the vehicle wheel, the distance traveled by one revolution of the vehicle wheel needs to be determined as shown in Fig. 3.5.

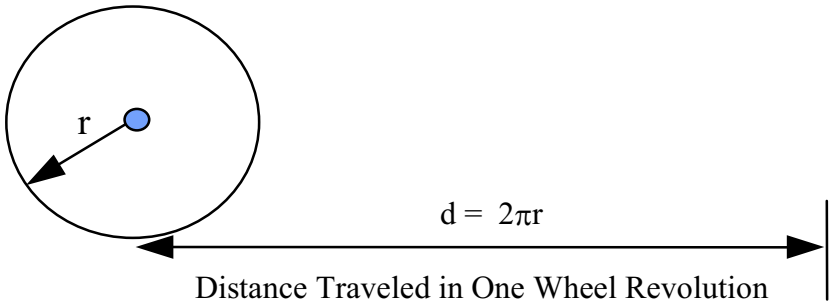


Fig. 3.5. Illustration of the distance traveled in one wheel revolution used in translating vehicle speed to shaft speed.

By translating wheel revolutions per second to radians per second and utilizing the relationship shown in Fig. 3.5, a formula, given by (3.4), is derived for translating the sampled vehicle speed command in km/h, $v^*[n]$, to the dynamometer commanded shaft speed in rpm, $rpm^*_{shaft}[n]$.

$$rpm^*_{shaft}[n] = v^*[n] \cdot \frac{0.2778\ \text{m/s}}{\text{km/h}} \cdot \frac{1\ \text{rot/s}}{2\pi\ \text{r m/s}} \cdot GR_{diff} \cdot \frac{2\pi\ \text{rad/s}}{\text{rot/s}} \cdot \frac{60\ \text{rpm}}{2\pi\ \text{rad/s}} = 2.653 \cdot v^*[n] \cdot \frac{GR_{diff}}{r} \tag{3.4}$$

Equations (3.3) and (3.4) form the discrete time equations that are implemented to model a vehicle driving profile on the dynamometer table. A simulation diagram that utilizes these equations is shown in Appendix B, Figure B-1. The arrangement shown in Fig. B-1 is repeatedly used later in this thesis to simulate the vehicle dynamics.

3.1.2. Description of the Qbasic Drive Profile Generation Program

A Qbasic program was written to use Equations (3.3) and (3.4), which were developed in the previous section, to determine the dynamometer torque and speed commands for the FUDS and HWFET profiles. The complete flowchart for the Qbasic program is shown in Appendix C, Figure C-1. The program written is text based and consists of the main menu shown below:

MAIN MENU

Choose an action from the following list:

(1) Configure vehicle characteristics

(2) Choose a drive profile

(3) Output drive profile commands to data file

(4) End session

Once the program is executed, the first step is to configure the vehicle characteristics by choosing option (1) on the Main Menu. After choosing (1), a submenu appears that prompts the user to choose from a list of predefined vehicles or to enter a custom configuration.

The next step is to choose one of the drive profiles listed under option (2) in the Main Menu. The standard options are either FUDS or HWFET. Choosing option (3) from the Main Menu first prompts the user to enter a name for the output data file generated by the program. Then, the program loads the sampled km/hr data points for the selected drive profile, chosen from option (2) of the Main Menu, into an array. Lastly, the torque and speed commands are computed and output to the data file specified.

A coding system was developed for the data that is output by the Qbasic program so that all the data required for a drive profile could be stored in one ASCII data file. Five different code words are used to indicate the desired action to the HP-VEE Drive Profile Implementation Program, which is discussed in the next section. The code words are shown below:

TORQUE <τ^*_{rotor}>

SPEED <rpm^*_{shaft}>

TREVERSE

TNUETRAL

SFORWARD

The command called TORQUE indicates that the dynamometer shaft torque reference is to be changed to the indicated value of τ^*_{rotor}. Similarly, the command SPEED indicates that the dynamometer shaft speed reference is to be changed to the indicated value of rpm^*_{shaft}. The command TREVERSE indicates that the torque reference should applied to the dynamometer shaft, and the command TNUETRAL means the torque reference should be removed. Lastly, the command SFORWARD indicates that the speed reference should be applied to the dynamometer shaft. An example Command Data File is shown below for the velocity profile in Fig. 3.6 and assuming the mid-size EV truck parameters listed in Appendix A.

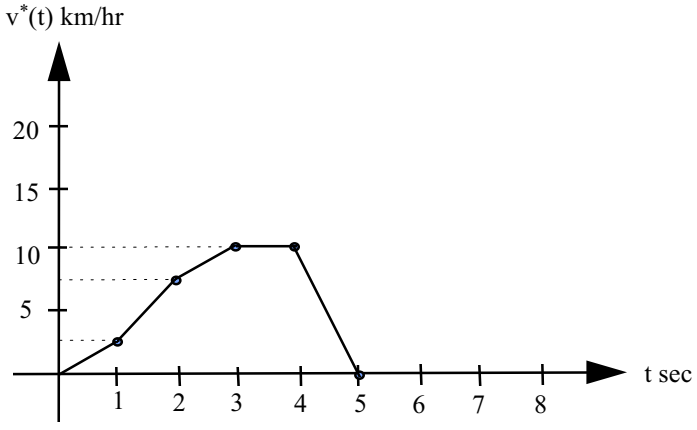


Fig. 3.6. Example vehicle velocity profile.

```
START   SPEED 100
        SFORWARD
        TREVERSE
        SPEED 158
        TORQUE 110.3
        SPEED 274
        TORQUE 218.0
        SPEED 332
        TORQUE 112.1
        SPEED 332
        TORQUE 5.45
        SPEED 100
        TORQUE 0
END     TNUETRAL
```

3.2. Development of Drive Profile Implementation Software Using the Graphical Programming Language HP-VEE

The Command Data Files generated by the Qbasic Drive Profile Generation Program, discussed in the previous section, are interpreted and sent to the drive controllers by the HP-VEE Drive Cycle Implementation Program [23]. The HP-VEE program has direct control of the host PC serial ports, which allows the program to send messages to the speed and torque inverters. Both inverters are equipped to receive messages from an RS-232 connection and to perform the action indicated by the commands given by the messages received.

In this section, the HP-VEE software system is first introduced. Next, the way that the HP-VEE program interprets the commands from the Qbasic Command Data File is discussed. Then, the timing of sending messages is explained and shown in flow chart form. Lastly, the development and description of the HP-VEE program graphical user interface is explained.

HP-VEE is a graphical programming language. The different programming conditional statements, branching statements, and loop control are all represented as objects that are connected together by data lines. As an example, a typical application usually requires push buttons that the user can click on with the mouse. The implementation of a push button in HP-VEE is shown in Fig. 3.7.

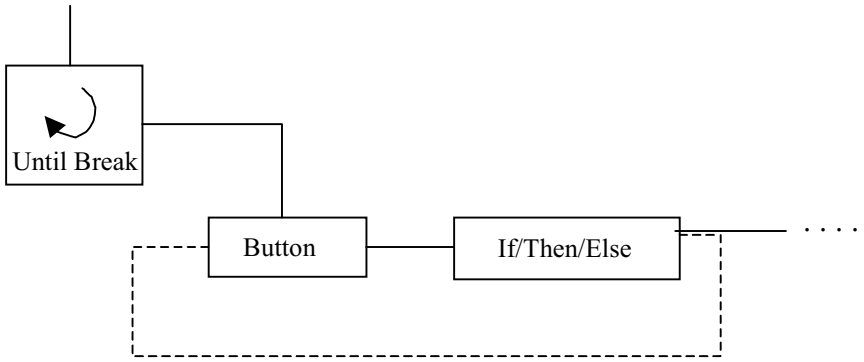


Fig. 3.7. Implementation of a push button in HP-VEE.

As shown in Fig. 3.7, the various HP-VEE programming elements are connected by data lines, which carry the flow of information between the blocks. An "Until Break" block is used to continually poll the data push button. When the push button is pressed, the data line into the If/Then/Else block is pulsed high, which allows the additional blocks in the thread to execute.

Several HP-VEE programming elements are attached to interpret the commands contained in the Qbasic Command Data File. As each command is read in from the Command Data File, the HP-VEE program determines whether a torque command is generated or a speed command is generated. If a command begins with a S, then the command is one that will be sent to the speed regulator. Similarly, if the command begins with a T, then the command will be sent to the torque regulator. The HP-VEE implementation of determining whether a torque or speed command is sent uses a string processing block called strPostChar to determine the first character in the data command. This implementation is shown in Fig. 3.8.

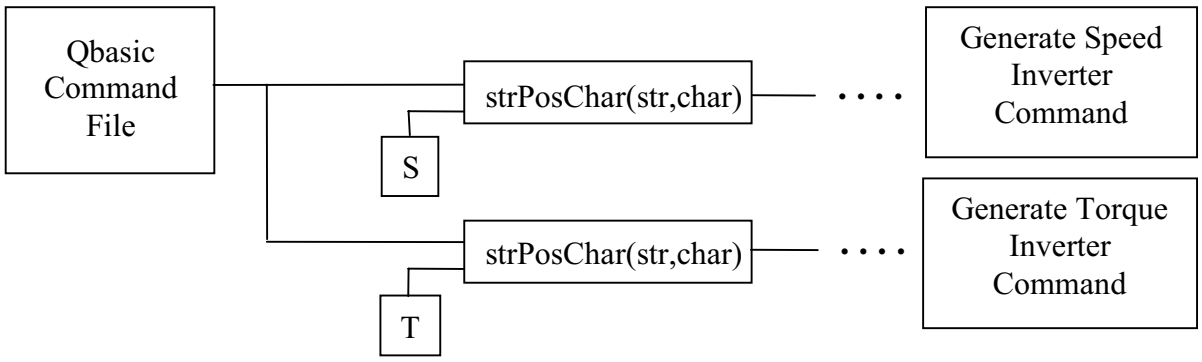


Fig. 3.8. Diagram of how HP-VEE interprets the Command Data File.

Once the inverter commands have been generated, they are sent to the appropriate serial ports of the PC. The speed reference inverter is setup to communicate at 1200 baud, and the torque reference inverter uses 19200 baud. In order to regulate the sending of data commands in one second intervals to the inverters, the HP-VEE program uses the now() function call to read the PC system clock. After sending a torque and speed update, the program loops until the system clock advances by one second, and when the clock has advanced, the next torque and speed updates are delivered. This procedure is shown in flow chart form in Fig. 3.9.

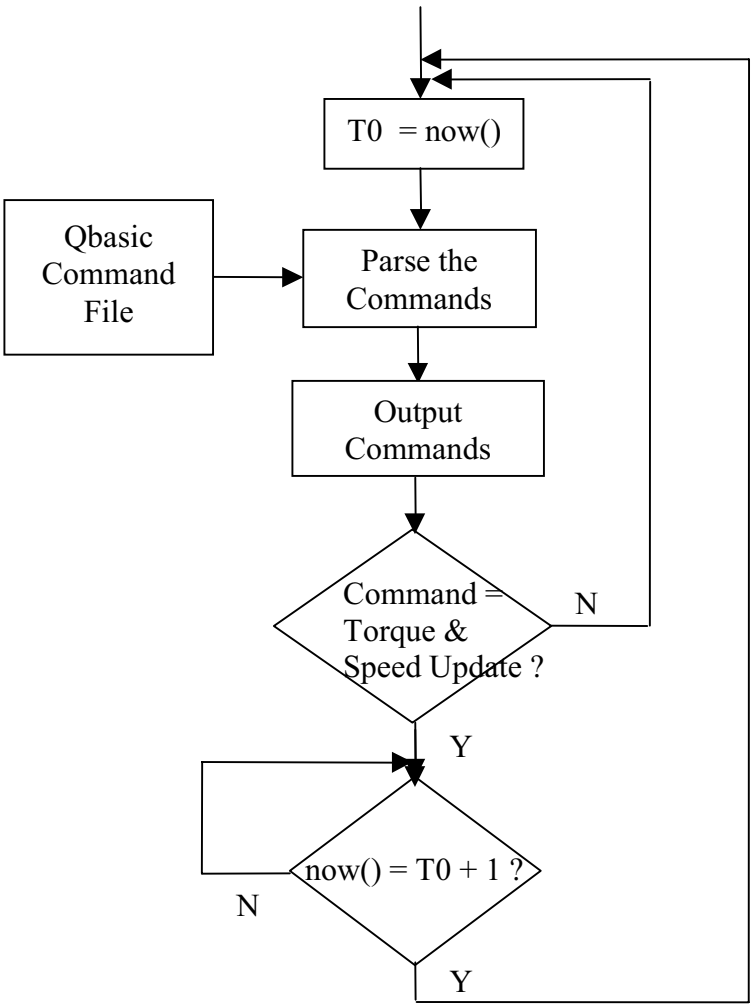


Fig. 3.9. Flow chart for the sending of commands by the HP-VEE program.

As shown in Fig. 3.9, the HP-VEE program outputs commands to the torque and speed inverters as they are read from the Qbasic Command Data File. Torque and speed commands are read consecutively, and then sent out. If non-torque/speed update commands are sent, the program bypasses the one second delay.

To allow for ease of operation, a graphical user interface was created for the HP-VEE Drive Profile Implementation Program. The interface was setup using the Panel View of HP-VEE [46]. The Panel View shows the buttons and screens necessary to operate the program on a colored background. The user interface developed is shown in Fig. 3.10.

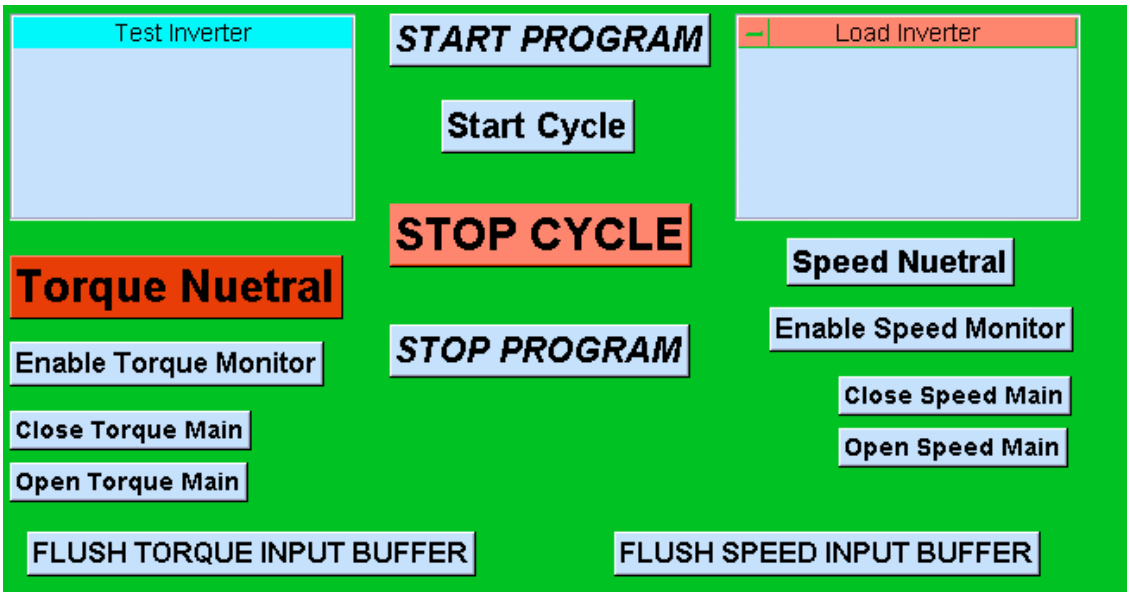


Fig. 3.10. HP-VEE drive cycle implementation program graphical interface developed.

In Fig. 3.10, the buttons in the middle are commands to the HP-VEE program itself concerning the execution of the current drive profile. For example, pressing the "START PROGRAM" button prompts the used to enter the location of the Qbasic Command Data File and initializes the PC COM ports. "Start Cycle" is used to begin the drive cycle on the dynamometer. The button "STOP CYCLE" is used in emergencies to halt the operation of drive profile and remove torque from the shaft as quickly as possible.

The buttons and displays on the left and right sides in Fig. 3.10 are used to send commands to the torque (Test) inverter and to the speed (Load) inverter, respectively. The large display boxes echo the commands sent to the inverters. Buttons are included to enable the monitor mode of each inverter, which enables the RS-232 special testing and remote diagnostics capabilities of the inverters. Buttons are also included to flush the serial buffers, control the contactors, and put each drive in neutral.

This chapter has described the development of a Drive Profile Generation Program written in Qbasic and a Drive Profile Implementation Program written with HP-VEE. These programs are used in Chapter 5 for dynamic performance testing of an inverter using SVM

control versus an inverter using hysteresis current control on the FUDS and HWFET profiles. In the next Chapter, modeling of the actual drive system in Visual C++ is introduced.

Chapter 4. Drive System Simulation and Analysis

Chapter 4 discusses how the entire drive system has been modeled using a higher level language: Visual C++. Different IM drive system component models have been developed through the use of the C++ class definition, which allows for modular program development and ease of use.

One of the reasons computer simulation has been used is to assist in the design and analysis of different modulation methods, specifically SVM and hysteresis (Chapter 5). The simulations developed will also be used to analyze a case study presented in this Chapter. The case study is the simulation of the induction motor stator phase currents. The currents obtained will be used in Chapter 5 to compare induction motor losses. Another use for the simulation models developed in this Chapter will be in Chapter 6, where simulation will assist in the development of the shudder compensator, which is used to damp the driving shaft oscillations.

4.1. Drive System Component Modeling in Visual C++ Using an Object Oriented Approach.

In this section, the way that various EV induction motor drive components have been modeled is described. The C++ programming language is shown to be versatile and powerful for use in drive system simulations. Program development time is decreased by using the object oriented approach, which allows the independent design of the different drive system components using C++ class definitions. Once the independent components are designed, they can be seamlessly connected together through controlled interfaces. Internal variables and parameters can then be isolated between the different objects created, and the objects do not require knowledge of the internal operation of the other objects they are connected to. The

switching model simulations performed in C++ have a fast execution speed with a step size as small as 1 μs. The models are useful in the control design of the drive system, and they are also useful in the design and comparison of modulation algorithms.

The first model described is the induction motor, whose model is derived by using the dynamic equations for the motor in the stationary dq reference frame. Since the equations are nonlinear, a 4th order Runge-Kutta numerical integration algorithm is used to solve the state space system equations. The induction motor can be represented by Equations (4.1)-(4.6), where the system is assumed to be balanced [14], [40], [41]. All of the symbols and variables shown in (4.1)-(4.6) have been defined in the List of Symbols section at the beginning of the thesis.

$$\begin{bmatrix} \dot{i}_{sq} \\ \dot{i}_{sd} \\ \dot{i}_{rq} \\ \dot{i}_{rd} \end{bmatrix} = \frac{-1}{\Delta} \begin{bmatrix} R_s L_r & \omega_r L_h^2 + k1 & -R_r L_h & \omega_r L_r L_h \\ -\omega_r L_h^2 - k1 & R_s L_r & -\omega_r L_r L_h & -R_r L_h \\ -R_s L_h & -\omega_r L_h L_s & R_r L_s & -\omega_r L_r L_s + k1 \\ \omega_r L_h L_s & -R_s L_h & \omega_r L_r L_s - k1 & R_r L_s \end{bmatrix} \begin{bmatrix} i_{sq} \\ i_{sd} \\ i_{rq} \\ i_{rd} \end{bmatrix} + \frac{1}{\Delta} \begin{bmatrix} L_r & 0 & -L_h & 0 \\ 0 & L_r & 0 & -L_h \\ -L_h & 0 & L_s & 0 \\ 0 & -L_h & 0 & L_s \end{bmatrix} \begin{bmatrix} v_{sq} \\ v_{sd} \\ 0 \\ 0 \end{bmatrix} \tag{4.1}$$

$$\Delta = L_s L_r - L_h^2 \tag{4.2}$$

$$k1 = 0 \text{ for stationary reference frame} \tag{4.3}$$

$$k1 = \omega_e \cdot \Delta \text{ for synchronous reference frame} \tag{4.4}$$

$$T_e = \frac{3}{4} P L_m (i_{sq} i_{rd} - i_{sd} i_{rq}) \tag{4.5}$$

$$\dot{\omega}_r = \frac{P}{2} \bullet \frac{T_e - T_L}{J} \bullet \frac{B}{J} \omega_r \tag{4.6}$$

Equations (4.1) are non-linear because some of the coefficients in the system matrix are multiplied by the rotor speed, ω_r, which is a state variable, as shown in (4.6). However, Equations (4.1) can be solved by using a numerical integration technique for their solution. Specifically, a 4th order Runge-Kutta numerical integration algorithm has been used to solve the equations, as shown in the motor.cpp source file listed in Appendix E. A C++ induction motor object was defined by using a class definition, which is shown in Appendix D. The class created is called Motor and is invoked by creating an object of type Motor with the appropriate IM parameters. A public member function is used as the interface to the Motor class. The interface function inputs are v_{sq}, v_{sd}, T_L, and ω_e, and the Motor outputs are i_{sq}, i_{sd}, ω_r, and T_e. The complete motor model can be represented as shown in Fig. 4.1.

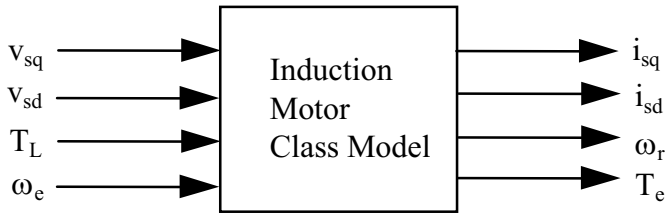


Fig. 4.1. Representation of the Motor class.

Class definitions for a variable band hysteresis current controller and a space vector modulation controller have also been developed. These two controllers utilize the equations developed in Chapter 2. The hysteresis current controller is defined as the ModHyst class, where the ModHyst class header and definition files are shown in Appendices F and G, respectively. The space vector modulation controller is defined as the ModSVM class, where the ModSVM class header and definition files are shown in Appendices H and I, respectively.

A variable band hysteresis controller is created by invoking an object of type ModHyst. A public member function is used as the interface to the ModHyst class. The interface function inputs are I_s^*, θ_s^*, i_{sq}, and i_{sd}, and the ModHyst outputs are the stator phase voltages, v_{sq} and v_{sd}, which can be input to a Motor object. In addition, the variables i_{sq} and i_{sd} are available as outputs

from a Motor object. The term $I_s^* \angle \theta_s^*$ is the commanded stator phase current vector, which must be calculated by the field oriented controller, as shown in Fig. 2.1 in Chapter 2. The complete hysteresis current controller model can be represented as shown in Fig. 4.2.

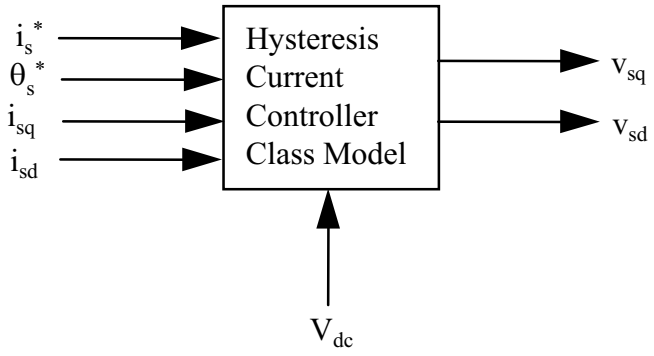


Fig. 4.2. Representation of the ModHyst class.

A space vector modulation controller is created by invoking an object of type ModSVM. For a ModSVM object, the switching frequency is variable, and edge or center-based PWM can be used. There are also three choices for the zero vector selection technique: optimal, direct-direct, and direct-indirect [2], [16]. A public member function is used as the interface to the ModSVM class. The interface function inputs are d_d, d_q, θ_s^*, i_{sq}, and i_{sd}, and the ModSVM outputs are the stator phase voltages, v_{sq} and v_{sd}, which can be input to a Motor object. The inputs i_{sq} and i_{sd} are utilized when optimal zero vector selection is used, as explained in Chapter 2, and are available from a Motor object. Additionally, the term θ_s^* is the commanded reference vector phase angle output from the field oriented controller. The variables d_d and d_q represent the d-axis and q-axis duty cycle commands, respectively. Figure 4.3 shows the representation of a space vector modulation controller.

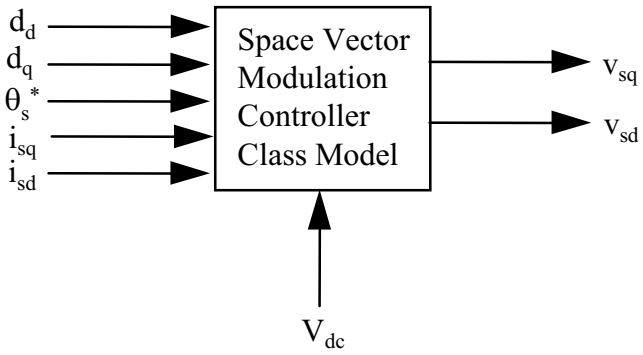


Fig. 4.3. Representation of the ModSVM class.

To create the d_d and d_q duty cycles input into a ModSVM object, a dq current regulator is created by invoking an object of type CurReg. The current regulators are of the PI type and implemented in discrete time in the synchronous reference frame [39]. A public member function is used as the interface to the CurReg class. The CurReg class header and definition files are shown in Appendices J and K, respectively. As shown in Appendices J and K, the interface function inputs to a CurReg object are I_{sq}^*, I_{sd}^*, i_{sq}, i_{sd}, and ω_e^*, and the CurReg outputs are d_q and d_d, which can be input to a ModSVM object. The inputs i_{sq} and i_{sd} are available from a Motor object, and ω_e^* is the commanded stator angular frequency, which is available from the field oriented controller. Figure 4.4 shows the representation of a dq current regulator object.

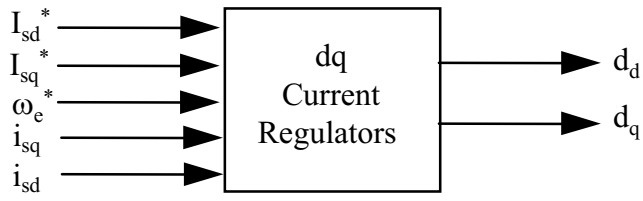


Fig. 4.4. Representation of the CurReg class.

The drive system models developed in this section using Visual C++ have many possible uses in induction motor drive system design. The next section will demonstrate some of the uses of the models developed.

4.2. Case Study: Analysis of Induction Motor Phase Currents Using Simulation

For maximum EV performance and range, the VSI must deliver quality ac current waveforms to the induction motor from the vehicle dc batteries with a minimum of losses. The function of the current controller is to reproduce sinusoidal phase currents in the induction motor based on the commanded torque and flux producing components of the stator phase current.

To see how well the current commands are reproduced, this section uses the drive system component models developed to obtain the induction motor stator phase currents. First, the simulation arrangement for obtaining the induction motor stator phase currents is explained, and then the simulated and experimental phase currents are compared at a particular operating point.

Using spectral analysis of the phase currents that will be obtained, it will be shown that the hysteresis inverter current outputs have many randomly distributed current harmonics. The inverter using optimal zero vector selection SVM at 10 kHz will be shown to yield current outputs that have relatively few current harmonics centered around the switching frequency, which is 10 kHz in this case. Decreased current harmonics will later be shown (Chapter 5) to improve both the inverter and induction motor efficiency.

The GE induction motor parameters, shown in Chapter 1, are used to create a Motor object. The dc-link voltage,V_{dc}, is set to 325 Volts, which is the same voltage used for the dynamometer bench, and the stator flux command is set to 0.304 Wb. The system simulation arrangement shown in Fig. 4.5 is used to obtain the switching waveforms for SVM. A similar arrangement is used to test hysteresis current control, except that a ModHyst object translates the current commands to the applied induction motor phase voltages. In Fig. 4.5, the field oriented controller in the stator flux reference frame, shown in Fig. 1.9, is used. A speed regulator has been added to regulate the rotor speed at the commanded value, and the load torque is applied to the induction motor modeled in the stationary reference, which is a common choice when the switching waveforms are simulated.

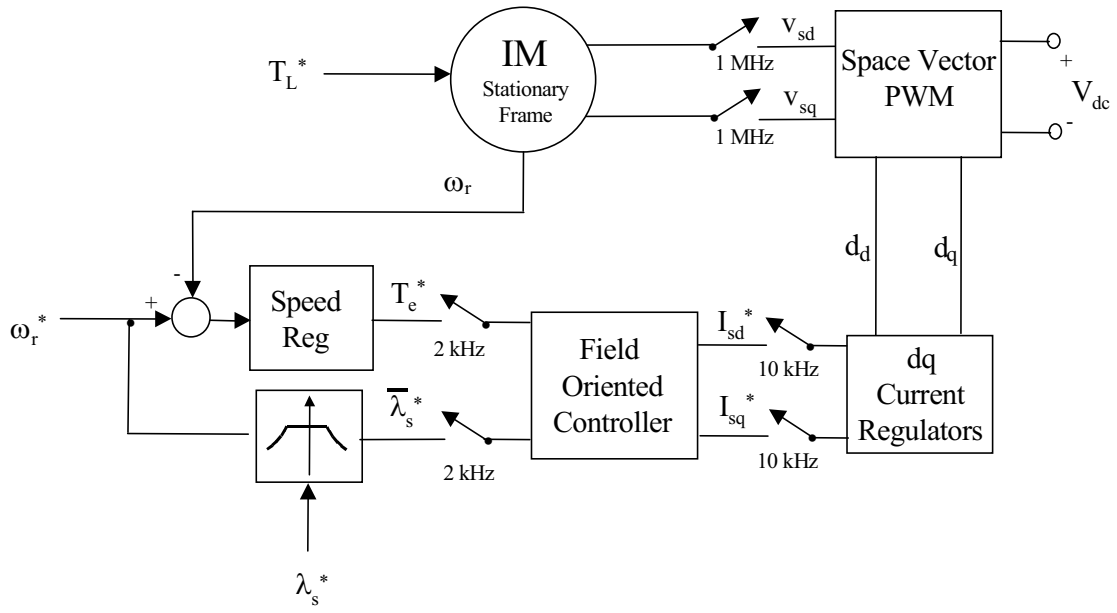


Fig. 4.5. Simulation arrangement for stator flux linkage selection.

Figure 4.6 shows the simulated phase voltages and currents at a rotor speed of 2500 rpm and 40 N-m rotor torque for SVM at 10 kHz, and in addition, the simulated phase voltages and currents for the hysteresis current control introduced in Chapter 2 are shown in the same plots. Figure 4.7 shows the corresponding experimental results measured on the dynamometer test bench. The simulated and experimental results are in general agreement, although the induction motor model used does not incorporate any inherent nonlinearities that could lead to more accurate results [40], [42]. However, for this particular operating point, the harmonic amplitudes are nearly the same, and there is only a slight difference in the location of the harmonics for the hysteresis current control results. From the results obtained, it is shown that the harmonic components, when using hysteresis control, are broadly spaced around an average switching frequency of about 5 kHz (~3.5 kHz from simulation). When using SVM control, the harmonic components are closely spaced around the switching frequency used, which was 10 kHz, and are fewer in number. In Chapter 5, experimental results will be shown that indicate that the lowered current harmonics produced by an inverter with SVM at 10 kHz significantly lowers the system

losses as compared to a the same inverter with hysteresis current control. The lowered system losses will be shown to lead to a large increase in the driving range of the vehicle.

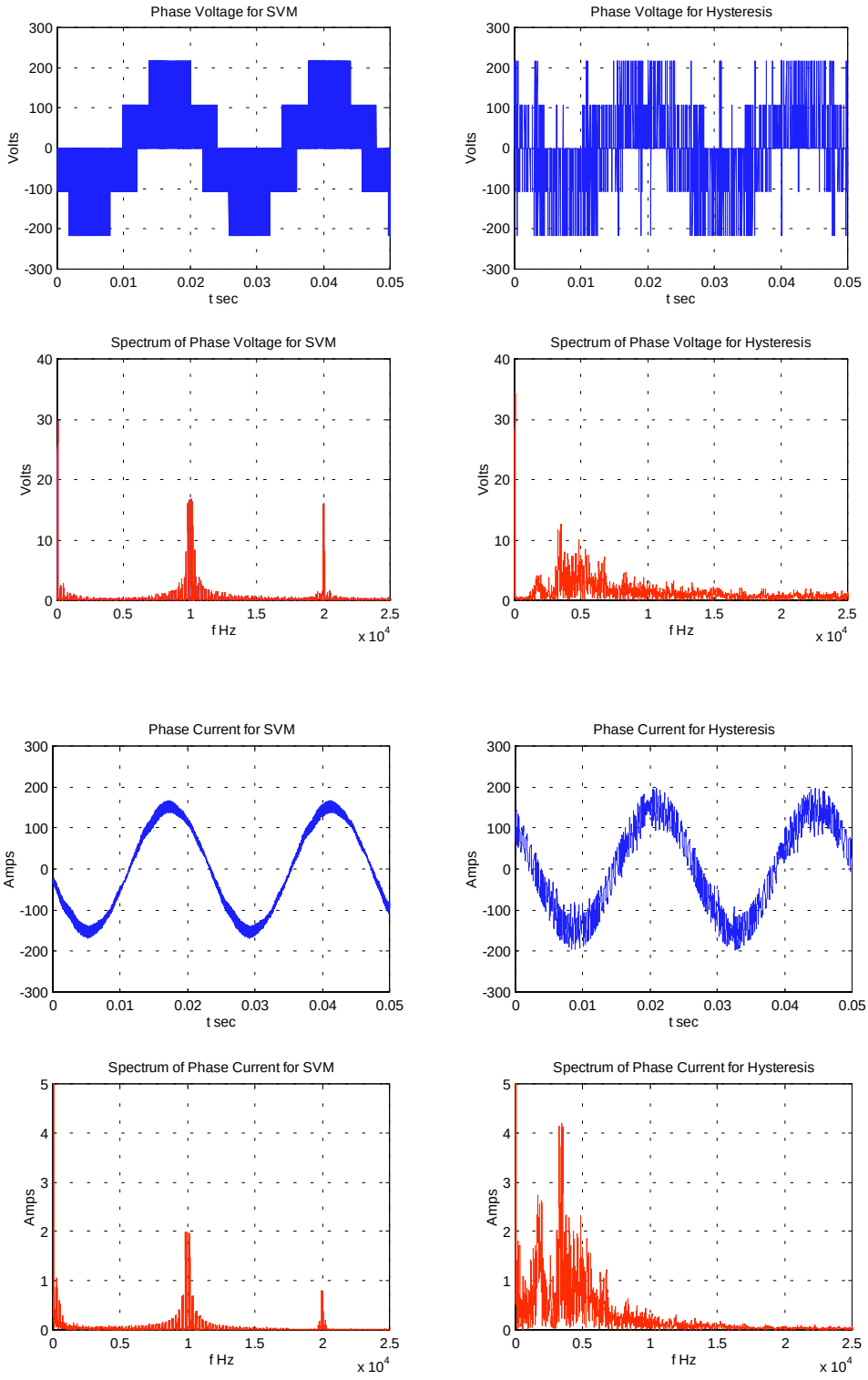


Fig. 4.6. Simulated IM phase voltage and current harmonics at 2500 rpm and 40 N-m.

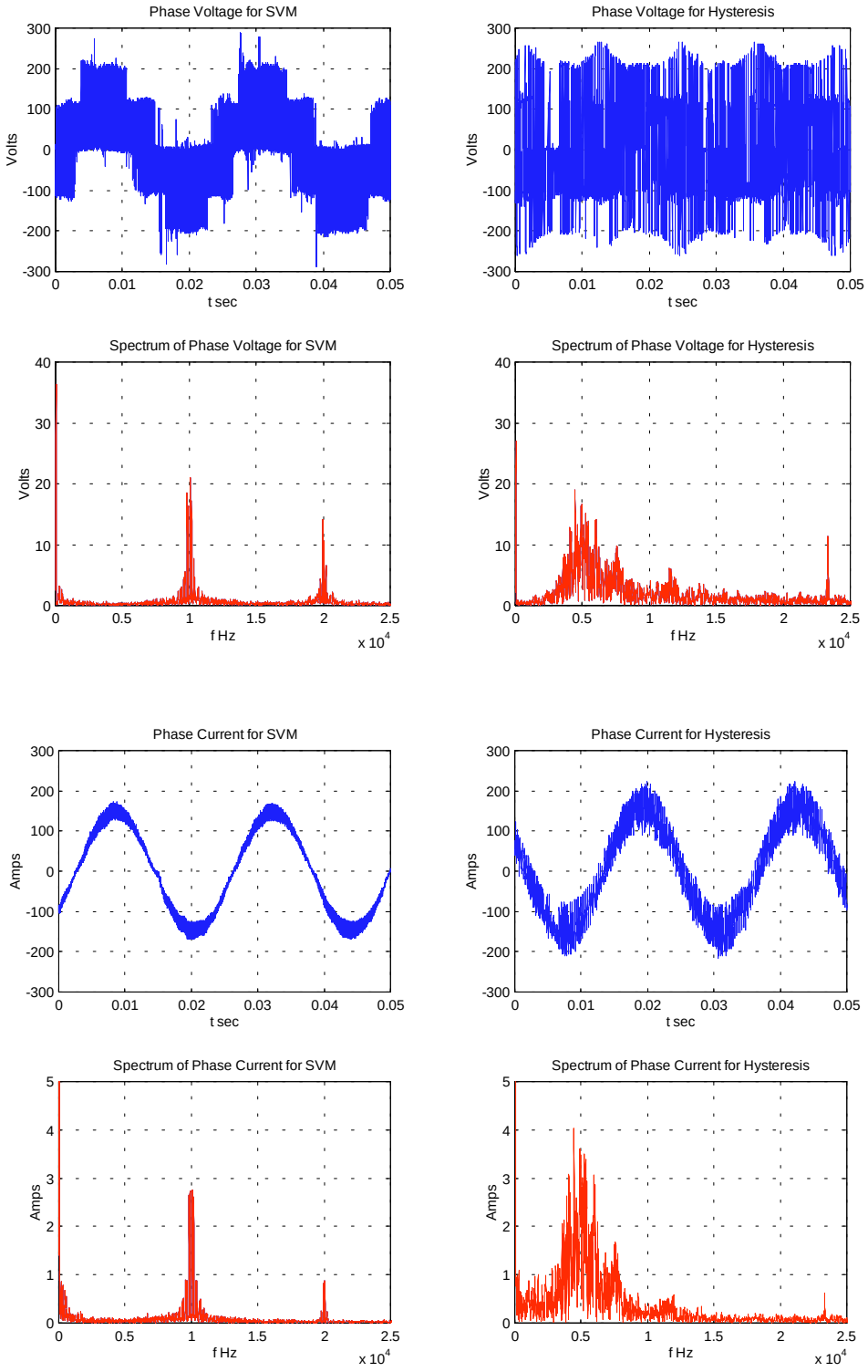


Fig. 4.7. Measured IM phase voltage and current harmonics at 2500 rpm and 40 N-m.

This Chapter has described the development of simulation models for IM drives, and in addition, this Chapter has utilized the models developed to compare the simulated and measured induction motor phase currents, which were in general agreement. The next Chapter, Chapter 5, will further use the simulation models developed in this Chapter when the performance analysis of the modulation algorithms in an EV are discussed.

Chapter 5. Comparison of EV Drive Performance with SVM vs. Hysteresis

In Chapter 5 the IM drive using optimal zero vector selection SVM, whose basic control software is shown in Fig. 1.14, is compared to the same drive with hysteresis current control, whose control software is shown in Fig. 2.1, to determine the benefits of using SVM [37]. Mostly, experimental measurements results are used from the actual drive system that was described in Chapter 1; however, section 5.5 will use the simulation program developed in Chapter 4 to perform an analysis of the impact of increasing the switching frequency with optimal SVM.

In EVs the induction motor torque and speed varies over a broad range depending on the type of driving. The modulation technique used should perform well over the entire load characteristic. This chapter will show that using an inverter with optimal zero vector selection SVM will lead to improved performance over the majority of the torque vs. speed plane when compared to a hysteresis controlled inverter. To help explain the performance improvements that will be obtained, complete steady state efficiencies are experimentally measured over the entire torque versus speed plane and compared. Harmonic loss comparisons are used to explain the improvement in steady state efficiencies that are found. Once all of the current harmonics have been found, induction motor harmonic losses are compared by using a semi-empirical loss model. The steady state results obtained will offer valuable insights into the dynamic, or drive profile, results obtained.

For dynamic testing, the inverter and induction motor losses are compared on the FUDS and HWFET profiles, which are implemented on the dynamometer bench as described in Chapter 3. The efficiency gains are large for both the FUDS and HWFET profiles, especially for the induction motor. The final results will emphasize the importance of the pulse width modulation control strategy on the system performance of an EV.

5.1. Comparison of Drive System Efficiency Maps

This section compares the measured steady state efficiencies of the inverter and induction motor with SVM control versus the measured steady state efficiencies with hysteresis control, where the SVM inverter uses a 10 kHz switching frequency. The efficiency points tested are within the range of a standard electric vehicle inverter on the FUDS or HWFET profiles. After the measured efficiency plots are introduced, the inverter and induction motor power losses are examined in section 5.2 at various steady state torques and speeds, which will offer explanations of the experimental results obtained in this section.

First, the experimental efficiency maps of the two different systems are shown. The efficiencies are shown in Figures 5.1-5.2 as contours on the rotor torque versus speed plane. Measured efficiencies are shown in the Figures within the standard GE induction motor and inverter operating ranges. The areas that are shaded in the Figures are beyond the standard operating region. Note that the shaded part grows as the rpm increases because the field weakening, or constant power, region begins at 5000 rpm, which is where the maximum rotor torque starts to be reduced inversely with speed.

The data to make Figures 5.1-5.2 was measured by using the dynamometer test bench that was described in Chapter 3. The efficiency contours shown in Figures 5.1-5.2 are linearly interpolated across a closely spaced grid of discrete torque and speed test points. To operate the dynamometer at each torque and speed test point, a constant shaft torque reference, τ^*_{rotor}, and shaft speed reference, rpm^*_{shaft}, were applied, and then power measurements were taken once steady state was reached. With the dc link voltage clamped to 325 Volts by an auxiliary dc power supply, the inverter input and output power were measured with a 3-phase power analyzer (Voltech PM3000A) [32]. The measured inverter efficiency was then found as the measured inverter output power divided by the measured inverter input power—See (5.1). The induction motor input power equals the inverter output power, and the induction motor output power was measured by a Himmelstein torque meter, where a non-contact strain gage torque transducer is used on the shaft that connects the two induction motors on the dynamometer [32]. The measured induction motor efficiency was then found as the measured induction motor output power divided by the measured induction motor input power (inverter output power)—See (5.2).

$$\eta_{\text{inverter}} = \frac{\text{Power Inverter}_{\text{out}}}{\text{Power Inverter}_{\text{in}}} \tag{5.1}$$

$$\eta_{\text{IM}} = \frac{\text{Power IM}_{\text{out}}}{\text{Power Inverter}_{\text{out}}} \tag{5.2}$$

The system efficiency was found by multiplying the inverter and induction motor efficiencies:

$$\eta_{\text{system}} = \eta_{\text{inverter}} \times \eta_{\text{IM}} \tag{5.3}$$

Figures 5.1(a)-(c) show the efficiency contours for the inverter with SVM, the induction motor with SVM, and the system efficiency with SVM, respectively. Figures 5.2(a)-(c) show the efficiency improvement contours obtained when SVM control is used rather than hysteresis control. Note that "improvement" is defined as the percent increase in efficiency that is obtained by using SVM as opposed to hysteresis for each of the torque and speed points that were tested.

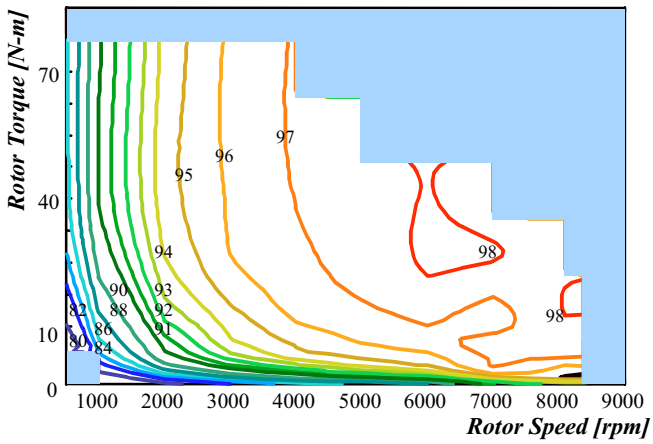


Fig. 5.1(a). SVM inverter measured efficiency plot.

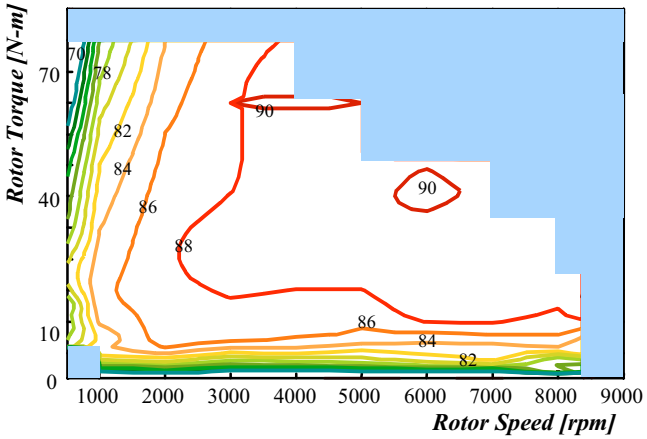


Fig. 5.1(b). SVM induction motor measured efficiency plot.

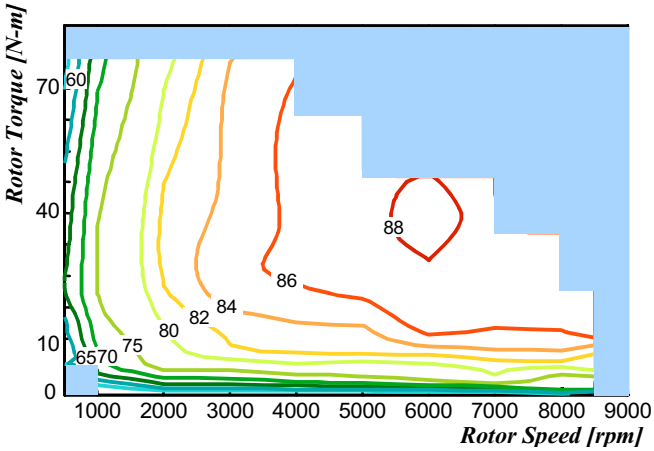


Fig. 5.1(c). SVM system measured efficiency plot.

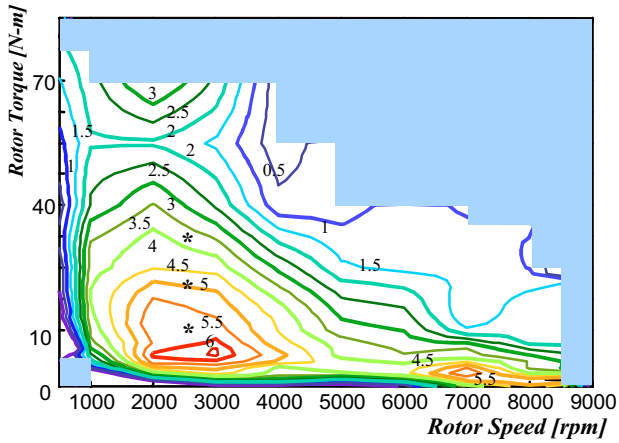


Fig. 5.2(a). Inverter efficiency improvement with SVM.

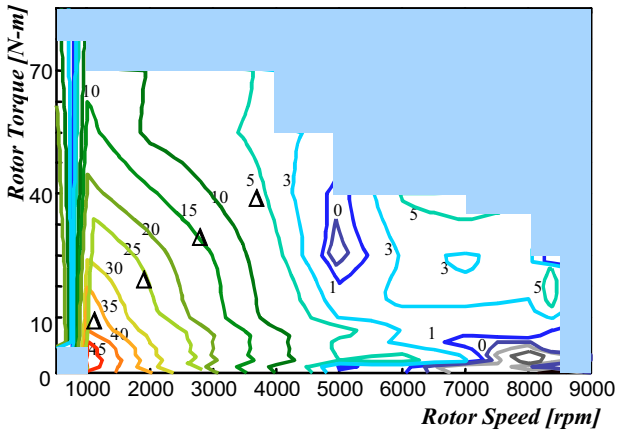


Fig. 5.2(b). Induction motor efficiency improvement with SVM.

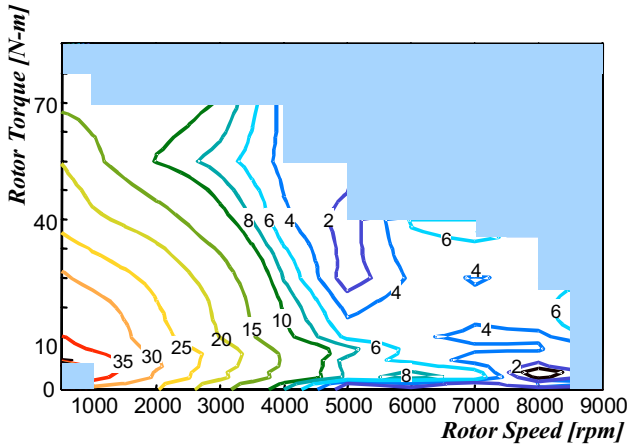


Fig. 5.2(c). System efficiency improvement with SVM.

From Figures 5.1-5.2 the system with SVM shows improvement in both the inverter and induction motor efficiency over the majority of the torque vs. speed plane. The largest inverter efficiency increase when using SVM are for the speeds between 2000 to 3000 rpm for all loads and also at 7000 rpm under light load. The largest induction motor efficiency increases when using SVM are for the rotor speed between 1000 to 4000 rpm for all loads, especially at light load.

5.2. Analysis of Drive System Losses

In this section, the percentage decrease in the inverter power losses when using SVM are analyzed at the group of test points shown in Fig. 5.2(a) that are marked with a star, *. The percentage decrease in the induction motor power losses when using SVM are analyzed at the group of test points shown in Fig. 5.2(b) that are marked with a triangle, Δ. The test points are located in the areas of the largest inverter and induction motor efficiency improvements. Each test point in the torque versus speed plane defines the induction motor shaft power for that particular test which is given by (5.4).

$$P_{rotor} = \text{Rotor Speed in rpm} \times \frac{2\pi}{60} \times T_e \tag{5.4}$$

All comparisons between the hysteresis and SVM inverter are made at the same commanded shaft power or particular location in the torque vs. speed plane.

5.2.1 Inverter loss comparison

The inverter power losses are compared on the grid of test points shown by a star in Fig. 5.2(a). Comparisons between total inverter losses on a varying dc link are examined and the THD of the induction motor phase currents are compared.

The inverter input and output powers are measured for several different values of the dc link voltage. The switching loss is expected to decrease in proportion to the decrease in the dc link voltage. The dc link voltage is varied from 330 volts to 250 volts. The inverter power loss will decrease as the dc link voltage is decreased. Figure 5.3 shows the comparison of the inverter power loss as the dc link voltage is varied for a rotor speed of 2500 rpm and 10, 20, and 30 N·m of rotor torque. From Fig. 5.3, it is shown that when hysteresis control is used, there is a steeper rolloff of the inverter power loss, indicating larger switching loss. Additionally, the inverter with hysteresis control is expected to have larger conduction losses in the diodes and IGBTs due to the increased THD of the phase currents. The comparisons between the rolloff rate, or slope, of the inverter power loss with decreasing dc link and the phase current THDs are given in Table 5.1. All comparisons are given as the percentage of the value when SVM control is used as opposed to hysteresis control.

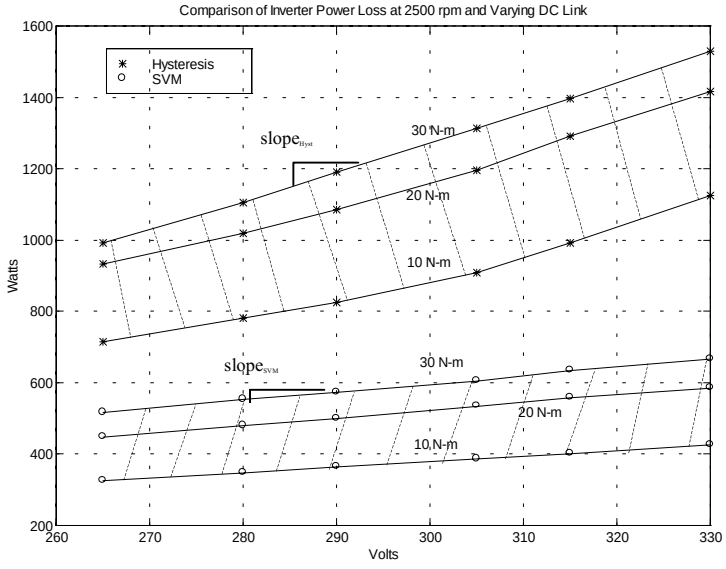


Fig. 5.3. Comparison of inverter power loss at 2500 rpm and varying DC link.

Table 5.1: Comparison of inverter loss rolloff rate with decreasing dc link and phase current THD at 2500 rpm and 10, 20, and 30 N·m.

rpm	τ N·m	THD_{SVM} THD_{Hyst}	$slope_{SVM}$ $slope_{Hyst}$
2500	10	45.3%	25.0%
2500	20	50.0%	28.5%
2500	30	45.5%	27.5%

Space vector PWM with an optimal zero vector selection routine has decreased both the conduction loss of the inverter, due to a reduced THD, and the switching power losses of the inverter, due to decreased switching, which is clearly indicated for the test points at 2500 rpm. The generalized result is improved inverter efficiency across the majority of the torque vs. speed plane as shown in Fig. 5.2(a).

Improved inverter efficiency can result in reduced heat sink size and cooling requirements of the semiconductor devices. The reduced cooling requirements are important for an EV because smaller inverter heatsink cooling pumps and fans can be used, which is a key issue due to the limited space that is available. The system is also more reliable because if the cooling system ever fails the EV can go farther before the inverter faults due to the inverter heatsink overheating.

5.2.2. IM Loss Comparison

When the induction motor is supplied by a nonsinusoidal voltage supply, there are added power losses due to the harmonic components applied that are in addition to the losses due to the fundamental components [24], [25]. Figure 5.4 is an example of a typical PWM voltage waveform. Clearly, the waveform is nonsinusoidal.

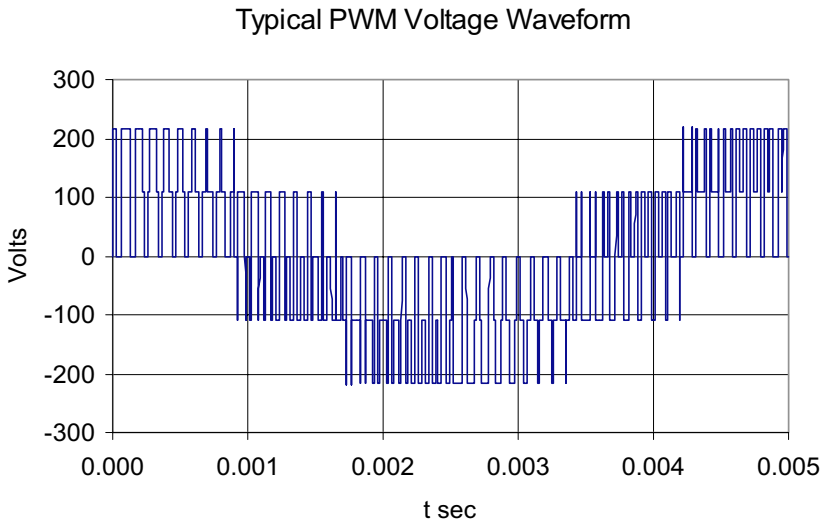


Fig. 5.4. Example of a PWM Voltage Waveform, F = 200 Hz.

The usual way to view the effect of the PWM waveform of Fig. 5.4 on the induction motor is to determine the loss due to each harmonic and then add the separate losses to get the total loss using the principle of superposition. The PWM voltage wave applied to the induction motor can then be viewed as shown in Fig. 5.5, where the separate voltage harmonics are connected in series across the motor terminals [24].

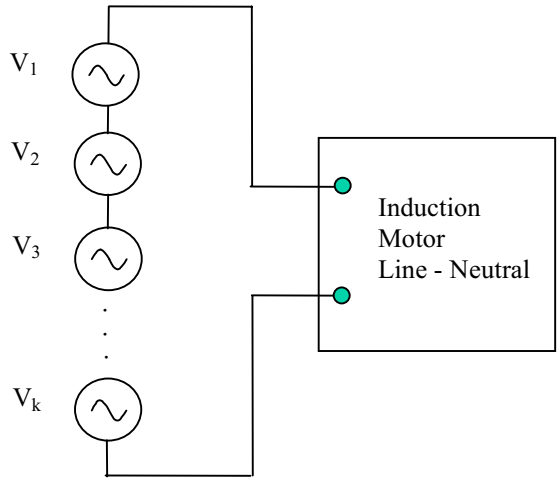


Fig. 5.5. Induction motor with nonsinusoidal voltage excitation equivalent circuit.

The voltage harmonics that are applied result in phase current harmonics that cause induction motor harmonic losses. Several different techniques have been proposed in the literature for the determination of the harmonic losses in inverter fed induction machines:

1. Finite Element Based Analysis
2. Direct Measurement [29], [30]
3. Semi-empirical Motor Loss Models [26], [27]

Technique 1 is considered to be beyond the scope of this thesis. Technique 2 is based on making precise measurements of the induction motor input and output power during locked rotor, no load, or arbitrary load. Technique 2 is not chosen due to the uncertainty of the induction motor parameters, as well as the unknown mechanical losses of the 4.29:1 planetary gear that couples the rotor to the output shaft of the GE induction motor. Therefore, technique 3 is used as the testing method for determination of induction motor harmonic losses. Technique 3 allows for a comparative study of different PWM modulation strategies without precise knowledge of the motor parameters or mechanical losses by using measured current harmonics to obtain loss estimates.

5.2.2.1 Development of a Procedure to Measure IM Phase Current Harmonics up to 30kHz

In this thesis, the induction motor losses are analyzed by measuring the phase currents and then using a harmonic analysis technique to find the current harmonics needed for use in semi-empirical motor loss model. The harmonic components of the motor phase current in steady state are determined up to 30 kHz. To find the harmonic currents, the first thing that is done is to take the FFT of the sampled and windowed induction motor stator phase current, which has a fundamental frequency F. This procedure is shown schematically in Fig. 5.6.

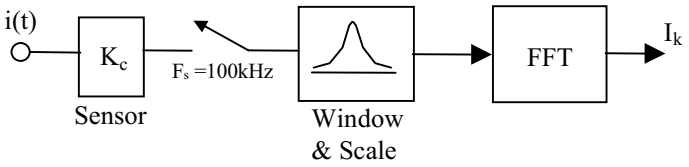


Fig. 5.6. Current sampling and determination of harmonics.

The window used is the Hamming window because of its relatively narrow mainlobe width and good sidelobe attenuation [36]. The following definitions are used for the harmonic analysis carried out:

- N Number of samples of the time waveform
- L FFT length after zero padding of the time waveform
- F_s Sampling frequency
- T_s Sampling period = $1/F_s$
- F Time waveform fundamental frequency
- T Time waveform fundamental period = 1/F
- n Time sample
- m Spectral sample number
- k Harmonic frequency sample number

Enough samples of the current waveform must be taken in order to get good frequency and amplitude resolution in the resulting spectrum and hence accurate measurement of the current harmonics. With a given sampling frequency, F_s, and fundamental current waveform frequency, F, the number of time samples, N, should be greater than or equal to the value given by (5.5) to have a good separation of the measured harmonics.

$$N >= \frac{2}{F / F_s} \tag{5.5}$$

Additionally, the sampling frequency, F_s, must be large enough to satisfy the Nyquist Criterion, which states that the minimum sampling frequency must be two times greater than the highest frequency component contained in the time domain waveform.

Once the correct samples of the current time waveform have been obtained, they are first windowed by a length N hamming window and then zero padded to a length L. L is chosen as some large power of two such as 2^{15} or 2^{16}. Zero padding means more spectral samples will be obtained, which will result in more accuracy in locating the harmonic peak amplitudes and frequencies [36]. A power of two is used because the FFT is optimized for processing input data lengths that are a power of two. The sampled, windowed, and zero padded current samples are next processed as shown in (5.6) to determine the frequency samples of the induction motor phase current spectrum. The expression in (5.6) is the DFT of the phase current and can be easily evaluated using the FFT algorithm in a software package such as MATLAB.

$$I_m = \frac{1}{SF} \sum_{n=0}^{L-1} i(nT_s)\ e^{-j\frac{2\pi}{L}mn} \qquad m=0,1,\ldots,L-1 \tag{5.6}$$

In (5.6) the term SF is a constant and is determined as shown in (5.7) for the Hamming window.

$$SF = \sum_{n=0}^{N-1} [0.54 + 0.46\cos(\frac{2\pi}{N-1}(n - \frac{(N-1)}{2}))] \tag{5.7}$$

The frequency values in Hz that correspond to the stator phase current spectral samples, given by (5.6), are given by (5.8).

$$f_m = \frac{F_s}{L} \cdot m \qquad m=0,1,\ldots,L-1 \tag{5.8}$$

When all of the I_m have been computed, the next step is to search for the peak values of current spectrum samples vs. the frequency samples, I_m vs. f_m, to determine the current harmonics. Finding the peaks in the spectral samples can be performed in a looping algorithm, whose flow chart is given in Fig. 5.7.

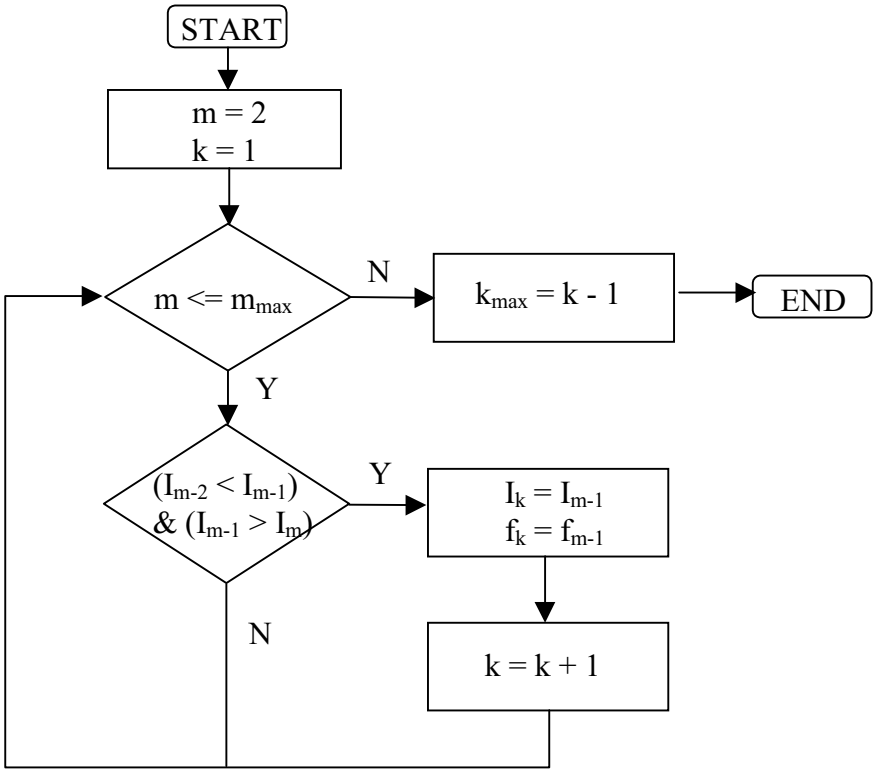


Fig. 5.7. Flowchart for extraction of current harmonics from the current spectrum samples.

In Fig. 5.7, m_{max} represents the maximum frequency where harmonics are searched for and is given as (5.9).

$$m_{max} = L \cdot \frac{F_{max}}{F_s} \tag{5.9}$$

In (5.9), F_{max} has been chosen as 30 kHz.

The sample values I_k are half of the harmonic sine wave peak amplitude because the FFT gives the two-sided spectrum between zero Hz and F_s. If a peak value of current is found that is below an appropriate noise level, it is not counted as a current harmonic. For this thesis, 0.1 A was chosen as the noise floor; therefore, all valid harmonics have to be greater than 0.1 A to be counted.

The accuracy of the current harmonics, I_k, that are computed can be checked by comparing the estimated power, determined from the current harmonics found, and the measured power, determined from the phase current samples measured. The estimated power across a 1Ω reference is found by using the computed I_k,, as shown in (5.10). The measured power across a 1Ω resistor, computed by using the original current samples, is found as shown in (5.11).

$$P_{est} = I_0^2 + 2 \sum_{k}^{k_{max}} I_k^2 \tag{5.10}$$

$$P_{meas} = \frac{F}{F_s} \sum_{n=1}^{int(\frac{T}{T_s})} (i(nT_s))^2 \tag{5.11}$$

In every case tested, P_{est} was always very slightly less than P_{meas} and is what was expected because only harmonics up to 30kHz were counted in (5.10). For any experiment, (5.10) must be within a certain tolerance of (5.11) or the harmonic samples are determined to be invalid. The expression in (5.12) must be satisfied for an appropriately small tolerance so that the current

harmonics computed are known to be valid. In this work, ε_{tol} was chosen to be very small, because the current harmonics were found accurately due to the proper selection of the sampling frequency, F_s, number of current waveform samples, N, and amount of zero padding.

$$\frac{P_{meas} - P_{est}}{P_{meas}} < \varepsilon_{tol} \tag{5.12}$$

Appendix L lists the complete stator phase current harmonic determination routine as a Matlab m-file.

Once all of the I_k have been computed, the current waveform THD can be estimated from Equation (5.13).

$$THD = \frac{\sqrt{\sum_{k=2}^{k_{max}} I_k^2}}{I_1} \tag{5.13}$$

5.2.2.2 Comparison of IM Losses Using a Semi-Empirical Loss Model

Using the harmonic analysis software described in the previous section, this section compares the induction motor power losses for hysteresis and SVM on the grid of test points shown by a triangle in Fig. 5.2(b). Comparisons between the total induction motor losses and the separate motor copper and iron harmonic losses are made using a semi-empirical motor loss model based on the knowledge of the phase current harmonics, as obtained by the procedure explained in section 5.2.2.1. The power losses in an induction motor between the line input to the shaft output consist of three main components as shown in (5.14).

$$P_{mot\ loss} = [P_{mech} + P_{Fe}] + [P_{Cu}] = [P(\omega_r, T_e) + P_{iron}] + [P_{rot} + P_{stat}] \tag{5.14}$$

The mechanical losses, P_{mech}, are due to the induction motor friction, windage, and planetary gear losses. The mechanical losses are assumed to depend in a non-linear way on the rotor speed and torque, where the functional form for the mechanical losses, $P(\omega_r, T_e)$, is unknown. The copper losses, P_{Cu}, are due to the I^2R power losses in the stator and rotor resistances, P_{stat} and P_{rot}, respectively, and these losses are increased when a nonsinusoidal voltage supply is used because of the high-frequency harmonics that are present. The iron losses, P_{iron}, are due to the hysteresis and eddy current losses in the machine iron as well as the stray load losses, such as end-leakage and skew-leakage fluxes. It has been suggested in the literature that PWM type waveforms can lead most significantly to increases in the iron losses [29], [30], [31].

Appropriate expressions for the increased copper and iron losses due to nonsinusoidal voltage supplies are frequency dependent. This is because the induction motor parameters change with the different applied harmonic stator frequencies [26], [28]. The semi-empirical induction motor loss model developed in [26] is used to determine appropriate expressions for the separate induction motor losses based on knowledge of the stator phase current harmonics.

First, the values for P_{rot} and P_{stat} can be calculated by Equations (5.15)-(5.16):

$$P_{rot} = 3 \cdot I_r^2 \cdot R_r + 3 \cdot \sum_{k=2}^{k_{max}} R_{rAC} \cdot I_{sk}^2 \tag{5.15}$$

$$P_{stat} = 3 \cdot I_s^2 \cdot R_s + 3 \cdot \sum_{k=2}^{k_{max}} R_{sAC} \cdot I_{sk}^2 \tag{5.16}$$

In (5.15) and (5.16), the Cu loss due to the fundamental current component is shown as the first loss component, and the second component, where a summation is used, determines the Cu losses due to the harmonic stator phase currents. The term I_s is the RMS value of the fundamental stator phase current, which can be directly measured, and the term I_r is the RMS value of the fundamental rotor current, which can be approximated by Equation (5.17) because near perfect current regulation is assumed:

$$I_r \cong \frac{I_{sq}^*}{\sqrt{2}} = \frac{\frac{2}{3} \cdot \frac{T_e^*}{\lambda_s^*}}{\sqrt{2}} \tag{5.17}$$

The terms R_{rAC} and R_{sAC} in Equations (5.15) and (5.16) represent the resistance that is seen by the high frequency harmonic components of the stator phase current, the I_{sk}. These resistances are larger than the DC resistances due to high frequency skin and proximity effects and can be estimated by (5.18) and (5.19):

$$R_{rAC} = R_r \cdot [1 + K_1(f)] \tag{5.18}$$

$$R_{sAC} = R_s \cdot [1 + K_2(f)] \tag{5.19}$$

In the above equations, the K terms are frequency dependent and represent how the resistances vary with frequency, and these terms depend on the induction machine size and geometry. Approximations for $K_1(f)$ and $K_2(f)$ for the GE induction motor are shown below [26]:

$$K_1(f) \approx k_{Cu1} \cdot \sqrt{f} = 0.1 \cdot \sqrt{f} \tag{5.20}$$

$$K_2(f) \approx k_{Cu2} \cdot f^2 = 1^{-7} \cdot f^2 \tag{5.21}$$

Next, the iron loss, P_{iron}, in Equation (5.14) can be approximated by the following Equation:

$$P_{iron} = \sum_{k=1}^{k_{max}} K_3(f) \cdot I_{sk}^2 \tag{5.22}$$

The term $K_3(f)$ in Equation (5.22) represents the scaling factor that is multiplied by the squared harmonic currents in order to find the harmonic iron losses. $K_3(f)$ must account for the influence of the core losses and their variation with applied frequency and voltage, the skew leakage losses, the stator and rotor end leakage losses, and the variation of the leakage inductance with frequency:

$$K_3(f) \approx k_{Fe} \cdot f^{1.18} \tag{5.23}$$

In (5.23), k_{Fe} is an unknown scaling constant.

Now that expressions for the various harmonic losses have been determined, the different losses between hysteresis and SVM are compared. The percentage decrease in harmonic losses when using SVM as opposed to hysteresis control are analyzed. Table 5.2 summarizes the percentage of the total induction motor power loss measured for SVM as compared to hysteresis control on the grid of test points in Fig. 5.2(b).

Table 5.2: Comparison of Total Induction Motor Loss = $P_{mot\ loss}$

rpm	T_e N-m	$P_{mot\ loss}$ Hyst	$P_{mot\ loss}$ SVM	$\frac{P_{mot\ loss SVM}}{P_{mot\ loss Hyst}}$
1000	10	1302 W	-----	-----
2000	20	2055 W	411 W	20.0%
3000	30	2994 W	678 W	22.6%
4000	40	3965 W	1066 W	26.9%

The ratio of harmonic losses (losses that are in addition to the fundamental applied waveform losses) with SVM control as compared to hysteresis control are shown as a percentage in Table 5.3.

Table 5.3: Comparison of Separate Induction Motor Harmonic Losses

rpm	T_e N·m	$P_{rot}(f_k)_{SVM}$ / $P_{rot}(f_k)_{Hyst}$	$P_{stat}(f_k)_{SVM}$ / $P_{stat}(f_k)_{Hyst}$	$P_{iron}(f_k)_{SVM}$ / $P_{iron}(f_k)_{Hyst}$
1000	10	3.78%	9.10%	5.30%
2000	20	12.2%	25.7%	17.7%
3000	30	19.0%	38.1%	28.1%
4000	40	23.5%	54.3%	39.7%

Table 5.3 shows that with SVM , all of the calculated harmonic losses are significantly lowered. Table 5.4 compares the total calculated Cu losses, which are equal to P_{rot} and P_{stat}, as determined from Equations (5.15) and (5.16), respectively.

Table 5.4: Comparison of Total Calculated Cu Losses, $P_{Cu} = P_{rot} + P_{stat}$

rpm	T_e N-m	P_{Cu} Hyst	P_{Cu} SVM
1000	10	450 W	132 W
2000	20	882 W	291 W
3000	30	1266 W	472 W
4000	40	1420 W	729 W

The total calculated Cu losses for SVM are at least half as large as the Cu losses for SVM. Additionally, the iron plus mechanical losses can now be determined as the total measured induction motor losses, $P_{mot\ loss}$, minus the calculated Cu losses, P_{Cu}. The results of this computation are shown in Table 5.5, where large decreases in the mechanical plus iron losses are predicted.

Table 5.5: Comp. of Calculated Mechanical plus Iron Losses ,$P(\omega_r, T_e) + P_{iron} = P_{mot\ loss} - P_{Cu}$

rpm	T_e N-m	P_{Cu} Hyst	P_{Cu} SVM
1000	10	852 W	-----
2000	20	1173 W	120 W
3000	30	1728 W	206 W
4000	40	2545 W	337 W

The results found in this section are not unexpected because the spectrums of the current, shown in section 4.2, revealed that with SVM control there is a large reduction in the harmonic content of the phase current of the induction motor. The reduction in current harmonics helps explain the reduced induction motor losses measured in this section.

5.3. FUDS and HWFET Profiles Performance Comparison

The first part of Chapter 5 has shown measurement and test results to explain the large system efficiency increases with SVM control as opposed to hysteresis current control. In this section, the average driving efficiencies on the FUDS and HWFET profiles, called the transient efficiencies (output energies divided by input energies), are determined. The comparison of the transient efficiencies gives an estimate of the improved vehicle range due to the change from hysteresis control to SVM control. A larger increase in transient efficiency obtained on the FUDS profile as opposed to the HWFET profile will be shown. The large FUDS transient efficiency increase is explained by showing the separate operating point densities for the FUDS and HWFET profiles and correlating them with the areas of largest efficiency gains when SVM control is used.

The required rotor speed and torque for the FUDS and HWFET profiles for a mid-size truck are computed from the simulation diagram in Appendix B, as described in Chapter 3, and plotted in Fig. 5.8-9.

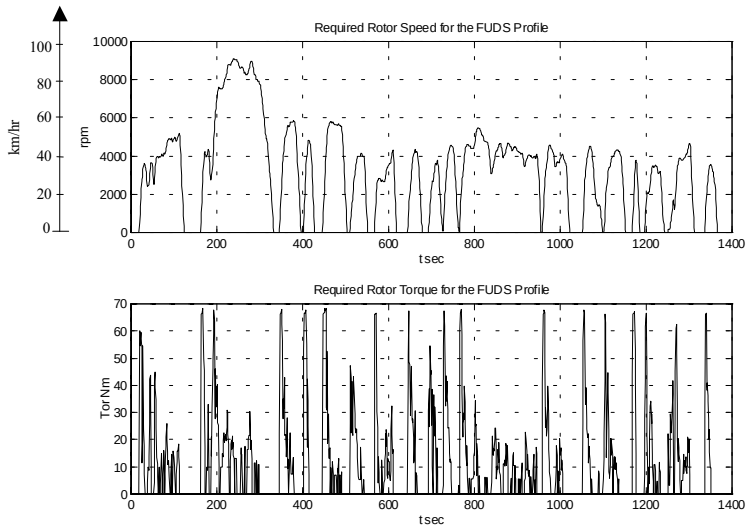


Fig. 5.8. Required rotor torque and speed for the FUDS profile.

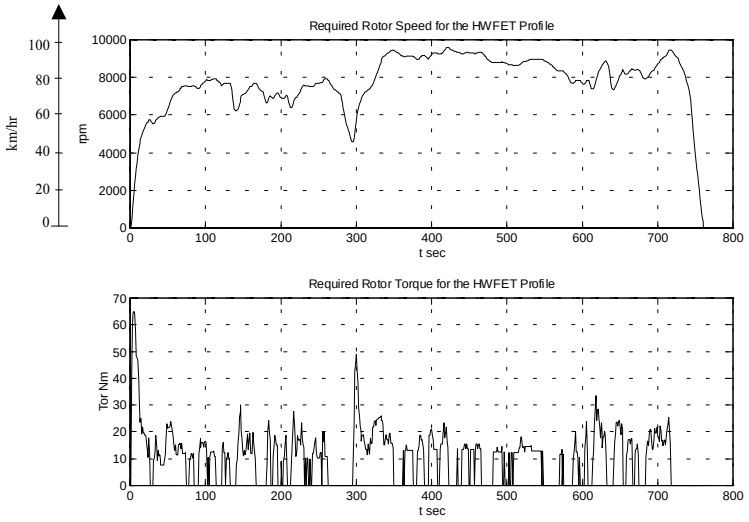


Fig. 5.9. Required rotor torque and speed for the HWFET profile.

As shown in Fig. 5.8, the FUDS profile consists mostly of low speed high torque accelerations. As shown in Fig. 5.9, the HWFET profile consists mostly of high speed low torque accelerations and constant speed cruising torques. Also, note that only positive torques have been considered. If regeneration was turned on, it could be modeled as a constant value of negatively applied torque when the vehicle decelerates; however, energy recovered through regeneration was considered to be a small fraction of the energy consumed and so was ignored.

To compare the drive system with SVM and hysteresis on the FUDS and HWFET profiles, the energy needs to be measured at each point in the system, because the transient efficiency is going to be defined as output energy divided by input energy:

$$\eta_{tran} = \frac{E_{out}}{E_{in}} \times 100 \tag{5.24}$$

To make transient efficiency measurements, the energy used at each point of the drive system needs to be either directly measured or estimated. The drive system can be represented as shown below in Fig. 5.10. The inverter input energy, $E_{inv\ in}$, and output energy, $E_{inv\ out}$, can be directly measured by the power analyzer (Voltech PM3000A) during a drive cycle test to determine the inverter transient efficiency [32]:

$$\eta_{tran\ \ inverter} = \frac{E_{inv\ \ out}}{E_{inv\ \ in}} \times 100 \tag{5.25}$$

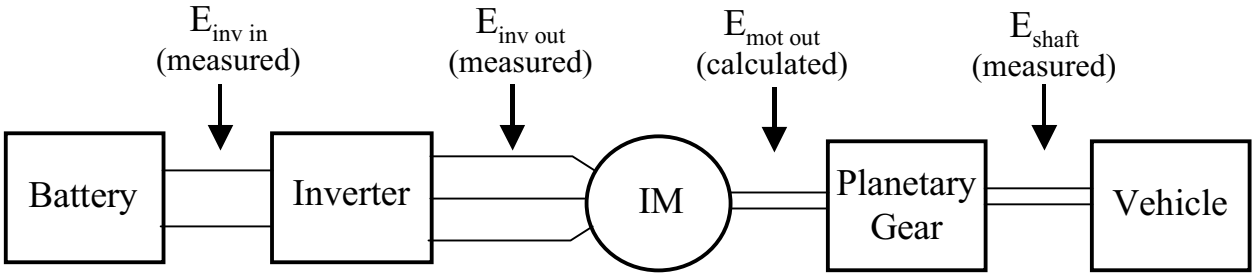


Fig. 5.10. Diagram showing what energies are used to compute the transient efficiencies.

The induction motor input energy is clearly the same as the inverter output energy measured, $E_{inv\ out}$; however, the induction motor output energy is not as easily found. Two ways can be used to determine the induction motor output energy in order to estimate the induction motor transient efficiency. First, the motor output energy can be considered to be the value that is determined by integrating the measured shaft torque times the measured shaft speed over a drive profile:

$$E_{shaft} = \int \tau_{shaft\ meas}(t) \cdot w_{shaft\ meas}(t) \cdot dt \tag{5.26}$$

The measurement errors in τ_{shaft} and w_{shaft} were found to be large, so a second way was created to estimate the induction motor output energy. The second way of finding the induction motor output energy uses the simulation diagram shown in Appendix B to determine the variable that is termed $E_{mot\ out}$, which is the induction motor output energy that results if there were not any gear losses. In determining $E_{mot\ out}$, it is assumed that the induction motor rotor torque and speed are maintained at the commanded values with very little deviation for the different modulation methods tested, which makes the test very repeatable.

Using the second method for determining the induction motor output energy, the induction motor transient efficiency has been defined as follows:

$$\eta_{tran}\,\text{motor} = \frac{E_{mot\ out}}{E_{inv\ out}} \times 100 \tag{5.27}$$

$E_{mot\ out}$ for the FUDS and HWFET profiles for a mid-size truck, as determined from the simulation diagram shown in Appendix B, are listed below.

$E_{mot\ out}$ for motor shaft on FUDS = 1.514 kW-hr
$E_{mot\ out}$ for motor shaft on HWFET = 1.508 kW-hr

Now that the techniques for measuring the transient efficiency have been explained, the results obtained will be shown next.

The FUDS and HWFET driving profiles were simulated on an electric motor dynamometer, as described in Chapter 3, to determine the energies required to compute the transient efficiencies over a given profile [23], [32]. Once the required energies were measured from a simulated drive profile on the dynamometer, the transient efficiency of the inverter and induction motor over the particular profile were computed by using Equations (5.25) and (5.27),

respectively. The averaged results obtained for the transient efficiencies when using optimal SVM control with a switching frequency of 10 kHz are summarized in Tables 5.6 and 5.7 for the FUDS and HWFET profiles, respectively.

Table 5.6 FUDS, SVM Control

kW-hr inverter in	1.882 kW-hr
kW-hr inverter out	1.701 kW-hr
η_{tran} inverter	90. 4%
η_{tran} motor	89.0%

Table 5.7 HWFET, SVM Control

kW-hr inverter in	1.898 kW-hr
kW-hr inverter out	1.782 kW-hr
η_{tran} inverter	93.9%
η_{tran} motor	84.6%

The average results for transient efficiency when using hysteresis control are shown in Tables 5.8 and 5.9 for the FUDS and HWFET profiles, respectively.

Table 5.8 FUDS, Hysteresis Control

kW-hr inverter in	2.562 kW-hr
kW-hr inverter out	2.175 kW-hr
η_{tran} inverter	84.9%
η_{tran} motor	69.6%

Table 5.9 HWFET, Hysteresis Control

kW-hr inverter in	2.070 kW-hr
kW-hr inverter out	1.898 kW-hr
η_{tran} inverter	91.7%
η_{tran} motor	79.5%

The transient efficiency results show that when SVM control is used, the required energy input of the inverter is reduced for the same given driving profile. Reduced energy input indicates increased driving range for the same available energy from the vehicle batteries. For example, for vehicle operation similar to the FUDS profile, a range increase of 26.5% is expected, as computed via Equation (5.28).

$$\%\text{range increase} = \frac{\text{kWhr in}_{\text{Hyst}} - \text{kWhr in}_{\text{SVM}}}{\text{kWhr in}_{\text{Hyst}}} \times 100 \tag{5.28}$$

Likewise, for vehicle operation similar to the HWFET profile, a range increase of 8.3% is expected. These results are summarized below:

Range Increase for the FUDS: **26.5%**
Range Increase for the HWFET: **8.3%**

The larger range increase for the FUDS profile as opposed to the HWFET profile is explained by the location of the largest densities of the torque versus speed operating points for the two different profiles. The operating point densities for the FUDS profile are shown in Fig. 5.11 to be concentrated in the areas of the greatest system efficiency improvements, as shown in Fig. 5.2(c). The largest operating point densities for the HWFET profile are concentrated in areas of less efficiency system efficiency gains, as shown in Fig. 5.12 and Fig. 5.2(c).

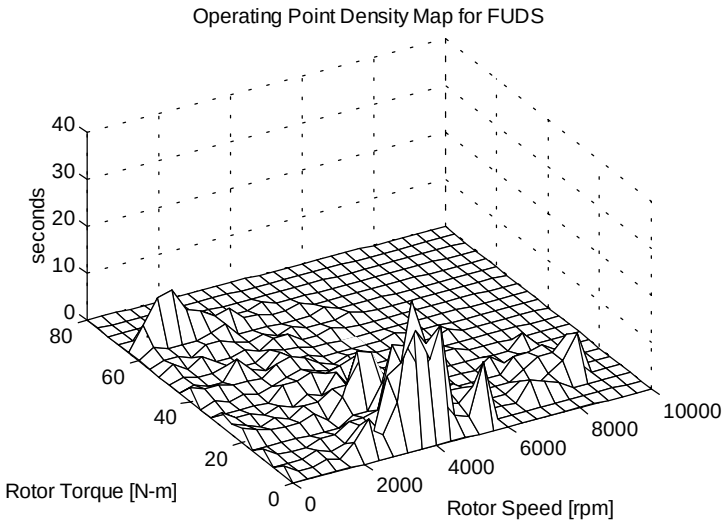


Fig. 5.11. Operating point density map for the FUDS profile for a mid-size EV truck.

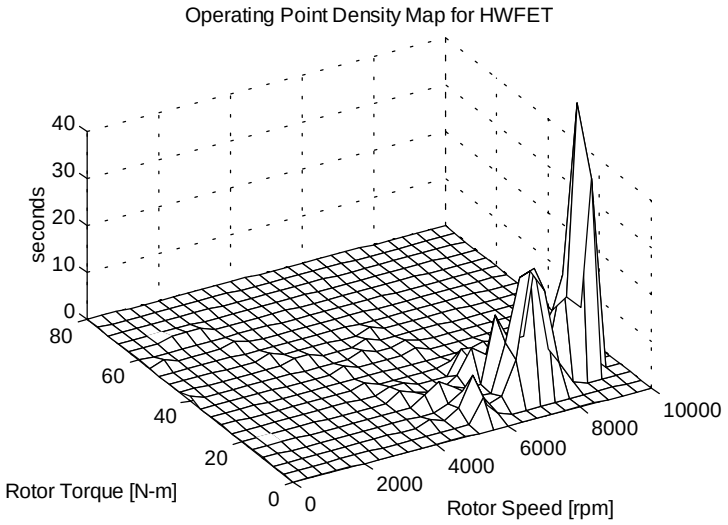


Fig. 5.12. Operating point density map for the HWFET profile for a mid-size EV truck.

5.4. Impact of Switching Frequency with Optimal SVM

Assuming that the reduced harmonic currents with SVM lowers the IM losses, which was demonstrated in sections 5.1 and 5.2, it is useful to analyze how much further the harmonics can be reduced by increasing the inverter switching frequency. This sections shows some initial results for varying the switching frequency with optimal zero vector SVM, where the simulation arrangement shown in Fig. 4.5 was used.

First, the phase currents of the IM are simulated at a particular operating point in the torque versus speed point. The rotor speed was chosen to be 2500 rpm, so w_r^* is set to be 262 rad/sec, as shown in Fig. 4.5. The IM shaft torque command, T_L^*, is set to be 30 N-m. Center based PWM was chosen as well as a 3 μs pulse deletion time. The results of this test are shown below:

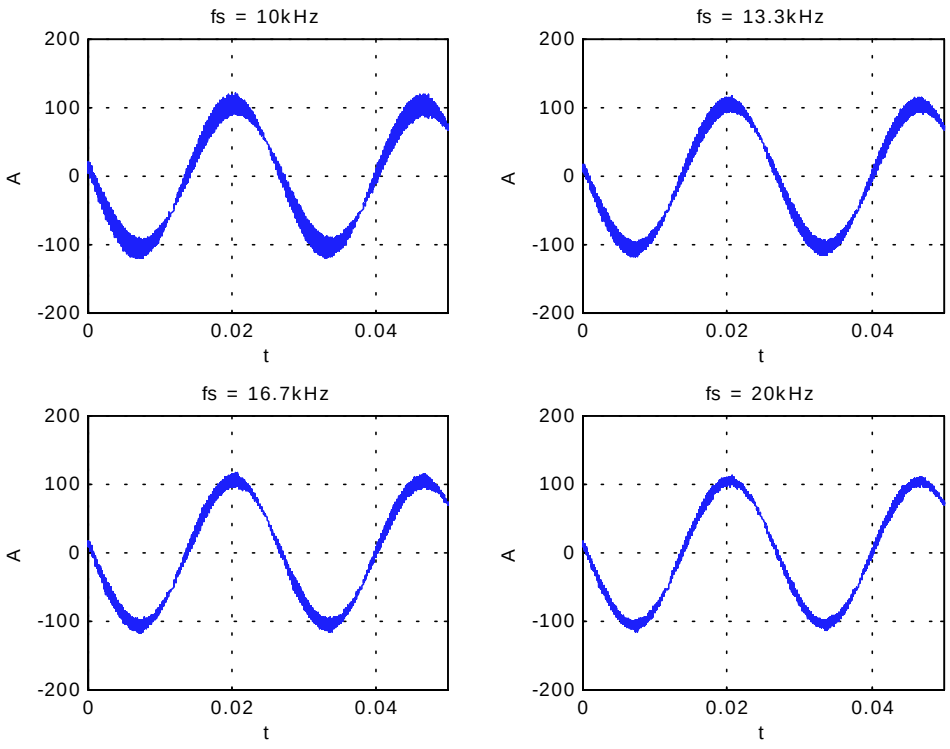


Fig. 5.13. Resultant stator phase currents for the 2500 rpm, 30 N-m test at various inverter switching frequencies.

As shown in Fig. 5.13, the harmonics in the simulated stator phase currents appear to be reduced as the switching frequency is increased. An estimation of the amount of reduction in the current harmonics in the waveforms can be found by computing the total harmonic distortion (THD) of the waveforms, as determined by Equation (5.13), where the current harmonics can be determined by the procedure outlined in section 5.2.2.1. The THD of the IM stator phase current versus frequency, at the operating point 2500 rpm and 30 N-m, is shown in Fig. 5.14.

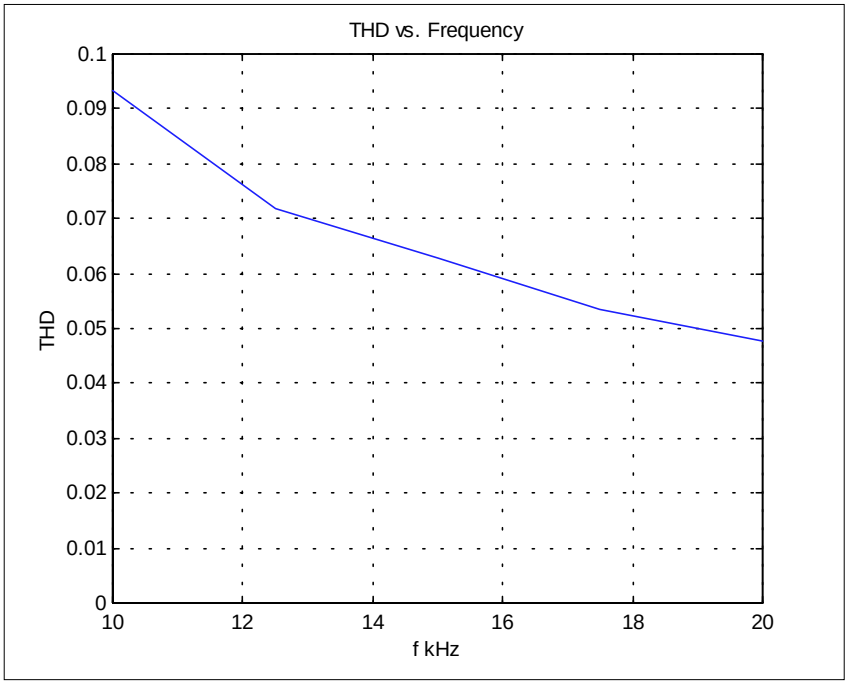


Fig. 5.14. Plot showing the reduction in THD versus switching frequency for the currents in Fig. 5.13.

Generally, additional simulations showed that the THD is reduced in roughly the same proportion shown in Fig. 5.14 across the entire torque vs speed plane.

5.5. Summary of System Efficiency Improvements with SVM

This chapter has shown that using optimal zero vector selection SVM control with a 10 kHz switching frequency as opposed to hysteresis control has increased the EV inverter and induction motor efficiency across the majority of the torque versus speed plane. The largest

efficiency increase was for the induction motor in the range 0 to 4000 rpm. Inverter efficiency improvement was justified by reduced switching and conduction losses with SVM. The motor efficiency was shown to increase due to a decrease in harmonic losses when SVM control was used, as found by using a semi-empirical motor loss model. SVM was shown to have less harmonic content of the stator phase currents leading to a reduction in harmonic losses. Large decreases in copper, iron, and mechanical losses were predicted. Simulation results have predicted an even further decrease in the harmonic content of the stator phase currents if the switching frequency is increased.

Once a valid torque/speed profile was determined for a given driving profile, the profile was simulated in real time on a dynamometer set, and driving profile transient efficiencies were determined. The largest transient efficiency increases in the inverter and induction motor were obtained on the FUDS profile, since its operating points are concentrated in the areas of the largest system efficiency improvements. The largest efficiency gain was for the induction motor on the FUDS profile. Increases in the inverter and induction motor efficiencies give an improved vehicle range of 26.5% for the FUDS profile and 8.3% for the HWFET profile.

The analysis procedures, including the simulations, used in comparing SVM control to hysteresis control for an EV are considered to be general and can be repeated in order to compare different modulation algorithms or changes in the power stage topology of the inverter.

This chapter has shown that space vector PWM control using an optimal zero vector selection has produced a very efficient driving system as compared to its predecessor. Additionally, it has been shown that further increasing the inverter switching frequency can further improve the quality of the induction motor phase currents. The next chapter discusses the addition of the shudder compensator to the control system, which will further enhance the drive performance.

Chapter 6. Development of a Drive Resonance Compensator

6.1 Description of Drive Resonance

This Chapter discusses improvement of the base controller, shown in Fig. 1.14, through the addition of a drive resonance compensator. The compensator developed, also termed the shudder compensator, can correct for rotor shaft oscillations at low speeds of the EV that are due to a resonance induced between the induction motor rotor and the vehicle mass caused by the spring like nature of the drive shaft. The shudder compensator added is shown schematically in Fig. 6.1.

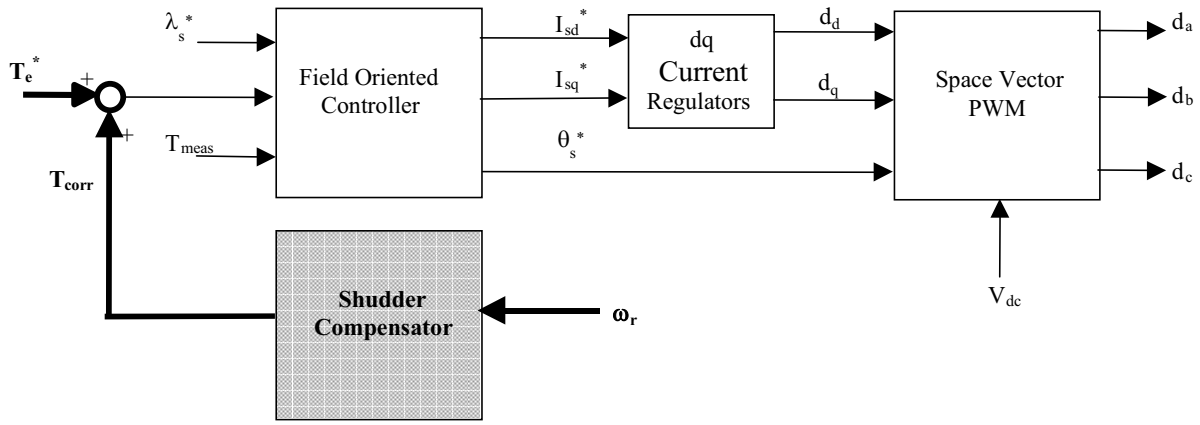


Fig. 6.1. Basic controller structure with added shudder compensator (highlighted).

Shudder was measured in the vehicle by logging the data samples of the rotor speed feedback while driving the test vehicle at low speeds. The measured oscillations are shown in Fig. 6.2.

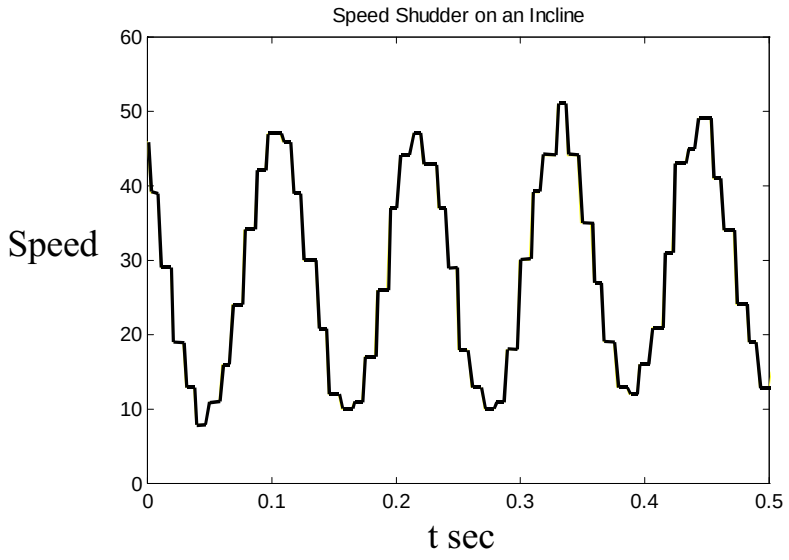


Fig. 6.2. Measured vehicle speed oscillations due to drive shaft resonance. T_{sample} *= 2ms*

To correct the objectionable oscillations that are shown in Fig. 6.2, a compensator was developed that corrects for speed shudder by feeding back measured rotor shaft speed, as shown in Fig. 6.1. The output of the shudder compensator is a correction torque that is added to the torque reference prior to being input into the field-oriented controller.

This chapter discusses the design and implementation of the shudder compensator. In section 6.2, a plant model is developed for simulating shudder compensation. Section 6.3 discusses the development of the type of shudder compensator used and the selection of the compensator gain. Section 6.3 also discusses the practical issues faced in the implementation of the shudder compensator in the DSP. Lastly, section 6.4 shows the simulation and experimental results, which indicate proper operation of the shudder compensator developed.

6.2 Modeling of EV Torque to Speed Transfer Function

In this section, the motor and drive system is modeled as a two mass system, and the shaft resonance (shudder) is simulated using the plant transfer function developed. The inertial

mass of the rotor is tied to the vehicle inertial mass through the motor gearing and drive shaft. The overall situation can be approximated as shown below in Fig. 6.3.

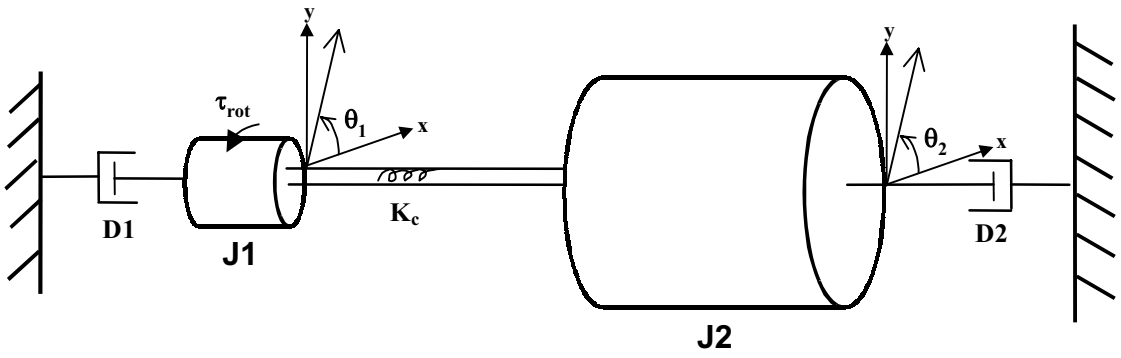


Fig. 6.3. Diagram of rotor mass connected to vehicle mass through the drive shaft.

The inertia of the induction motor rotor, shown as J1 in Fig. 6.3, can be determined from Equation (6.1).

$$J1 = \frac{m_{rot} k^2}{2} = 0.11\ N \bullet m \bullet s^2 \qquad (6.1)$$

Rotor mass = $m_{rot} = 18.1$ kg
Rotor radius of gyration = k = .11 m

The rotor damping ratio, shown as D1 in Fig. 6.3, is chosen to represent the amount of damping torque applied to the rotor from the friction and windage of the driveline. A value of 3 N·m is chosen to represent the damping torque at the maximum rotor rpm of 15000 rpm, which equals 1571 rad/sec. Hence, D1 can be determined by Equation (6.2).

$$D1 = 3/1571 = .002\ N \cdot m \cdot s \qquad (6.2)$$

The spring constant of the vehicle drive shaft, shown as K_c in Fig. 6.3, is going to be determined by experimental measurements that will be discussed later in this section.

Lastly, the reflected inertia of the vehicle, J2, and the vehicle damping ratio, D2, are now determined. The applied torque on the rotor due to the vehicle dynamics is given by (6.3), which has been described in detail in Chapter 3.

$$\tau^*_{rotor} = \frac{r}{GR} \cdot [m\frac{dv^*}{dt} + mg(C_{R0} + \frac{C_{R1}}{v_{ref}} v^* + \frac{C_A}{v_{ref}^2} v^{*2})] \tag{6.3}$$

The last terms in the equation, within the parenthesis, represent the vehicle rolling resistance and aerodynamic drag forces. Since the shudder compensation is meant for slower speeds, these terms are ignored because the aerodynamic drag is small at slow speed and the rolling resistance is usually small. Therefore, the damping ratio D2 that is shown is Fig. 6.3 is set to zero:

$$\therefore D_2 \cong 0$$

The first term in Equation (6.3) represents the torque required to accelerate the mass of the vehicle. For an acceleration of $1 m/s^2$, the rotor torque required is given by Equation (6.4):

$$\tau_{rot} = \frac{r \times m \times 1}{GR} = \frac{.32 \times 1600}{12} = 42.67 \text{ N} \bullet \text{m} \tag{6.4}$$

As shown in (6.5), a 1 m/s^2 acceleration of the vehicle can be translated to a 35 rad/s^2 acceleration of the rotor shaft.

$$\frac{rad}{s^2} = \frac{m}{s^2} \times \frac{GR}{r} = \frac{m}{s^2} \times 35 \tag{6.5}$$

Therefore, the term J2 can be determined as shown (6.6).

$$\begin{gathered} J2 \times 35 = 42.67 \text{ N} \cdot \text{m} \\ \therefore J2 \cong 1.33 \text{ N} \cdot \text{m} \cdot \text{s}^2 \end{gathered} \tag{6.6}$$

The equations of motion in the complex domain of the two mass system of Fig. 6.3 can now be written as shown in (6.7).

$$\begin{bmatrix} J_1 s^2 + D_1 s + K_c & K_c \\ -K_c & J_2 s^2 + K_c \end{bmatrix} \begin{bmatrix} \theta_1(s) \\ \theta_2(s) \end{bmatrix} = \begin{bmatrix} \tau_{rot}(s) \\ 0 \end{bmatrix} \tag{6.7}$$

Solving for the $\theta_1(s)/\tau_{rot}(s)$ transfer function leads to Equation (6.8):

$$\frac{\theta_1(s)}{\tau_{rot}(s)} = \frac{J_2 s^2 + K_c}{s(J_1 J_2 s^3 + J_2 D_1 s^2 + (J_2 + J_1)K_c s + D_1 K_c)} = \frac{1.33s^2 + K_c}{s(0.146s^3 + 0.00266s^2 + 1.44K_c s + 0.002K_c)} \tag{6.8}$$

To complete the plant model given by (6.8), the frequency of the shaft oscillations were measured in the vehicle to determine the shaft spring constant, K_c, which is discussed next.

The spectrum of the speed shudder measured in Fig. 6.2 is shown in Fig. 6.4. This spectrum plot is the sample values of the shudder waveform Discrete Time Fourier Transform (DTFT), where a rectangular window was used on the sampled data points. The sample spacing of the DTFT is every 2Hz, and the samples were found by using the FFT. From Fig. 6.4, it can be seen that the main shudder components are found at 8 and 10 Hz.

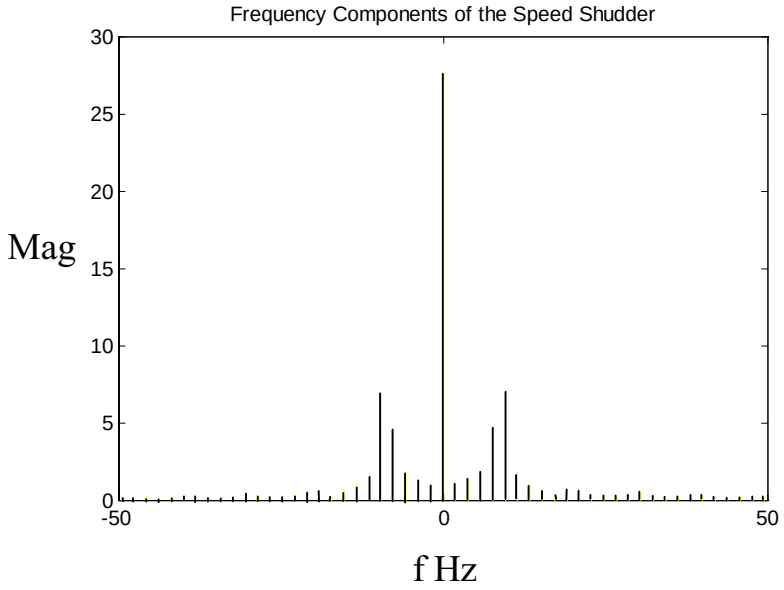


Fig. 6.4. Spectral samples of the speed shudder.

The roots of the characteristic equation of the $\theta_1(s)/T(s)$ transfer function of Equation (6.8) are given as Equations (6.9).

$$P_1 = 0$$
$$P_{2,3,4} = \text{roots}(0.146s^3 + 0.00266s^2 + 1.44K_cs + 0.002K_c) \quad (6.9)$$

The imaginary roots are nearly undamped so the frequency of oscillation should be close to the natural frequency that was measured in the vehicle:

$$w_n = 2\pi 10 = 62.8 \text{ rad/sec}$$

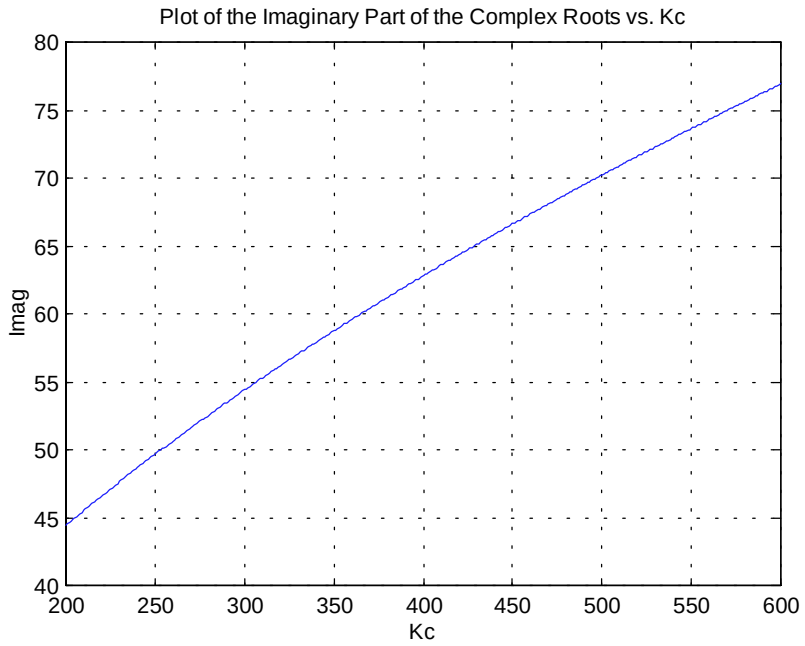


Fig. 6.5. Plot of the Imaginary part of the complex roots vs. spring constant.

From the plot of the imaginary part of the complex plant poles versus the spring constant, shown in Fig. 6.5, for K_c about 400, the required resonant frequency is correct. The resulting transfer function is given in (6.10).

$$K_c \cong 400$$

$$\frac{\theta_1(s)}{\tau_{rot}(s)} = \frac{J_2 s^2 + K_c}{s(J_1 J_2 s^3 + J_2 D_1 s^2 + (J_2 + J_1)K_c s + D_1 K_c)} = \frac{1.33s^2 + 400}{s(0.146s^3 + 0.00266s^2 + 576s + 0.8)} \tag{6.10}$$

To obtain the desired transfer function, the above transfer function is multiplied by s to obtain the $w_1(s)/\tau_{rot}(s)$. The resulting plant model is given in (6.11):

$$\frac{w_1(s)}{\tau_{rot}(s)} = \frac{1.33s^2 + 400}{(0.146s^3 + 0.00266s^2 + 576s + 0.8)} \tag{6.11}$$

The uncompensated system plant model, given by Equation (6.11), is tested by applying a torque pulse of 25 N·m at 0 seconds and removing it at 0.25 seconds. The system should show slightly damped oscillations at a frequency of about 10 Hz. This is shown to be the case in Fig. 6.6:

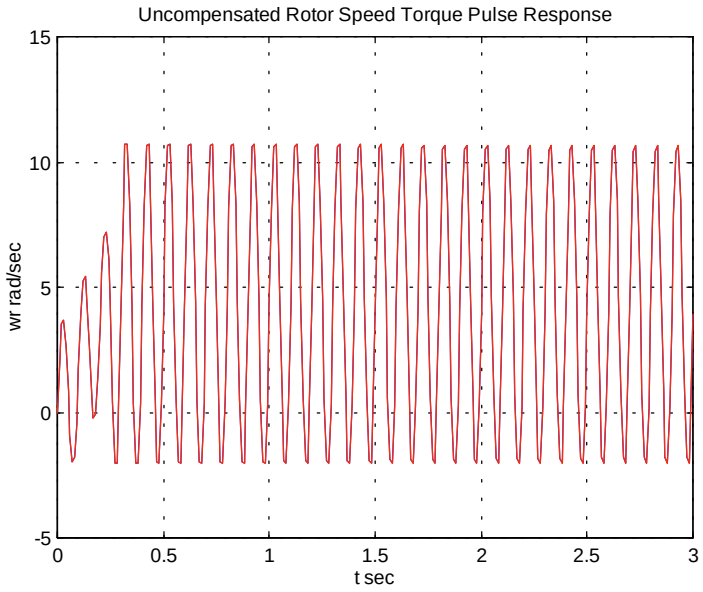


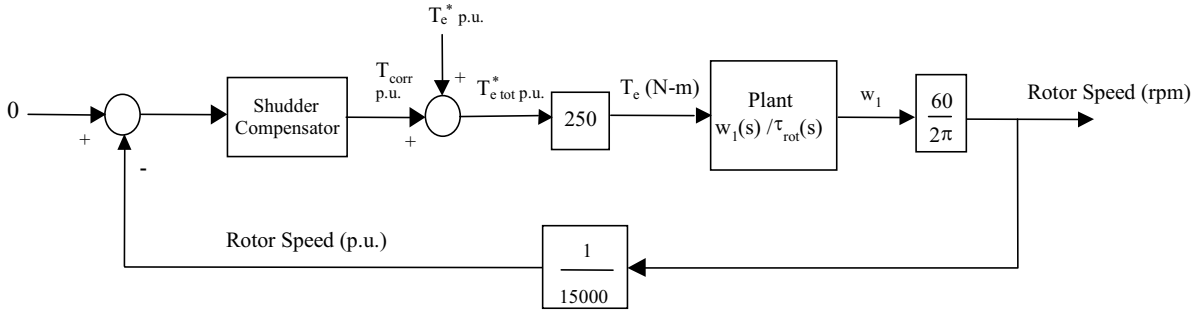
Fig. 6.6. Test of the plant model for a pulse torque input.

6.3 Compensator Development

In this section, a lead type compensator will now be shown to adequately compensate the rotor shaft oscillations shown in Figures 6.2 and 6.6. The proper compensator gain will be found by using a root locus and also by using frequency response techniques. The compensated system will show a reduced resonant peak and improved gain and phase margins. The actual shudder compensator implementation is discussed, as well as the relevant compensator implementation issues, such as the manner in which the rotor speed is computed in the DSP. The clamping of the shudder compensator output is explained, and the way that the shudder compensator output is rolled of with increasing speed is also discussed.

6.3.1 Type of Compensator and Gain Setting

The previous section developed the plant model transfer function between the rotor torque and rotor speed in radians per second. The various components of the compensated system are shown below in Fig. 6.7.



Note: The 250 gain block translates the torque command to actual torque in N-m

Fig. 6.7. Compensated system control diagram.

Figure 6.7 shows that the speed shudder can be compensated by feeding back the rotor speed through a compensator and then adding the compensator output, T_{corr}, to the torque command, T_e^*. In Fig. 6.7, the torque regulator, which executes at 500 μs, is assumed to have a constant gain transfer function and is modeled as a change from p.u. torque to actual torque, which is a gain of 250 since the base torque is 250 N·m. T_e^* and T_{corr} are both in p.u. on a 250 N·m base in Fig. 6.7, because this is how they are actually represented in the DSP. To help show that it is a valid assumption to model the change from commanded torque to actual as a constant gain in Fig. 6.7, the step response of the torque is simulated using the simulation models developed in Chapter 4. Fig. 6.8 shows the step response of the q-current and q-current command in the synchronous reference frame. The rotor speed command was 50 rad/sec, which translates to the same vehicle speed range where shudder is a problem. After an initial startup, the q-current command was stepped by 100 Amps at t = 0.125 seconds.

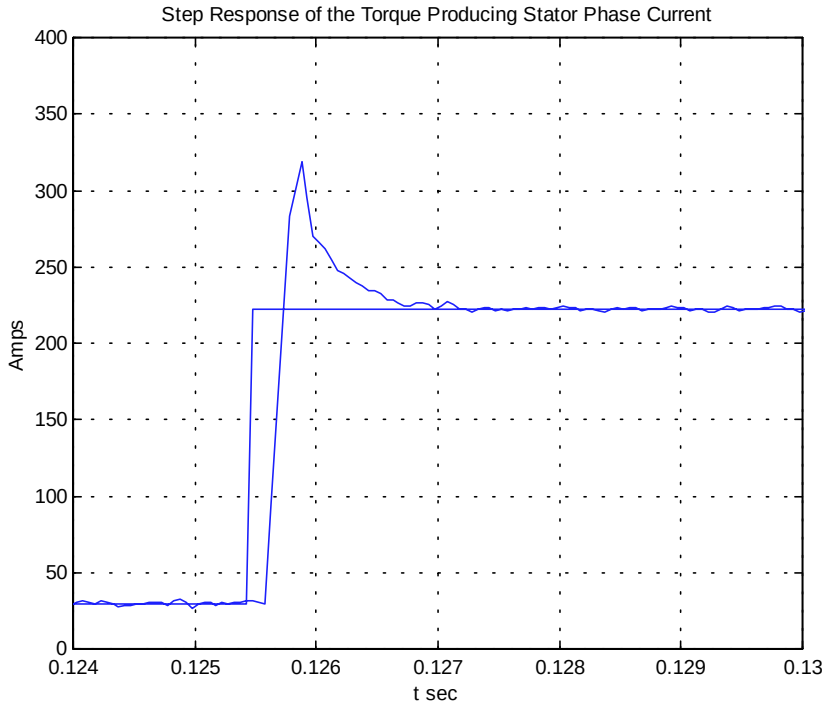


Fig. 6.8. Simulated torque command step response.

Figure 6.8 shows that the torque current rises and settles in about 1 ms, which indicates a much faster dynamic response than the plant, whose frequency response is shown below in Fig. 6.9 as determined from Equation (6.11) and the scaling constants shown in the block diagram in Fig. 6.7.

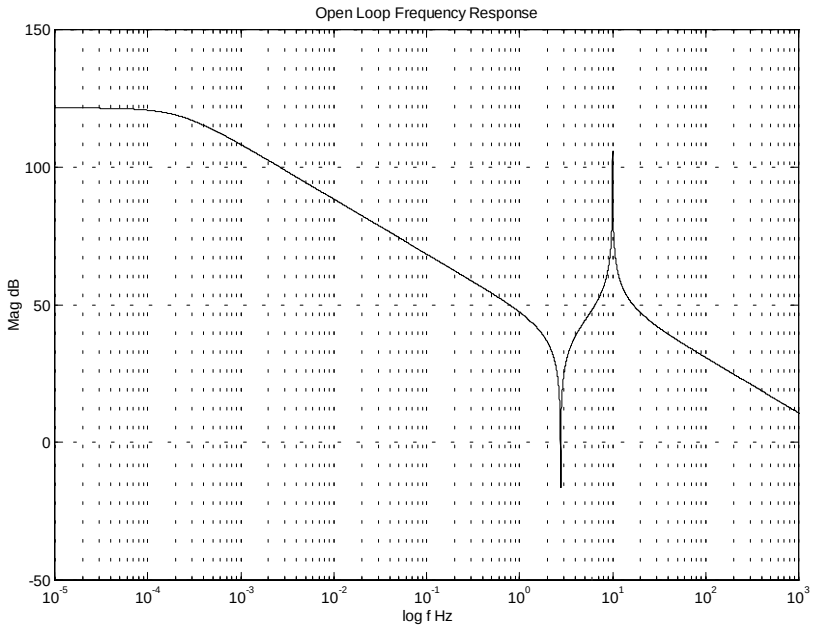


Fig. 6.9. Open loop frequency response of the plant model, where the 250 N-m torque command to torque gain and scaling constant to convert to rpm have been included

According to Fig. 6.9, the plant dynamics are much less than 1 Hz except for the resonance at 10 Hz. Therefore, the torque command to torque can be modeled as the constant gain shown in Fig. 6.7 without a large expected modeling error, because the step response of the torque in Fig. 6.8 clearly showed that torque command to torque response has a much faster response than the plant.

The next step is to determine the form of the shudder compensator in Fig. 6.7. A good compensator has been determined as (6.12):

$$\frac{K \bullet s}{(s + p_c)} \tag{6.12}$$

The two parameters to be determined are p_c and K. The value p_c is chosen as 4 because a good closed loop response was obtained with this value. This leaves the shudder compensator gain, K, which can be determined via root locus techniques. It will be important that the closed loop frequency response has good damping of the resonant peak. However, the closed loop system bandwidth should not extend far beyond the plant bandwidth or else large control signals could result, limited to the clamp value that will be placed after the compensator. Fig. 6.10 shows the root locus plot of the system. The large arc is the eventual location of the closed loop poles. To provide good damping, the closed loop poles should be as far into the left half plane as is feasible.

Figure 6.11 shows the damping verses gain for the complex pole pair, where the damping has been determined by taking the cosine of the angle between the negative Re axis and the complex pole location. As shown in Fig. 6.11, a large gain is required for good damping. However, a practical upper limit on the gain is determined by numerical issues in the DSP and unmodeled plant dynamics. If possible, saturation and limiting should be avoided in the range of operation of the shudder compensator. As will be shown in section 6.3.2.2, a ±15 N·m clamp, which equals ±15/250 p.u., is included after the shudder compensator in the actual implementation to avoid the application of overly large correction torques. Therefore, the practical upper limit on the shudder compensator gain, K, is chosen to be 10, which will be shown to be acceptable in the experimental results.

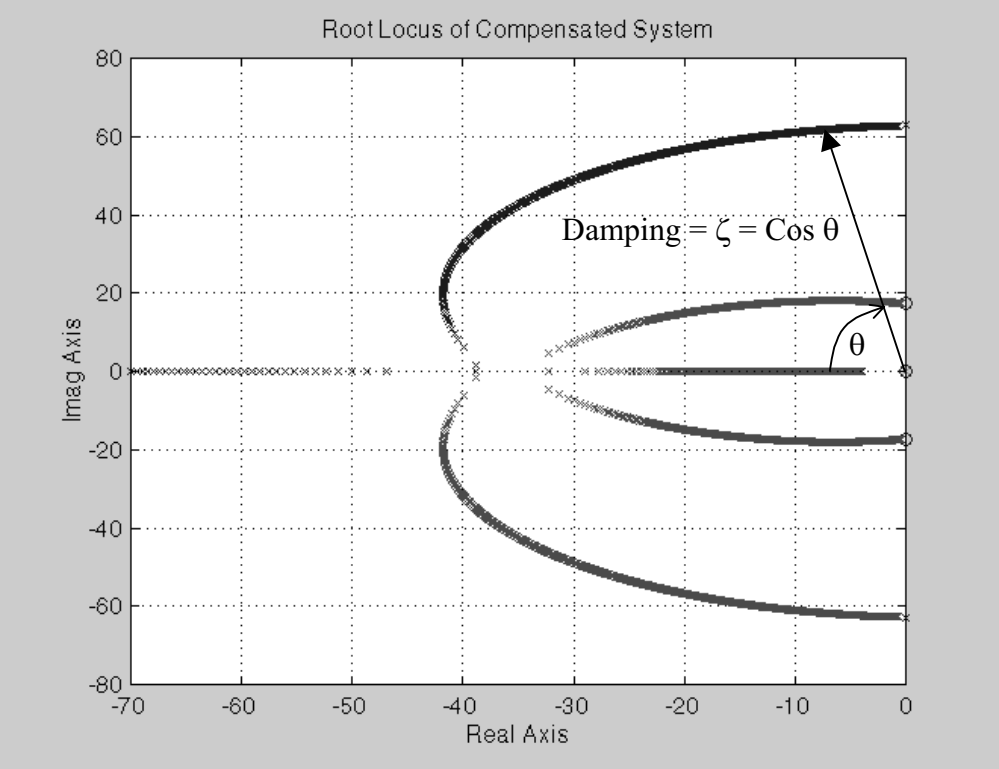


Fig. 6.10. Root Locus plot for the system shown in Fig. 6.7 and the compensator given by (6.12).

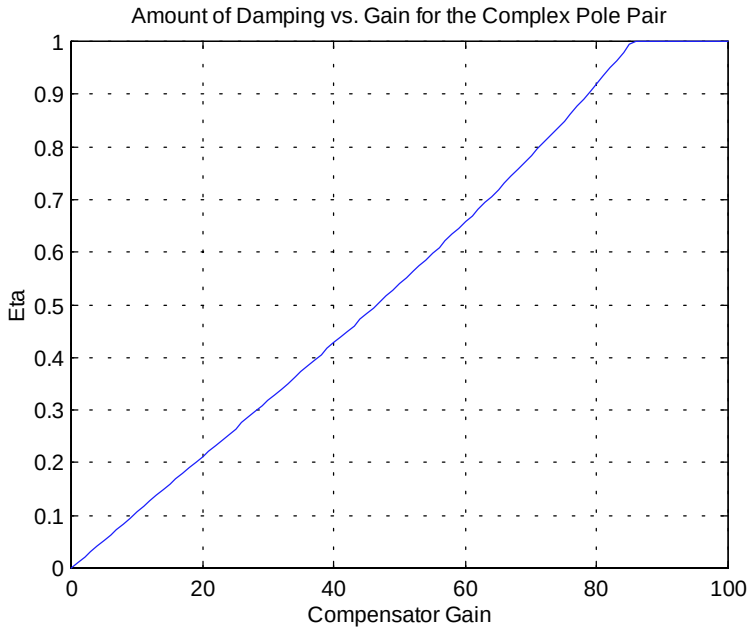


Fig. 6.11. Amount of damping of the system complex pole pair vs. compensator gain K.

Compensator gains of K = 10 and K = 1 are used in determining frequency response of the closed loop system transfer function, which is Rotor Speed(s)/T_e^*(s) in Fig. 6.7. This closed loop transfer function is shown in Fig. 6.12 and Fig. 6.13, respectively, for the two different compensator gains tested.

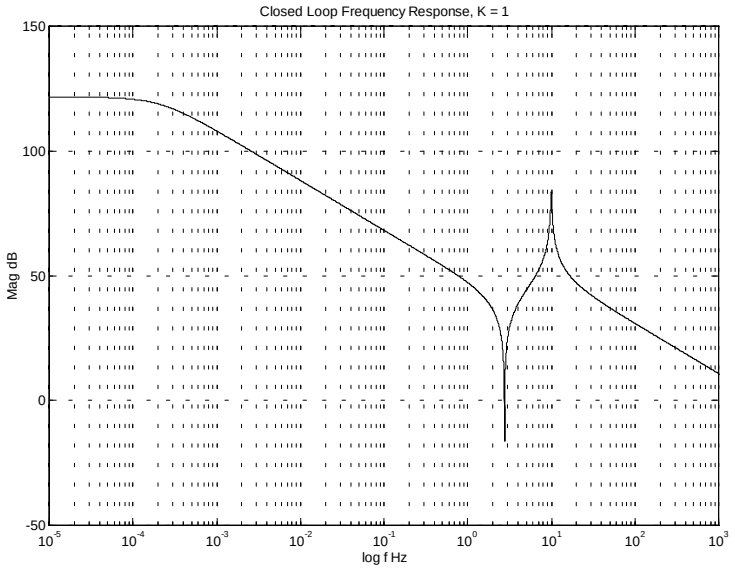


Fig. 6.12. Rotor Speed(s)/T_e^* closed loop for K = 1.

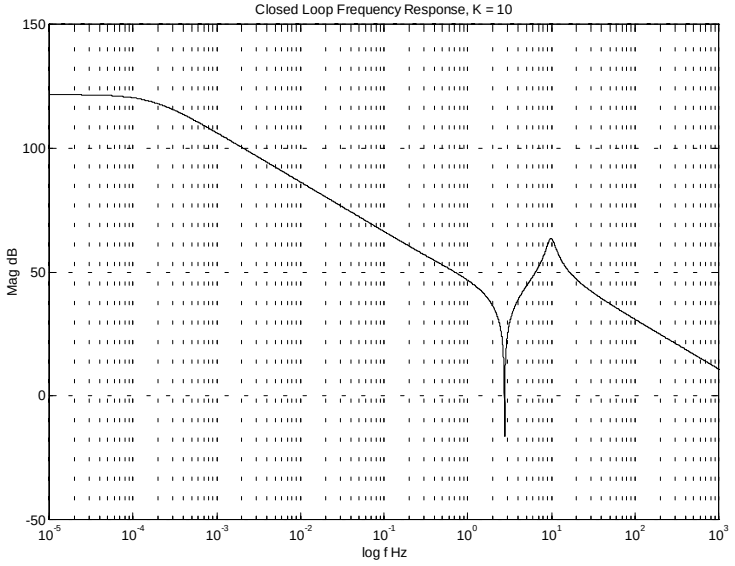


Fig. 6.13.Rotor Speed(s)/T_e^* closed loop for K = 10.

The larger compensator gain flattens out the resonant peak more, but also will require a larger control output signal.

To determine the limits of the control output signal of the shudder compensator and hence the maximum compensator gain, the clamping and saturation in the control loop are considered next.

6.3.2 Compensator Implementation Issues

To access the issues of clamping and saturation in the shudder compensator, and hence the maximum compensator gain, the manner in which the speed feedback is computed in the DSP and the implementation of the compensator must be considered. First, the speed feedback is analyzed.

6.3.2.1 Speed Feedback Computations

In this section, it is determined whether quantization of the measured speed feedback signal, which is shown in Fig. 6.7, can affect the shudder compensator. Quantization of the speed feedback signal can be a problem at very slow speeds of the EV, caused by transport lag, as well as very high speeds, due to the limited amount of teeth that are on the tachometer wheel that is mounted on the induction motor rotor. For the shudder compensator, the low speed performance is the most critical because the shudder is primarily a low speed problem, especially when the vehicle is just starting to accelerate from standstill.

The speed is represented in the DSP using a 15000 rpm rotor base speed, where 15000 rpm would correspond to 1 p.u. or hexadecimal $7fff. Therefore, 15000 rpm translates to about 1.0 decimal when using a 1.15 fractional binary format [35]. The hardware used for the rotor speed feedback includes the incremental encoder attached to the induction motor that sends quadrature pulses to the encoder interface port of the control board. The DSP receives the quadrature pulses and processes them in the encoder interface unit [34]. For the tachometer used, there are 64 slotted teeth on the circular wheel that is attached to the rotor; see Fig. 6.14. This slotted wheel revolves through an optical sensor to produce the quadrature pulses sent to the EV control board, as shown in Fig. 6.14.

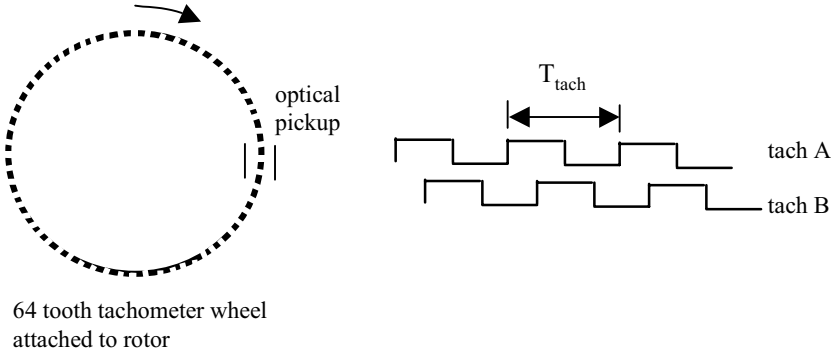


Fig. 6.14. Illustration of the generation of the tachometer quadrature pulses.

The frequency of the tach A and tach B square waves shown in Fig. 6.14 are related to the rotor speed in rpm as shown in (6.13).

$$f_{tach} = \frac{\text{Rotor Speed in rpm}}{60} \times 64 \tag{6.13}$$

The tachometer signals produced by the encoder mounted on the induction motor are processed through a controller interface circuit (comparator, buffer, an opto-coupler) and are then sent to the DSP—refer to Fig. 1.6. The built in DSP encoder interface hardware, which receives the final tach signals, is shown in Fig 6.15 [34].

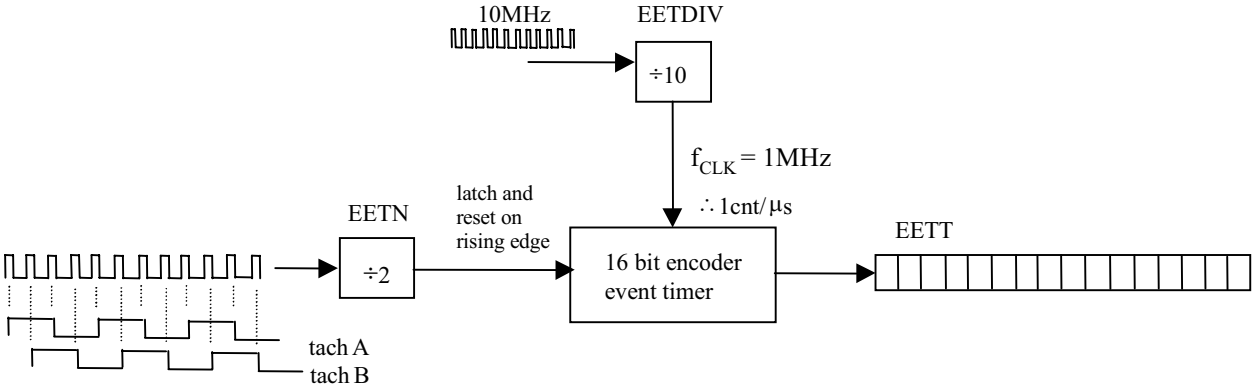


Fig. 6.15. DSP encoder interface hardware for the ADMC300 Processor.

The value $f_{CLK,}$ shown in Fig. 6.15 and computed as shown in (6.14), determines how fast the count value in the 16-bit encoder event timer is incremented or decremented.

$$f_{CLK} = \frac{10\text{MHz}}{\text{EETDIV}} \tag{6.14}$$

The minimum half period of one of the tach A or tach B signals needs to be derived to find an equation for the p.u. Rotor Speed shown in Fig. 6.7. As shown in Fig. 6.15, the EETT register contains the 16 bit up down counter result that is latched in from the event timer every time the event timer receives a rising edge from the output of the EETN decimator. The half period of the tach A or tach B signals is the amount of time that passes before the binary number in EETT is incremented or decremented. The EETN decimation value is set to 2 in the present application, which means that the latch and reset rate is $T_{tach}/2$. From the above equation for f_{tach}, Equation (6.13), the maximum tach signal frequency is for the rotor speed in rpm = 15000 rpm, which is about 90 mph in a standard electric vehicle:

$$\begin{aligned} &f_{tach.max} = 15000/60 \times 64 = 16000 \text{ Hz} \\ &\therefore T_{tach.min} = 62.5\mu\text{s} \end{aligned} \tag{6.15}$$

The minimum count value for register EETT for a rotor shaft speed of 15000 rpm is given by (6.16).

$$\text{EETT}_{\text{min.cnts}} = \frac{T_{\text{tach. min}}}{2} f_{CLK} = 31.25 \text{ cnts} \tag{6.16}$$

The DSP speed computation algorithm uses the value given in (6.16) to compute the rotor rpm at all speeds by using the equations given in (6.17) and (6.18). Equation (6.17) is the actual rotor speed in rpm, and Equation (6.18) is the per unit rotor speed in rpm on a 15000 rpm base.

$$\text{Rotor Speed in rpm} = \frac{\text{EETT}_{\text{min.cnts}} \times 15000}{\text{EETT}} = \frac{31.25 \times 15000}{\text{EETT}} \tag{6.17}$$

$$\text{Rotor Speed in p.u.} = \frac{\text{EETT}_{\text{min.cnts}}}{\text{EETT}} = \frac{31.25}{\text{EETT}} \tag{6.18}$$

The smallest possible rotor rpm that can be represented is when the EETT register value is equal to 32768_{10}, which is where the rotor speed equals 14.19 rpm as determined from (6.17). Given the rotor rpm as determined from (6.17), the speed of the vehicle in miles per hour can be determined by the approximation given in (6.19).

$$mph_{vehicle} \cong Rotor\ rpm * (\,^{1}/_{12}) * (1ft*2*pi) * (\,^{1}/_{60}) * (\,^{60mph}/_{88\ ft/sec}) \tag{6.19}$$

Total Gearing Between the Rotor and the Wheel

Translate Rotations into Distance Traveled

Convert from ft/min to ft/sec

Convert to mph

In (6.19), the term 1/12 represents the gearing ratio between the rotor shaft and the vehicle wheel due to the 4.29:1 planetary gear ratio of the induction motor output and the additional gearing of the vehicle differential, which is about 3:1. As determined from (6.19), the smallest mph that can be represented in the DSP is about 0.08 mph, where the rotor speed equals 14.19 rpm as found previously.

It is important to know the quantization steps in the rotor speed representation when the rotor speed is at its slowest to see the effect on the shudder compensator and whether or not the minimum measurable speed of 0.08 mph will be acceptable. The smallest step size possible in the 1.15 fractional format is given as 1/32768 = 30.5 μunits, corresponding to the equivalent decimal amount of the least significant bit (LSB) which is 2^{-15}. Hence, EETT must change from 32768 to 31742 to get the next change in the speed representation as shown in (6.20).

$$\frac{\text{EETT}_{\text{min .cnts}}}{31742} - \frac{\text{EETT}_{\text{min.cnts}}}{32768} = \frac{31}{31742} - \frac{31}{32768} = 30.5\mu \tag{6.20}$$

When EETT is equal to 31742 counts, the rotor speed is ≅ 14.65 rpm or about 1.53 rad/sec. This is a difference of 0.04 rad/sec from when EETT equaled 32768 counts and represents the

quantization step in the rotor angular speed. As shown in Fig. 6.16, the quantization intervals in the EETT count values become smaller as the rotor speed is increased.

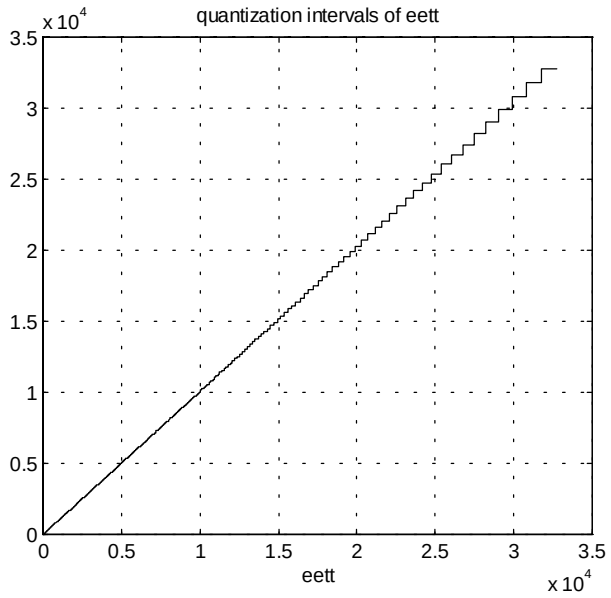


Fig. 6.16. Quantization intervals of EETT.

The quantization step in rpm_{rotor}, determined to be 0.04 rad/sec for the first quantization step, is independent of the EETT count value; the quantization step is always 0.04 rad/sec regardless of the rotor speed.

For the shudder compensator to work properly, the quantization steps in the rotor speed feedback signal can not be too large at low speeds. From Fig. 6.16, the quantization interval gets very small for EETT equal to 15000 cts where:

Rotor Speed in rpm = 31 rpm, 3.25 rad/sec
Vehicle Speed in mph ≅ 0.18 mph ≅ 3 in/sec

Therefore, for a vehicle speed of 3 in/sec, the resolution of the speed feedback is adequate. This amount of resolution is shown in the experimental results to not severely limit the shudder compensator at the slowest vehicle speeds.

In this section, the way that rotor speed is computed in the DSP was described, and the quantization step in the rotor speed feedback was determined to be about 0.04 rad/sec. This

amount of quantization is not expected to limit the operation of the shudder compensator, but the amount of quantization needed to be derived to rule out all of the possiblilities.

6.3.2.2 Controller Structure, Magnitude of Feedback Gain, and Clamping

The previous section has shown that the speed feedback resolution is adequate for use by the shudder compensator. In this section, the shudder compensator, determined previously to be in the form given by (6.21), is designed.

$$\frac{Ks}{(s+4)} \tag{6.21}$$

The transfer function shown in (6.21) is rewritten in (6.22), where the filter time constant ,T, is one quarter of a second.

$$\frac{K \bullet s}{s+\frac{1}{T}} = K\left(1 - \frac{1/T}{1/T + s}\right) = K\left(1 - \frac{1}{1+Ts}\right) \tag{6.22}$$

The shudder compensator is implemented from Equation (6.22) using the structure shown in Fig. 6.17.

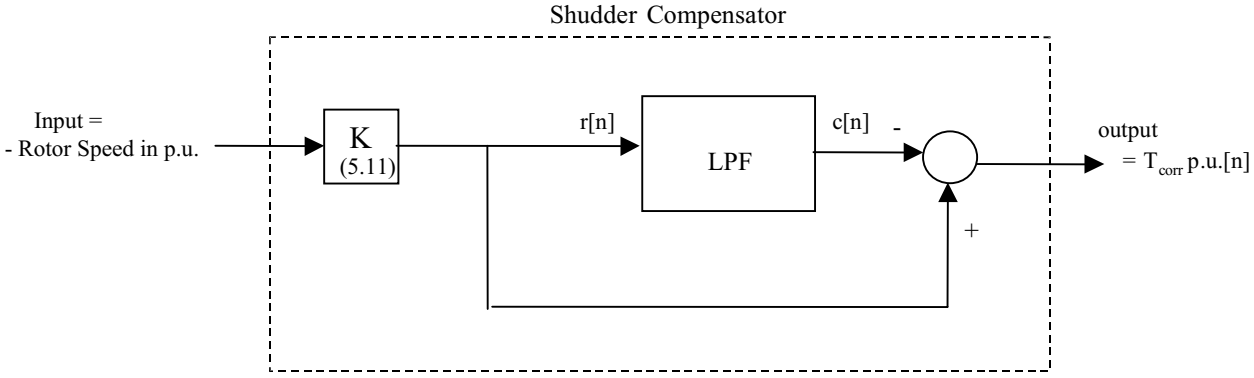


Fig. 6.17. Form of implementation of the shudder compensator.

The compensator shown in Fig. 6.17 implements Equation (6.21) by subtracting out the filtered average value of the rotor speed feedback signal from the rotor speed feedback signal. Once the average value is removed, the resulting negated signal, T_{corr} p.u., is used to correct the rotor oscillations by adding T_{corr} p.u. to the p.u. torque reference, as shown in Fig. 6.7 (Note: T_{corr} is limited by a clamp and speed rolloff which are discussed shortly). The lowpass filter in Fig. 6.17, $^{1}/_{(1+Ts)}$, is implemented digitally using the structure shown in Fig. 6.18, where the execution period is Δt = 2ms, T is 0.25sec as previously indicated, and z^{-1} is $e^{-js\Delta t}$.

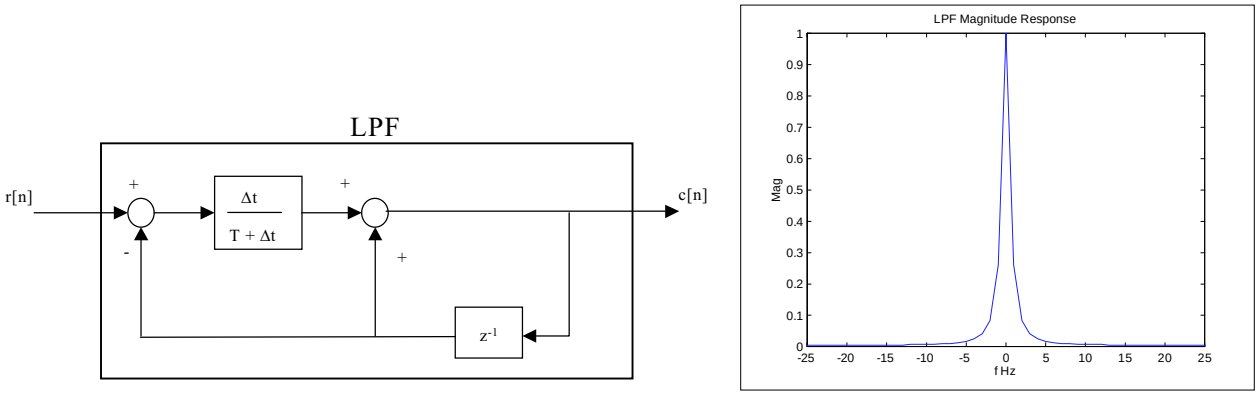


Fig. 6.18. Structure used and frequency response for the digital implementation of the lowpass filter.

From Fig. 6.18, the difference equation that represents the input to output transfer function of the low pass filter structure is given by (6.23).

$$c[n] = \frac{\Delta t}{T + \Delta t}(r[n] - c[n-1]) + c[n-1] \tag{6.23}$$

The frequency response of the LPF that is shown in Fig. 6.18 indicates that there is a large attenuation of the high frequency rotor speed input components, which is what is desired for obtaining the average rotor speed.

The numeric representation in the DSP for the feedback gain K, located as shown in Fig. 6.17, is now determined. For a feedback gain K less than 16, the numeric representation in the DSP for K can be in 5.11 binary fractional format [35]. The equation for determining this number in hexadecimal is (6.24).

$$K_{(5.11)} = \text{round}\left(\frac{K}{16} \times 32768\right) \Longrightarrow \text{Hexadecimal} \tag{6.24}$$

Once the rotor speed in p.u. is multiplied by $K_{(5.11)}$, all of the signals in the low pass filter structure are in 5.11 format. For a 5.11 binary fractional number, the LSB represents 2^{-11}, which equals 488μ. This magnitude of the LSB is 0.12 N·m on a 250 N·m base. A resolution of 0.12 N·m is shown in the experimental results to be enough resolution for this application.

In the DSP implementation, the shudder compensator output, T_{corr} p.u., is not allowed to increase without bound, but is clamped. On a base of 250 N·m for the torque and a clamp value of ±15 N·m, the clamp is ±0.06 p.u. Clamping of the shudder compensator output is depicted in Fig. 6.19.

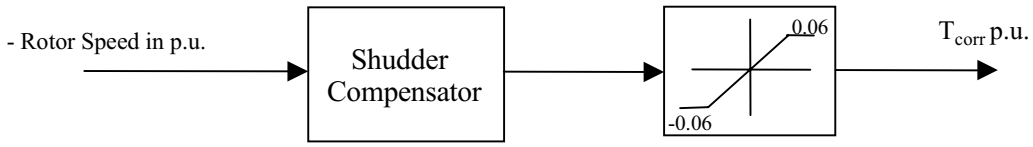


Fig. 6.19. Insertion of a clamp following the shudder compensator output.

If the output of the compensator does become clamped, a clamp reset mechanism has been employed that keeps the LPF output set at the correct value. Figure 6.20 shows the arrangement and definition of variables used for employing the clamp reset mechanism.

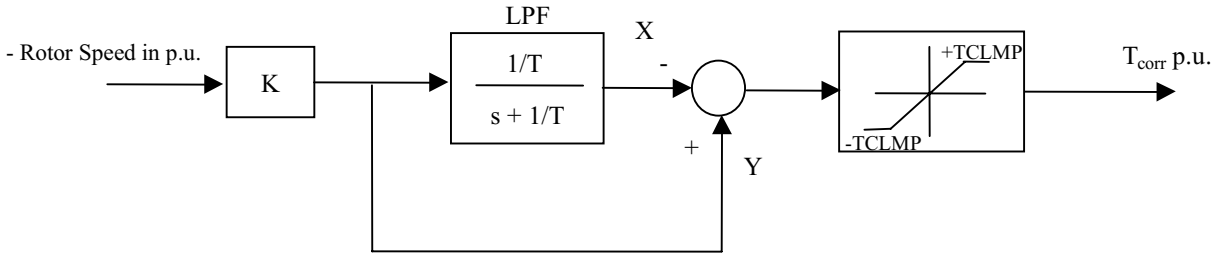


Fig. 6.20. Illustration of the variables used in the reseting of the shudder compensator output when the system becomes clamped.

When the output is not clamped, the correction torque is Y minus X:

$$T_{corr}\ \text{p.u.} = Y - X \tag{6.25}$$

If the system enters clamp, X is set to $Y - T_{corr}$ p.u. For positive clamp, this translates to X = Y – TCLMP, and for negative clamp, this translates to X = Y + TCLMP. In this way, the compensator does not have to reset by integrating out the LPF output error when it comes out of clamp.

6.3.2.3 Compensation Rolloff with Increasing Vehicle Speed

The speed shudder is mostly a low speed problem, which is when the speed of the vehicle is less than 10 mph. Therefore, the output of the shudder compensator can be rolled off at the higher speeds of the rotor to avoid unnecessary correction torques. Additionally, the shudder compensator output is rolled off so that above base speed, the correction torque is reduced inversely with speed because a large torque is more significant at a higher speed. The speed that the shudder compensation should start to rolloff is about 1200 rpm, or 7.14 mph, as determined from Equation (6.19). A speed inverse rolloff is employed, but a faster rolloff can be used if desired. The equation for a speed inverse rolloff of T_{corr} is given by (6.26).

$$T_{corr}\ \text{p.u.} = 5 \times \text{rpm}_{\text{rolloff}}\ \text{p.u.} \times \frac{\widetilde{T}_{corr}\ \text{p.u.} \times 0.2}{\text{rpm}_{\text{rot}}\ \text{p.u.}} \tag{6.26}$$

The scaling employed in Equation (6.26) ensures that the numerator is less than the denominator, which is required for the result of the divide to not overflow. If regular inverse speed rolloff is not fast enough, then the speed squared rolloff shown in Equation (6.27) can be used.

(6.27)

In both (5.26) and (5.27), $rpm_{rolloff}$ is the speed where the rolloff begins and is set to 1200 rpm, or 0.08 p.u. on a 15000 rpm base. Figure 6.21 is a graphical comparison of the inverse speed rolloff, given by (6.26), and the inverse speed squared rolloff, given by (6.27).

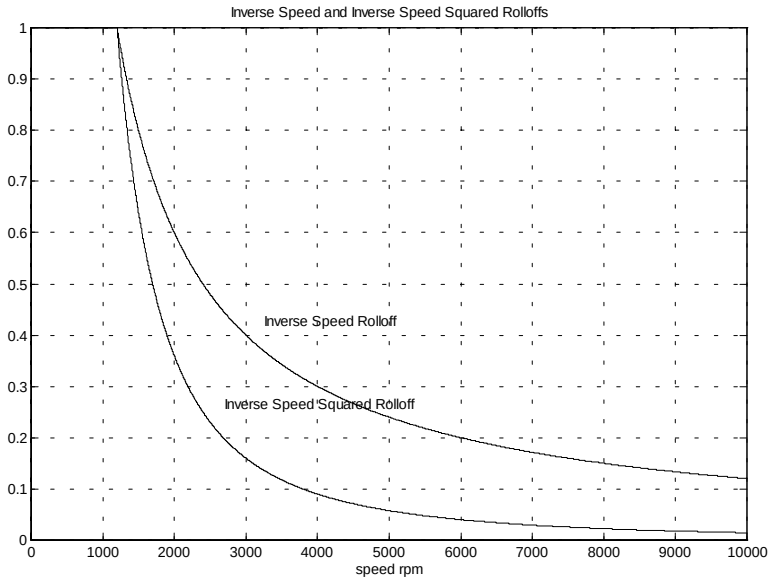


Fig. 6.21. Comparison of shudder compensator output rolloff rates.

Figure 6.21 shows that squared rolloff of the compensator output is much faster than regular inverse speed rolloff. However, in the vehicle, the regular speed rolloff was found to be acceptable; even so, it is beneficial to have both options available. The final form of the shudder compensator to be implemented in the DSP is shown schematically in Fig. 6.22.

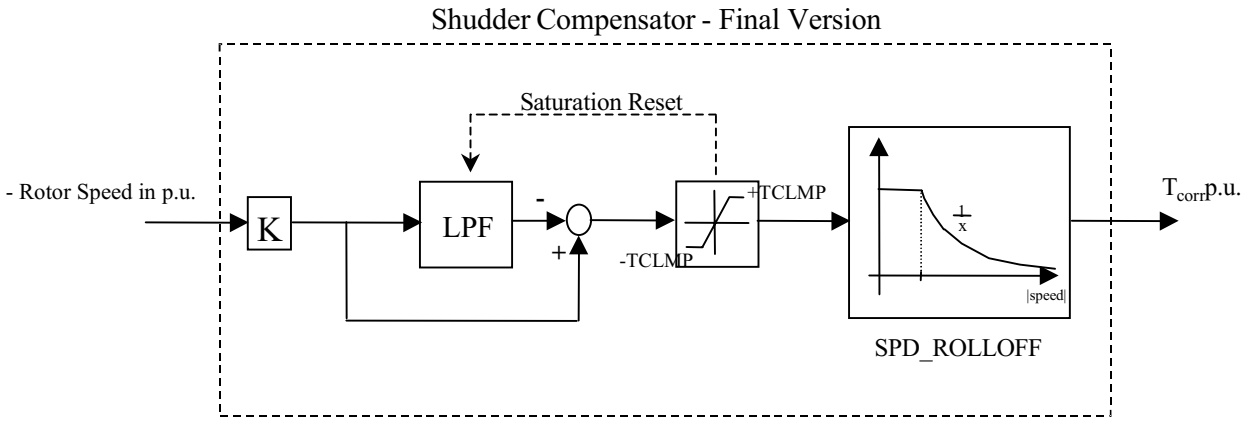


Fig. 6.22. Final form of the implemented shudder compensator.

6.4 Evaluation of the Implemented System

This section evaluates the performance of the compensator developed through both simulation and experiment. The compensator is shown to work well in all cases.

6.4.1 Simulation of Shudder Compensation

The compensator developed is implemented in the control system shown in Fig. 6.7 and tested with a PC using the software program Simulink. The next section will discuss the performance of the compensator placed in the DSP and tested in the vehicle. The simulations done and the corresponding graphs are given in the following pages. Figure 6.23 shows the open loop response, which shows that the rotor rpm has about a 500 rpm pk-pk ripple. Figure 6.24 shows the results of the system simulation when the shudder compensation is enabled. From Fig. 6.24, the graph of the low pass filter output shows that the high frequency components of the speed feedback have been removed for the most part. The output of the compensator is shown to be larger than the clamp value for the simulation. With the added correction torque, the rotor shaft oscillations are largely damped. The high speed torque reversal shows the highest oscillations with the compensator activated, but a high speed torque reversal is unlikely in the real vehicle.

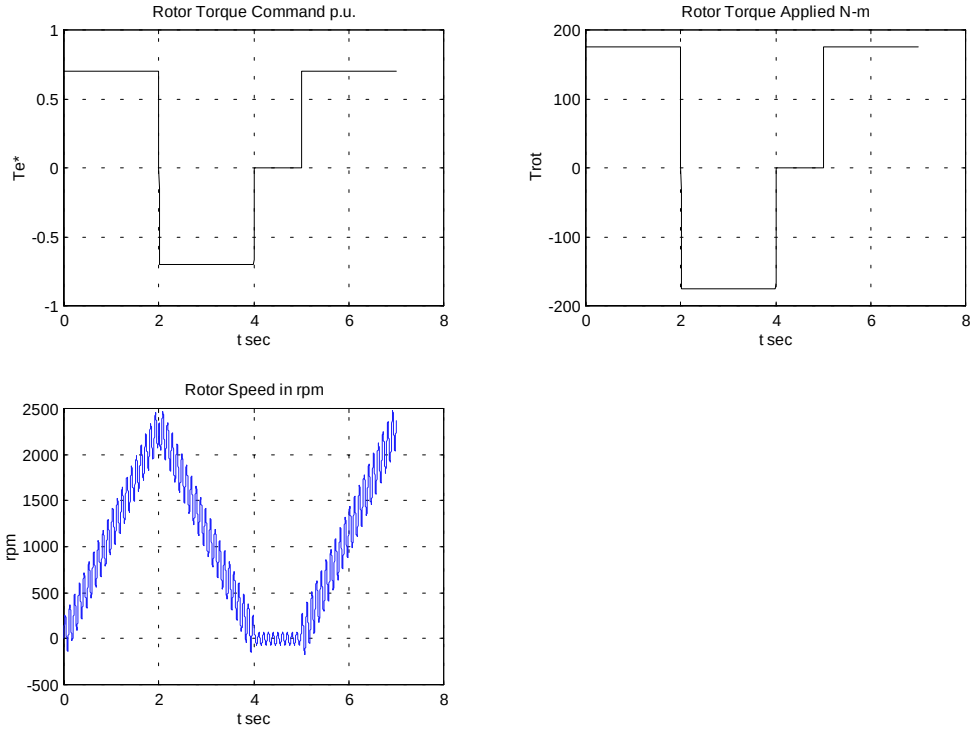


Fig. 6.23. Simulation of the rotor speed response without shudder compensation enabled.

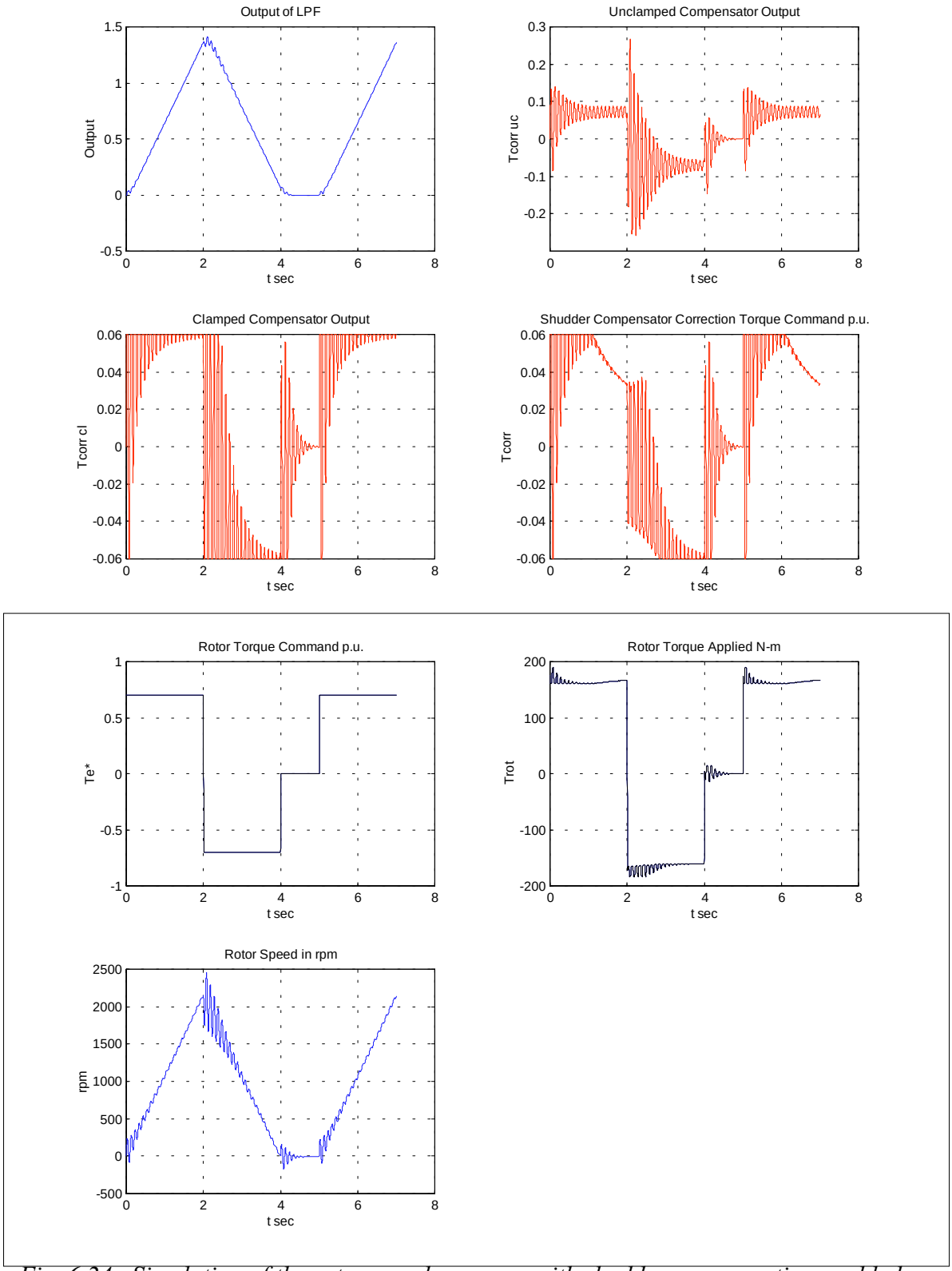


Fig. 6.24. Simulation of the rotor speed response with shudder compensation enabled.

6.4.2 Shudder Compensation Experimental Results

The shudder compensator has been added to the DSP software and thoroughly tested in a mid-size EV truck. The DSP code developed is listed in Appendix M. Figure 6.25 shows the comparison of the rotor shaft oscillations with and without the shudder compensation enabled. When compensation is enabled, the oscillations have been largely reduced. Figure 6.26 shows that the added correction torque, for the case shown in Fig. 6.25, is well within the limit of ±15 N-m. This is better than predicted by the simulation results. The shudder compensator developed has been found satisfactory for eliminating shudder and operates properly under varying driving conditions. The next and final chapter discusses the final added control block for increased EV drive performance: DSP based flux feedback compensation.

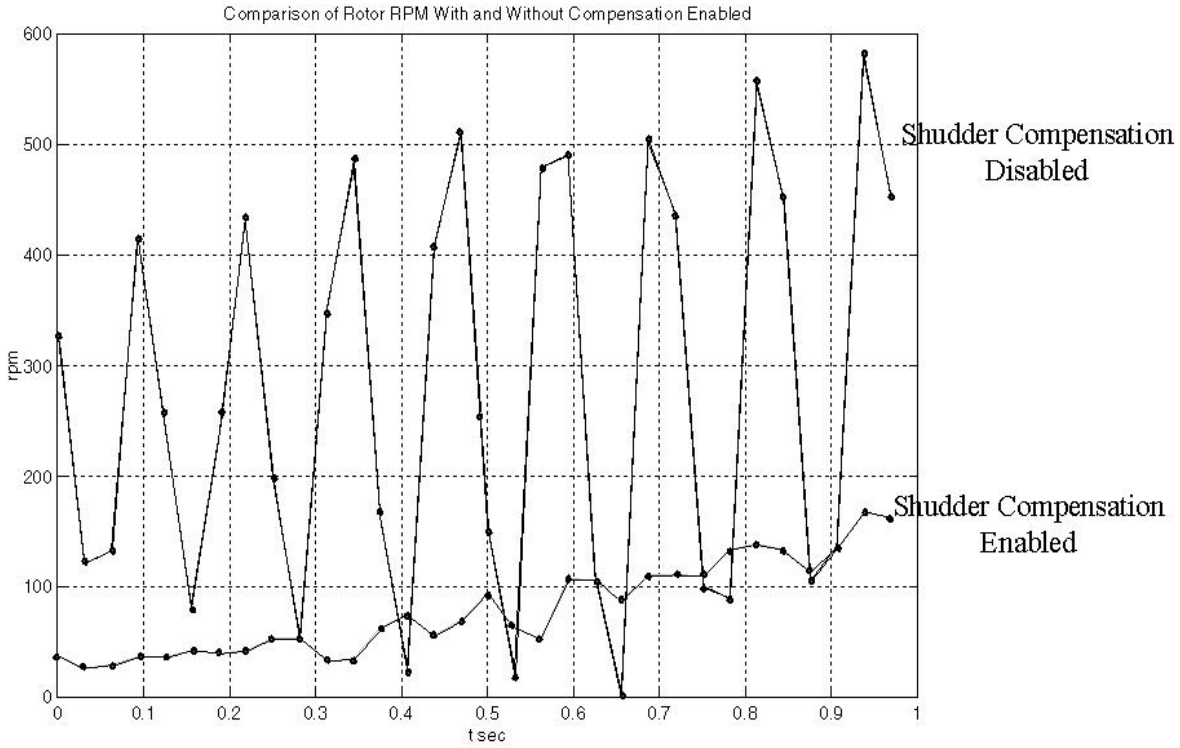


Fig. 6.25. Oscillations in the speed feedback before and after shudder compensation.

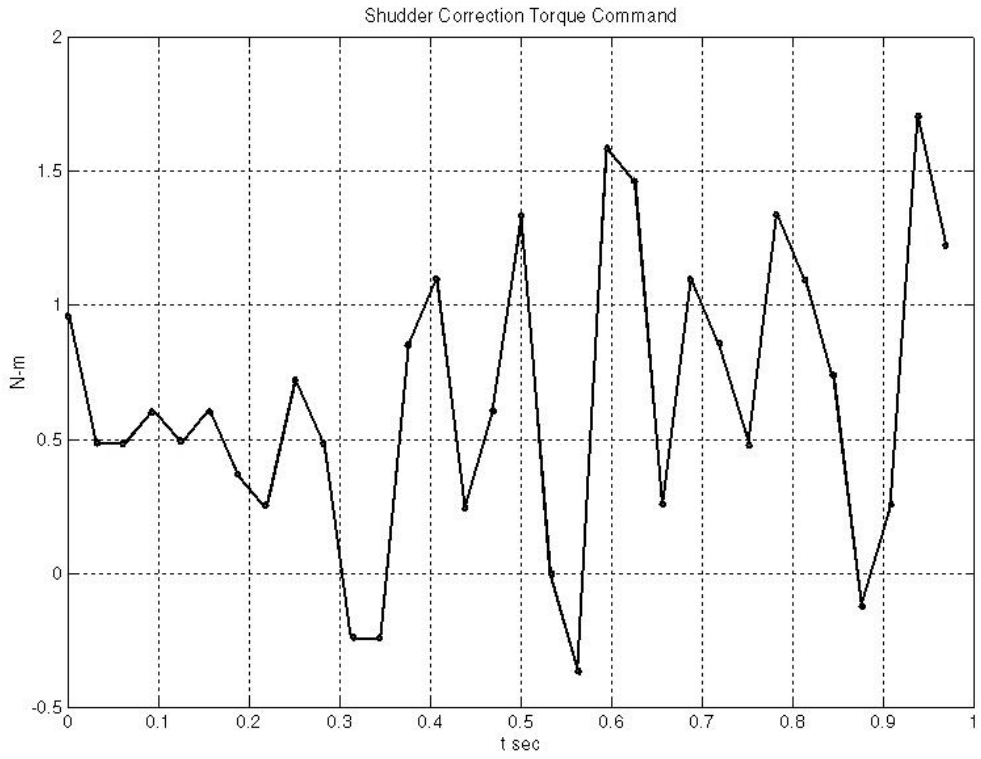


Fig. 6.26. Amount of added correction torque due to shudder compensation.

Chapter 7. Development of a DSP Based Flux Feedback Compensator

7.1 Description of Flux Feedback Problem

This chapter discusses the addition of the final control block to the EV induction motor control system that will further enhance the drive performance. The control block added will be a stator flux linkage measurement compensator. The compensator is required to correct for the gain and phase error in the measurement of the stator flux linkage from the installed flux coils of the GE induction motor. The compensator developed will also be applicable to drives that use integration of the motor terminal voltage to compute the flux linkage. Correction of the stator flux gain and phase error will allow proper operation of the torque regulator down to very low speeds resulting in reduced parameter sensitivity of the field oriented controller.

Additionally, the added compensation is considered to have possibilities for use in a direct field oriented controlled drive due to the improved ability to measure the field angle at low speeds [10]. The placement of the added stator flux linkage measurement compensator is indicated in Fig. 7.1.

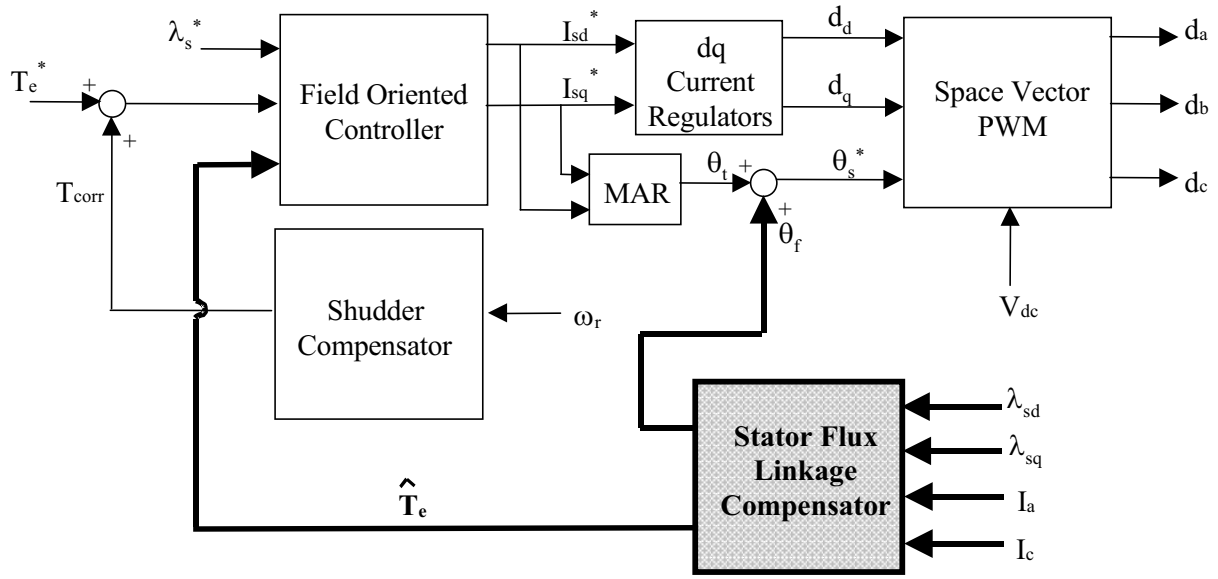


Fig. 7.1. Final controller structure showing added stator flux linkage compensator (highlighted).

Section 7.2 is an analysis of the existing hardware that is used to determine the stator flux linkage d and q components. The existing circuit is shown to behave like an ideal integrator down to approximately 30 rad/sec of applied angular frequency. Below 30 rad/sec, the circuit develops significant phase and then gain error. The gain and phase errors will be determined and plotted.

In section 7.3, the manner in which the feedback electromagnetic torque is computed in the DSP is introduced. Then, the error in the feedback electromagnetic torque caused by just the phase error of the stator flux feedback circuit will be determined and plotted. The next section, section 7.4, discusses the development of the gain and phase correction algorithm. Gain correction is performed by using a gain multiplier, and phase correction is performed by a vector rotation in dq coordinates. Then, the practical implementation of the gain and phase corrections in the DSP stator flux linkage compensator are discussed.

In section 7.5, the experimental results obtained from the implemented compensator are shown.

7.2 Determination of Flux Magnitude and Phase Error at low Frequencies from the Existing Circuit

The stator flux linkage of the GE induction motor is determined by integrating the voltage produced across a special pair of flux coils in the d and q axis of the stator winding. The integration circuit is located on the interface board between the induction motor and the DSP, as shown in Fig. 1.6. This integrator is composed of the analog circuitry shown in Fig. 7.2.

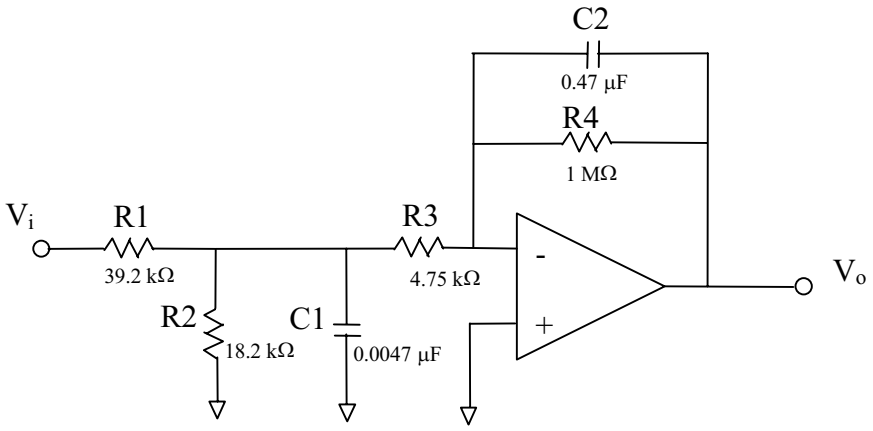


Fig. 7.2. Integration circuit used for flux coil voltage integration.

The transfer function for the circuit shown in Fig. 7.2 is solved and given in Equation (7.1):

$$\frac{V_o}{V_i} = \frac{-R_2}{R_1 + R_2} \bullet \frac{R_4}{R_3 + R_1 // R_2} \bullet \frac{1}{1 + R_4 C_2 s} \bullet \frac{1}{1 + (R_1 // R_2) // R_3 C_1 s} \tag{7.1}$$

The second pole, given by $1/((R_1//R_2)//R_3C_1)$, is far into the LHP and does not affect the low frequency dynamics. However, this pole is helpful in filtering the high frequency components of the induced flux coil voltage due the PWM power supply. The low frequency dynamics can then be estimated by Equation (7.2) for the component values shown in Fig. 7.2.

$$\frac{V_o(s)}{V_i(s)} = \frac{18.46}{1+0.47s} = \frac{39.27}{s+2.13} \tag{7.2}$$

Ideally, the transfer function should be as shown in Equation (7.3), which is the equation for a perfect integrator.

$$\frac{V_o(s)}{V_i(s)} = \frac{39.27}{s} \tag{7.3}$$

The comparison between the actual transfer function, given by (7.2), and the ideal transfer function, given by (7.3), is shown in Fig. 7.3.

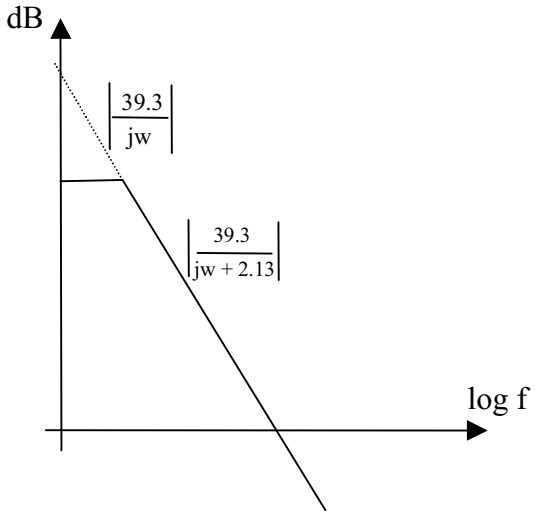


Fig. 7.3. Comparison of ideal and actual integrator transfer functions.

The dc gain of the ideal integrator is infinite. Also, the ideal integrator has a constant phase lag of 90 degrees. The gain and phase differences between the ideal integrator and the actual integrator are shown in Fig. 7.4.

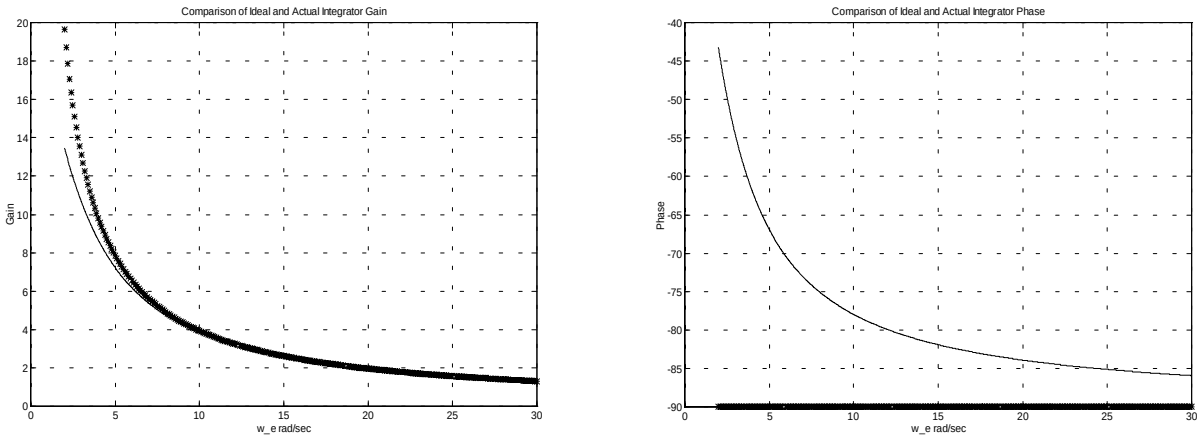


Fig. 7.4. Illustration of the gain and phase differences of the ideal and actual integrators.

7.3 Analysis of Torque Feedback Error at Low Frequencies

In this section, the error in the measured torque feedback, T_{meas}, is determined based on the gain and phase errors in the stator flux linkage measurements that were shown in the previous section.

For the 2-pole GE induction machine, the electromagnetic torque is computed form the following equation, which is given in terms of the complex stator current and flux linkage vectors.

$$T_e = \frac{3}{2} \mathrm{Im}\{\underline{i}_{qds} \underline{\lambda}^*_{qds}\} = \frac{3}{2} (\lambda_{sd} i_{sq} - \lambda_{sq} i_{sd}) \tag{7.4}$$

The complex stator flux linkage vector lags the complex stator voltage vector by 90°, because the flux linkage is the integral of the applied voltage. This situation is shown in Fig. 7.5.

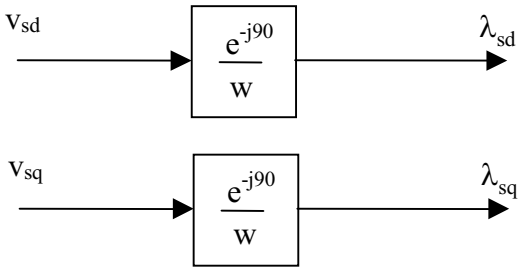


Fig. 7.5. Illustration of stator flux linkage lagging the stator voltage.

The complex stator current vector lags the complex stator voltage vector by the power factor angle, α. The current vector then leads the stator flux linkage vector by $90°-\alpha$. This is shown in Fig 7.6 for positive sequence rotation of the current vector.

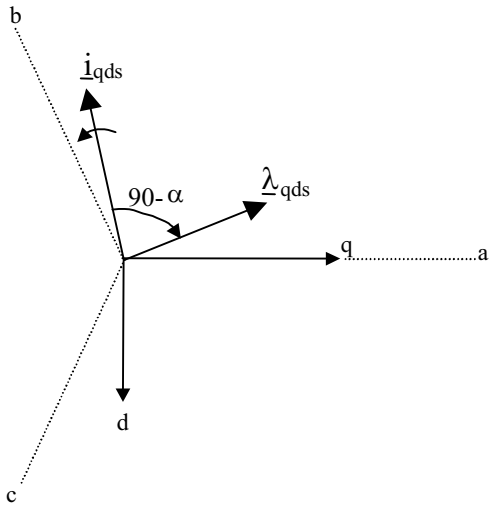


Fig. 7.6. Location of stator current and stator flux linkage vectors for forward motoring.

Assuming a balanced system, the d,q components of the complex stator current vector are given by (7.5)-(7.6) in the rotating reference frame.

$$i_{sq} = \frac{2}{3}[i_{as}\cos\omega_e t + i_{bs}\cos(\omega_e t - \frac{2\pi}{3}) + i_{cs}\cos(\omega_e t + \frac{2\pi}{3})] \tag{7.5}$$

$$i_{sd} = \frac{2}{3}[i_{as}\sin\omega_e t + i_{bs}\sin(\omega_e t - \frac{2\pi}{3}) + i_{cs}\sin(\omega_e t + \frac{2\pi}{3})] \tag{7.6}$$

In (7.5)-(7.6), the phase currents for positive sequence are given by (7.7)-(7.9):

$$i_{as} = I_s\cos\omega_e t \tag{7.7}$$

$$i_{bs} = I_s\cos(\omega_e t - \frac{2\pi}{3}) \tag{7.8}$$

$$i_{cs} = I_s\cos(\omega_e t + \frac{2\pi}{3}) \tag{7.9}$$

Similarly, the complex stator flux linkage vector is determined in the rotating reference frame by (7.10)-(7.11):

$$\lambda_{sq} = \frac{2}{3}[\lambda_{as}\cos(\omega_e t) + \lambda_{bs}\cos(\omega_e t - \frac{2\pi}{3}) + \lambda_{cs}\cos(\omega_e t + \frac{2\pi}{3})] \tag{7.10}$$

$$\lambda_{sd} = \frac{2}{3}[\lambda_{as}\sin(\omega_e t) + \lambda_{bs}\sin(\omega_e t - \frac{2\pi}{3}) + \lambda_{cs}\sin(\omega_e t + \frac{2\pi}{3})] \tag{7.11}$$

In (7.10)-(7.11) the stator flux linkages for positive sequence are given by (7.12)-(7.14), where the power factor angle has been denoted by α.

$$\lambda_{as} = \lambda_s \cos(\omega_e t - \frac{\pi}{2} + \alpha) \tag{7.12}$$

$$\lambda_{bs} = \lambda_s \cos(\omega_e t - \frac{2\pi}{3} - \frac{\pi}{2} + \alpha) \tag{7.13}$$

$$\lambda_{cs} = \lambda_s \cos(\omega_e t + \frac{2\pi}{3} - \frac{\pi}{2} + \alpha) \tag{7.14}$$

With the applied stator angular frequency, ω_e, small, the computed torque will be less than the actual torque due to the finite integrator gain and the integrator phase error. With added phase error, the measured location of the complex stator flux linkage vector becomes phase advanced as shown in Fig. 7.7.

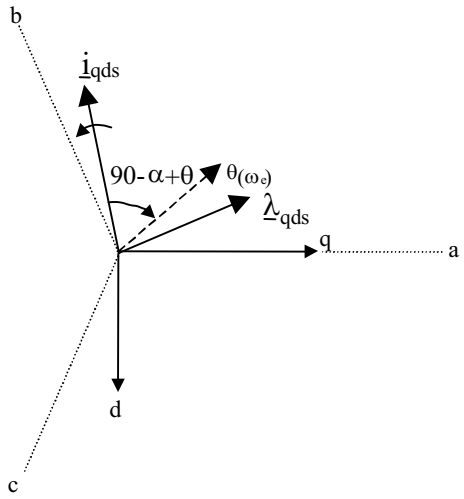


Fig. 7.7. Illustration of phase error advancing of the measured stator flux linkage vector.

With an added phase advance due to measurement error, the expressions for the stator flux linkages become (7.15)-(7.17):

$$\lambda_{as} = \lambda_s \cos(\omega_e t - (\frac{\pi}{2} - \theta(\omega_e)) + \alpha) \tag{7.15}$$

$$\lambda_{bs} = \lambda_s \cos(\omega_e t - \frac{2\pi}{3} - (\frac{\pi}{2} - \theta(\omega_e)) + \alpha) \tag{7.16}$$

$$\lambda_{cs} = \lambda_s \cos(\omega_e t + \frac{2\pi}{3} - (\frac{\pi}{2} - \theta(\omega_e)) + \alpha) \tag{7.17}$$

Nominal values for the GE induction motor are I_s = 424.3 A, λ_s = 0.304 Wb, and α = 0.4 rad. When these numbers are used in Equation (7.4), the nominal electromagnetic torque is found to be 178 N·m. With $\theta(\omega_e)$ allowed to vary between 0 to 50 degrees, or 0 to .873 radians, the reduction in computed electromagnetic torque is plotted in Fig. 7.8.

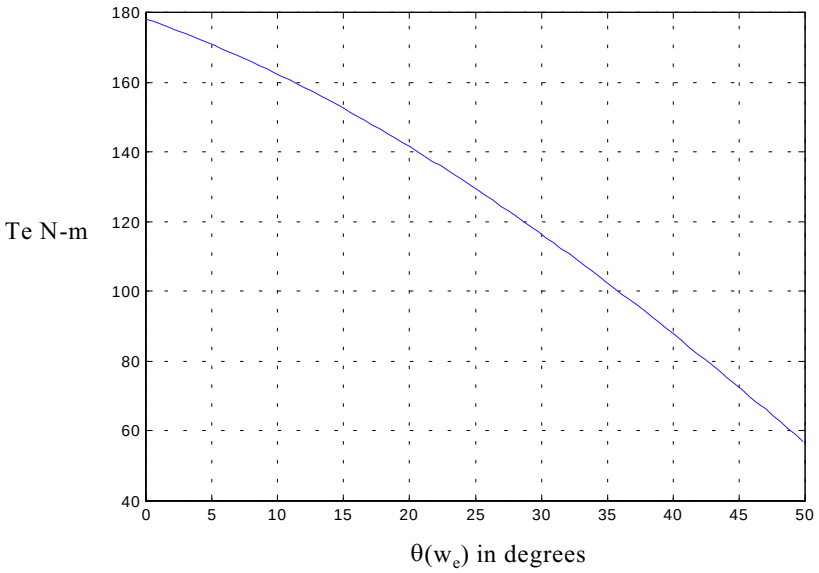


Fig. 7.8. Reduction in computed electromagnetic torque due to phase error in the stator flux linkage measurement.

Additional reduction in the measured electromagnetic torque occurs due to the gain error of the flux integrators, especially when the applied stator angular frequency is less than 5 rad/sec, as shown by Fig. 7.4.

7.4 Development of a Low Speed Flux Magnitude and Phase Error Compensator

The reduction in the computed electromagnetic torque that was shown in the previous section causes the drive controller to generate a larger torque than required because the computed torque is less than the actual torque. Therefore, a flux linkage measurement compensation is required in order to have proper operation at low speed.

In this section, formulas for the correction of the measured flux linkage gain and phase error are derived based on the stator angular frequency and integrator parameters. Then, the form that the developed stator flux linkage compensator takes when implemented in the DSP is shown.

7.4.1 Magnitude Error Compensation

The actual flux integrator transfer function was shown previously to be Equation (7.2), and the desired transfer function was Equation (7.3). To correct the gain error introduced, an adjustable gain correction factor, $G(\omega_e)$, is multiplied by the actual transfer function magnitude to obtain the ideal transfer function magnitude. The placement of the gain correction factor is shown in Equation (7.18):

$$\frac{G(\omega_e)\times 18.46}{\sqrt{1+(0.47\omega_e)^2}} = \frac{39.27}{|\omega_e|} \tag{7.18}$$

From (7.20), the gain correction factor can be determined as:

$$G(\omega_e) = \frac{\sqrt{1+(0.47\omega_e)^2}\ \ \frac{1}{0.47}}{|\omega_e|} \tag{7.19}$$

The gain correction factor is a function of the applied stator angular frequency and the RC time constant of the feedback impedance of the opamp integrator, which for the circuit under test is 0.47. A plot of the gain correction factor for $2 < |\omega_e| < 30$ is shown in Fig. 7.9.

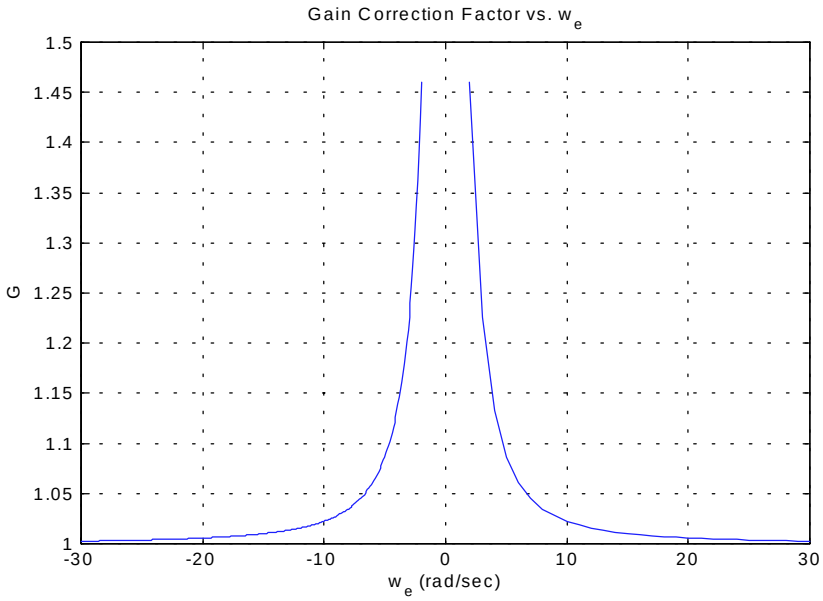


Fig. 7.9. Plot of the gain correction factor vs. applied stator angular frequency.

The same gain correction factor can be used for positive and negative applied stator angular frequency. In general, the applied angular frequency is positive for forward rotation of the rotor and negative for reverse rotation. The gain correction factor is computed as a 5.11 binary format number, as shown in Appendix N. Since a 5.11 number is used, the maximum possible gain correction is 16, which is for an applied angular frequency of about 0.13 rad/sec. Practically, one would not attempt to correct the measurement flux linkage below 1 rad/sec or so due to the noise offsets in the analog integrator and error in the determination of ω_e.

7.4.2 Phase Error Compensation

The stator flux feedback phase error is caused by the difference between the actual transfer function angle and 90°. The phase error introduced by the actual flux feedback transfer function is as follows:

$$\theta(\omega_e) = \frac{\pi}{2} - \text{atan}(0.47\omega_e) \tag{7.20}$$

The phase correction factor, $\theta(\omega_e)$, is a function of the applied stator angular frequency, ω_e, and the RC time constant of the feedback impedance of the opamp integrator, which for the circuit under test is 0.47. The phase correction factor is plotted in Fig. 7.10.

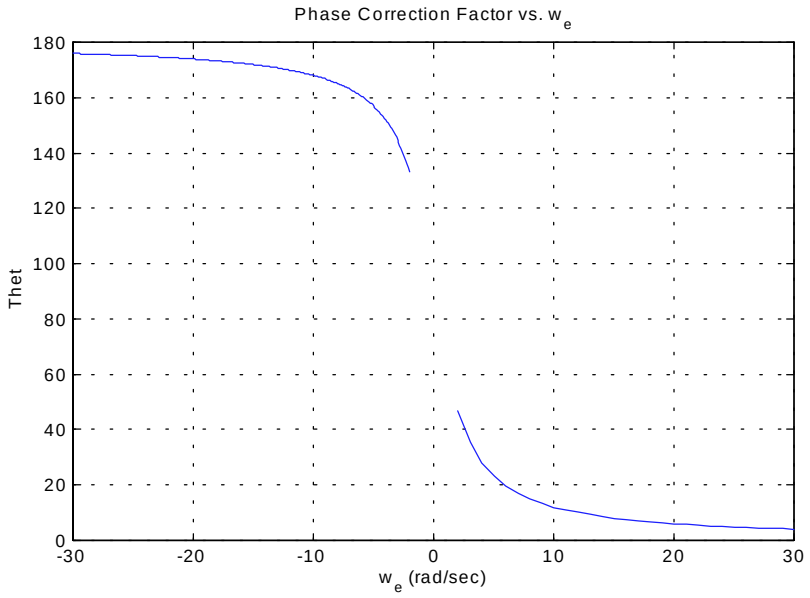


Fig. 7.10. Plot of the phase correction factor vs. applied stator angular frequency as determined from Equation (7.20).

The phase correction can be implemented via a vector rotation that adds phase lag to the measured complex stator flux linkage vector. This vector rotation is implemented via the matrix multiply shown in (7.21) for positive applied stator angular frequency.

$$\begin{bmatrix} \lambda_{sd\_est} \\ \lambda_{sq\_est} \end{bmatrix} = \begin{bmatrix} \cos\theta(\omega_e) & \sin\theta(\omega_e) \\ -\sin\theta(\omega_e) & \cos\theta(\omega_e) \end{bmatrix} \begin{bmatrix} \lambda_{sd\_meas} \\ \lambda_{sq\_meas} \end{bmatrix} \tag{7.21}$$

For negative applied stator angular frequency, the d and q components of the measured stator flux linkage are negated, but the matrix multiply stays the same as shown in (7.22).

$$\begin{bmatrix} \lambda_{sd\_est} \\ \lambda_{sq\_est} \end{bmatrix} = \begin{bmatrix} \cos\theta(\omega_e) & \sin\theta(\omega_e) \\ -\sin\theta(\omega_e) & \cos\theta(\omega_e) \end{bmatrix} \begin{bmatrix} -\lambda_{sd\_meas} \\ -\lambda_{sq\_meas} \end{bmatrix} \tag{7.22}$$

For negative applied stator angular frequency, the phase correction factor can be expressed as shown in (7.23).

$$\theta(\omega_e) = e^{j(\pi+\tilde{\theta}(\omega_e))} \quad \omega_e < 0 \tag{7.23}$$

Where

$$\tilde{\theta}(\omega_e) = |\tan^{-1}(.47\omega_e)| - \frac{\pi}{2} \quad \omega_e < 0 \tag{7.24}$$

For negative ω_e, the values λ_{sd_meas} and λ_{sq_meas} are negated, or multiplied by $e^{j-\pi}$, to cancel the offset of π radians (180°) in the phase correction factor that is shown in Fig. 7.10 for $\omega_e < 0$. Also, for negative ω_e the vector rotation will be in the opposite direction for a net equivalent negative phase correction factor, which is shown to be correct based on the vector diagram shown in Fig. 7.11 for negative phase rotation:

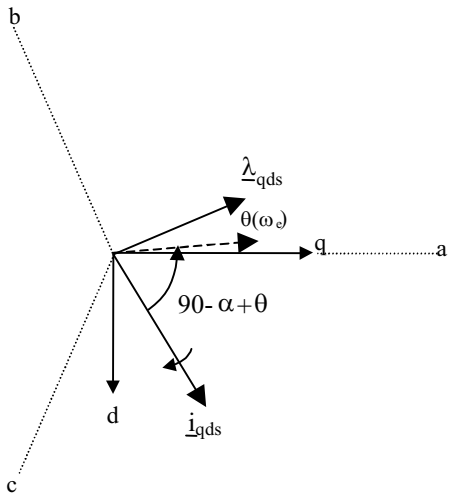


Fig. 7.11. Illustration of phase error advancing of the measured stator flux linkage vector for negative phase rotation.

If the phase correction is properly implemented, the estimated stator flux linkage d and q components, obtained after phase and gain correction, will always lag in time the measured stator flux linkage d and q components for both positive and negative applied stator angular frequency. That the estimated stator flux linkages lag the measured stator flux linkages will be shown to be the case for the system under test in the verification and measurements section.

7.4.3 Implementation Issues

The stator flux linkage measurement compensator, as described previously, is shown in Fig. 7.12.

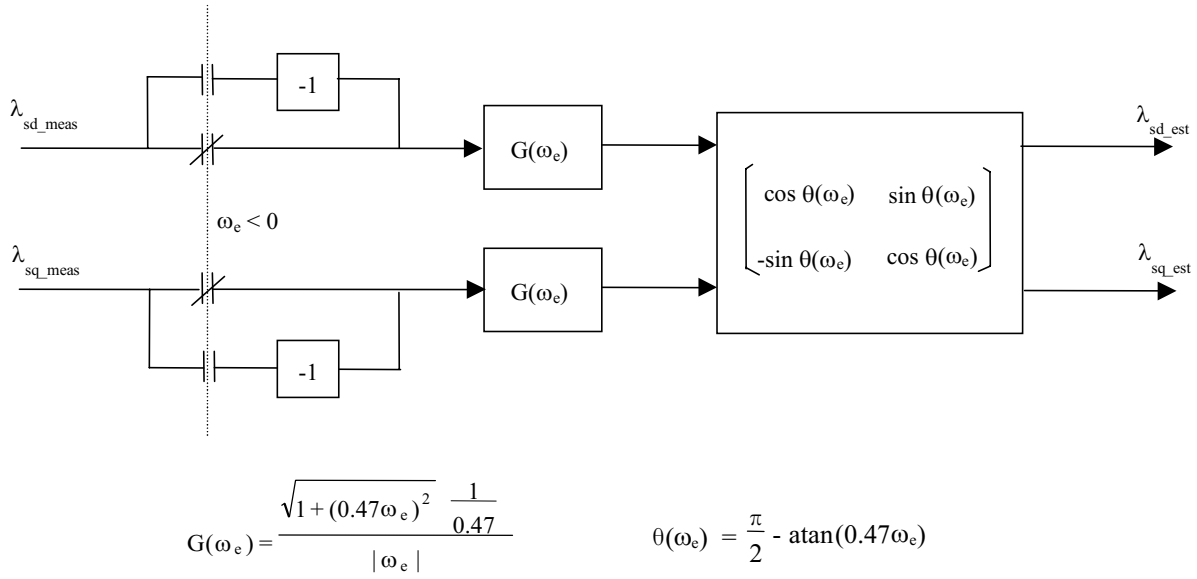


Fig. 7.12. Illustration of implemented stator flux linkage compensator.

The ACMC300 DSP, which was used for this project, has built in ROM utilities for computing sin, cos, and atan [34]. To use the stator flux linkage compensator, the estimated or measured stator angular frequency, ω_e, must also be determined. Assuming that a speed sensor is used, the stator angular frequency can be computed as shown in (7.25).

$$\omega_e = \omega_r + \omega_{slip} \text{ rad/sec} \tag{7.25}$$

In (7.25), ω_r is the measured rotor angular frequency, and ω_{slip} is the commanded slip angle that is output from the field oriented controller, as shown in Fig. 1.9.

Other methods for measuring the applied stator angular frequency without direct measurement of the rotor shaft speed have been researched [11]. With the corrected stator flux linkages obtained, the feedback electromagnetic torque used by the torque regulator is now determined as shown in Equation (7.28):

$$\hat{T}_e = \frac{3}{2} \mathrm{Im}\{\underline{i}_{qds} \underline{\lambda}^*_{qds}est\} = \frac{3}{2}(\lambda_{sd}est \cdot i_{sq} - \lambda_{sq}est \cdot i_{sd}) \tag{7.28}$$

7.5. Flux Feedback Compensator Test Results

A complete DSP algorithm was written to implement the stator flux linkage compensator shown in Fig. 7.12. The assembly code for this algorithm is listed in Appendix N. The compensator was tested down to 3 rad/sec and shown to be fully functional. Measurements of the inverter dc link current confirmed that when the compensation was enabled, the inverter input power was reduced as a result of correct torque regulator operation. A torque transducer was not available on the dynamometer used, so more precise shaft torque measurements could not be obtained.

The experimental waveforms shown in Figures 7.13 and 7.14 verify that the estimated flux always lags the measured flux by the proper amount in both forward and reverse in the actual system. At 6 rad/sec of applied stator angular frequency, the phase correction is about 19.5 degrees, as determined by Equation (7.22), and the gain correction is minimal, as determined by Equation (7.21). A phase lag of 19.5 degrees translates to about 0.06 seconds for a waveform with a 6 rad/sec frequency.

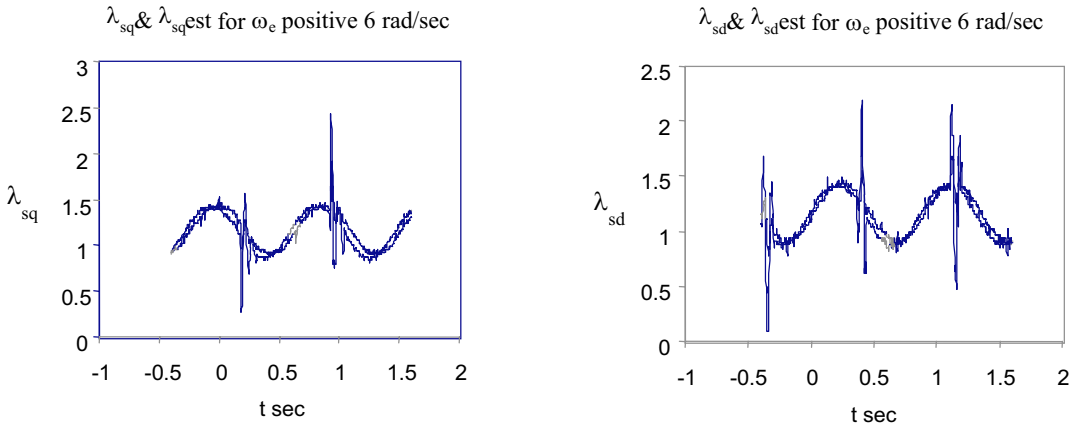


Fig. 7.13. Stator flux linkages for positive stator angular frequency. (The corrected flux linkage lags)

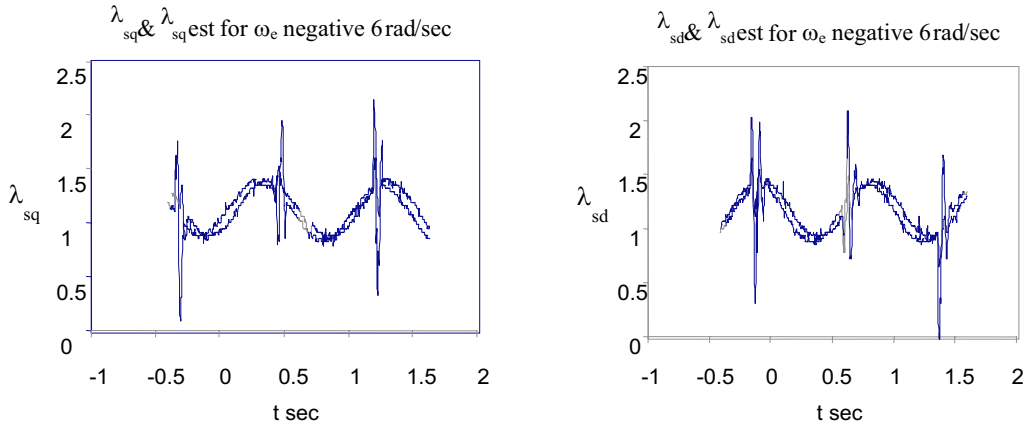


Fig. 7.14. Stator flux linkages for negative stator angular frequency. (The corrected flux linkage lags)

Chapter 8. Summary and Conclusion

This thesis has analyzed and solved various control design and performance issues of an induction motor electric vehicle drive with field oriented control in the stator flux reference frame. The fundamental operation of the drive hardware and control software have been shown. Space vector modulation was shown to be an effective modulation strategy for increasing drive system efficiency throughout the torque vs. speed operating region of an electric vehicle. Test procedures for drive system analysis were created, and new control algorithms were derived for shudder compensation and flux feedback compensation.

In Chapter 2, the different modulation algorithms tested were described. The implementation of hysteresis current control in the EV2000 was shown. Then, the implementation of space vector modulation in the drive system was introduced. The computation of duty cycles was discussed based on using a change of basis matrix multiplication. By choosing the zero vector based on the instantaneous phase current magnitudes, it was shown that the inverter switching losses could be reduced by 50% for most of the operating region of the GE induction motor. Lastly, Chapter 2 explained that the inverter power stage must allow for up to 166ms of continous current conduction if optimal zero vector selection is used down to zero speed. Otherwise, a different zero vector selection routine should be used near zero speed so that the current is more evenly shared by the switch and diode.

In Chapter 3, the development of drive profile generation and implementation software programs were explained. Discrete time equations were developed for the dynamometer shaft commanded torque and speed with the sampled velocity data points from the FUDS and HWFET profiles as inputs. A Qbasic program was developed to generate an ASCII text command file using five different coded commands. The command file generated by the Qbasic program was used by a HP-VEE program that was developed to control the operation of drive profiles on the dynamometer bench. How HP-VEE programs work in general was introduced, and the

development of the program for drive profile implementation was shown using schematics and flow charts. Lastly, Chapter 3 described the construction of a graphical user interface for the HP-VEE program by using the Panel View window.

Next, Chapter 4 introduced the drive system simulation models developed in Visual C++. Models for induction motors, hysteresis current controllers, SVM controllers, and dq current regulators were described. The object oriented design approach was shown to produce modular, reusable, and easily maintainable program code. Lastly, the drive system component models developed were used to demonstrate the sensitivity of the reference vector duty cycle to changes in the stator flux linkage command and the dc link voltage.

In Chapter 5, systematic loss analysis procedures were developed to compare the performance of different EV modulation strategies. The models developed in Chapter 4 were shown to closely approximate the induction motor phase voltage and current at a rotor speed of 2500 rpm and rotor torque of 40 N-m. Next, the operating efficiency points of the drive system with SVM were measured experimentally and plotted as a closely spaced grid of linearly interpolated contours across the torque versus speed plane. Large efficiency improvements were shown when SVM control was used as opposed to hysteresis current control for both the inverter and induction motor. Increases in inverter efficiency with SVM were explained by measuring and comparing the inverter power losses over a varying DC link at 2500 rpm. These measurements showed reduced switching and conduction losses that could be observed in the data plot created. Next, the development of detailed harmonic analysis procedures and the use of a semi-empirical induction motor loss model were used to explain the sharp increases in motor efficiency with SVM. Lastly, energy consumption measurements made on simulated driving profiles gave range increase estimates of 26.5% for FUDS type of driving and 8.3% for HWFET type of driving when SVM control is used as opposed to hysteresis control. These range increases show the impact that the modulation strategy can have on the performance of an EV.

Shudder compensation, which is an absolute necessity for vehicle driveabiltity, was developed in Chapter 6. An experimentally obtained data plot showed the large oscillations present in the rotor shaft without compensation. Next, a plant model was developed for doing simulations by modeling the induction motor and vehicle as a two mass system. The equation for vehicle dynamics was used to determine the two mass system parameters, and spectral analysis of the rotor speed oscillations determined the location of the plant complex pole pair. A

phase lead compensator was developed to compensate for shudder and was placed in the speed feedback path. It was determined that a large compensator gain was desirable but was limited by practical implementation issues, such as controller structure, magnitude of the feedback gain, and the clamping of the shudder compensator output. Quantization of the rotor speed feedback signal was shown to not be a major limitation. The final shudder compensator, which included speed rolloff, was thoroughly tested using both simulation and experiment and shown to be an effective solution.

In Chapter 7, a stator flux linkage gain and phase error compensator was developed. The exact errors were determined from the existing analog integrator circuit and plotted. Next, the effects of the measurement errors were shown by illustrating the movement of the stator flux linkage vector in the complex dq plane and showing that this movement caused incorrect measurement of the feedback electromagnetic torque. Compensation methods were then derived to correct for both the gain and phase errors. A gain correction formula was derived that depended in a non-linear way on the applied stator angular frequency and RC time constant of the integrator circuit. To correct phase error, a trigonometric function was derived based again on the applied stator angular frequency and RC time constant of the integrator circuit. The actual phase rotation was done by using a matrix multiply on the flux linkage vector. Experimental results showed that the compensation procedure developed worked as expected for both positive and negative applied stator angular frequencies.

The test procedures and systems developed in this thesis are expected to have a wide range of applicability as EVs continue to replace the cars, trucks, and buses of today. The dynamometer test bench test procedures developed can be used to test different inverter and induction motor designs in a dynamic fashion. The space vector modulation algorithm shows much promise for increasing vehicle range and decreasing the cost and size of both the inverter and induction motor. Shudder compensation will be required for any vehicle with a long driving shaft. The flux feedback compensation developed is directly applicable to drive systems using torque regulation which include direct field oriented drives that use flux coils or measured phase voltages to find the stator flux linkage.

REFERENCES

[1] M. Ehsani, K. M. Rahman, and H. Toliyat, "Propulsion system design of electric and hybrid vehicle," *IEEE Trans. on Ind. Elect*., vol. 44, no. 1, Feb. 1997, pp. 19-27.

[2] V. Himamshu Prasad, Dushan Boroyevich, and Stephen Dubovsky, "Comparison of high frequency PWM algorithms for a voltage source inverter," *Proc VPEC 1996 Ann. Meet*, pp. 115-122.

[3] D. M. Brod and D. W. Novotny, "Current control of VSI-PWM inverters," in *IEEE/IAS Annu. Meet*., 1984, pp. 418-425.

[4] R. Krishnan, Aravind S. Bharadwaj, "A review of parameter sensitivity and adaptation in indirect vector controlled induction motor drive systems," in *IEEE Trans. on Power Elec*., vol. 6, no. 4, Oct. 1991, pp. 695-703.

[5] K. B. Nordin, D. W. Novotny, D. S. Zinger, "The influence of motor parameter deviations in feedforward field orientation drive systems," *in IEEE Trans. on Ind. Appl*., vol. IA-21, no. 4, July/Aug. 1985, pp. 1009-15.

[6] R. W. De Doncker, "Parameter sensitivity of indirect universal field-oriented controllers," in *IEEE Trans. on Power Elec*., vol. 9, no. 4, July 1994, pp. 367-376.

[7] X. Xu, R. De Doncker, and D. W. Novotny, "A stator flux oriented induction machine drive," in *PESC 1988 Conf. Rec*., pp. 870-876.

[8] X. Xu, R. De Doncker, and D. W. Novotny, "A stator flux oriented control of induction machines in the field weakening region," in *IEEE-IAS Conf. Rec*., 1988, pp. 437-443.

[9] R. W. De Doncker and D. W. Novotny, "The universal field oriented controller," in *IEEE Trans. Ind. App*., vol 30, Jan/Feb 1994, pp. 92-100.

[10] T. G. Habetler, F. Projumo, M. Pastorelli, "Direct torque control of induction machines over a wide speed range," in IEEE-IAS Conf. Rec., 1991, pp. 600-606.

[11] B. K. Bose, M. G. Simoes, D. R. Crecelius, K. Rajashekara, and R. Martin, "Speed sensorless hybrid vector controlled induction motor drive," in *Conf. Rec. IEEE-IAS Annu. Mtg.*, 1995, pp. 137-143.

[12] B. K. Bose and Nitin R. Patel, "A programmable cascaded low-pass filter-based flux synthesis for a stator flux-oriented vector-controlled induction motor drive," *in IEEE Trans. on Ind. Elect*., vol. 44, no. 1, Feb. 1997, pp. 140-143.

[13] T. Rowan and R. Kerkman, "A new synchronous current regulator and an analysis of current-regulated PWM inverters," in *IEEE Trans. Ind. Appl*., vol. IA-22, no. 4, July/Aug. 1986, pp 678-690.

[14] D. W. Novotny and T. A. Lipo, *Vector Control and Dynamics of AC Drives*. Oxford University Press, 1996.

[15] H. Van Der Broeck, H. Skudelny, and G. Stanke, "Analysis and realization of a pulse width modulator based on voltage space vectors," *in IEEE-IAS Conf. Rec*., 1986, pp. 244-251.

[16] Victor R. Stefanovic, and Slobodan N. Vukosavic, "Space vector PWM voltage control with optimized switching strategy," in *IEEE IAS-1992 Ann. Meet*., pp. 1025-1033.

[17] S. Ogasawara, H. Akagi, and A. Nabae, "A novel PWM sheme of voltage source inverters based on space vector theory," in *European Power Electronics Conf. Rec*., 1989, pp. 1197-1202.
[18] V. Himamshu Prasad, Dushan Boroyevich, and Richard Zhang, "Analysis and comparison of space vector modulation shemes for a four-leg voltage source inverter," in *Conf. Rec. of IEEE APEC 1997*, pp. 864-871.
[19] Ahmet M. Hava, Russel J. Kerkman, and Thomas A. Lipo, "Carrier-based PWM-VSI overmodulation strategies: analysis, comparison, and design," *in IEEE Trans. on Power Elec*., vol. 13, no. 4, July 1998, pp. 674-689.
[20] J. Holtz, "Pulsewidth modulation – a survey," in *IEEE Trans. on Ind. Elec.*, vol 39, no. 5, Oct. 1992, pp. 410-420.
[21] S. Bolognani, M. Zigliotto, "Novel digital continuos control of SVM inverters in the overmodulation range," in *IEEE-APEC Conf. Rec*., 1996, pp. 219-223.
[22] K. M. Rahman, M. Rezwan Khan, M. a. Choudhury, and M. A. Rahman, "Variable-band hysteresis current controllers for PWM voltage-source inverters," in *IEEE Trans. on Power Elec*., vol. 12, no. 6, Nov. 1997, pp. 964-970.
[23] D. Katsis, M. Herwald, Jae-Young Choi, Dushan Boroyevich, and F. C. Lee, "Drive cycle evaluation of a soft-switched electric vehicle inverter," in *Conf. Rec. of IEEE IECON 1997*, pp. 658-663.
[24] E. Klingshirn and H. E. Jordan, "Polyphase induction motor performance and losses on nonsinusoidal voltage sources," *IEEE Trans. on Power Apparatus and Systems*, vol PAS-87, Mar. 1968, pp. 624-631.
[25] B. J. Chalmers, B. R. Sarkar, "Induction motor losses due to nonsinusoidal supply," Proc. IEE, vol. 115, no. 12, pp. 1777-1782, 1968.
[26] F. De Buck, P. Gistelinck, and D. De Backer, "A simple but reliable loss model for inverter-supplied induction motors," *IEEE Trans. on Ind. Appl*., Jan/Feb 1984, pp. 190-202.
[27] Tore M. Undeland and Ned Mohan, "Overmodulation and loss considerations in high-frequency modulated transistorized induction motor drives," IEEE Trans. on Power Elec., vol. 3, no. 4, Oct. 1988, pp. 447-452.
[28] K. Kawagishi, M. Udaka, and M. Akamatsu, "Frequency dependency of induction motor parameters and their measuring method," in *1983 JIEE IPEC*, Tokyo, Japan, pp. 202-213.
[29] M. Sokola, V. Vuckovic, and E. Levi, "Iron losses in current-controlled PWM inverter fed induction machines," in *Records of 8th Mediterranean Electrotechnical Conference*, 1996, pp.361-364.
[30] A. Boglietti, "PWM inverter fed induction motors losses evaluation," in *Electric Machines and Power Systems*, vol. 22, no. 3, May/June 1994, pp. 439-449.
[31] A. Boglietti, P. Ferraris, M. Lazzari, and F. Profumo, "Iron losses in magnetic materials with six-stip and PWM inverter supply," in *IEEE Trans. on Magnetics*, vol. 27, no.6, Nov. 1991, pp. 5334-5336.
[32] D. Katsis, Development of a Testbed for Evaluation of Electric Vehicle Drive Performance, M.S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, August 1997.
[33] R.W. Boothe, "EV2000 Control Dictionary," General Electric Drive Systems, Salem VA.

[34] "ADMC300 DSP Motor Controller Developer's Reference Manual," Analog Devices, Inc. 1997.
[35] "ADSP – 2100 Family User's Manual," Analog Devices, Inc. 1995.
[36] Sanjit K. Mitra, *Digital signal processing a computer-based approach*. New York: McGraw-Hill, 1998.
[37] M. Herwald, Jae-Young Choi, and Dushan Boroyevich, "Comparison of Modulation Methods for Induction Motor Electric Vehicle Drives," *Proc VPEC 1998 Ann. Meet*, pp. 77-84.
[38] L. Baghli, H. Razik, A. Rezzoug, "A stator flux oriented drive for an induction motor with extra (alpha, beta) coils," in *Conf. Rec. of IEEE IECON 1998*, vol 4. pp. 2522-6.
[39] Ying-Yu, Ming-Fa Tsai, Yuh-Farn Lin, and Hwa Wu, "Dual DSP Based Fully Digital Control of an AC Induction Motor," *IEEE ISIE Conf.* , 1996, vol. 2, pp. 673-78.
[40] Chee-Mun Ong, *Dynamic Simulation of Electric Machinery*. Prentice Hall, 1998.
[41] Ramesh V. Kanekal, Modeling, Simulation and Analysis of an Indirect Vector Controlled Induction Motor Drive, M.S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, 1987.
[42] Vas, Peter, *Sensorless Vector and Direct Torque Control*. New York: Oxford University Press, 1998.
[43] Motorola, "M68HC11 Literature," *http://www.mcu.motsps.com/lit/hc11.html* 1998.
[44] Krause, Paul C., *Analysis of Electric Machinery*. New York: McGraw Hill, 1995.
[45] Kazmierkowski, Marian P, Dzieniakowski, Maciej A., "Review of Current Regulation Techniques for Three-Phase PWM Inverters," in *Conf. Rec. of IEEE IECON 1994*, pp. 567-575.
[46] Hewlett Packard Web Site, "HP VEE Home," *http://www.tmo.hp.com/tmo/pia/HPVEE/PIATop/English/* 1999.
[47] Zoran Mihailovic, Himamshu V. Prasad, and Dushan Boroyevich, "Computer Modeling and Analysis of VSI Fed Permanent Magnet Synchronous Motor Drive Systems with Adjustable Levels of Complexity," in *Conf. Rec. of IEEE APEC 1997*, pp. 728-735.

APPENDIX

Appendix A: Vehicle Parameters for a Mid-Size EV Truck

$m = 1600$ kg
$r = 0.32$ m
$GR = 12$
$C_{R0} = 0.008$
$C_{R1} = 0.004$
$A = 2.2\ m^2$
$C_D = 0.42$

Appendix B: Drive Profile Simulation Diagram

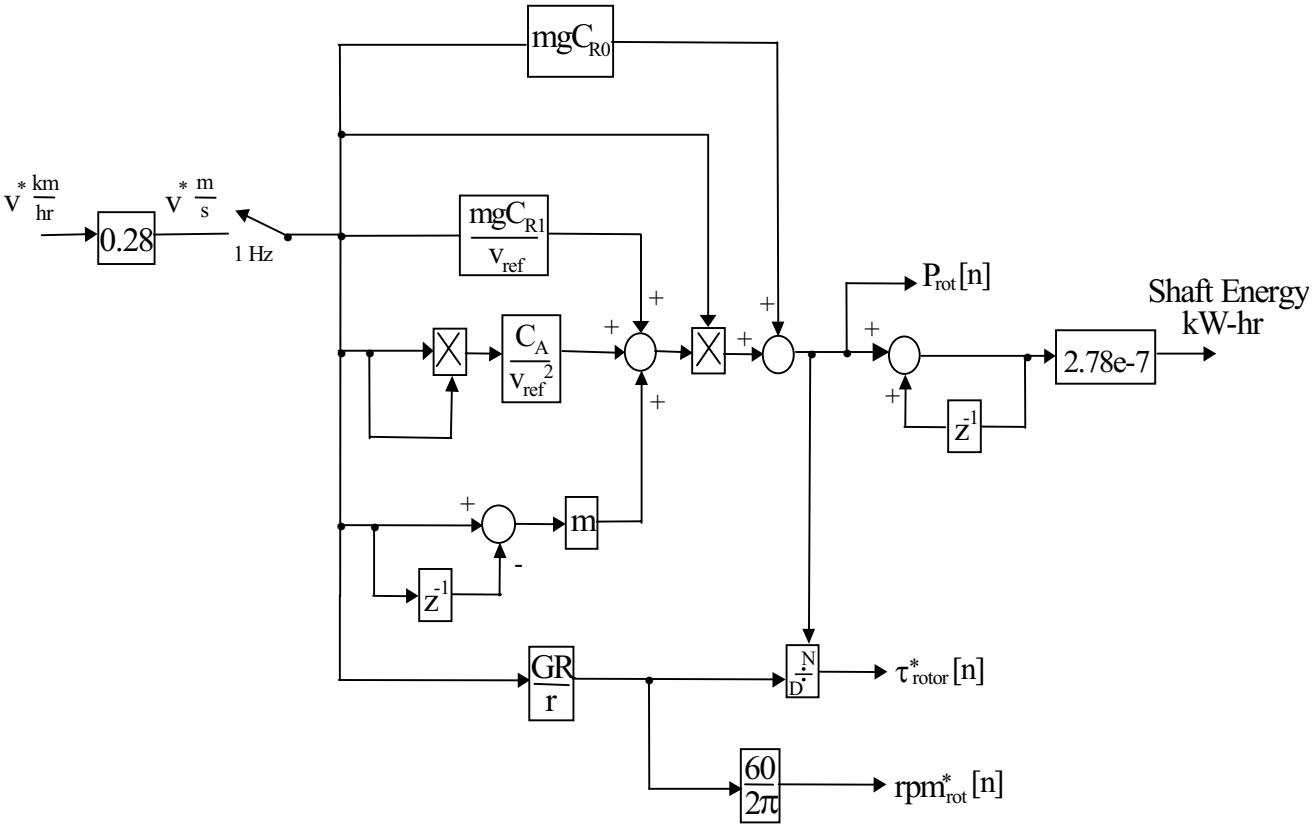


Fig. B-1. Diagram that shows the method used to determine the shaft energy of the IM over a given velocity profile.

Appendix C. Flow Chart for Qbasic Drive Profile Generation Program

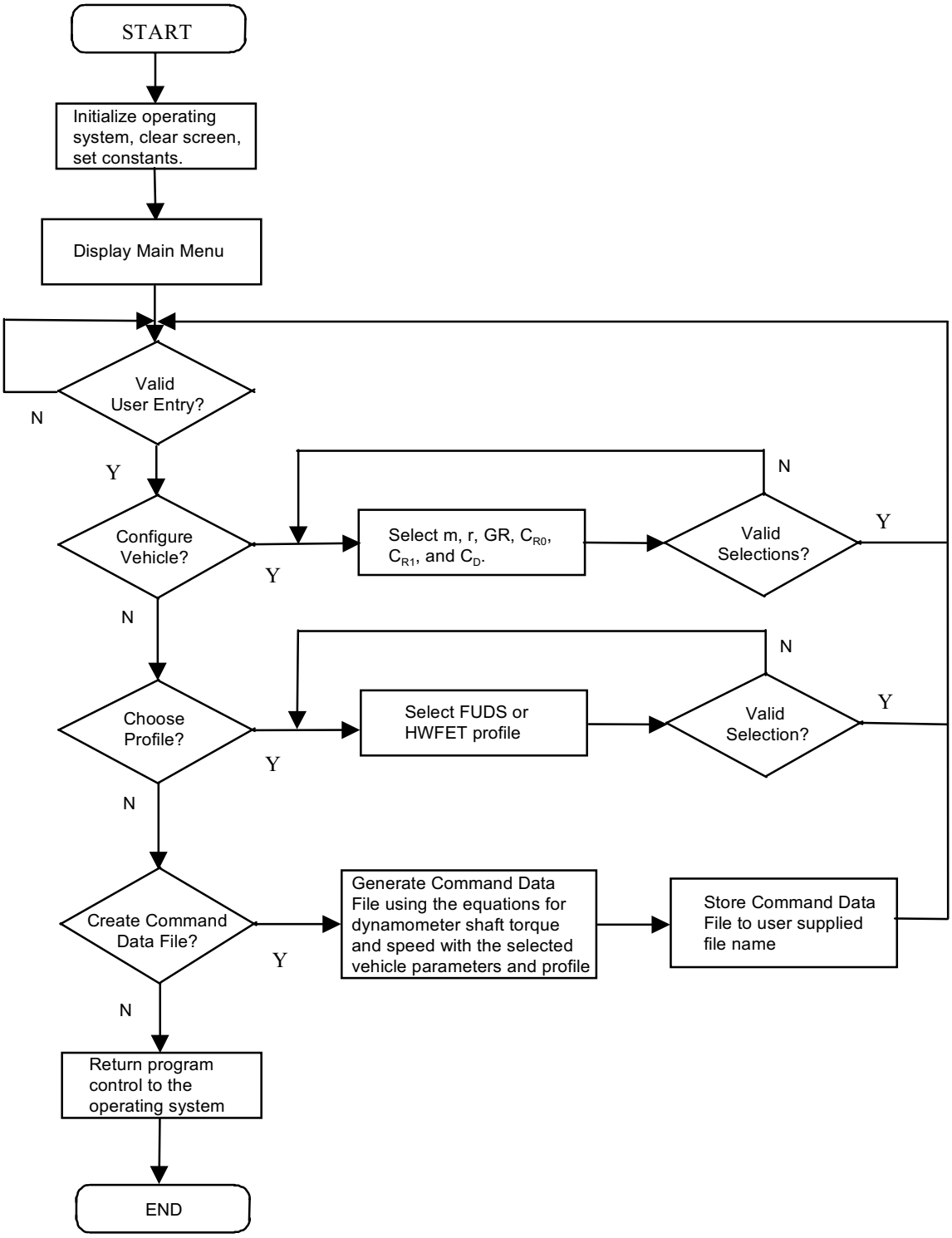


Fig. C-1. Flow Chart for Qbasic Drive Profile Generation Program.

Appendix D: Header File for Induction Motor Model

```
/*****************************************************************
***
*     Author:  Marc Herwald                                Date:  Spring 1999
*
*
*
*   This is the definition and source listing for the induction motor model.  The main public
*
*    function interface is called comp_next_state.
*
*
*
*****************************************************************************
***


//motor.h
#ifndef __motor_h
#define __motor_h

class Motor { public:
//Constructor function prototype
  Motor(double _Rs=.277,
          double _Rr=.183,
          double _Lm=.05383,
          double _Ls=.0553,
          double _Lr=.05606,
          double _J=.01667,
          double _B=.001,
          double _P=4.0);
//Public interface function prototype
  void comp_next_state(const double& _vqs, const double& _vds,
                       double _Tl, double _we);

//Access functions for private data members
  double& wr() {return x[4];}
  const double& iqs() {return x[0];}
  const double& ids() {return x[1];}
  const double& lamdaqr() {return lamqr;}
  const double& lamdadr() {return lamdr;}
  const double& lamdaqs() {return lamqs;}
  const double& lamdads() {return lamds;}
  const double& Te()  {return Tem;}
```

```
//Private function and data member definitions
private:
  void comp_state_derivs(double s[]);
  void calcDel();
  void calcTe(double s[]);
  double Rs, Rr, Lm, Ls, Lr, J, P, Del, h, Tem, Tl, B;
  double vqs, vds;
  double x[5];
  double xdot[5];
  double s[5];
  double K[4][5];
  double lamqr, lamdr, lamqs, lamds;
  double we, k1;
  };
#endif


The state vector, x, is defined as follows:
/*   x =  iqs
          ids
          iqr
          idr
          wr     */
```

Appendix E: Source File for Induction Motor Model

```
//motor.cpp
#include "motor.h"
#include <math.h>

Motor::Motor(double _Rs,      //constructor function
        double _Rr,
        double _Lm,
        double _Ls,
        double _Lr,
        double _J,
        double _B,
        double _P) : Rs(_Rs), Rr(_Rr), Lm(_Lm), Ls(_Ls),
Lr(_Lr),
                     J(_J), B(_B), P(_P)
        { calcDel();
          h = 1e-6;
          for (int cnt=0;cnt < 5;cnt++)
            { x[cnt] = 0;} //Initialize beginning state vector
          calcTe(x);
        }


inline void Motor::calcDel()  {
       Del = (Ls*Lr - Lm*Lm);  }

inline void Motor::calcTe(double s[])   {
       Tem =.75*P*Lm*(s[0]*s[3]-s[1]*s[2]);  }


//Public interface function
void Motor::comp_next_state(const double& _vqs, const double&
_vds,
                            double _Tl, double _we)
  {

   int cnt;

//Assign values to private data members
   vqs = _vqs;
   vds = _vds;
   Tl = _Tl;
   we = _we;

//Compute next state.  See Chapter 4.
```

```
   comp_state_derivs(x);
   for (cnt = 0; cnt < 5; cnt++)
      K[0][cnt] = xdot[cnt];
   for (cnt = 0; cnt < 5; cnt++)
      s[cnt] = x[cnt] + K[0][cnt]*h/2;
   comp_state_derivs(s);
   for (cnt = 0; cnt < 5; cnt++)
      K[1][cnt] = xdot[cnt];
   for (cnt = 0; cnt < 5; cnt++)
      s[cnt] = x[cnt] + K[1][cnt]*h/2;
   comp_state_derivs(s);
   for (cnt = 0; cnt < 5; cnt++)
      K[2][cnt] = xdot[cnt];
   for (cnt = 0; cnt < 5; cnt++)
      s[cnt] = x[cnt] + K[2][cnt]*h;
   comp_state_derivs(s);
   for (cnt = 0; cnt < 5; cnt++)
      K[3][cnt] = xdot[cnt];
   for (cnt = 0; cnt < 5; cnt++)
     x[cnt] += (h/6)*(K[0][cnt] + 2*K[1][cnt] + 2*K[2][cnt] +
K[3][cnt]);
   calcTe(x);

//Compute rotor and stator flux linkage components
   lamqr = Lm*(x[0]+x[2])+(Lr-Lm)*x[2];
   lamdr = Lm*(x[1]+x[3])+(Lr-Lm)*x[3];
   lamqs = Ls*x[0] + Lm*x[2];
   lamds = Ls*x[1] + Lm*x[3];

}

void Motor::comp_state_derivs(double s[])   {

   /*k1 = 0 for stationary reference frame
     k1 = we*(Ls*Lr - Lm*Lm) for synchronous reference frame */

 //k1 = we*(Ls*Lr - Lm*Lm);
 k1 = 0;

   xdot[0] = -(1/Del)*(Rs*Lr*s[0]+(k1+s[4]*Lm*Lm)*s[1]-Rr*Lm*s[2]+s[4]*Lr*Lm*s[3]-
Lr*vqs);
   xdot[1] = -(1/Del)*((-k1-s[4]*Lm*Lm)*s[0]+Rs*Lr*s[1]-s[4]*Lr*Lm*s[2]-Rr*Lm*s[3]-
Lr*vds);
   xdot[2] = -(1/Del)*(-Rs*Lm*s[0]-s[4]*Lm*Ls*s[1]+Rr*Ls*s[2]+(k1-
s[4]*Lr*Ls)*s[3]+Lm*vqs);
   xdot[3] = -(1/Del)*(s[4]*Lm*Ls*s[0]-Rs*Lm*s[1]+(-
k1+s[4]*Lr*Ls)*s[2]+Rr*Ls*s[3]+Lm*vds);
   xdot[4] = (P/2)*((.75*P*Lm*(s[0]*s[3]-s[1]*s[2]))-Tl)/J - (B/J)*s[4];
   //xdot[4] = 0;
```

```
}
```

Appendix F: Header File for Hysteresis Current Controller Model

```
/*****************************************************************
***
*     Author:  Marc Herwald                                Date:  Spring 1999
*
*
*
*  This is the definition and source listing for the hysteresis current controller.  The main public
*
*   function interface is called comp_phase_volt.
*
*
*
*****************************************************************************
***


//modhyst.h
#ifndef __modhyst_h
#define __modhyst_h

//Constructor function prototype
class ModHyst {  public:
  ModHyst(double _ihyst=15, double _vdc=325);
//Public interface function prototype
   void comp_phase_volt(double _iscomm, double _thetas, const
                        double& _iqs, const double& _ids);
//Functions used to access the private data members
   const double& vqs() {return v_qs;}
   const double& vds() {return v_ds;}
private:
   void comp_invert_phase_volts();
   double ihyst, vdc, iascomm, ibscomm, icscomm, van, vbn, vcn,
vn;
   double v_qs, v_ds, ias, ibs, ics, vas, vbs, vcs;
   double ias_err, ibs_err, ics_err;
   };

#endif
```

Appendix G: Source File for Hysteresis Current Controller Model

```
//modhyst.cpp
#include "modhyst.h"
#include <math.h>

const double pi = 3.1415926536;

ModHyst::ModHyst(double _ihyst,         //constructor function
              double _vdc) : ihyst(_ihyst), vdc(_vdc)
              {  }

//public function interface
void ModHyst::comp_phase_volt(double iscomm, double thetas,
const
                              double& iqs, const double& ids)
  {

//Translate commanded current vector into the individual phase
//current commands.  See Chapter 2.
    iascomm = iscomm*cos(thetas);
    ibscomm = iscomm*cos(thetas - 2*pi/3);
    icscomm = iscomm*cos(thetas + 2*pi/3);
    ias = iqs;
    ibs = -.5*iqs - (sqrt(3)/2)*ids;
    ics = -.5*iqs + (sqrt(3)/2)*ids;

//Determine the current errors
    ias_err = ias - iascomm;
    ibs_err = ibs - ibscomm;
    ics_err = ics - icscomm;

//Perform hysteresis control logic
    if (ias_err > ihyst)
       van = -vdc/2;
    else if (ias_err < -ihyst)
       van = vdc/2;
    if (ibs_err > ihyst)
       vbn = -vdc/2;
    else if (ibs_err < -ihyst)
       vbn = vdc/2;
    if (ics_err > ihyst)
       vcn = -vdc/2;
    else if (ics_err < -ihyst)
       vcn = vdc/2;
```

```
//Compute stator phase voltages
    vn = (van + vbn + vcn)/3;
    vas = van - vn;
    vbs = vbn - vn;
    vcs = vcn - vn;

//Assign the final vqs and vds outputs
    v_qs = vas;
    v_ds = (vcs - vbs)/sqrt(3);

}
```

Appendix H: Header File for SVM Controller Model

```
/*****************************************************************
***
*     Author:  Marc Herwald                                  Date:  Spring 1999
*
*
*
*     This is the definition and source listing for SVM.  The main public function interface
*
*     is called comp_phase_volt.
*
*
*
*****************************************************************************
***


//modsvm.h
#ifndef __modsvm_h
#define __modsvm_h

class ModSVM { public:
//Constructor function prototype
   ModSVM(double _vdc=325, int _Ts=100);
//Public interface function prototype
   void comp_phase_volt(const double& _dd, const double& _dq, const double&
_iqs,
                        const double& _ids, const double& _thetas);
//Functions used to access private data members
   const double& vqs() {return v_qs;}
   const double& vds() {return v_ds;}
   const double& Getvas() {return vas;}
   const double& Getvbs() {return vbs;}
   const double& Getvcs() {return vcs;}
   const double& Getda()  {return da;}
   const double& Getdb()  {return db;}
   const double& Getdc()  {return dc;}
   const double& Getd1()  {return d1;}
   const double& Getd2()  {return d2;}
   const double& Getd0()  {return d0;}
   const double& Getdbeta() {return dbeta;}
   const int& Getvsect() {return v_sector;}
   const int& Getisect() {return i_sector;}
   const int& Getzerovect() {return zero_vect;}
   int& SetTs() {return Ts;}
```

```
   int& SetPD() {return PD;}
   int& Setzerovect() {return zerovect_sel;}
   double& SetVdc() {return vdc;}
//Private function prototypes and data members
private:
   void comp_vsect();
   void comp_duty_cycles();
   void comp_zerovect_direct_direct();
   void comp_zerovect_optimal();
   void comp_zerovect_alternating();
   void comp_i_sector();
   void comp_dadbdc();
   void comp_center_based_pwm();
   void comp_edge_based_pwm();
   void comp_vqs_vds();
   double vdc, dalpha, dbeta, d1, d2, d0;
   double da, db, dc, van, vbn, vcn, vas, vbs, vcs;
   double ias, ibs, ics;
   double v_qs, v_ds;
   double thetas;
   int time_ctr, a_big, b_big, c_big, v_sector, i_sector,
zero_vect,
       Ts, PD;
   int spacing_sel, zerovect_sel;
   };

#endif
```

Appendix I: Source File for SVM Controller Model

```
//modsvm.cpp
#include "modsvm.h"
#include <math.h>

const double pi = 3.1415926536;

ModSVM::ModSVM(double _vdc, int _Ts) : vdc(_vdc), Ts(_Ts)
{time_ctr = 0;  zero_vect = 0; PD = 0;}   //constructor function

//public function interface
void ModSVM::comp_phase_volt(const double& dq, const double& dd, const
                 double& iqs, const double& ids, const double&
_thetas)
  {
    thetas = _thetas;

    if (time_ctr == 0)  {

       //Assign phase currents
       ias = iqs;
       ibs = -.5*iqs - (sqrt(3)/2)*ids;
       ics = -.5*iqs + (sqrt(3)/2)*ids;

       //compute reference vector duty cycles in stationary
frame
       dalpha = dd*cos(thetas) - dq*sin(thetas);
       dbeta = dd*sin(thetas) + dq*cos(thetas);

       comp_vsect();       //Determine location of reference
vector
       comp_duty_cycles(); //Find d1,d2, and d0
       comp_i_sector();    //Determine where current is

//Select type of zero vector selection based on setting of
//zerovect_sel
       switch (zerovect_sel)
       {
       case 1:
          comp_zerovect_direct_direct();  //zero vector depends
on
          break;                          //the voltage sector
       case 2:
          comp_zerovect_optimal();        //Optimal zero vector
```

```
          break;                              //selection.  See Chap
2
       case 3:
          comp_zerovect_alternating();    //zero vector
alternating
          break;
       default:
          comp_zerovect_direct_direct();
       }
       comp_dadbdc();          //Compute inverter duty cycles
    }
    comp_center_based_pwm();  //Compute IM voltage waveforms
    comp_vqs_vds();           //Find stator phase voltages
    time_ctr++;
    if (time_ctr == Ts)
       time_ctr = 0;
  }



void ModSVM::comp_vsect()
{

if (dalpha*dbeta > 0)
   {
   if (sqrt(3)*fabs(dalpha) > fabs(dbeta))
      {
      if (dbeta > 0)
         v_sector = 1;
      else
         v_sector = 4;
      }
   else
      {
      if (dbeta > 0)
          v_sector = 2;
       else
          v_sector = 5;
      }
   }
else
   {
   if (sqrt(3)*fabs(dalpha) > fabs(dbeta))
      {
      if (dbeta > 0)
         v_sector = 3;
      else
```

```
         v_sector = 6;
      }
   else
      {
      if (dbeta > 0)
          v_sector = 2;
       else
          v_sector = 5;
      }
   }

return;
}


/* This function computes the d1, d2, and d0 duty cycles as
described in Chapter 2, where a change of Basis definition was
shown */

void ModSVM::comp_duty_cycles()
{

if (v_sector == 1)
   {
   d1 = (float)(dalpha - dbeta*(1/sqrt(3)));
   d2 = (float)(dbeta*(2/sqrt(3)));
   d0 = 1-d1-d2;
   if (d0 < 0)
      d0 = 0;
   }
if (v_sector == 2)
   {
   d1 = (float)(dalpha + dbeta*(1/sqrt(3)));
   d2 = (float)(-dalpha + dbeta*(1/sqrt(3)));
   d0 = 1-d1-d2;
   if (d0 < 0)
      d0 = 0;
   }
if (v_sector == 3)
   {
   d1 = (float)(dbeta*(2/sqrt(3)));
   d2 = (float)(-dalpha - dbeta*(1/sqrt(3)));
   d0 = 1-d1-d2;
   if (d0 < 0)
      d0 = 0;
   }
if (v_sector == 4)
```

```
   {
   d1 = (float)(-dalpha + dbeta*(1/sqrt(3)));
   d2 = (float)(-dbeta*(2/sqrt(3)));
   d0 = 1-d1-d2;
   if (d0 < 0)
      d0 = 0;
   }
if (v_sector == 5)
   {
   d1 = (float)(-dalpha - dbeta*(1/sqrt(3)));
   d2 = (float)(dalpha - dbeta*(1/sqrt(3)));
   d0 = 1-d1-d2;
   if (d0 < 0)
      d0 = 0;
   }
if (v_sector == 6)
   {
   d1 = (float)(-dbeta*(2/sqrt(3)));
   d2 = (float)(dalpha + dbeta*(1/sqrt(3)));
   d0 = 1-d1-d2;
   if (d0 < 0)
      d0 = 0;
   }

return;
}


/*****************************Zero
Vector***********************/
void ModSVM::comp_zerovect_alternating()
{
   if (zero_vect == 0)  {
      zero_vect = 1;
      return;   }
   zero_vect = 0;
   return;
}

void ModSVM::comp_zerovect_direct_direct()
{

if (v_sector == 1)
   zero_vect = 1;
if (v_sector == 2)
   zero_vect = 0;
if (v_sector == 3)
```

```
   zero_vect = 1;
if (v_sector == 4)
   zero_vect = 0;
if (v_sector == 5)
   zero_vect = 1;
if (v_sector == 6)
   zero_vect = 0;

return;
}

void ModSVM::comp_i_sector()
{
     if (ias*ibs > 0)         /*Phase c current is largest*/
     {
          if (ics < 0)
          {
               if ((fabs(ibs)-fabs(ias)) >= 0)
                    i_sector=3;
               else
                    i_sector=2;
          }
         if (ics >= 0)
          {
               if ((fabs(ibs)-fabs(ias)) >= 0)
                    i_sector=9;
               else
                    i_sector=8;
          }
     a_big = 0;
     b_big = 0;
     c_big = 1;
     return;
     }

     if (fabs(ias) > fabs(ibs))
     {
          if (ias < 0)
          {
               if ((fabs(ics)-fabs(ibs)) >= 0)
                    i_sector=7;
               else
                    i_sector=6;
          }
          if (ias >= 0)
          {
               if ((fabs(ics)-fabs(ibs)) >= 0)
```

```
                    i_sector=1;
               else
                    i_sector=12;
          }
     a_big = 1;
     b_big = 0;
     c_big = 0;
     return;
     }

     if (fabs(ibs) >= fabs(ias))
     {
          if (ibs < 0)
          {
               if ((fabs(ias)-fabs(ics)) >= 0)
                    i_sector=11;
               else
                    i_sector=10;
          }
          if (ibs >= 0)
          {
               if ((fabs(ias)-fabs(ics)) >= 0)
                    i_sector=5;
               else
                    i_sector=4;
          }
     a_big = 0;
     b_big = 1;
     c_big = 0;
     return;
     }
}

//Compute optimal zero vector based on the sector you are in and
//which current is largest

void ModSVM::comp_zerovect_optimal()
{

    if (v_sector==1)
     {
          if (a_big == 1)
               zero_vect = 1;
          else if ((b_big == 1)&&(ias > ics))
               zero_vect = 1;
          else if (c_big ==1)
              zero_vect = 0;
```

```
      else
          zero_vect = 0;
 }

 if (v_sector==2)
 {
      if ((a_big == 1)&&(ibs > ics))
           zero_vect = 1;
      else if (b_big == 1)
           zero_vect = 1;
      else if (c_big == 1)
           zero_vect = 0;
      else
          zero_vect = 0;
 }

 if (v_sector==3)
{
      if (a_big == 1)
           zero_vect = 0;
      else if (b_big == 1)
           zero_vect = 1;
      else if ((c_big == 1)&&(ias > ibs))
           zero_vect = 0;
      else
           zero_vect = 1;
 }

 if (v_sector==4)
{
      if (a_big == 1)
           zero_vect = 0;
      else if ((b_big == 1)&&(ias > ics))
           zero_vect = 0;
      else if (c_big == 1)
           zero_vect = 1;
      else
           zero_vect = 1;
 }

 if (v_sector==5)
 {
      if ((a_big == 1)&&(ibs > ics))
           zero_vect = 0;
      else if (b_big == 1)
           zero_vect = 0;
      else if (c_big == 1)
```

```
                zero_vect = 1;
           else
              zero_vect = 1;
      }

      if (v_sector==6)
      {
           if (a_big == 1)
                zero_vect = 1;
           else if (b_big == 1)
                zero_vect = 0;
           else if ((c_big == 1)&&(ias > ibs))
                zero_vect = 1;
           else
                zero_vect = 0;
      }

return;
}

/*************************************************************
**/


//Find IGBT gating duty cycles.  See Chapter 2.
void ModSVM::comp_dadbdc()
{

if (v_sector == 1)
   {
   if (zero_vect == 0)
      {
      da = d1 + d2;
      db = d2;
      dc = 0;
      }
   else
      {
      da = 1;
      db = d0 + d2;
      dc = d0;
      }
   }
if (v_sector == 2)
   {
   if (zero_vect == 0)
      {
```

```
      da = d1;
      db = d1 + d2;
      dc = 0;
      }
   else
      {
      da = d0 + d1;
      db = 1;
      dc = d0;
      }
   }
if (v_sector == 3)
   {
   if (zero_vect == 0)
      {
      da = 0;
      db = d1 + d2;
      dc = d2;
      }
   else
      {
      da = d0;
      db = 1;
      dc = d0 + d2;
      }
   }
if (v_sector == 4)
   {
   if (zero_vect == 0)
      {
      da = 0;
      db = d1;
      dc = d1 + d2;
      }
   else
      {
      da = d0;
      db = d0 + d1;
      dc = 1;
      }
   }
if (v_sector == 5)
   {
   if (zero_vect == 0)
      {
      da = d2;
      db = 0;
```

```
      dc = d1 + d2;
      }
   else
      {
      da = d0 + d2;
      db = d0;
      dc = 1;
      }
   }
if (v_sector == 6)
   {
   if (zero_vect == 0)
      {
      da = d1 + d2;
      db = 0;
      dc = d1;
      }
   else
      {
      da = 1;
      db = d0;
      dc = d0 + d1;
      }
   }

return;

}




void ModSVM::comp_center_based_pwm()
{

//Edge Based pwm
  /*if (time_ctr < (int)100*da)
      van = vdc;
  else
      van = 0;
  if (time_ctr < (int)100*db)
      vbn = vdc;
  else
      vbn = 0;
  if (time_ctr < (int)100*dc)
      vcn = vdc;
  else
```

```
      vcn = 0;    */




/*****************************************************************
****
*
*
*   The following algorithm was invented to simulate the voltage
* *    pulse spacings for center based PWM.  The resolution is 1
us. *
*
*
*****************************************************************
***/

if (time_ctr < Ts/2)
   {
   if (time_ctr < (int)(Ts*(1-da)/2))
      van = 0;
   else
      van = vdc;

   if (time_ctr < (int)(Ts*(1-db)/2))
      vbn = 0;
   else
      vbn = vdc;

   if (time_ctr < (int)(Ts*(1-dc)/2))
      vcn = 0;
   else
      vcn = vdc;
   }

else
   {
   if ((time_ctr-Ts/2) < (int)(Ts*da/2))
      van = vdc;
   else
      van = 0;

   if ((time_ctr-Ts/2) < (int)(Ts*db/2))
```

```
      vbn = vdc;
   else
      vbn = 0;

   if ((time_ctr-Ts/2) < (int)(Ts*dc/2))
      vcn = vdc;
   else
      vcn = 0;
   }


 //Pulse Deletion Logic
 if(Ts*da < PD)
    van = 0;
 if(Ts*da > (Ts-PD))
    van = vdc;
 if(Ts*db < PD)
    vbn = 0;
 if(Ts*db > (Ts-PD))
    vbn = vdc;
 if(Ts*dc < PD)
    vcn = 0;
 if(Ts*dc > (Ts-PD))
    vcn = vdc;

return;

}



void ModSVM::comp_vqs_vds()
{
   vas = (2.0*van - vbn - vcn)/3.0;
   vbs = (2.0*vbn - van - vcn)/3.0;
   vcs = (2.0*vcn - van - vbn)/3.0;

   v_qs = (1./sqrt(3))*(vbs-vcs);
   v_ds = (2.0/3)*(vas-.5*vbs-.5*vcs);

return;
}
```

Appendix J: Header File for dq Current Regulator Model

```
/*****************************************************************
***
*     Author:  Marc Herwald                                Date:  Spring 1999
*
*
*
*     This is the definition and source listing for the dq current regulators.  The main public
*
*     function interface is called comp_duty_cycles.
*
*
*
*****************************************************************************
***


//curreg.h
#ifndef __curreg_h
#define __curreg_h

class CurReg { public:
//Constructor function prototype
   CurReg(double _Kp=1,          //default proportional gain
          double _Ki=.04883,     //default integral gain
          double _Rs = .015);    //default stator resistance
//Public interface function prototype
   void comp_duty_cycles(const double& _isq_comm, const double&
_isd_comm,
                  const double& _isq, const double& _isd, const
double& _thetas);
//Functions used to access private data members
   const double& dq() {return usq;}
   const double& dd() {return usd;}
   const double& Getiqs() {return iqs;}
   const double& Getids() {return ids;}
private:
   double Kp, Ki, Rs, h;
   double usq, usd;
   double iqs_iterm, iqs_iterm_old, usq_uc;
   double ids_iterm, ids_iterm_old, usd_uc;
   double iqs_err, ids_err;
   double iqs, ids;
   };
```

```
#endif
```

Appendix K: Source File for dq Current Regulator Model

```
//curreg.cpp
#include "curreg.h"
#include <math.h>

const double pi = 3.1415926536;


//Constructor Function
CurReg::CurReg(double _Kp,
               double _Ki,
               double _Rs) : Kp(_Kp), Ki(_Ki), Rs(_Rs)
//Do the initializations required for when an object is built
              {iqs_iterm_old = 0;      //Reset q integrator
               ids_iterm_old = 0;      //Reset d integrator
               h = 1e-6;       }       //Set step size

//public function interface
void CurReg::comp_duty_cycles(const double& iqscomm, const double& idscomm,
const
                        double& _iqs, const double& _ids, const double&
thetas)
  {

//When IM in stationary frame, convert currents to synchronous
frame
      ids = _ids*cos(thetas) + _iqs*sin(thetas);
      iqs = -_ids*sin(thetas) + _iqs*cos(thetas);

//When IM in synchronous frame, leave currents as they are
      /*     ids = _ids;
             iqs = _iqs;  */

//Compute the current errors
      iqs_err = iqscomm - iqs;
      ids_err = idscomm - ids;

//Determine q-axis PI regulator output
      iqs_iterm = iqs_iterm_old + Ki*iqs_err*h;
      iqs_iterm_old = iqs_iterm;
      usq_uc = Kp*iqs_err + iqs_iterm;
//Clamp regulator output.  This is not necessarily the best way
//When output gets clamped, instability can result; See Chapter
4
      if (usq_uc > (sqrt(3)/2))
```

```
         usq = sqrt(3)/2;
      else if (usq_uc < -sqrt(3)/2)
         usq = -sqrt(3)/2;
      else
         usq = usq_uc;

  //Determine d-axis PI regulator output
      ids_iterm = ids_iterm_old + Ki*ids_err*h;
      ids_iterm_old = ids_iterm;
      usd_uc = Kp*ids_err + ids_iterm;
//Clamp regulator output.  This is not necessarily the best way
//When output gets clamped, instability can result; See Chapter
4
      if (usd_uc > sqrt(3)/2)
         usd = sqrt(3)/2;
      else if (usd_uc < -sqrt(3)/2)
         usd = -sqrt(3)/2;
      else
         usd = usd_uc;

   }
```

Appendix L: Matlab Induction Motor Loss Analysis Software

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%Author:  Marc A. Herwald                                   Date:  June 1998
%
%
%
%   Purpose:  The code is a Matlab m file program.  The purpose of the code is to find the
%
%   harmonic components of a steady state stator phase current waveform.  Once the harmonics
%
%   are found, the induction motor harmonic losses are estimated.
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

%Setup the constants
L = 65536;                        %FFT length after zero padding
F = 42;                           %Waveform fundamental frequency
Fs = 100000;                      %Sampling frequency
Flim = 30000;                     %Maximum frequency to where harmonics are counted
inoise = 0.1;                      %Minimum harmonic amplitude to count in
computations
Wi=hamming(5000);                  %Create a length 5000 Hamming Window

auw = a;                          %Save un-windowed samples for later
a=a.*Wi;                          %Multiply current samples by Window function
sf=1/sum(Wi);                    %Scale factor needed
Wa=fft(a,L);                      %compute an L length FFT
Wa=sf.*abs(Wa);                  %Scale harmonic currents to actual values
f=0:(2*pi)/L:(2*pi)*(L-.5)/L;    %Determine frequency points
f=f./pi.*50000;
t=0:.00001:.04999;

%Initialize parameters for extraction of current harmonics algorithm
Wa_m1 = 0;
Wa_m2 = 0;
isum = 0;
count = 0;
I = 1;

%Loop used to find the current harmonics and store them in vector ihar:  See Chapter 5
```

```
cnt_stop=round(L*Flim/Fs)
for n=2:1:cnt_stop
  if (Wa_m2 < Wa_m1)&(Wa_m1 > Wa(n))
     if Wa_m1 > inoise
       isum = isum + 2*Wa_m1^2;
       ihar(l) = Wa_m1;
       fhar(l) = f(n);
       l = l + 1;
     end
  end
  if count > 1;
    Wa_m2 = Wa_m1;
    Wa_m1 = Wa(n);
  end
count=count+1;
end


%Compute the estimated power across a 1 ohm resistor and compare to the measured
value %that is obtained from the original current samples:  See Chapter 5
Pest=Wa(1)^2+isum
a_one_cyc=auw(1:round(Fs/F));
v_one_cyc=vuw(1:round(Fs/F));
Pmeas=(F/Fs)*sum(a_one_cyc.^2)
Pinv_out=3*(F/Fs)*sum(a_one_cyc.*v_one_cyc)

if ihar(1) > 15
  m = 2;
elseif ihar(2) > 15
  m = 3;
else
  m = 4;
end


%Estimate IM Harmonic Losses:  See Chapter 5
ihar=ihar(m:l-1);
fhar=fhar(m:l-1);
i_sqr_sum = sum(ihar.^2)
Prc=sum(fhar.^.5.*ihar.^2)
Psc=sum(fhar.^2.*ihar.^2)
Pic=sum(fhar.^1.18.*ihar.^2)


figure(2)
plot(f,Wa)
```

```
grid on
```

Appendix M: Shudder Compensation DSP Assembly Code Routine for the ADMC300 Chip

```
{*************************************************************************
***
*       Author:  Marc Herwald
*
*
*
*       Function Name  :   SHUDDER_COMP_FUNC
*
*
*
*
*
*       Inputs                  speed, SHUD_EN, COMP_GAIN, TCOMP_LAG, TCOMP_CLMP
*
*                               SPD_ROLLOFF, SPD_ROLL_REG, SPD_ROLL_SQR,
*
*                               SPD_ROLL_SEL
*
*
*
*
*
*      Outputs          t_ref_shudder
*
*
*
*
*
*      Variables defined local to TRR_FR.DSP:
*
*
*
*                       lpf_out_msw
*
*                       lpf_out_lsw
*
*                       t_comp_uc
*
*                       t_comp_cl
*
```

```
*
*
*
*
*     Variables defined external to TRR_FR.DSP:
*
*
*
*                         speed
*
*                         speed_abs
*
*                         t_ref_shudder
*
*
*
*
*
*     Description:  The shudder_comp_func is used to compensate for rotor
*
*                       shaft oscillations.
*
*
*
*     Revision History
*
*       3/??/98  Marc Herwald   Added shudder compensation
*
*       3/19/98  Matt Boothe     Added flag to disable shudder
*
*       4/17/98  Marc Herwald   Added some things
*
*       6/11/98  Marc Herwald   Began making the final adjustments for shudder.  Changed
method    *
*                                of speed feedback.  Made the compensator gain a 5.11 number.
*
*                                Added option flag to choose whether 1/x or 1/x^2 speed rolloff.
*
*                                Revised the TUNEUP Parameter names and the variable names.
*
*************************************************************************
***}

shudder_comp_func:

  ax0 = dm(SHUD_EN);                { if shud_en = 0)                }
```

```
   none = pass ax0;                 {    t_ref_shudder = 0            }
   if ne jump go_shudder;           {    lpf_out_msw = 0          }
   ar = pass 0;                     {    return                   }
   dm(t_ref_shudder) = ar;          { else                        }
   dm(lpf_out_msw) = ar;            {    execute shudder compensation   }
   rts;

go_shudder:
   mx0 = dm(COMP_GAIN);          {Compensator Gain in (5.11) format}
   my0 = dm(speed);              {Rotor Speed in pu computed in pwm isr routine}

   mr = mx0*my0 (rnd);           {COMP_GAIN * speed}

   ax0 = mr1;                    {save in ax0 so that can subtract it from the}
                                 {LPF output and use it at other places}

   mx0 = mr1;
   my0 = dm(TCOMP_LAG);
   mr0 = dm(lpf_out_lsw);
   mr1 = dm(lpf_out_msw);
   mr = mr + mx0*my0 (ss);       {Cn-1 + Rn*TCOMP_LAG}
   if mv sat mr;
   mx0 = dm(lpf_out_msw);
   mr = mr - mx0*my0 (ss);       {Cn-1 + Rn*TCOMP_LAG - Cn-1*TCOMP_LAG}
   if mv sat mr;
   dm(lpf_out_lsw) = mr0;
   dm(lpf_out_msw) = mr1;


   ay0 = mr1;                    {get LPF filter output}
   ar = ax0 - ay0;               {COMP_GAIN * speed - LPF output}
   sr = ashift ar by 4 (hi);     {Rescale to (1.15)}
   dm(t_comp_uc) = sr1;          {save ouput of compensator in sr1 and check}
                                 {for clamping next}

   ar = abs sr1;
   ay0 = dm(TCOMP_CLMP);         {ay0 = TCOMP_CLMP}
   ar = ar - ay0;
   if lt jump not_clamped;

   ar = pass sr1;
   if pos jump pos_saturation;
   ar = ay0;
   sr=ashift ar by -4 (hi);
   ay1 = sr1;
   ar = ax0 + ay1;               {negative saturation so}
```

```
                                    {LPFmsw = COMP_GAIN * speed + TCOMP_CLMP}
   dm(lpf_out_msw) = ar;
   ar = -ay0;
   dm(t_comp_cl) = ar;              {t_comp_cl = -TCOMP_CLMP}
   jump speed_rolloff;

pos_saturation:
   ar = ay0;
   sr=ashift ar by -4 (hi);
   ay1 = sr1;
   ar = ax0 - ay1;                  {positive saturation so}
                                    {LPFmsw = COMP_GAIN * speed - TCOMP_CLMP}
   dm(lpf_out_msw) = ar;
   dm(t_comp_cl) = ay0;             {t_comp_cl = TCOMP_CLMP}
   jump speed_rolloff;

not_clamped:
   ar = dm(t_comp_uc);
   dm(t_comp_cl) = ar;


speed_rolloff:
   ay0 = dm(speed_abs);
   ax0 = dm(SPD_ROLLOFF);
   ar = ax0 - ay0;
   if ge jump set_tcomp;            {if abs(speed) < SPD_ROLLOFF skip rolloff}

   ar = dm(SPD_ROLL_SEL);
   ar = pass ar;
   if eq jump reg_rolloff;

   mx0 = dm(t_comp_cl);
   my0 = 0x03d7;                    {0.03 decimal}
   mr = mx0*my0 (ss);
   ay1 = mr1;                       {t_comp_cl * 0.03}
   ay0 = mr0;
   mx0 = dm(speed_abs);
   mr = mx0*mx0 (rnd);
   ax0 = mr1;                       {speed^2}
   divs ay1, ax0;
   cntr = 15;
   do speed_roll_div1 until ce;
speed_roll_div1:  divq ax0;
   mx0 = dm(SPD_ROLL_SQR);          {SPD_ROLLOFF^2*(1/.03)}
   my0 = ay0;
   mr = mx0*my0 (rnd);              {rolloff with P.U. speed^2}
```

```
                                        {base = SPD_ROLLOFF}
   ar = -mr1;
   dm(t_ref_shudder) = ar;          {save shudder compensation}
rts;

reg_rolloff:
   mx0 = dm(t_comp_cl);
   my0 = 0x1999;                    {0.2 decimal}
   mr = mx0*my0 (ss);
   ay1 = mr1;
   ay0 = mr0;
   ax0 = dm(speed_abs);
   divs ay1, ax0;
   cntr = 15;
   do speed_roll_div2 until ce;
speed_roll_div2:  divq ax0;
   mx0 = dm(SPD_ROLL_REG);      {SPD_ROLLOFF*(1/.2)}
   my0 = ay0;
   mr = mx0*my0 (rnd);              {rolloff with P.U. speed}
                                    {base = SPD_ROLLOFF}
   ar = -mr1;
   dm(t_ref_shudder) = ar;          {save shudder compensation}
rts;



set_tcomp:
   ay0 = dm(t_comp_cl);
   ar = -ay0;
   dm(t_ref_shudder) = ar;          {save shudder compensation}
rts;

.endmod;
```

Appendix N: Integrator Compensation DSP Assembly Code Routine for the ADMC300 Chip

```
{*************************************************************************
***
*       Author:  Marc Herwald                                   July 1998
*
*
*
*       Purpose:  Correct the Gain and Phase error in the flux feedback measurements if it
*
*       is enabled.  See Chapter 7.
*
*************************************************************************
***}


ar = dm(INT_COMP_EN);       {If INT_COMP_EN == 0 skip compensation}
ar = pass ar;
if ne jump go_int_comp;

jump int_comp_bypass;

go_int_comp:
mx0 = dm(speed_uf);         {unfiltered measured rotor speed 15krpm/u}
my0 = 0x622c;               {2*pi*250 in (12.4) format}
mr = mx0*my0 (ss);
mx0 = dm(wtm);              {dynamic slip frequency 200rad/s / u}
my0 = 0x0c80;               {200 in (12.4) format}
mr = mr + mx0*my0 (rnd);    {mr = w_e in (12.20)}
dm(w_e_est) = mr1;          {w_e in (12.4)}

ax0 = dm(w_e_est);
ar = abs ax0;
ay0 = dm(INT_COMP_HIGH);        {Upper Limit of Compensation}
ar = ar - ay0;
if gt jump int_comp_bypass;     {skip compensation if |w_e| > INT_COMP_HIGH}

ar = abs ax0;
ay0 = dm(INT_COMP_LOW);         {Lower limit of compensation}
ar = ar - ay0;
if lt jump int_comp_bypass;     {skip compensation if |w_e| < INT_COMP_LOW}
```

```
my0 = dm(RC_CONS);          {RC_CONS (3.13)}
mr = mr1*my0 (ss);          {RC_CONS*w_e (14.18)}
sr = lshift mr0 by -2 (lo);
sr = sr or ashift mr1 by -2 (hi);
mr1 = sr1;                  {integer part of RC_CONS*w_e}
mr0 = sr0;                  {fractional part of RC_CONS*w_e}
m4 = 1;
l4 = 0;

call arctan;

ax0 = 0x4000;               {pi/2 on a per unit base of pi}
ay0 = ar;
ar = ax0 - ay0;             {ar = pi/2 - atan(RC_CONS*w_e)}
dm(phas_corr_fact) = ar;    {Phase correction factor on a pi/u base}
ax0 = ar;
m5 = 1;
l5 = 0;

call admc_sin;

mx0 = ar;                   {mx0 = sin_thet}
m5 = 1;                     {Note: the call to admc_cos does}
l5 = 0;                     {not destroy mx0}

call admc_cos;

mx1 = ar;                   {mx1 = cos_thet}
my0 = dm(flux_d);           {my0 = flux_d}
my1 = dm(flux_q);           {my1 = flux_q}
ax0 = dm(PHAS_SEQ);

ar = dm(w_e_est);
ar = pass ar;
if ge jump w_e_positive;

ar = pass ax0;
if eq jump rev_seq_neg;
mr = mx1*my0 (ss);
mr = mr + mx0*my1 (rnd);    {Applied Stator Frequency is Negative, abc seq.}
ar = -mr1;
dm(flux_d_est) = ar;        {Fdest} {=}  {cos(thet)  + sin(thet)} {-Fd}
mr = mx1*my1 (ss);          {Fqest}      {-sin(thet) + cos(thet)} {-Fq}
mr = mr - mx0*my0 (rnd);
ar = -mr1;
dm(flux_q_est) = ar;
```

```
jump gain_corr;

rev_seq_neg:
mr = mx1*my0 (ss);
mr = mr - mx0*my1 (rnd);       {Applied Stator Frequency is Negative, acb seq.}
ar = -mr1;
dm(flux_d_est) = ar;          {Fdest} {=}  {cos(thet) - sin(thet)} {-Fd}
mr = mx1*my1 (ss);             {Fqest}      {sin(thet) + cos(thet)} {-Fq}
mr = mr + mx0*my0 (rnd);
ar = -mr1;
dm(flux_q_est) = ar;
jump gain_corr;


w_e_positive:
ar = pass ax0;
if eq jump rev_seq_pos;
mr = mx1*my0 (ss);             {Applied Stator Frequency is Positive, abc seq.}
mr = mr + mx0*my1 (rnd);
dm(flux_d_est) = mr1;          {Fdest} {=} {cos(thet)  + sin(thet)} {Fd}
mr = mx1*my1 (ss);             {Fqest}     {-sin(thet) + cos(thet)} {Fq}
mr = mr - mx0*my0 (rnd);
dm(flux_q_est) = mr1;
jump gain_corr;

rev_seq_pos:
mr = mx1*my0 (ss);             {Applied Stator Frequency is Positive, acb seq.}
mr = mr - mx0*my1 (rnd);
dm(flux_d_est) = mr1;          {Fdest} {=} {cos(thet) - sin(thet)} {Fd}
mr = mx1*my1 (ss);             {Fqest}     {sin(thet) + cos(thet)} {Fq}
mr = mr + mx0*my0 (rnd);
dm(flux_q_est) = mr1;


gain_corr:

mx0 = dm(RC_CONS);
mr = mx0*mx0 (rnd);           {RC_CONS^2 (3.13)*(3.13) = (5.11)}
my0 = dm(w_e_est);            {w_e (12.4)}
mr = mr1*my0 (rnd);           {w_e*RC_CONS^2 (16.16)}
mr = mr1*my0 (ss);            {w_e*w_e*RC_CONS^2 (27.5)}

sr = lshift mr0 by 11 (lo);
sr = sr or ashift mr1 by 11 (hi);  {sr = w_e^2*RC_CONS^2 (16.16)}

ay0 = 0x0001;                  {add one before taking square root}
```

```
ar = sr1 + ay0;
mr1 = ar;
mr0 = sr0;

call sqrt;                          {answer will be in sr1 in (8.8)}

my0 = dm(INT_COMP_GAIN);       {1/RC_CONS in (5.11)}
mr = sr1*my0 (ss);              {(1+(.47*w_e)^2)^.5*(1/.47) (12.4)}

ay1 = mr1;                      {numerator in 12.20}
ay0 = mr0;

ar = dm(w_e_est);               {w_e in (12.4)}
ar = abs ar;                    {|w_e|}
sr = ashift ar by 4 (hi);       {denominator, |w_e| in (8.8)}

ax0 = sr1;

divs ay1,ax0;
cntr = 15;
do intcomp_div1 until ce;
intcomp_div1:  divq ax0;

dm(gain_corr_fact) = ay0;      {ay0 = gain_corr_fact (5.11)}

mx0 = dm(flux_d_est);
my0 = dm(gain_corr_fact);
mr = mx0*my0 (rnd);             {gain correction*flux_d_est (5.11)}
sr = ashift mr1 by 4 (hi);
dm(flux_d_est) = sr1;           {flux_d_est in (1.15)}

mx0 = dm(flux_q_est);
mr = mx0*my0 (rnd);             {gain correction*flux_q_est (5.11)}
sr = ashift mr1 by 4 (hi);
dm(flux_q_est) = sr1;           {flux_q_est in (1.15)}
jump int_comp_end;

int_comp_bypass:
ar = dm(flux_d);
dm(flux_d_est) = ar;
ar = dm(flux_q);
dm(flux_q_est) = ar;

int_comp_end;1
```

VITA

Marc A. Herwald was born in Richmond, VA on March 21rst 1974. Soon after, he and his parents and brothers moved to Highland County, VA, where he grew up. The author attended school in the Highland County Public School System. When he wasn't in school, the author usually worked on the farm on various projects. In the fall of 1992, the author started his BSEE at Virginia Tech, which he completed in the spring of 1996. From August 1996 to the present, the author has been working towards his MSEE degree.