Harmonic Control of Multiple-Stator Induction Machines for Voltage Regulation

by

Jack Wade Holloway

Bachelor of Science of Mathematics,
Massachusetts Institute of Technology, 2003

Bachelor of Science of Electrical Engineering,
Massachusetts Institute of Technology, 2003

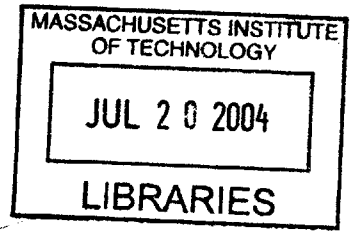
Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering of Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2004



Author .....
Department of Electrical Engineering and Computer Science
June 1, 2004

Certified by .....
Steven B. Leeb
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Harmonic Control of Multiple-Stator Induction Machines for Voltage Regulation

by

Jack Wade Holloway

Submitted to the Department of Electrical Engineering and Computer Science
on June 1, 2004, in partial fulfillment of the
requirements for the degree of
Master of Engineering of Electrical Engineering and Computer Science

Abstract

Small, one to a few horsepower, three-phase induction machines with three sets of electrically-isolated, magnetically-coupled stator winding circuits are described. A voltage inverter is developed and used to drive one set of the machine stator winding circuits. The second set of machine stator winding circuits is connected to a three-phase rectifier in which a path for zero-sequence current is provided from the winding circuits to the rectifier. The last set of stator winding circuits is connected to another three-phase rectifier, however, the stator circuit star point is floating, not providing a zero-sequence current path. By controlling the phase of the third harmonic on the machine drive stator circuits, and thus the waveforms present on the secondary and tertiary stator winding circuits, the output voltage of the rectifier with a zero-sequence current path can be tune above or below the rectifier output without third harmonic injection. The rectifier connected without a zero-sequence current path does not display this tunability with respect to third harmonic phase.

Thesis Supervisor: Steven B. Leeb
Title: Associate Professor of Electrical Engineering and Computer Science

Acknowledgments

I would like to thank my LEES lab mates and the members of Professor Leeb's group. Specifically, I would like to thank Jim Paris, Rob Cox, Riad Wahby, Joshua Phinney, and Mariano Alvira for their help, advice, confessional debugging time, and friendship during this and other work. In addition, I owe a great deal to Professor Steven Leeb, Professor James Kirtley, and Professor David Perreault for their technical advice on this project. My brother, Tanner J. Holloway proofread this work, so I think any reasonable person would blame him for any mistakes[1]. This project owes much of its progress to Mr. John Meszaros of Goodland, Kansas who wound induction machines for these experiments when no one else could.

The United States Navy's Office of Naval Research, the Ford Motor Company, and the Grainger Foundation all contributed funding for this wok. I am indebted to them for their support.

[1] technical or not

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 System Description

THIS THESIS develops a system in which a one voltage drive can be used to simultaneously excite a three-phase induction machine and adjust the output DC voltage of a three-phase rectifier. A specially-constructed polyphase AC induction motor is connected to a voltage drive and to a set of three-phase rectifiers. Through proper control of the spectral content of the drive waveforms, the electric machine can be excited and operated in a traditional fashion while the rectifier circuitry provides an adjustable DC bus.

AC machines are important in the role as servo and traction drives. Generally, the cheapest and most popular class of AC motors are induction machines. These machines are inexpensive to manufacture and can be very simple to operate [3]. Applications involving an electromechanical drive require drive electronics, control, and often other power electronic components. The system investigated in this thesis is a proof-of-concept of a means of eliminating much of the power electronics components not directly involved in the generation of the electromechanical drive excitation.

1.1.1 Multiple-Stator Induction Machine

The work presented in this thesis hinges on an induction machine wound with several parallel[1], bifilarly-wound stator windings. This type of machine can be simultaneously used as both an electromechanical machine and as a polyphase transformer. The transformer action of the multiple stator windings is used to drive three-phase rectifiers, providing the output DC busses of the system.

1.2 Document Organization

Chapter 2 discusses three-phase systems, and addresses the constraints on a three-phases system with respect to third harmonic components. Chapter 3 contains an overview of small horsepower, three-phase, squirrel cage induction machines. The specifics of a multiple-stator induction machine are discussed. Polyphase-rectifier circuits are treated in Chapter 4. Constraints for various operating regimes are discussed, and specific considerations for operation with third harmonic current injection is discussed.

After the individual system components have been discussed, Chapter 5 outlines the system setup, containing a spectrally-pure drive source, the multiple-stator induction machine, and three-phase rectifiers. The concepts developed in previous chapters are experimentally verified using a programmable polyphase lab supply.

Using the system in Chapter 5, a programmable voltage inverter is designed in Chapter 6. Chapter 7 demonstrates the entire system, voltage inverter, machine, and rectifier, functionality. Proper usage of the system is described in this chapter as well.

Chapter 8 concludes this thesis with an overview of the system, results, and a discussion of future work and improvements.

[1] in the electrical sense

Chapter 2

Three-Phase Systems

POLYPHASE systems have a number of advantages over single-phase systems. A polyphase power system is cheaper to build than independent single-phase systems of equivalent capacity. In addition, polyphase electric machinery is both cheaper to manufacture and lighter than an equivalent power single-phase machine. Three-phase systems are the most common type of polyphase system [19], and thus are of considerable commercial importance.

A three phase system could consist of three sinusoidal voltage sources connected in a wye or delta configuration (Figure 2-1). Each of these voltage sources are 120° out of phase with the other two sources. Wye-connected sources are easier to generate from a bridge-based voltage inverter. In addition, a wye-connected three-phase *load* inherently rejects triple-n *current* harmonics when the source and load commons nodes are not shared. The triple-n current harmonics rejected by properly-connected wye-configuration loads are useful for the system examined in this work. We consider wye-connected systems almost exclusively in this work.

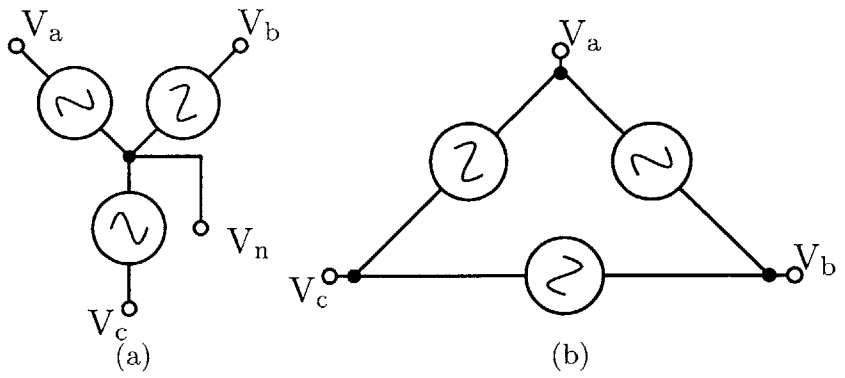


Figure 2-1: Three-phase (a)Wye- and (b)Delta-connections.

2.1 Wye-Connected Systems

2.1.1 Floating Load

Consider the wye-connected system of Figure 2-2. Note that the load, consisting of three identical wye-connected load impedances Z_l, is balanced. In this system, we see that the load common, or *star-point*, is floating; that is, the central nodes of the source and load are unconnected. Kirchoff's current law shows that the phase currents into the load must sum to zero

$$I_a + I_b + I_c = 0. \tag{2.1}$$

If we define the phase currents as a sum of harmonics, we find that there can be no triple-n (or integer multiples of 3) harmonics in the current waveform [9]. See Appendix A for a complete derivation of this requirement.

Given the load phase current harmonic constraints, we know that the *phase to neutral* voltages can be described by

$$V_{an} = \sum_{k=3\cdot i\pm 1}^{\infty} V_k \cdot \cos(k\omega t) \tag{2.2}$$

$$V_{bn} = \sum_{k=3\cdot i\pm 1}^{\infty} V_k \cdot \cos\left(k\left(\omega t + \frac{2\pi}{3}\right)\right) \tag{2.3}$$

$$V_{cn} = \sum_{k=3\cdot i\pm 1}^{\infty} V_k \cdot \cos\left(k\left(\omega t - \frac{2\pi}{3}\right)\right), \tag{2.4}$$

for $i \in \mathbb{Z}$ and V_k the kth harmonic voltage amplitude.

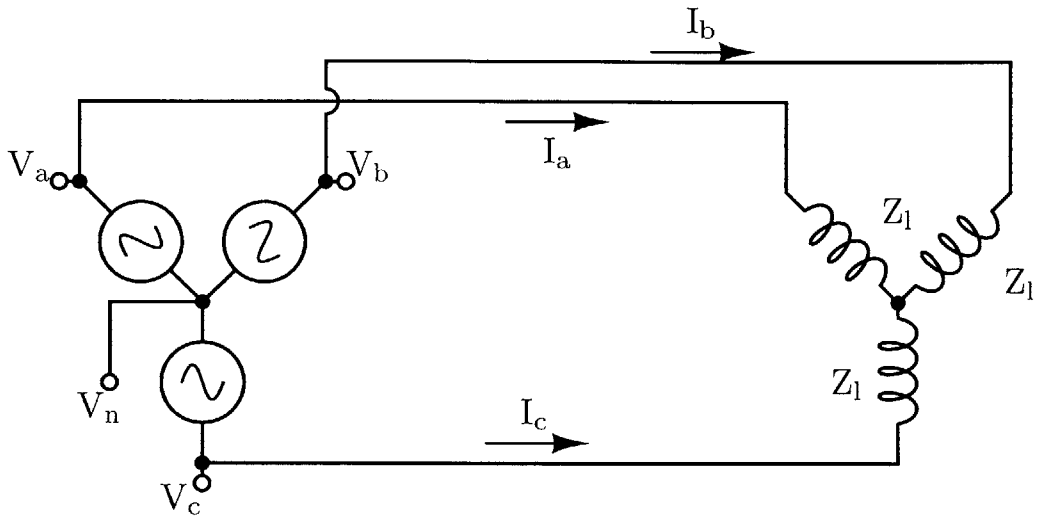


Figure 2-2: Three-phase Wye-connected source and load with disconnected star-points.

2.1.2 Shared Commons

If the three-phase system in Figure 2-2 is modified to include a current path from the load star-point to the driving source neutral, as is shown in Figure 2-3, the restrictions on the harmonic content of the phase currents change. This shared common allows a *zero-sequence current*, I_0, to flow from the source to the load [3, 9, 19].

If zero-sequence current can flow between the source and load, the current constraint in Equation 2.1 becomes

$$I_a + I_b + I_c = I_0.$$

Given this constraint, we find that triple-n current harmonics *can* flow into the load [2, 9]. In addition, the voltage drop across each phase of the load is exactly the same as the corresponding source phase to neutral voltage, neglecting any voltage drops from the source

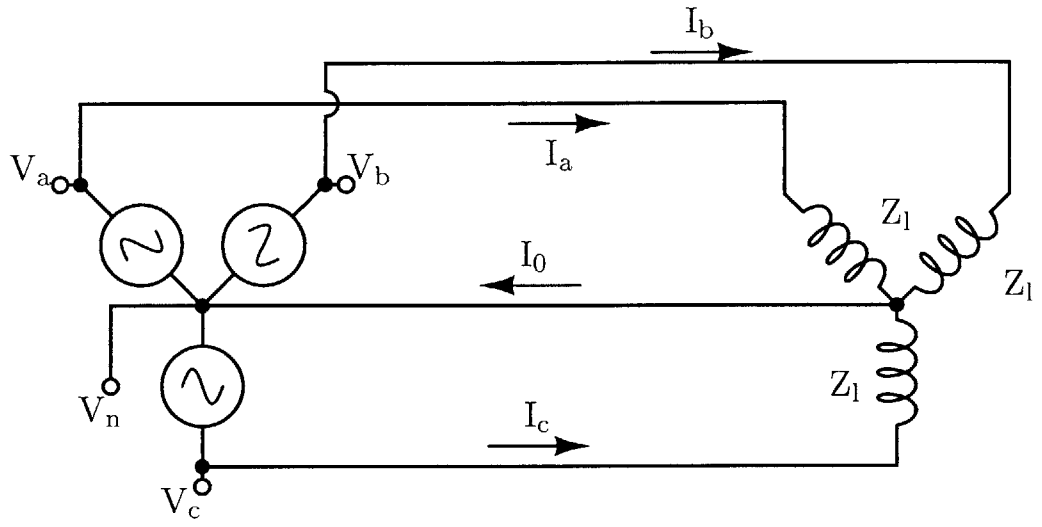


Figure 2-3: Three-phase Wye-connected source and load with connected star-points.

to load. The load current harmonic content can easily be controlled by driving the load with the appropriate voltage harmonics.

The phase voltages and currents are quite simple in this system. For arbitrary amounts of any harmonic, we find

$$V_{an} = I_a \cdot Z_l = \sum_{k=1}^{\infty} V_k \cdot \cos(k\omega t) \tag{2.5}$$

$$V_{bn} = I_b \cdot Z_l = \sum_{k=1}^{\infty} V_k \cdot \cos\left(k\left(\omega t + \frac{2\pi}{3}\right)\right) \tag{2.6}$$

$$V_{bn} = I_c \cdot Z_l = \sum_{k=1}^{\infty} V_k \cdot \cos\left(k\left(\omega t - \frac{2\pi}{3}\right)\right), \tag{2.7}$$

Note that a wye- to delta-connected transformation is topologically impossible in this case [5]. If the zero-sequence current is known a-priori, an equivalent delta-connected circuit can be imagined in which independent time-varying sources are used to provide the needed equivalent I_0.

2.2 Phasor Descriptions of Polyphase Systems

We can describe the sinusoidal quantities, voltage and current, in a polyphase system in terms of the equivalent complex magnitude and angle. Given a voltage component

$$V_k = \overline{V_k} \cos\left(k\omega t + \phi_k\right)$$

for the complex voltage $\overline{V_k}$ and arbitrary phase ϕ_k, we rewrite

$$V_k = \mathrm{Re}\left\{\underline{V_k}\left(\phi_k\right) e^{jk\omega t}\right\},$$

where we define the voltage *phasor* [2, 24],

$$\underline{V_k}\left(\phi_k\right) = \overline{V_k} e^{j\phi_k}. \tag{2.8}$$

Similarly, for a three-phase system with balanced line-to-neutral source voltages

$$V_{an} = V_1 \cdot \cos\left(\omega t + \phi_1\right), \tag{2.9}$$

$$V_{bn} = V_1 \cdot \cos\left(\omega t + \frac{2\pi}{3} + \phi_1\right), \tag{2.10}$$

$$V_{bn} = V_1 \cdot \cos\left(\omega t - \frac{2\pi}{3} + \phi_1\right), \tag{2.11}$$

we find the representative voltage phasors

$$\underline{V_{an}}\left(\phi_1\right) = V_1 e^{j\phi_1}, \tag{2.12}$$

$$\underline{V_{bn}}\left(\phi_1\right) = V_1 e^{j\left(\phi_1 + \frac{2\pi}{3}\right)}, \tag{2.13}$$

$$\underline{V_{cn}}\left(\phi_1\right) = V_1 e^{j\left(\phi_1 - \frac{2\pi}{3}\right)} \tag{2.14}$$

The same procedure yields current phasors for this same system.

The phasors in Equations 2.12-2.14 are complex quantities that can be plotted in the

complex plane at a given time t. The arbitrary phase angle ϕ_1 is included for generality and corresponds to a rotation of *all* the voltage phasors about the origin. Figure 2-4 shows the three-phase phasors from Equations 2.12-2.14 with no phase offset and with $\phi_1 = \frac{\pi}{6}$ at time $t = 0$. Note that the these phasors rotate around the origin in time, as well. At a given time t, all of the phasors will have rotated counterclockwise by ωt.

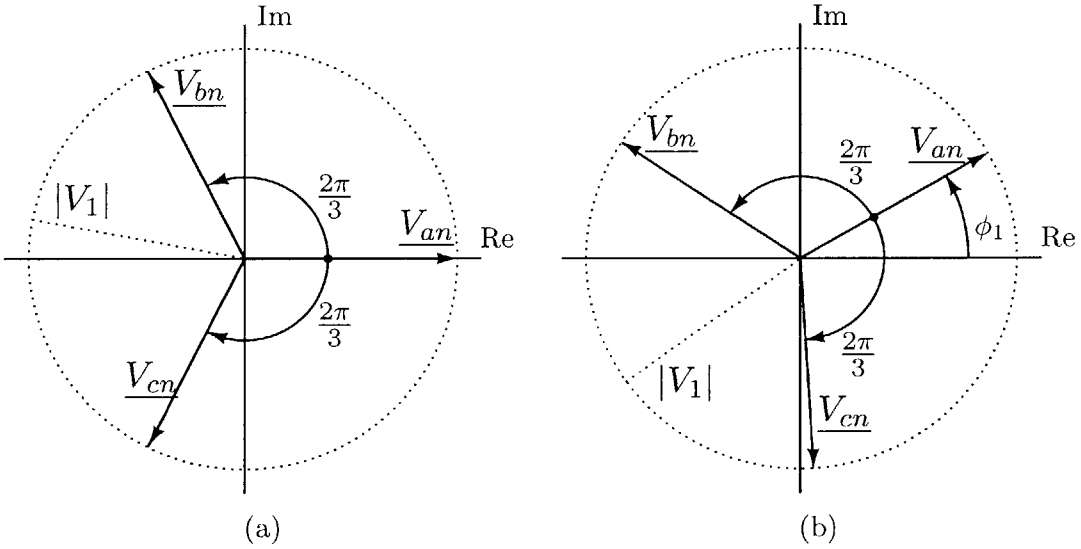


Figure 2-4: Three-phase voltage phasors with (a) $\phi_1 = 0$, and with (b) $\phi_1 = \frac{\pi}{6}$

Phasor concepts can be applied to geometrically determine the allowable harmonic content in a three-phase system, as presented in Section 2.1. We showed that in a balanced-load wye-connected system where the source and load do not share star-points, triple-n current harmonics could not exist. We can see this in that, for all k such that $k = 3 \cdot n \pm 1, n \in \mathbb{Z}$, the phasors vectorially sum to zero. This is perhaps seen most easily by showing the $(k = 3 \cdot n \pm 1)$th harmonic phasor angles[1] become

$$\begin{aligned} (3 \cdot n \pm 1) \cdot (0) &= 0 \\ (3 \cdot n \pm 1) \cdot \left(\frac{2\pi}{3}\right) &= 2\pi n \pm \frac{2\pi}{3} \end{aligned}$$

[1]The arbitrary phase angles ϕ_k have been omitted for simplicity, however their inclusion does not change the angles of the phasors with respect to each other.

$$(3 \cdot n \pm 1) \cdot \left(-\frac{2\pi}{3}\right) = -2\pi n \mp \frac{2\pi}{3}$$

for phase A, B, and C respectively. This corresponds to a rotation of all the phasors by an integer multiple of 2π. In this orientation, the phasors all retain their relative angles with respect to each other, and as such, still sum to zero.

In the case of triple-n harmonics, the harmonic phasors do not sum to zero. For the $(k = 3 \cdot n, n \in \mathbb{Z})$th harmonic, the harmonic phasor angles are

$$\begin{aligned}(3 \cdot n) \cdot (0) &= 0 \\ (3 \cdot n) \cdot \left(\frac{2\pi}{3}\right) &= 2\pi n \\ (3 \cdot n) \cdot \left(-\frac{2\pi}{3}\right) &= -2\pi n\end{aligned}$$

for phase A, B, and C respectively. These angles all align with the positive real axis, summing vectorially to thrice the harmonic phasor magnitude. If these phasors do indeed describe phase currents in the three-phase system, the triple-n harmonic case indicates that a net current flows *into* the load. This can only happen if zero-sequence current can circulate from the load back to the source (Figure 2-3).

Chapter 3

Induction Machines

INDUCTION machines have their beginnings in 1885 with an Italian Professor of Physics, Galileo Ferraris. Ferraris demonstrated a crude 2-phase induction machine, although he didn't fully understand the principles of its operation. In 1886, Nikola Tesla conceived and, then in 1888, built his own 2-phase version of the machine. Tesla understood the technical and commercial merits of this new technology, and Tesla was eventually granted the rights to the induction machine [13].

Small power (fractional to a few horsepower) 3-phase squirrel-cage induction machines are specifically considered in this thesis. These machines have a number of beneficial characteristics inherent in the machine construction. In addition, squirrel cage rotors are cheap and very reliable, making them popular in industrial applications [3].

3.1 Electromechanics of Induction Machines

This class of electric motor has a stator winding consisting of a number of poles per phase. In the case of the work described here, the machines are always three-phase, with several sets of pole pairs. For the sake of tractability, we will describe the fundamental electromechanics of the system through an example machine with two magnetic poles per phase. Figure 3-1 shows this machine schematically. Let us assume that the machine shown in Figure 3-1 is

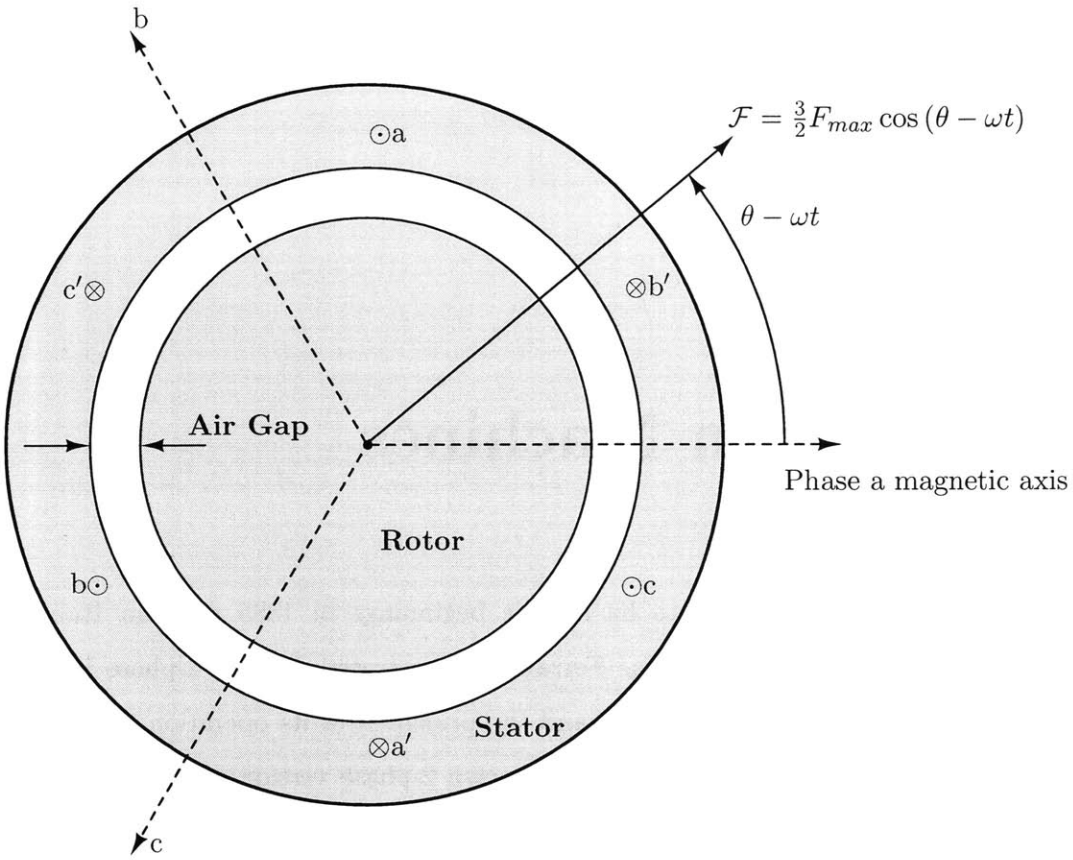


Figure 3-1: A simple two pole three-phase machine [3].

energized by three phase currents.

$$i_a = I\cos(\omega t) \tag{3.1}$$

$$i_b = I\cos\left(\omega t - \frac{2\pi}{3}\right) \tag{3.2}$$

$$i_c = I\cos\left(\omega t + \frac{2\pi}{3}\right) \tag{3.3}$$

The stator winding connections would be connected as described in Chapter 2, Figure 2-2. That is, the stator windings share a neutral connection, imagine a',b', and c' are shorted. Similarly, the source currents share a common node, and i_a, i_b, and i_c are connected to

a, b, and c in the machine diagram. Many wye-connected machines are constructed such that the stator winding neutral node is left floating inside the machine housing. Indeed, in many induction machines, this star point is not accessible. As such, the source and load *cannot* share star-points, and therefore, no zero-sequence current flows into the machine stator windings (exactly as described in Section 2.1 with floating loads).

Time-varying magnetic fields arise in the stator through the excitation currents (Equations 3.1-3.3). These drive currents give rise to a rotating *magnetic flux* wave. This flux crosses an air gap between the surface of the stator to rotor [3, 11, 24, 27].

In many alternating current machines, the rotor produces its own opposing flux through permanent magnets or through one or many sets of electrically-excited windings. An induction machine, however, is distinct in that its rotor is energized not directly, but through induction or transformer action [3, 13, 16, 19]. The rotor conductor surface links some of the magnetic flux from the stator windings [3, 27]. This linked flux gives rise to phase currents that energize the rotor conductors [9]. This process is very similar to a transformer. Indeed, one can model an induction machine as a transformer with the stator winding connected to the primary and the rotor to the transformer secondary [11].

In an induction machine, the armature, or stator, flux spatially leads the rotor flux. The rotor and stator *flux waves* rotate at the same steady-state rates, but there always exists an angular displacement between these waves. This difference in flux wave fronts gives rise to a torque on the machine rotor. The rotor does *not* rotate in synchronism with the armature flux wave; instead, the rotor is constantly slipping through the flux wave. This slip gives rise to induced rotor currents, and, in turn, the rotor currents give rise to the rotor flux wave [3, 11, 13, 16, 19].

This rotor slip through the stator flux wave is the reason an induction machine runs. We define slip, s, as the ratio of the difference in angular velocity between a fixed point on the rotor surface and the stator flux wavefront to the stator flux wave speed. Given an induction machine with P magnetic poles *per phase* being exited by currents at an electrical

frequency of ω, we define the *spatial* flux wave angular velocity

$$\Omega_s = \frac{2 \cdot 60 \cdot (2\pi\omega)}{P} \text{ RPM}.$$

If the stator flux wave front is rotating around the machine at Ω_s RPM and the rotor is rotating at Ω_r RPM, we define the slip

$$s = \frac{\Omega_s - \Omega_r}{\Omega_s}. \tag{3.4}$$

The idea of slip will be useful when we formulate a more tractable[1] model of these machines [3].

3.2 Machine Construction

3.2.1 Rotor Construction and Behavior

Squirrel-cage induction machines are characterized by a set of fixed conductive rotor bars shorted together at the rotor ends. It is in these rotor bars that currents are induced. We view the rotor bars as a set of three equivalent wye-connected impedances. The shorted rotor ends form the star-point of the equivalent wye-connected rotor-load [3].

As discussed in Chapter 2, a wye-connected load without a zero-sequence current path does not allow the injection of triple-n current harmonics. Therefore, given the rotor construction, we can reason that no triple-n current harmonics can be induced in the rotor. Furthermore, if the rotor conductors cannot have triple-n current harmonics, the induced rotor MMF and flux waves cannot contain triple-n harmonics.

3.2.2 Stator Construction and Behavior

As was discussed in Section 3.1, induction machines can be modeled using transformers [10,11]. The rotor, electrically isolated from the stator, is energized by the stator excitation

[1] Read: engineer-friendly

through induction or transformer action. We can see this in Figure 3-2. The rotor bar impedance, while complex in reality, is modeled by a lumped Z_r, and the stator winding inductances and resistances are not included.

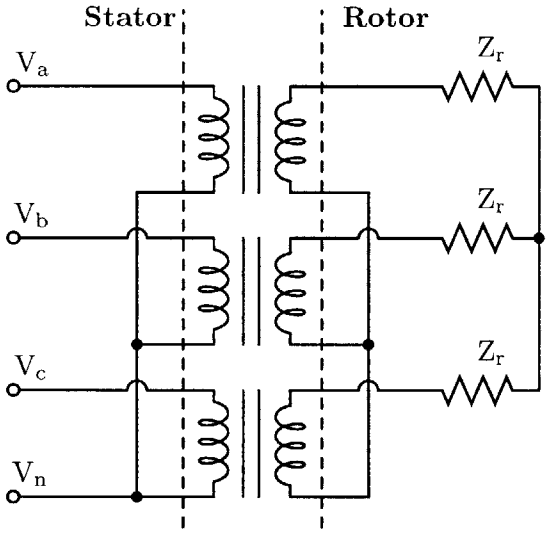
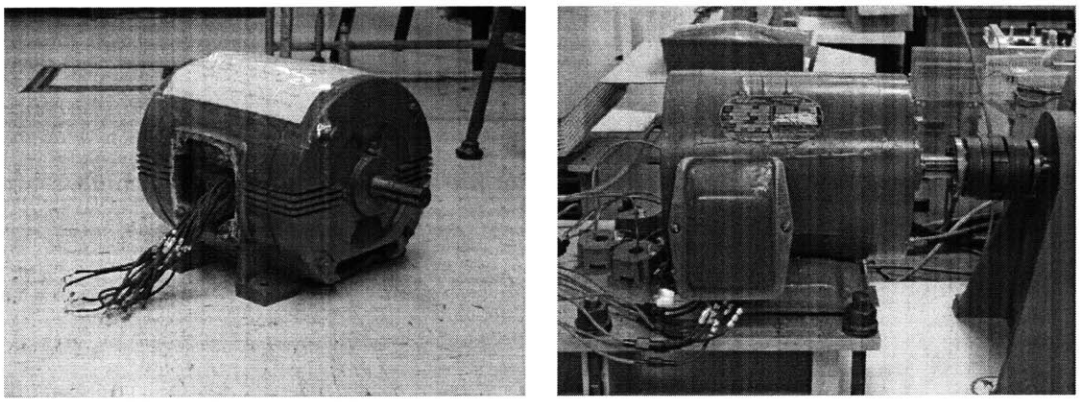


Figure 3-2: An ideal transformer model for a 3-phase induction machine.

We have made the stator star-point node, V_n explicit and seemingly available in Figure 3-2, however, this may not the case. Induction machines may be designed without physical access to this star-point. The star-point could be buried between stator windings inside the machine housing. This is done intentionally to eliminate zero-sequence current from the machine. Zero-sequence current (triple-n harmonics) requires larger stator winding conductors to handle the extra current, as well as another conductor to provide a path for the triple-n current to flow back to the load. In addition, by assuming triple-n harmonics, specifically third harmonic, are zero, classical induction machine models become simpler [11].

3.3 Multiply-wound Stator Induction Machines

Two, small-horsepower, three-phase machines (Figure 3-3) were wound for this system. Both machines were wound to run at 1800 RPM, nominally, from $120V_{RMS}$ and $230V_{RMS}$, line-to-neutral. These machines differ from commercially-available induction machines in two ways. The stators of the motors are wound "three-in-hand," which is to say that there are three electrically-isolated circuits wound tri-filarly in the same stator slots for each pole of each phase. To avoid magnetic saturation in the stator steel, the machine is wound to support less than the saturation flux density when all three stator windings are driven in parallel at the rated voltage. In addition, the star-points of all the stator circuits is made readily available outside of the machine housing. Note the 12 stator connection leads coming out of the machine housing (four per stator winding) in Figure 3-3. An equivalent transformer-model schematic is shown for this machine in Figure 3-4).



(a) (b)

Figure 3-3: (a) $120V_{RMS}$ and (b) $230V_{RMS}$ Multiple-stator Induction Machines.

This machine structure is interesting in two ways. First, access to the star-points of the stator windings allows triple-n harmonics to be driven into the machine on the appropriately-connected winding circuits. For example, if the source star-point, V_n, were connected to the stator winding star-points, V_{n1}, V_{n2}, and V_{n3}, then zero-sequence current could be pushed into the machine. Note that this zero-sequence current cannot be induced in the

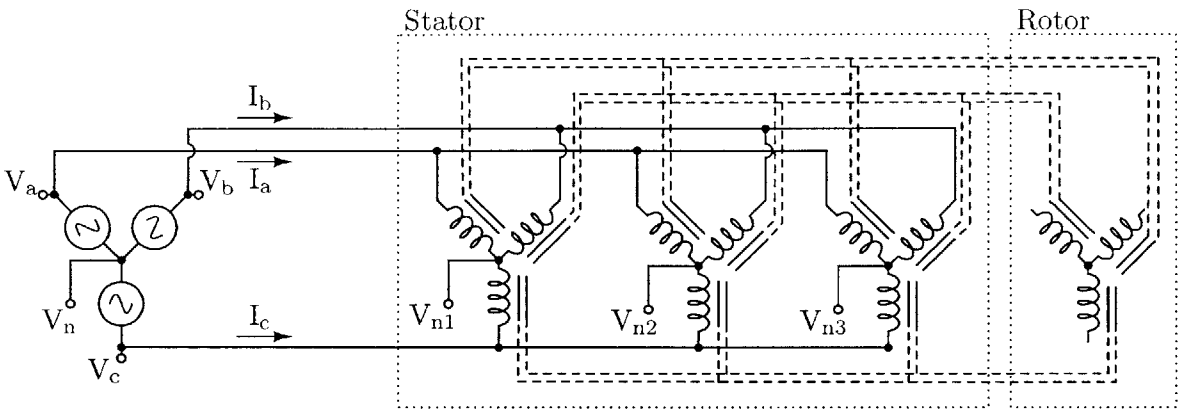


Figure 3-4: Multiple-stator Induction Machine Model.

rotor winding due to it's physical construction.

In addition, these parallel stator windings exhibit very high coupling. The transformer action between stator windings has been verified experimentally and is shown in Figure 3-5. The first set of stator windings, V_{a1}, V_{c1}, and V_{b1} are driven with a relatively spectrally pure $230V_{RMS}$ 60Hz sinusoid. The second and third stator windings, V_{a2}, V_{b2}, and V_{b3} and V_{a2}, V_{b2}, and V_{b3} are left floating. The top plot in Figure 3-5 shows the drive voltage, the middle and bottom plots show the same phase voltage on the second and third set of stator windings.

3.4 Equivalent Circuit Model

Steady-state operation of induction machines is often modeled on a per-phase basis in terms of equivalent circuits. One such equivalent circuit is shown in Figure 3-6. Here, I_a is the phase current, R_a is the equivalent stator winding resistance, and X_1 is the leakage[2] of the stator winding. The reactance X_2 is the equivalent rotor leakage. The reactance X_m represents the stator to rotor magnetizing inductance. The leakages and the magnetizing inductance are involved functions of the machine construction. Lastly, the rotor equivalent resistance $\frac{R_2}{s}$ is a function of the machine slip, s, as defined in Equation 3.4 [11].

[2]All the reactive components of this equivalent circuit are denoted by $X = \omega L$.

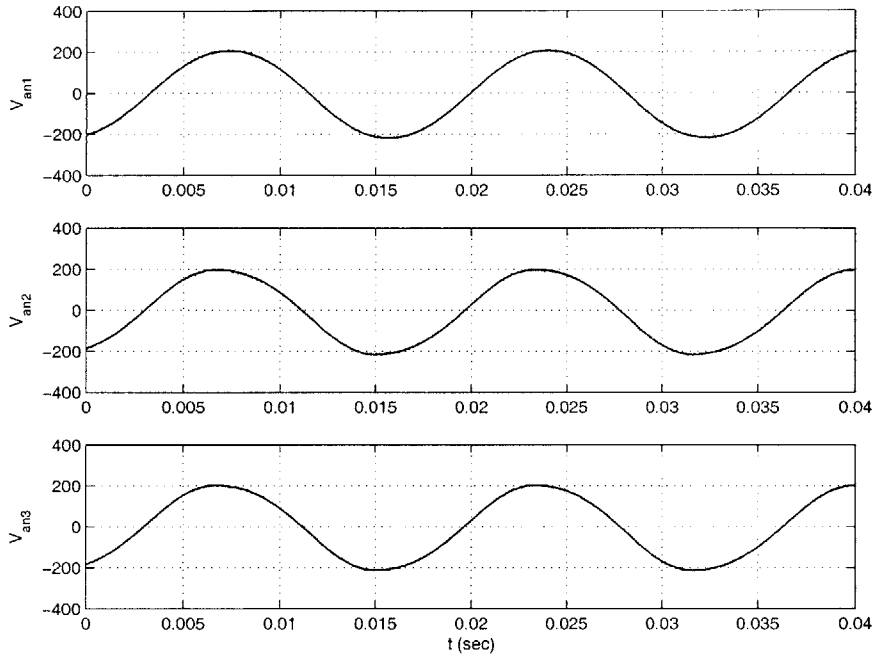


Figure 3-5: Stator Voltage Drive and Transformer Waveforms.

We can used this equivalent circuit to model a machine in steady-state, as well as empirically determine machine parameters. The total terminal phase to neutral impedance is

$$Z_{eq} = jX_1 + R_a + Z_g$$

for the impedance of the air gap and the rotor as seen from the stator

$$Z_g = jX_m \| \left(jX_2 + \frac{R_s}{s} \right).$$

Given this, we know that the terminal current, I_a is simply

$$I_a = \frac{V_{an}}{Z_{eq}}$$

and the rotor current

$$I_2 = \frac{I_a \cdot jX_m}{jX_2 + \frac{R_2}{s}}$$

from the current divider between the magnetizing reactance and the rotor impedance [11].

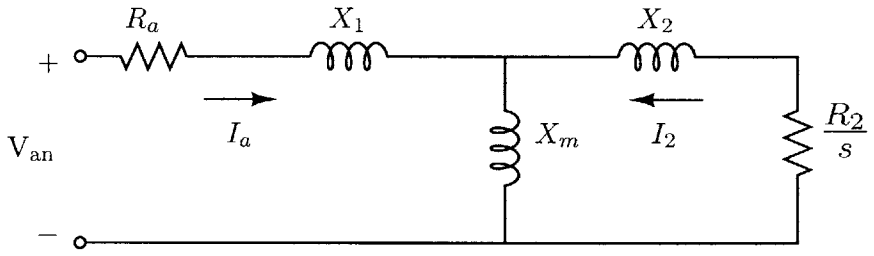


Figure 3-6: Induction Machine Phase-to-Neutral Equivalent Circuit.

3.4.1 Empirically Determining Machine Parameters

In order to roughly determine the machine equivalent circuit element values, we perform *no-load* and *blocked-rotor* tests [3, 11]. In the no-load test, the machine is allowed to start and spin freely without a load. In this situation, the rotor requires little torque to maintain speed. The torque production required in a machine is

$$T \propto \sin(\delta)$$

for the the angle between the rotor MMF wave and the induced rotor flux wave, δ. Given that the torque, T, is negligible, we conclude that $\delta \approx 0$. This condition requires that the rotor and stator flux wave are virtually in synchronism. From Equation 3.4, we see that

$$s = \frac{\Omega_s - \Omega_r}{\Omega_s},$$

and thus, for $\Omega_s \approx \Omega_r$, the slip $s \approx 0$ [3, 11].

Given that the machine slip is approximately zero, the equivalent rotor resistance,

$$\frac{R_s}{s} \approx \infty.$$

Looking into the machine from, phase to neutral, with the nearly-infinite rotor impedance, we see R_a, X_1, and X_m, the stator winding resistaxnce, leakage, and magnetizing inductance, respectively.

Using this approximation for the per-phase stator input impedance, we drive the machine from a voltage source and measure the input current that flows.

$$Z_{nl} = \frac{V_{nl}}{\sqrt{3}I_{nl}} \tag{3.5}$$

for the magnitude of the no-load per-phase impedance. V_{nl} and I_{nl} are the voltage and current magnitudes, again taken per phase. The power into the machine, P_{nl}, is related to the stator winding resistance

$$R_a \approx R_{nl} = \frac{P_{nl}}{3I_{nl}^2}. \tag{3.6}$$

Using this and

$$X_1 + X_m \approx X_{nl} = \sqrt{Z_{nl}^2 - R_{nl}^2}, \tag{3.7}$$

R_a and X_{nl} are uniquely defined.

In a blocked-rotor test, the machine rotor is held rigidly in place and the stator is allowed to freely energize from a voltage source. If the rotor is fully blocked, that is, it does not turn at all, we see from Equation 3.4 that the slip is unity. For the blocked-rotor power, voltage, and current magnitudes, P_{br}, V_{br}, and I_{br}, we have

$$Z_{br} = \frac{V_{br}}{\sqrt{3}I_{br}} \tag{3.8}$$

$$R_a + R_s \approx R_{br} = \frac{P_{br}}{3I_{br}^2} \tag{3.9}$$

$$X_{br} = \sqrt{Z_{br}^2 - R_{br}^2}. \tag{3.10}$$

Table 3.1: 230 V_{RMS} Induction Machine Circuit Model Values

Measured Values		Calculated Values	
V_{nl}	205.3 V	R_a	9.5875 Ω
I_{nl}	7.749/3 A	X_{nl}	44.887 Ω
Z_{nl}	45.889 Ω	R_{br}	5.1091 Ω
P_{nl}	191.9 W	X_{br}	5.5966 Ω
V_{br}	33.3 V	X_1	2.7998 Ω
I_{br}	7.609/3 A	X_2	2.7998 Ω
Z_{br}	7.5801 Ω	X_M	42.087 Ω
P_{br}	98.6 W	R_2	3.8778 Ω

Lastly, we need to relate the stator and rotor leakages to each other. It has been empirically determined, in a variety of low-power induction machines, that the rotor and stator leakages are approximately equal [3]. For the calculation of the individual leakages, we will assume $X_1 \approx X_2$.

For the high-voltage multiple-stator machine (Figure 3-3(b)), the parameters in Table 3.1 were calculated at a 60 Hz electrical drive frequency. In addition, it must be noted that this machine was driven by only one of the three sets of parallel stator winding circuits. The other two stator winding circuits were left floating. The star-points of all the stator windings were unconnected for these tests. Given the construction of the machine, we assume that the per-phase circuit model element values are virtually identical.

3.4.2 Determining Machine Parameters Using Transient Data

Using machine startup transient current and voltage transient data along with a numerical simulation, we can refine the rough parameter estimates found using the blocked-rotor and no-load tests.

Using a well-known *direct-quadrature* (dq) axis model[3] of the machine, we can formulate a simulation of the machine during startup and steady-state operation. Using the block-rotor and no-load equivalent circuit model element values in Table 3.1 as an initial guess, we can use multidimensional nonlinear least-squares curve fitting routines to find more accurate

[3]See [3, 11] for complete treatments of dq-axis models and Park's Transform.

Table 3.2: 230 V_{RMS} Induction Machine Circuit Model Values From Data Fitting

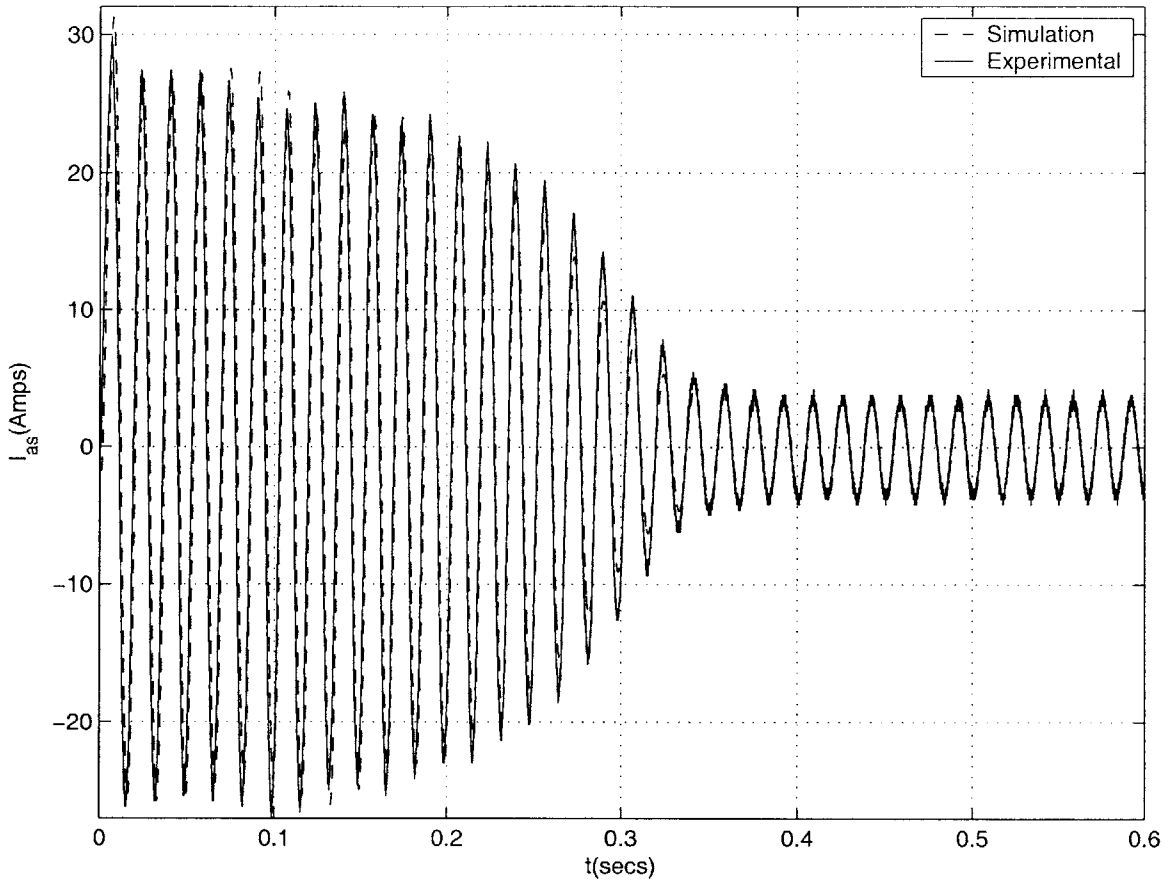
Simulation-Fitted Values	
R_a	2.0 Ω
X_1	2.8 Ω
X_2	2.8 Ω
X_M	42.09 Ω
R_2	1.5 Ω
J	0.0168 kg·M^2

estimates to these model parameters, See Appendix B for a full listing of the necessary Matlab code. Using the initial guesses from Table 3.1 and the code from Appendix B, the circuit parameters in Table 3.2 were found to be optimum.

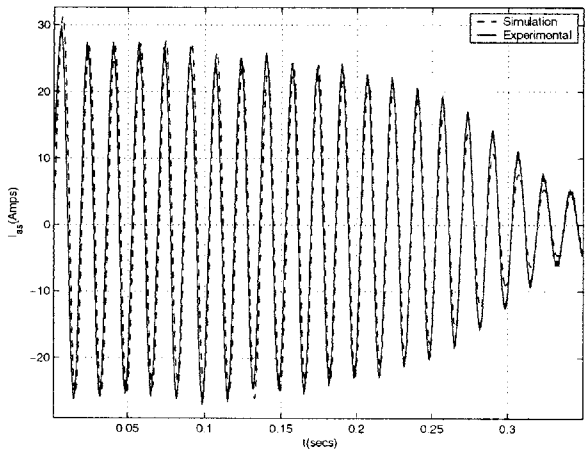
We note here, that the dynamic model used in these simulations also makes use of the rotor moment of inertia, J. Given our initial guess of $J = 0$ kg·M^2, the simulations converge on something decidedly more realistic than our first assumption[4]. It is interesting to note that the final values and those of the initial block-rotor and no-load estimates are quite close. The only notable changes are the stator winding and rotor resistances.

The most telling result, however, is in the simulated phase current waveforms versus the laboratory-observed phase currents. Figure 3-7 shows these results. We can see that the steady-state estimate is perfect, and the transient behavior is almost ideal.

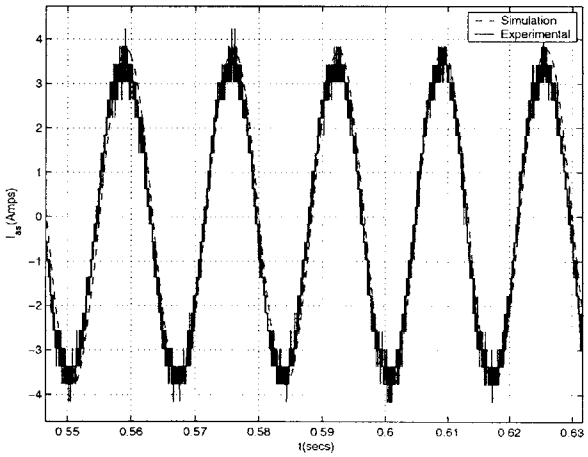
[4]which was simply a massless rotor



(a) Transient and Steady-State Simulated and Observed Phase Current Data.



(c) Transient Waveforms



(d) Steady-State Waveforms.

Figure 3-7: Simulated and Observed 230 V_{RMS} Machine Phase Currents.

Chapter 4

Three-phase Rectifiers

RECTIFIER circuits consist of one or two rectifier devices[1] per phase. Generally speaking, rectifier circuits come in two flavors—*half-wave* and *full-wave* rectifiers. Half-wave rectifier circuits present only one diode drop from AC source to DC output. These circuits connect the input AC to the output DC during only one half-cycle of the source. Full-wave rectifiers present two diode drops from source to output, and connect the AC input to the DC output during both positive and negative portions of the AC waveform [1, 9, 21].

In the work presented here, we deal with polyphase full-wave rectifier circuits, and in particular, three-phase circuits. Polyphase rectifier circuits consist of a number of rectifiers, e.g. diodes, whose cathodes or anodes are connected together. The devices sharing a common cathode are connected to a number of voltage sources by their anodes. The devices sharing a common anode and connected, by their cathodes to the these same voltage sources. The common anode and common cathode busses form the output DC bus [9]. Figure 4-1 shows an example n-phase full-wave rectifier.

[1] vacuum diodes, semiconductor diodes, SCRs, or even transistors

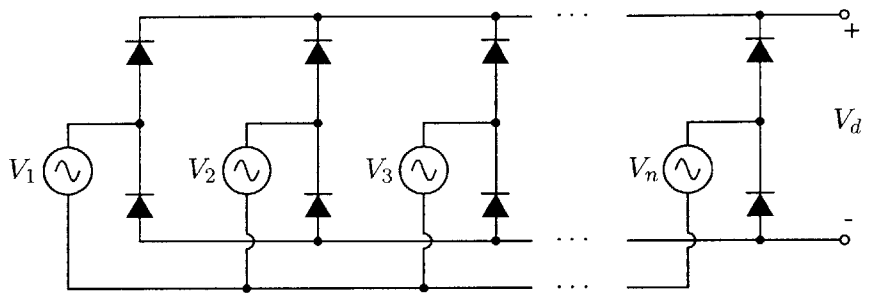


Figure 4-1: A general n-phase full-wave rectifier topology.

4.1 Modes of Operation

Rectifier circuits operate in three distinct modes, *continuous conduction mode* (CCM), *discontinuous conduction mode* (DCM), and on the border between CCM and DCM. During CCM, there is at least one rectifier device conducting at any given time. Discontinuous conduction mode is characterized by periods of time in which all of the rectifier circuit diodes are reverse biased, and thus not conducting. The rectifier circuit output filter, the load being driven, and the AC-side driving-point reactances determine the mode of operation [2].

Figure 4-2 shows a generic three-phase rectifier circuit with a generic second-order, resonant, LC filter, and a generic load impedance, Z_l. Note that AC-side reactances have been omitted from the schematic. These reactances are normally present in the form of mains wiring inductance or transformer secondary leakages.

The output EMI (electromagnetic interference) filter[2], or tank, has some Q, or quality factor, which we can define a number of ways. Assuming that the load, Z_L, is purely resistive[3],

$$Z_L = R$$

[2]EMI filters attenuate current harmonics that might otherwise interfere with other devices connected to the driving source

[3]This is a fairly reasonable assumption given the output operating frequencies—very near DC with much smaller amplitude harmonic content.

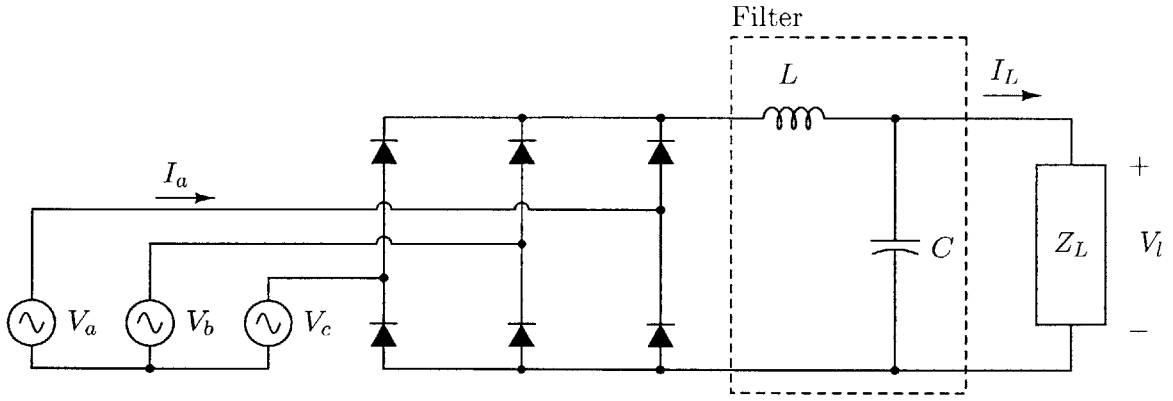


Figure 4-2: A generic three-phase rectifier with generalized output filter.

we can define the Q as

$$\begin{aligned} \mathrm{Q} &= R\sqrt{\frac{C}{L}} \\ &= \frac{R}{Z_0} \end{aligned}$$

where we define the filter characteristic impedance

$$Z_0 = \sqrt{\frac{L}{C}}$$

for this LC filter, and indeed for the parallel LC as well [2, 15].

A given network's Q is most generally described by

$$Q \equiv \omega \frac{\text{energy stored}}{\text{average energy dissipated}}.$$

Large values of Q, corresponding to L being small, or non-existent, correspond to operation in DCM. Smaller values of Q correspond to CCM [1, 2, 6, 15, 26].

4.1.1 Continuous Conduction Mode

In this mode of operation, the tank resonance is such that the input current must be reasonably close, in terms of frequency content, to the output voltage ripple. The behavior is readily seen in Figure 4-3(a). These simulated waveforms for an LC output filter with a Q of more than 5, which is much lower than needed to enter DCM, show that there is indeed constant conduction into the output filter. In addition, we can see the voltage ripple frequency from the current ripple. The current ripple is twice the AC-side voltage frequency, f_L, and thus the voltage waveform will have a $2f_L$ ripple frequency.

4.1.2 Discontinuous Conduction Mode

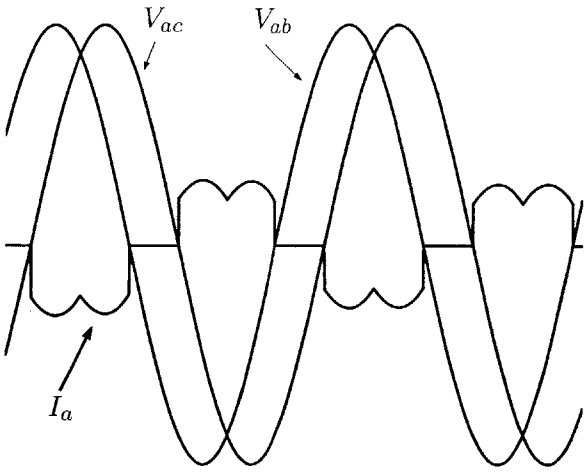
In this regime, the voltage across the tank will appear sinusoidal regardless of the current flowing into the tank. For extremely large values of Q, the input current can approach a train of delta functions [2, 14]. Any AC-side inductance decreases the magnitude and increases the duration of the phase current pulses during DCM [1, 14].

In DCM, the rectifier acts, in effect, as a peak detector. Given the output filter connection shown in Figure 4-2[4], we see that the phase currents are pulses at the peaks in line-to-line voltage. For example, given an output filter consisting of only a shunt capacitor, that is $L = 0$ and thus $Q \rightarrow \infty$, we see the waveforms in Figure 4-4.

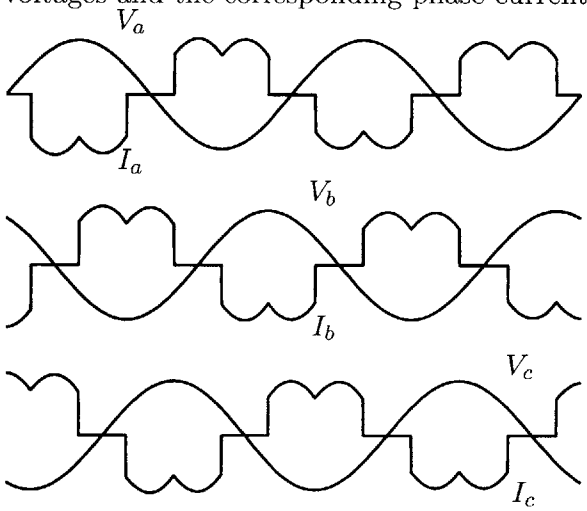
Intuitively, any given diode becomes forward biased and conducts when the line voltage exceeds the filter capacitor voltage. The capacitor voltage rises with the phase voltage now being conducted. The line voltage then reverse biases the diode, and the AC-line current stops conducting. Now the capacitor is discharged through the load resistance [9, 21]. For large values of the load resistance R, the capacitor remains charged, and thus phase current conduction doesn't happen until the line voltage is sufficient to forward bias the diode. This is another way of looking at the phase current dependence on Q.

It is of interest to note that AC-line side inductances can drastically change the operating characteristics of rectifier circuits. Given an inductance in series with each of the AC-side

[4]again, we are concerned with DCM, so the filter Q is large



(a) Line-to-line voltages and the corresponding phase current pulses in CCM.

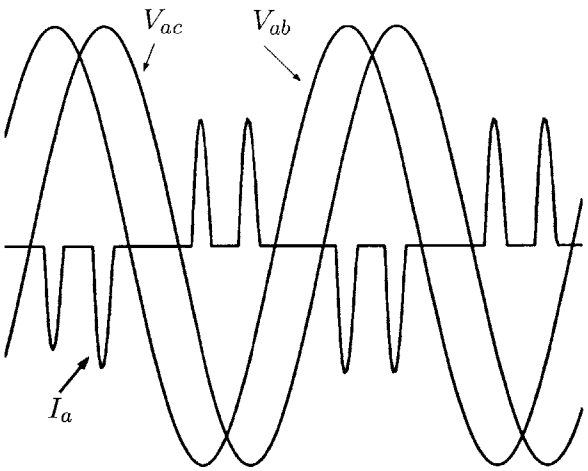


(b) Phase-to-neutral voltages and phase currents in CCM.

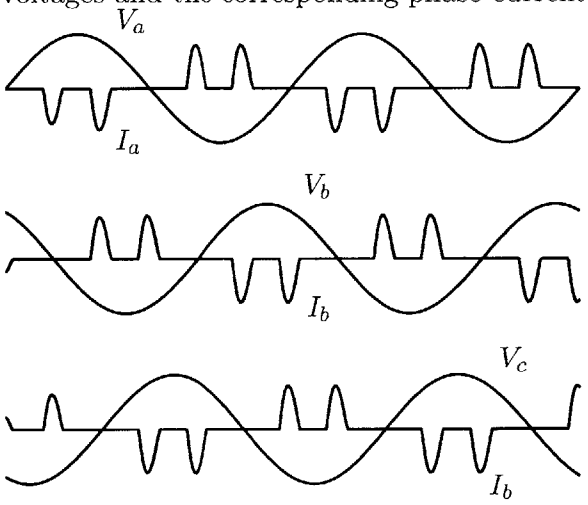
Figure 4-3: CCM rectifier circuit waveforms for small Q.

sources, the phase-current cannot change quickly[5]. This causes the line currents to be spread out and, given enough AC-side inductance, can push a rectifier into CCM from DCM. AC-line inductances can be either intentional, in the case of chokes, or non-ideal parasitics, such as transformer leakages or simple line inductance [1,2,9,21]. In this thesis, the AC-side line reactances are physically realized in the equivalent stator leakages of an induction machine. For a sufficiently large machine with large series leakages, a DCM rectifier may operate in

[5]This is the idea behind chokes.



(a) Line-to-line voltages and the corresponding phase current pulses in DCM.



(b) Phase-to-neutral voltages and phase currents in DCM.

Figure 4-4: DCM rectifier circuit waveforms for large Q.

CCM when connected with these machines.

Discontinuous conduction mode rectifier circuits are used primarily in light-load, constant voltage situations [2]. Given the intuitive description of a DCM rectifier operation given in this section, we can see this. If an excessive amount of current were drawn from the filter capacitor, the capacitor voltage would droop quickly, causing the next phase[6] to conduct immediately. This increased load affectively lowers the Q of the output filter and

[6] in terms of phase current conduction orders

pushes the rectifier into CCM operation.

The system described in this work makes exclusive use of three-phase rectifiers using a shunt filter capacitor and a relatively large Q filter, thus DCM operation is of particular interest.

4.2 DCM Rectification with Center-Tapped DC Busses

Consider the center-tapped rectifier filter in Figure 4-5. The load voltage, V_l is split across the shunt filter capacitance. This rectifier topology, while still a DCM operation circuit, operates differently than the DCM circuits.While the circuit in Figure 4-2 qualitatively acts

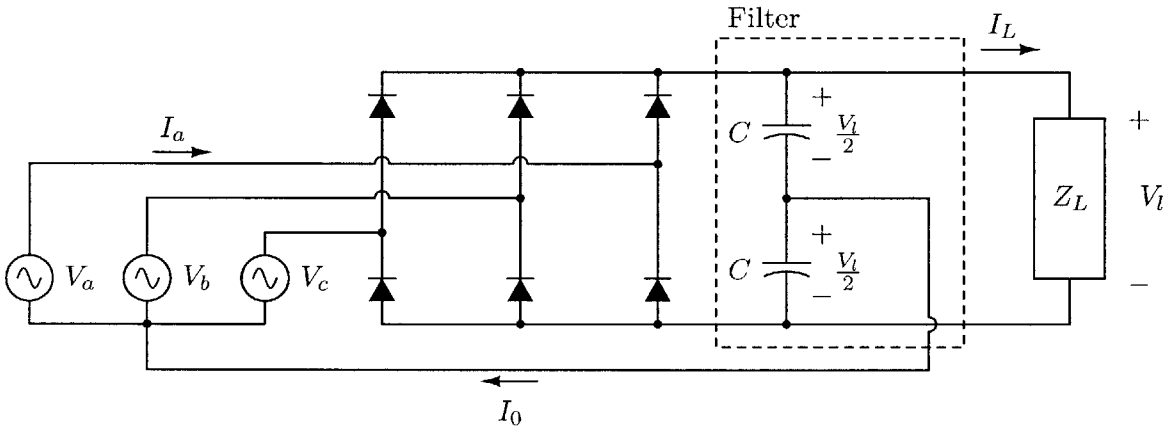


Figure 4-5: A center-tapped DC bus three-phase rectifier.

as a line-to-line voltage peak detector, the center-tapped rectifier, when operating in DCM, acts as a *line-to-neutral* voltage peak detector. This can be seen in the waveforms presented in Figure 4-6. The phase-to-neutral voltage, phase current, and corresponding line-to-line voltages are shown. In this plot, it is clear that the AC-side line currents happen during phase-to-neutral peaks.

Intuitively, this is can be explained in the same fashion as was used to explain the line-to-line peak detection observed in the DCM operation rectifier in Figure 4-2. In this case however, the top diodes will conduct if the device anodes are raised above $\frac{V_l}{2}$ with respect to

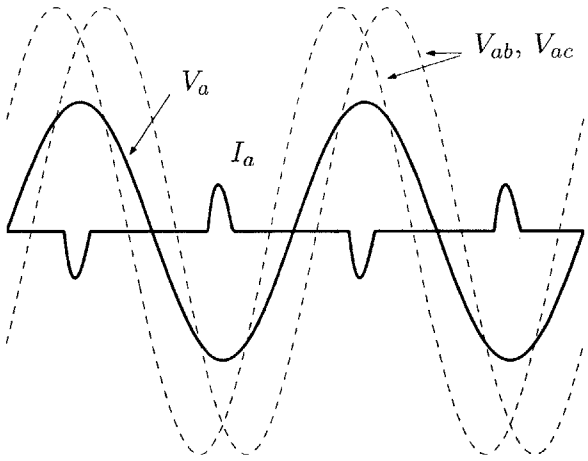


Figure 4-6: Center-tapped DC bus three-phase rectifier waveforms.

the center node[7] between output capacitors. By the same token, the bottom diodes conduct only when the voltage source connected to the device cathodes drops below $\frac{V_l}{2}$.

We have assumed this circuit is operating in DCM, therefore the output resistance must be large, such that the RC time constant of the filter is significantly longer than an AC-side voltage period. In addition, we are assuming a low-load condition, as was discussed in Subsection 4.1.2.

The circuit presented in Figure 4-5 is of particular interest because it provides a path for zero-sequence current (Chapter 2), I_0. Triple-n harmonic currents present in the driving source can circulate from the source to the load and back again.

4.3 Phase-Controlled Rectifier Load Voltage

Consider the circuit in Figure 4-5 driven with the line-to-neutral voltages

$$V_{an} = V_1 \sin(\omega t) + V_3 \sin(3\omega t + \phi_3) \tag{4.1}$$

$$V_{bn} = V_1 \sin\left(\omega t + \frac{2\pi}{3}\right) + V_3 \sin(3\omega t + \phi_3) \tag{4.2}$$

[7] we will ignore diode drops in analysis. At the high voltages we will be dealing with, one diode drop is insignificant.

$$V_{cn} = V_1 \sin\left(\omega t - \frac{2\pi}{3}\right) + V_3 \sin\left(3\omega t + \phi_3\right). \tag{4.3}$$

It is clear that increasing the *magnitude*, V_3, of the third harmonic[8], with respect to the fundamental magnitude, injected into the rectifier circuit in Figure 4-6 will change the DC output voltage, V_l. Less obviously, changing the *phase*, ϕ_3, of the third harmonic with respect to the fundamental will change the center-tapped rectifier output voltage.

SPICE simulations[9] show that the average output voltage, $\overline{V_l}$, varies with the third harmonic phase, ϕ_3 as shown in Figure 4-7. The system has been simulated with and without commutating inductance[10] over a range of $\phi_3 = 0$ to 2π. In the same plot, we can see the average output voltage for the same rectifier given no third harmonic voltage component.

In this simulation, we assumed

$$\begin{aligned} V_1 &= 120\ \mathrm{V_{RMS}} \\ V_3 &= 10\% \times V_1 \\ L_{AC} &= 7.427\ \mathrm{mH/phase\ or\ 0\ H/per\ phase} \\ C &= 750\ \mu\mathrm{F} \\ Z_l &= 300\Omega \end{aligned}$$

It is important to note that the waveforms in Figure 4-7 are *not* simply sinusoids. A sinusoidal fit matching perfectly at the extremes, near $\phi_3 = 0$ and π, deviates, at $\phi_3 = \frac{\pi}{2}$, from the simulated results by approximately 15% of the $\overline{V_l}$ swing.

The simulations in Figure 4-7 indicate that, with the appropriate control, the DC output voltage of the rectifier in Figure 4-6 can be adjusted in a controlled manner. A control scheme in which the three-phase fundamental voltage and a third harmonic are generated,

[8] We will only work with the third harmonic and ignore all other triple-n harmonics.
[9] See Appendix C for a complete SPICE netlist and associated code listing.
[10] AC-side line inductance

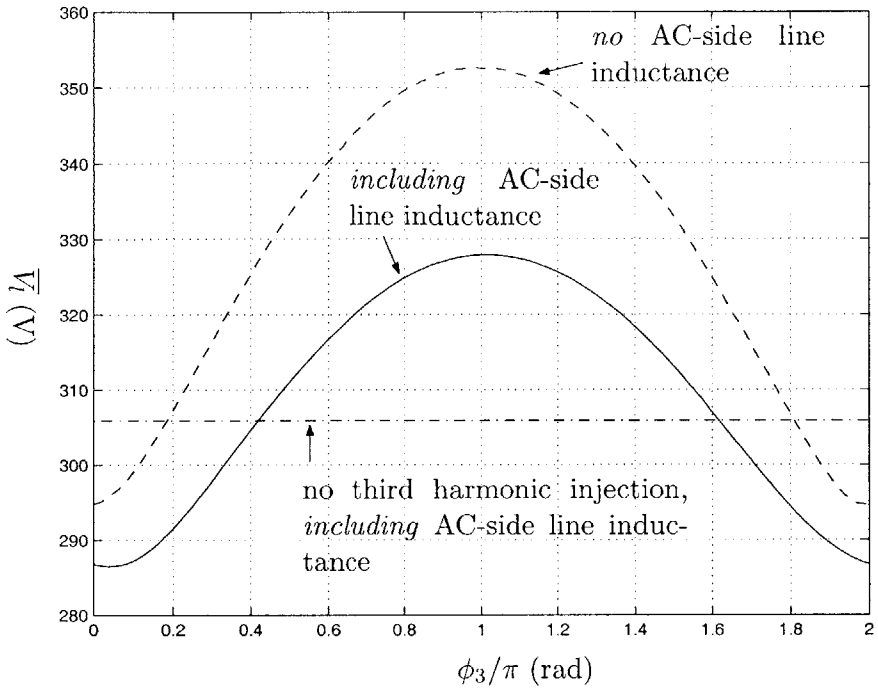


Figure 4-7: Center-tapped rectifier $\overline{V_l}$ variation versus third harmonic phase, ϕ_3.

and the third harmonic added with varying phase, with respect to the fundamental component of phase A, would provide a means for adjusting a rectifier circuit output for changing regulation needs.

4.3.1 Phasor Representation of Phase-Control

Given the drive waveforms in Equations 4.1 - 4.3 and the observation that polyphase rectifier circuits operating in DCM act as peak detectors[11], we argue that we need a method of determining the time-domain waveform amplitudes as functions of the third harmonic phase, ϕ_3.

To determine the time-domain waveform amplitude maximums, first find the time at

[11] See Subsection 4.1.2.

which the waveforms are maximized. Simply differentiating with respect to time, t,

$$\frac{\mathrm{d}}{\mathrm{d}t}V_{an} = 0.$$

Manipulation yields the expression

$$\frac{-V_1}{3V_3} = \frac{\cos(3\omega t + \phi_3)}{\cos(\omega t)}. \tag{4.4}$$

The manipulations for finding the time at which V_{bn} and V_{cn} are maximized are similar.

Equation 4.4 does not yield an analytic expression for the time, t, in terms of arbitrary ϕ_3, V_1, and V_3. We must, therefore, numerically determine t, then, using Equations 4.1 - 4.3, determine the relative voltage swing at the rectifier circuit input for varying ϕ_3. In so doing, we can arrive at a qualitative description of the behavior simulated and shown in Figure 4-7.

An alternative approach involves the use of the phasor description of polyphase systems outlined in Chapter 2. If we plot a number of line-to-neutral phasors as the time argument is swept, we see the third harmonic component modifies the phasor trajectories of *all* the phase phasors as shown in Figure 4-8. For $\phi_3 = 0$ and $V_3 = 10\% \times V_1 = 0.1V$, the full voltage waveform phasors have been plotted along with the third harmonic phasor rotations at integer multiples of $\frac{\pi}{4}$. Note that this phasor trajectory is the same for *all* phases, A, B, and C. The phasor trajectory plots differ only in their $t = 0$ beginning point. All of the trajectories lie in the exact same positions on the plane.

As ϕ_3 is increased from 0, the curve in Figure 4-8 rotates clockwise, with $\phi_3 = \pi$ setting this curve orthogonally to the plotted orientation. We can use this curve and its rotations to describe the behavior seen in Figure 4-7. In the simulated waveforms, the rectifier circuit is being driven with sine waves[12]. In this case, we are concerned with the the *sine wave* time-domain amplitude at a given third harmonic phase offset, ϕ_3. Using phasor plots, we simply find the magnitude of the curve projection on the *imaginary axis*.

[12] as opposed to cosine waves

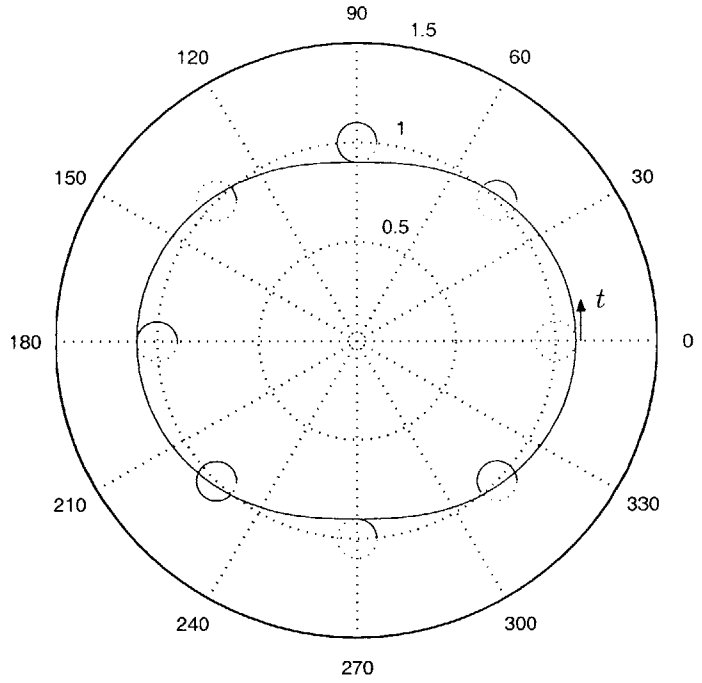


Figure 4-8: Line-to-neutral input voltage phasors.

Figure 4-9 shows the line-to-neutral voltage phasors for $\phi_3 = 0$, $\frac{\pi}{2}$, and π. These plots were generated assuming 120 V_{RMS} fundamental amplitude and $V_3 = 10\%$ of the fundamental amplitude.

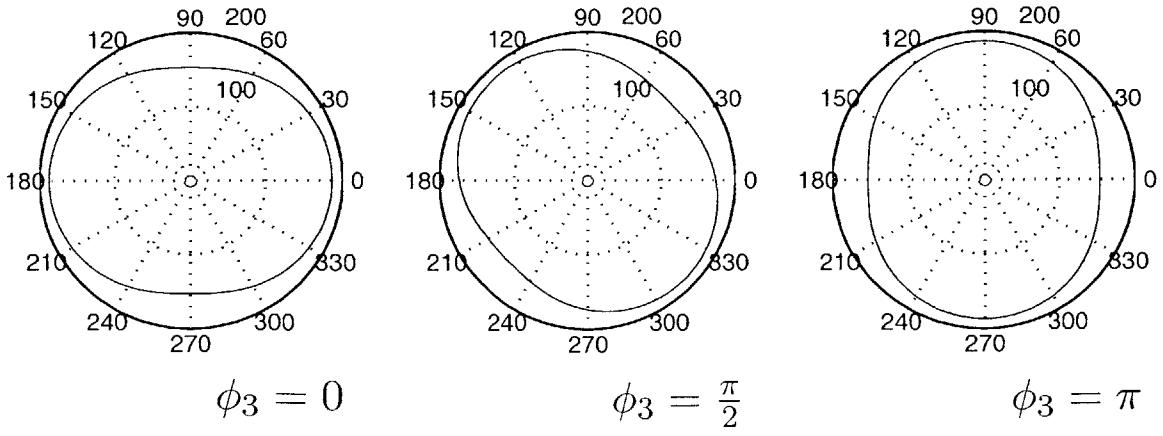


Figure 4-9: Line-to-neutral input voltages, parameterized by time at varying ϕ_3.

The phasor projections on to the imaginary axis qualitatively show the relationship between $\overline{V_l}$ and ϕ_3 demonstrated in the simulations of Figure 4-7. This same technique can be used with more general systems, see Appendix D.

This tool is useful enough to show us when the third harmonic-excited rectifier output $\overline{V_l}$ is the same as a rectifier driven with no third harmonic. For example, in the case of *no* AC-side inductance, imagine a circle of radius 120 V_{rms} plotted on the curves of Figure 4-9. We rotate the phasor trajectories by changing ϕ_3. When the phasor curve and the 120 V_{RMS} radius circle intersect at the *imaginary* axis, the ϕ_3 angle corresponds to $\overline{V_l}$ with third harmonic injected at ϕ_3 being equal to $\overline{V_l}$ without third harmonic injection.

Unfortunately, because the AC-DC conversion process that takes place through any given rectifier diode is inherently nonlinear, it is very difficult to derive an analytic relationship between the AC-side wave shape and amplitude and the average output voltage. From the phasor plots of Figure 4-9 it may be possible to develop an analytic framework for quantitatively determining the average output voltage.

4.4 Phase-Controlled Rectifier Input Impedance

Given the rectifier circuit in Figure 4-5 and the drive waveforms presented in Equations 4.1 - 4.3, we may be interested in the rectifier input impedance[13]. From a SPICE simulation we can determine the fundamental component of the rectifier input impedance as a function of the the third harmonic phase, ϕ_3. The SPICE simulation, from which the data in Figure 4-10 comes from, does not include the AC-side inductances. See Appendix E for a full code listing of the simulation tools.

Figure 4-10 shows that this rectifier, to a first-order approximation, has a significant amount of reactive power flowing out of the circuit and back into the source[14]. The input impedance angle, $\phi(Z_{in})$ is quite large, varying by as much as $\frac{\pi}{4}$ over the range of ϕ_3.

Also of interest is that the input impedance magnitude is minimized near $\phi_3 = \pi$. This

[13] Here, we mean to describe the input impedance in the *describing function* sense of the term [15, 22, 25].

[14] This point is especially important if the rectifier circuit is connected in parallel to an induction machine whose operation will be effected by any reactive power components pushed into the stator MMF.

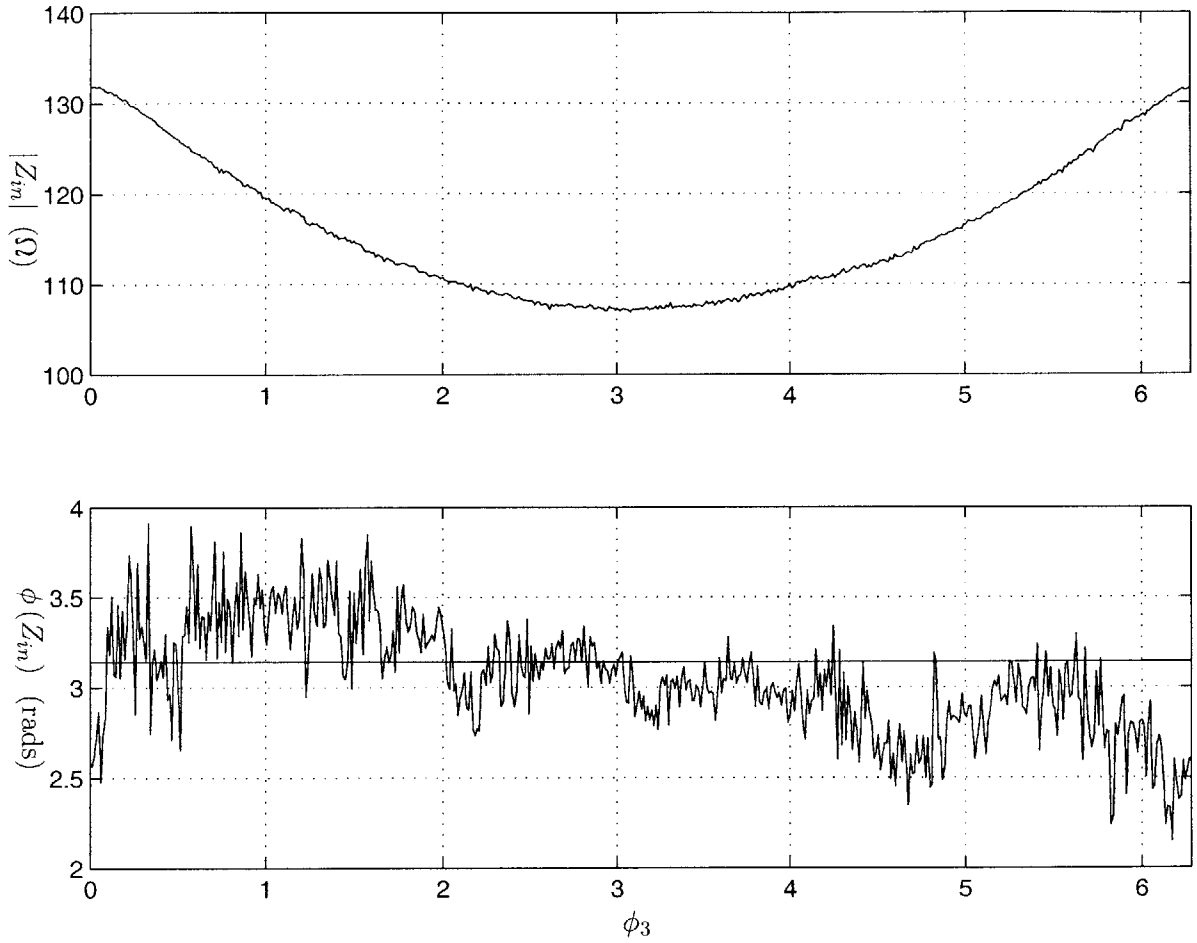


Figure 4-10: Line-to-neutral input impedance versus varying ϕ_3.

point corresponds to the maximum average output voltage, $\overline{V_l}$ from Figure 4-7. We find that the power *into* the rectifier, and thus the load, is maximized near $\phi_3 = \pi$ and minimized near $\phi_3 = 0$.

Chapter 5

Multi-stator Winding Induction Machines as DC Voltage Regulators

USING THE ideas developed in Chapters 2, 3, and 4, we can design a system in which both a specially-wound induction machine is used both as a electromechanical conversion device and as a transformer. By driving this machine appropriately, we can provide adjustable voltage regulation through a number of rectifier circuits.

5.1 System Description

By driving one set of stator windings in the multiple-stator induction machines described in Chapter 3, and connecting the other stator windings to center-tapped rectifier circuits, as described in Chapter 4, we can control the machine operating dynamics and regulate the rectifier average output voltage.

Given the physical construction of the machines, as described earlier, it is our contention that, using a triple-n harmonic excitation in addition to the polyphase fundamental drive,

the machine control and rectifier circuit voltage regulation can be completely decoupled.

5.1.1 Machine Stator Connections

Consider a three-stator three-phase induction machine. The first set of stator windings, labeled V_{a1}, V_{b1}, V_{c1}, and V_{n1} for the phase A, B, C, and neutral point, is driven with a wye-connected voltage source. The machine stator winding neutral and the drive source neutral are connected. In doing so, we allow the presence of triple-n harmonics, specifically, we allow the third harmonic to excite the system.

The machine construction specifically rejects the induction of third harmonics in the rotor bars[1], and as such, to first order, the third harmonic should not affect the rotor MMF and flux waves. In addition, the tri-filar nature of the stator windings gives rise to transformer action between stator windings. We can therefore, use the second and third set of stator windings as the secondaries of a set of three-phase transformers.

The second set of stator windings, labeled V_{a2}, V_{b2}, V_{c2}, and V_{n2}, are connected to a center-tapped three-phase rectifier circuit in which V_{n2} is shorted to the center-tap on the DC bus. This configuration allows any third harmonics to circulate from the source[2] into the rectifier load and back.

The third set of stator windings, labeled V_{a3}, V_{b3}, V_{c3}, and V_{n3}, are connected to a normal three-phase rectifier circuit. The rectifier circuit DC bus may be center-tapped, but the center-tap must be allowed to float. In doing so, the rectifier will reject third harmonic currents into the load.

This system configuration allows us to demonstrate control of the average output voltage of one rectifier, V_2, independently of V_3. This functionality will be demonstrated solely through the driving of the induction machine by its first stator winding.

We denote the line-to-neutral voltages of the first set of stator windings as V_{an1}, V_{bn1}, and V_{cn1}. The line-to-neutral voltages for the second set of stator windings is denoted by V_{an2}, V_{bn2}, and V_{cn2}. The third set of stator circuit line-to-neutral voltages are similarly

[1] as explained in Subsection 3.2.1

[2] where "source" is used to define the stator windings not being driven

denoted. We denote the first set of stator circuit phase currents by I_{a1}, I_{b1}, I_{b1}, and the zero sequence current I_{01}. Similar notation is used for the second and third parallel stator circuits.

Given the system connection described herein and schematically presented in Figure 5-1, we can control the induction machine electromechanical conversion process by any number of well-known drive schemes, such as constant V-Hz, field oriented, or flux-weakening control [3, 11, 12].

In addition to the voltage waveforms necessary for these control schemes, we excite the machine with addition third harmonic with a controllable phase shift, ϕ_3. Using this phase as our control variable, we are able to shift the average output voltage, V_2, around a nominal operating point as described in Chapter 4, and specifically in Figure 4-7. In contrast, because the third set of stator windings does not allow triple-n harmonics to circulate, the average output voltage V_3 remains relatively constant with respect to changes in ϕ_3.

5.2 Experimental Verification

An HP-6834B 3 Phase Power Source/Analyzer was connected to one set of stator windings of a 230 V_{RMS}, multiple-stator induction machine. The remaining stator windings were connected to three-phase rectifiers as shown in Figure 5-1. The rectifiers both used filter and load components

$$\begin{aligned} R_{load} &= 300\ \Omega \\ C_{filt} &= 750\ \mu\text{F} \end{aligned}$$

Initially, the first set of stator windings, V_{a1}, V_{b1}, V_{c1}, and V_{n1}, was driven without third harmonic excitation, that is only 120 V_{RMS} line-to-neutral fundamental. The line-to-neutral voltage for phase A of all three stator circuits was measured. Figure 5-2 shows the voltages of these line-to-neutral voltages with a purely fundamental drive. The top panel in

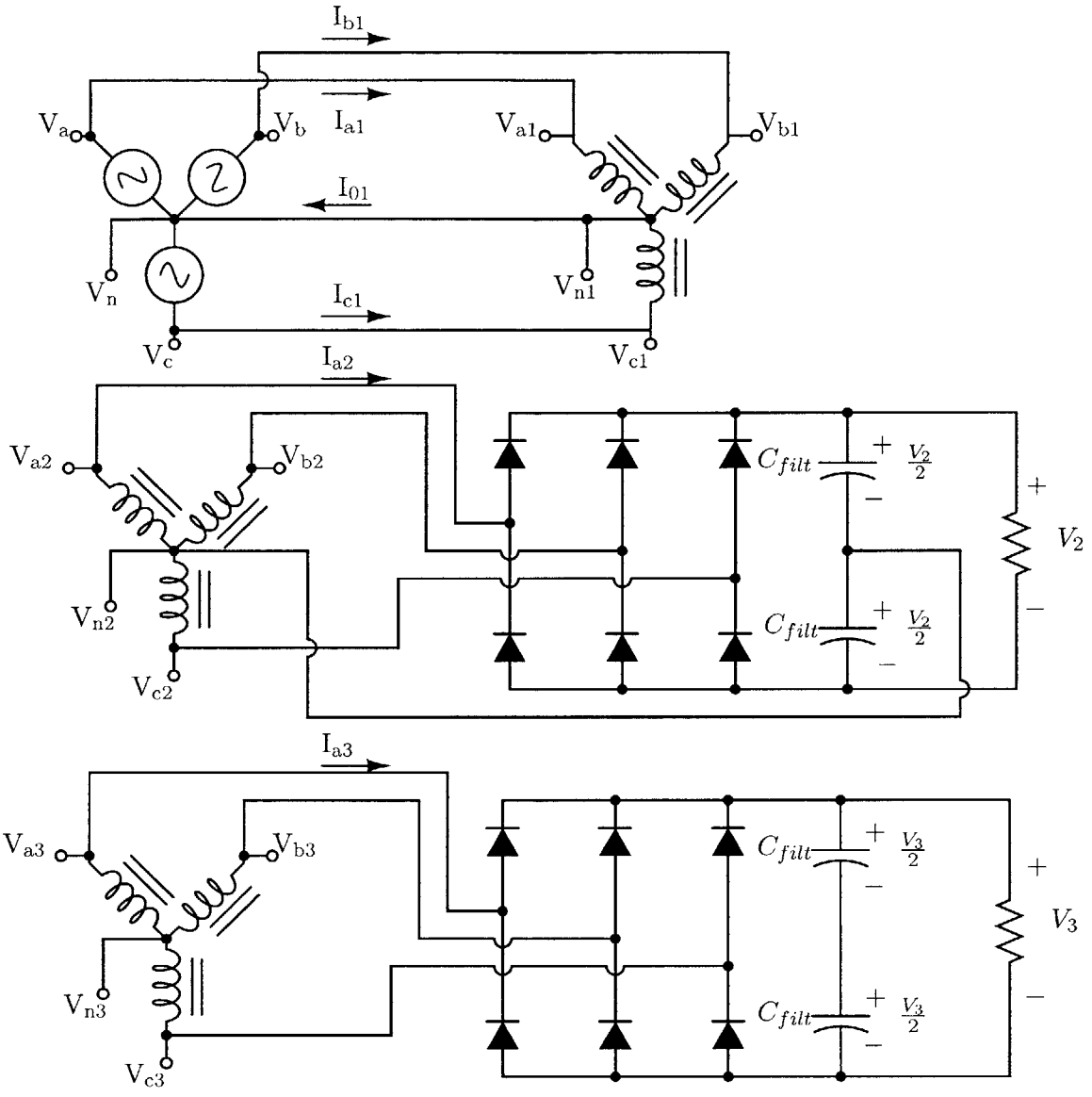


Figure 5-1: Multiple-stator Induction Machine Driving Three-Phase Rectifiers.

Figure 5-2 corresponds to the drive line-to-neutral waveform from the HP-6834B. The next two waveforms, labeled V_{an2} and V_{an3}, are the phase A line-to-neutral voltages presented to the inputs of the three-phase rectifiers shown in Figure 5-1.

We can see some distortion in the line-to-neutral voltages V_{an2} and V_{an3} in Figure 5-2.

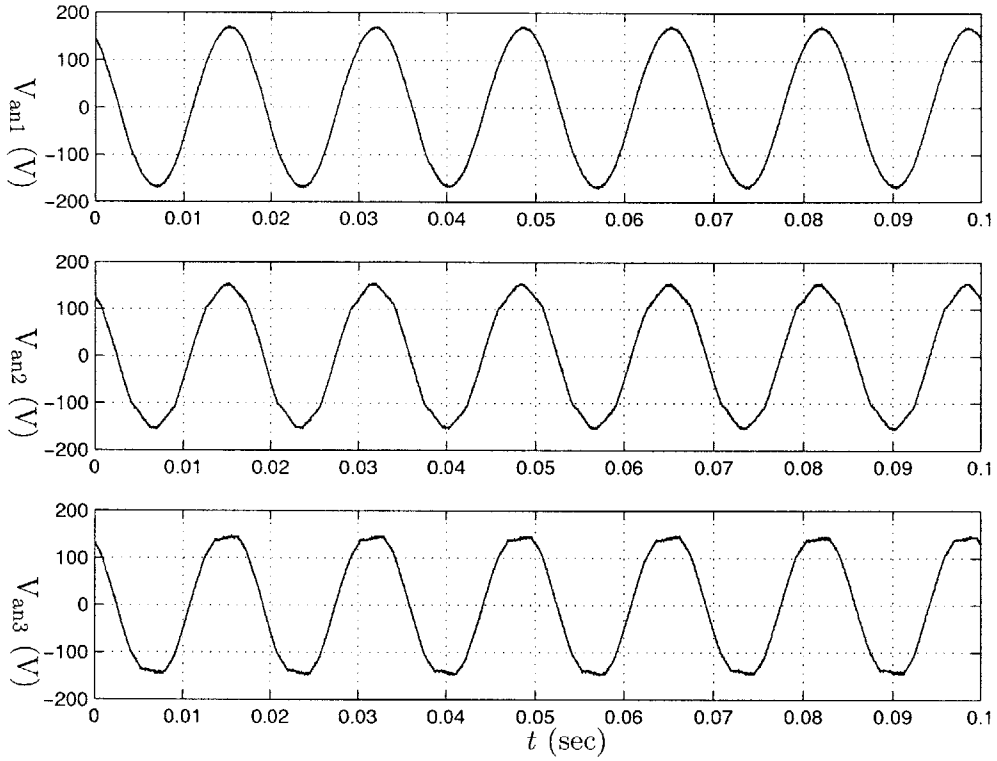


Figure 5-2: Rectifier-loaded Stator Line-to-neutral Voltages With Purely Fundamental Drive.

This distortion was not seen in same winding line-to-neutral voltages when the second and third parallel stator circuits are left floating, as seen in Figure 3-5. That is, connecting the rectifier circuits as shown in Figure 5-1 causes some distortion in the line-to-neutral waveforms despite the fact that these same harmonics are not present in the drive waveform, V_{an1}. This is somewhat expected as these rectifiers are operating in DCM and are pulling a number of higher-order current harmonics out of the machine stator circuits[3].

Figure 5-3 shows the experimentally-collected phase A current waveforms for this drive. The machine was excited with the same 120 V_{RMS}, purely fundamental, voltage drive as shown in Figure 5-2 We can clearly see the rectifiers are operating in DCM, as predicted

[3]see Subsection 4.1.2

in Chapter 4. In addition, it is immediately obvious from I_{a2}[4] that the second set of stator windings is connected to a center-tapped rectifier circuit with a path for zero-sequence current between the rectifier and stator windings. It is also interesting to note the total input phase current into the machine, I_{a1}, has very little harmonic content.

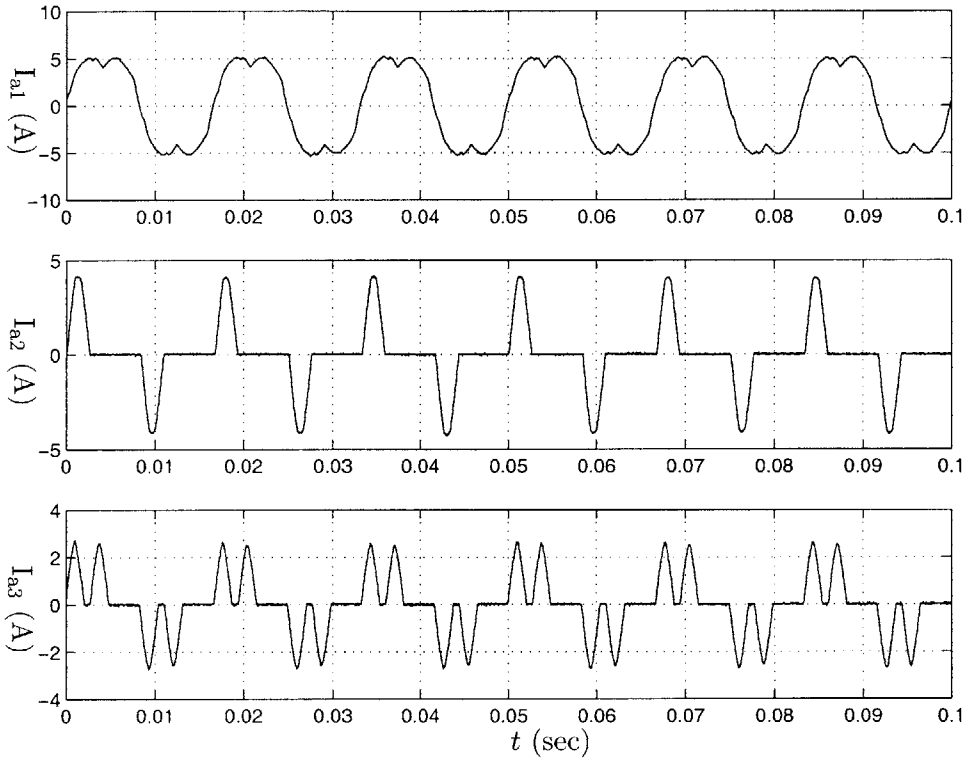


Figure 5-3: Rectifier-loaded Stator Line Currents With Purely Fundamental Drive.

Next, a drive voltage made up of fundamental and third harmonic with $\phi_3 = 0$ was applied to the machine stator using the HP-6834B. The third harmonic amplitude was 10% of that of the fundamental, itself 120 V_{RMS} line-to-neutral. Figures 5-4 and 5-5 show the line-to-neutral voltages for phase A of each of the stator circuits and phase currents of all the stator winding circuit A phases.

Again, we can see that the rectifier circuits remain in DCM. The I_{a2} waveforms have

[4]from the tell-tale impulse shape of the phase current

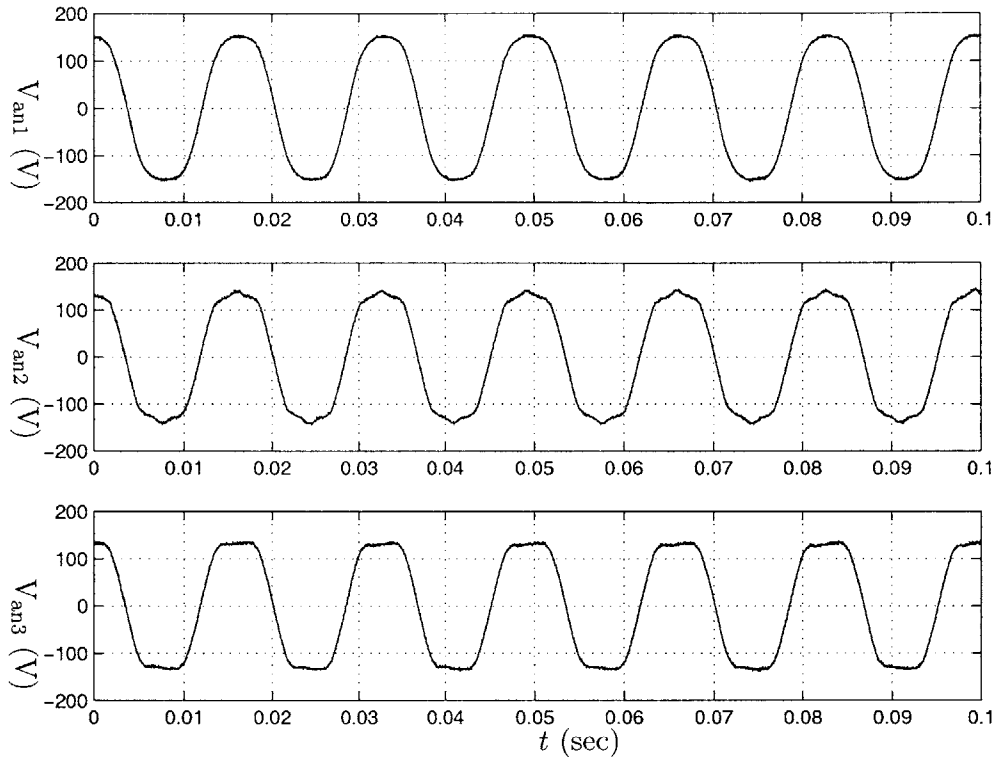


Figure 5-4: Rectifier-loaded Stator Line-to-neutral Voltages With 10% Third Harmonic Drive.

become shorter and wider. The theory and simulations of Chapter 4 predict that the output of this rectifier, V_2, should be lower than the case of a drive sans third harmonic excitation. This would correspond to a smaller impulse of current being delivered to the rectifier output filter during line-to-neutral voltage peaks. This seems to be the case in this experimental data.

Additionally, the shape of non-center-tapped rectifier line currents have stayed the same with the injection of additional harmonic content. Note, however, that the location of the I_{a2} peaks with respect to the I_{a3} conduction has shifted to later in a given cycle with the addition of triple-n harmonics. This has lead to a distortion of the drive-stator line currents, I_{a1}, in comparison to the purely-fundamental drive.

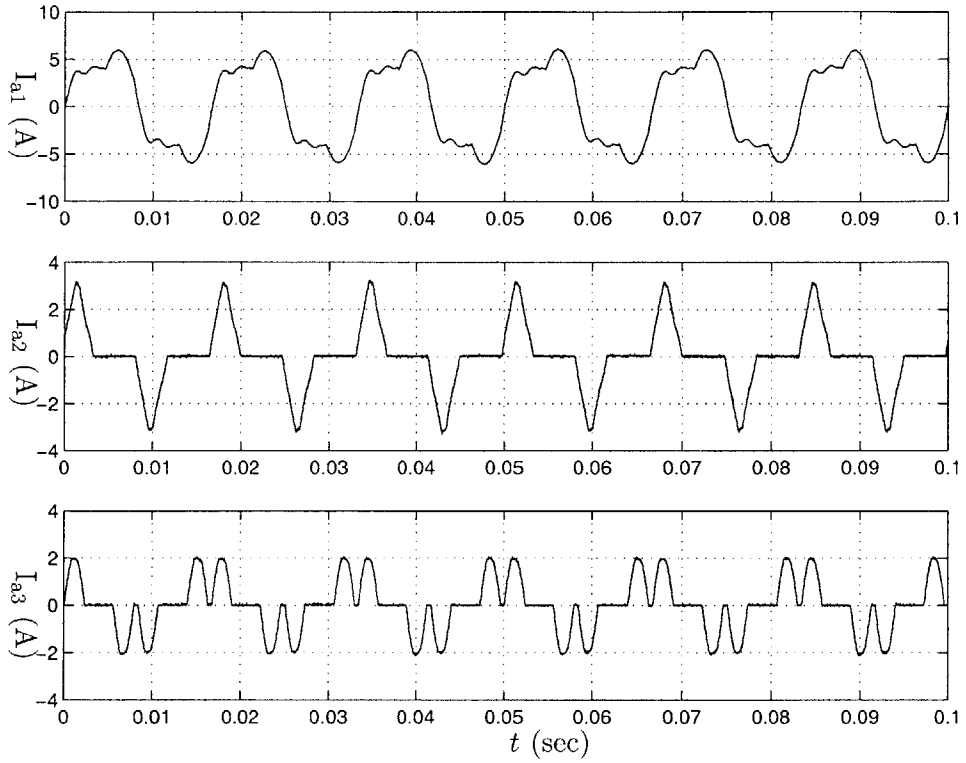


Figure 5-5: Rectifier-loaded Stator Line Currents With 10% Third Harmonic Drive.

Lastly, the first set of stator windings, V_{a1}, V_{b1}, V_{c1}, and V_{n1}, were driven with fundamental plus third harmonic while varying ϕ_3 values. The fundamental, V_1, in this experiment was 120 V_{RMS} and the third harmonic amplitude was 10% of the fundamental amplitude. The outputs of the rectifier circuits were measured, and the data are presented in Figure 5-6.

In this plot, we see that the output from the center-tapped rectifier[5], V_2, does indeed change as we would expect as ϕ_3 is adjusted. In addition, the three-phase rectifier *not* sharing a neutral connection with its driving stator winding, the third set of stator winding circuits, does not show significant variation on the rectifier output voltage, V_3, as the third harmonic is added to the drive waveform at time $t \approx 15$ sec, nor when the phase of the

[5]see Figure 5-1

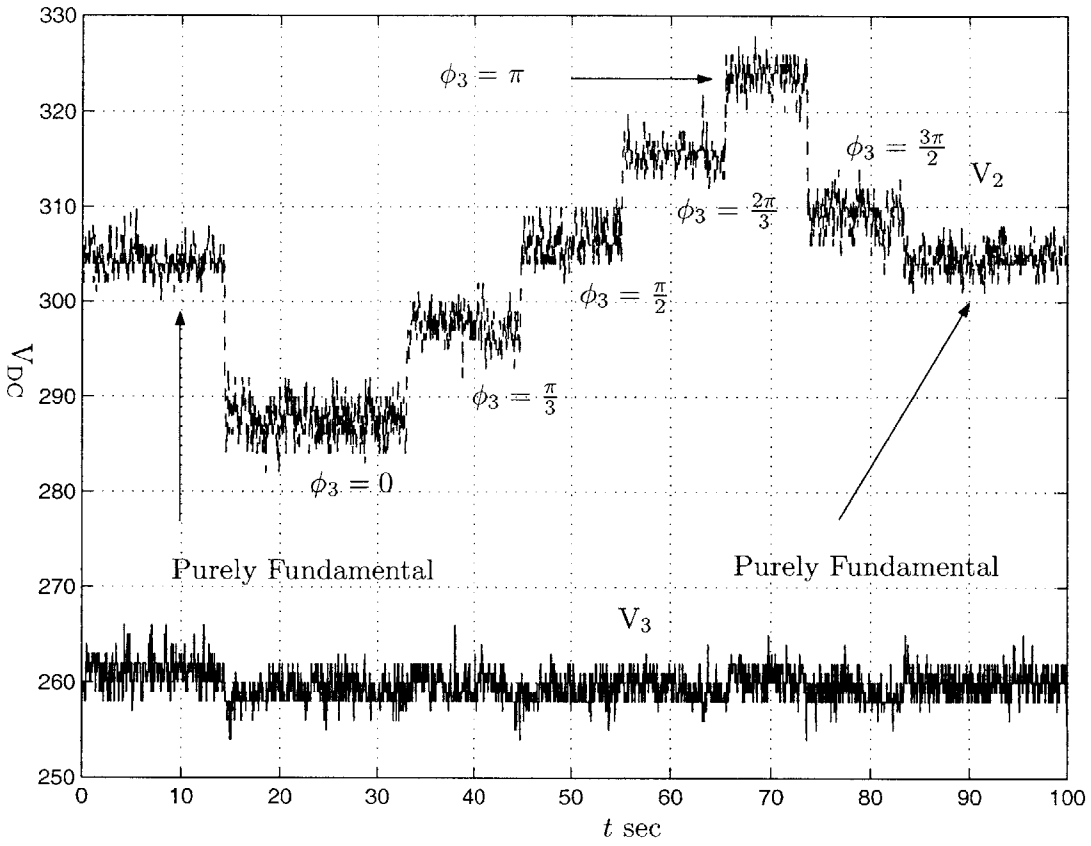


Figure 5-6: Varying Third Harmonic Excitation Using an HP-6834B.

third harmonic is changed. This also agrees with the theory developed earlier.

5.3 Drive Scheme

For simplicity, imagine we can implement a rudimentary V-Hz control scheme at the voltage source, denoted by V_{an}, V_{bn}, and V_{cn} in Figure 5-1. Let us assume you can generate spectrally pure voltage sinusoids of the appropriate size and frequency. We can express the drive voltages for all time,

$$V_{an}(t) = V_1(t)\sin(\omega t) + V_3(t)\,U(t-T)\sin(3\omega t + \phi_3) \tag{5.1}$$

$$V_{bn}(t) = V_1(t)\sin\left(\omega t + \frac{2\pi}{3}\right) + V_3(t)\,U(t-T)\sin(3\omega t + \phi_3) \tag{5.2}$$

$$V_{cn}(t) = V_1(t)\sin\left(\omega t - \frac{2\pi}{3}\right) + V_3(t)\,U(t-T)\sin(3\omega t + \phi_3), \tag{5.3}$$

for the time-varying

$$\omega = \Omega(t), \tag{5.4}$$

$$\phi_3 = \Phi(t), \tag{5.5}$$

and time-varying fundamental and harmonic amplitudes, $V_1(t)$ and $V_3(t)$, respectively. Note the Heaviside step, $U(t-T)$ [1,5,18,23]. This is explicitly included because we do not intend to inject third harmonic into the machine during the startup transient. Therefore, we choose a value of T larger than the startup transient time. For the experiments carried out in this work, we also generate constant-amplitude third harmonics, so

$$V_3(t) = V_3.$$

5.4 Drive Electronics

Generating AC waveforms of varying amplitudes and frequencies can be realized in myriad architectures. All of these systems can be classified as DC-AC voltage converters, or *inverters* [9].

Multi-level inverters create voltage waveforms of varying amplitude, and as such, can provide excellent spectral purity. For low frequency operation[6] at the required voltage levels, multi-level inverters are difficult to realize [9].

Bridge inverters come in two flavors, full and half bridge. These circuits act in reverse of the AC-DC rectifier bridges we examined in Chapter 4. Switches are turned on and off to conduct either positive of negative DC sources, producing a rough time-varying voltage.

[6]Small induction machines are rated for 50/60Hz, which is *very* low frequency

With appropriately chosen switching times or angles, these architectures are provide very high spectral purity at low frequencies. In addition, these circuits are quite easy to build in comparison to multi-level inverters. Full bridge inverters make use of twice as many devices as their half-bridge brethren, and the full bridge architecture allows the output voltage three levels: the DC positive supply voltage, the DC supply negative voltage, and zero volts [2, 9, 12].

The inverter configuration used in this system makes use of a half-bridge topology. Amplitude control is affected by the use of *pulse width modulation*, or PWM, of the output voltage [9, 12].

Chapter 6

Voltage Inverter

AS DISCUSSED in Chapter 5, the three-phase voltage drive used in this system makes use of a half-bridge voltage inverter. An International Rectifier PIIPM15P12D007X[1] programmable isolated IPM (Integrated Power Module). This module incorporates a three-phase bridge inverter, a three-phase diode rectifier, a TI TMS320LF2406A DSP, and a number of power supplies and isolation in one package [4, 20].

6.1 Integrated Power Module

Figure 6-1 shows the bridge inverter setup. External three-phase AC is supplied from mains at 60Hz, $120V_{RMS}$, line-to-neutral. This mains supply is connected directly to the PIIPM rectifier input pins. The DC output bus is made up of two large off-board capacitors connected in series at the rectifier output.

The center of these caps provides a neutral for the output AC waveforms. The capacitors split the DC voltage fairly evenly, and thus, V_n floats nearly halfway between the positive and negative DC rails. The symmetry of this output DC bus can be improved by placing the capacitors in parallel with two large, matched resistors. Protection diodes are provided to provide a discharge path through a bleeder resistor, and ensure that neither capacitor

[1] the trailing "X" denotes an experimental part. Documentation for this part can be found under the moniker PIIPM15P12D007.

charge negatively during discharge.

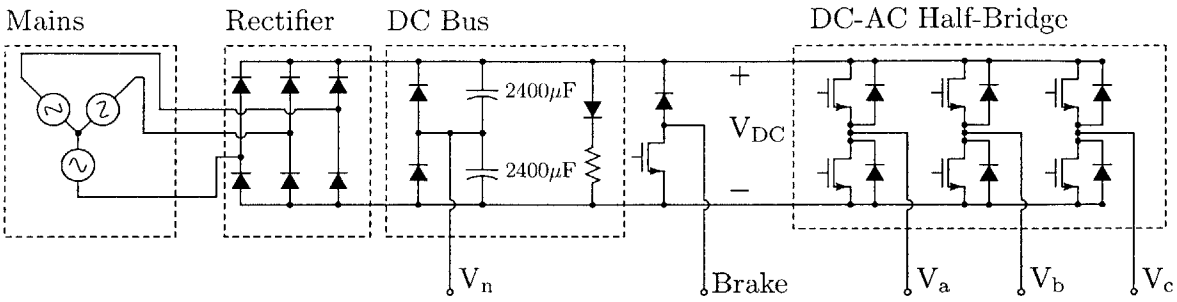


Figure 6-1: Half-bridge Voltage Inverter.

The bridge is made up of three half-bridge IGBT inverters. The IGBT gates are driven with on-board gate drives, and the switching waveforms are commanded by the incorporated TI DSP. The phase voltages, V_a, V_b, and V_c are switched between the positive and negative DC supply voltages to produce AC waveforms. By using PWM control at the IGBT gates, we can dither the output average amplitude and frequency.

The brake control is not used in the current experimental setup.

6.1.1 PIIPM Connections

The PIIMP module has a number of off-board connections that must be made in order to operate the module. The connections are shown in Figure 6-2. See the Power Module Frame Pins Mapping section in [20] for more information. The capacitor bank that forms the DC bus is connected across the the DC OUT+ and DC IN− pins. The DC OUT+ and DC IN+ pins are shorted together. The three-phase mains is connected to the IN1, IN2, and IN3 spade lugs on one of the short sides of the package. The AC output from the bridge is available on the spade lugs opposite the input lugs, OUT1, OUT2, and OUT3.

Two small daughter cards connect to the JTAG and RS-485 connector sockets[2], J2 and J1, respectively, in the middle of the module. The J1 and J2 socket pin 1 locations are marked in Figure 6-2 with a small white dot. Figure 6-3 shows one such daughter card,

[2]The connectors are Molex 53916-0204 connectors. They mate with Molex 54167-0230 or 52991-0208 connectors [20].

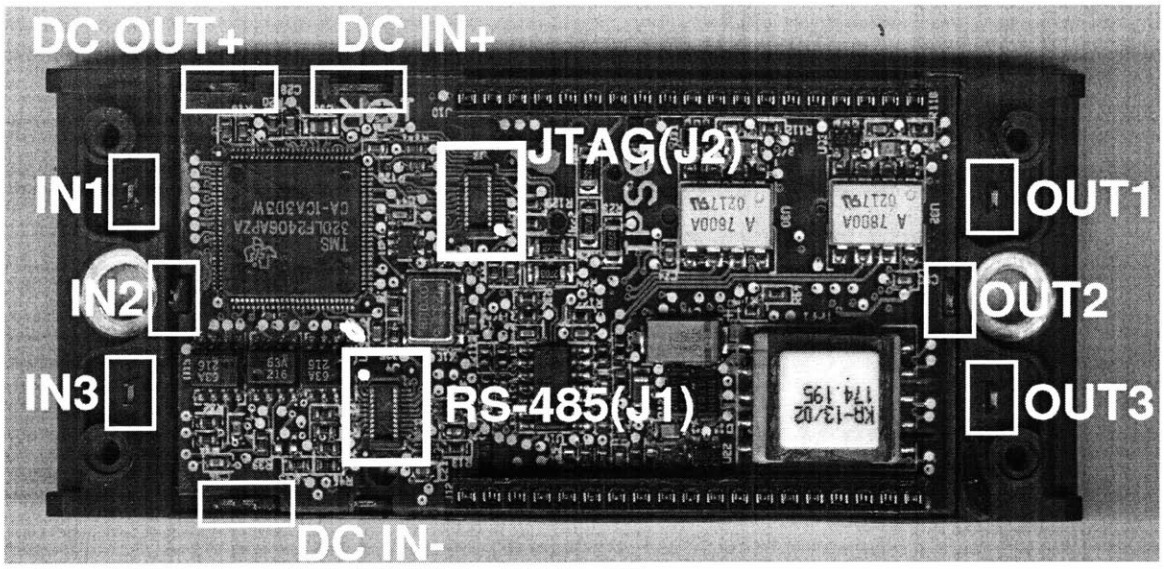


Figure 6-2: PIIPM15P12D007 Embedded Driving Board.

with pin 1 highlighted in the upper-left hand corner[3]. The J1 and J2 connectors do *not* have an intrinsic orientation, but pin 1 of the daughter card and pin 1 of the PCB socket must be properly aligned.

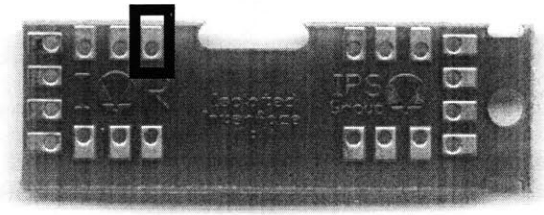


Figure 6-3: PIIPM JTAG and RS-485 Daughter Card.

The JTAG connector pins 19 and 11 (or 19 and 1) must be shorted together to enable firmware programming via the serial communication interface. In the experimental setup, a switch has been installed and labelled to enable either DSP firmware programming or DSP execution of the loaded firmware.

[3]The daughter card pin 1 is denoted here by a black box. The pin numbering continues in a counter-clockwise fashion.

Table 6.1: PIIPM J1 Connection Table.

PIIPM J1 Pin	**485OT9L Serial Converter**
1	RDB+ (receive +)
2	RDA− (receive -)
3	TDA− (transmit -)
4	TDB+ (transmit +)
6	+12 VDC **ISOLATED**
7, 10	GND **ISOLATED**
PIIPM J1 Pin	**Isolated DC Supply**
17, 18	fuse, diode, +15 VDC
19, 20	GND
PIIPM J1 Pin	**Misc.**
13	General purpose digital I/O pin.

The RS-485 connector, J1, handles the serial communications. A B&B Electronics 485OT9L RS-232 to RS-485 isolated converter [17] is used to converter the PIIPM differential serial signals to single-ended RS-232 levels. The converter positive and negative receive pins, RDB+ and RDA− respectively, are connected to the PIIPM RS-485 connector pins 1 and 2, respectively. Pin 3 is connected to the converter TDA−, and the PIIPM RS-485 connector pin 4 is shorted to the converter TDB+. The converter and the RS-485 connector share an isolated 12 V supply (GND connected to pins 7 and 10 on the PIIPM and +12 V connected to pin 6). Pins 17 and 18 on PIIPM RS-485 connector are connected through a diode and a 200 mA fuse to an isolated 15 V supply. Pins 19 and 20 are connected to the common of the same supply. Pin 13 of the PIIPM RS-485 connector is a general-purpose digital I/O pin, and was used for debugging purposes.

Converter Settings

The B&B Electronics converter used is configured for 38.4 kbps operation with termination enabled. The converter is wired to the PIIPM J1 connected as shown in Table 6.1, and the serial converter configuration switches are set as is shown in Table 6.2

Table 6.2: 485OT9L Serial Converter Configuration DIP Switches.

Switches/Jumpers	State
Switch 1	Off
Switch 2	Off
Switch 3	Off
Switch 4	**On**
Switch 5	Off
Switch 6	Off
Switch 7	Off
Switch 8	**On**
SD	**Open**
RTS	**Open**

6.2 Bridge Control

This IRF power module makes use of the TI TMS320LF2406A 16-bit fixed point DSP. The Texas Instruments CodeComposer Studio^{TM} was used for development purposes. The inverter software is written in a mixture of C and assembly. In addition, the software makes use of the Texas Instruments C initialization and runtime routines provided for the TMS320LF2406A. A complete code listing is included in Appendix F, Listing F.1.

6.2.1 DSP Configuration

Preparing the PIIPM DSP for use involves

1. **Disabling interrupts:** This can be done setting the `intm` bit.

2. **Setting the Watchdog Control register:** The `WDCR` register, at memory location `0x7029`, controls the watchdog timer. We set is to `0x68` in order to disable the watchdog.

3. **Configuring the processor clock PLL:** This allows the system to select the DSP system clock frequency. In the case of the voltage inverter described herein, the CPU clock frequency is set to 40 MHz.

4. **Setting up the PWM generation hardware:** Before running the PWM hardware, the timers, various polarity registers, and dead-band[4] units must be setup.

5. **Masking off the appropriate interrupts:** The TMS320LF2406A allows the programmer a small number of hardware interrupts with a secondary level of software flags to determine which peripherals triggered an interrupt.

6. **Enabling interrupts:** Clearing the `intm` bit re-enables interrupts.

6.2.2 PWM Generation

The ISR, `periodic_isr` in Listing F.1 is serviced every 600 machine clock cycles, or every

$$600 \times \frac{1}{40 \text{ MHz}} = 15\mu \text{ s}.$$

Each time this ISR is serviced, the previous location in a sine table is incremented for the fundamental and third harmonic. That is, the table is stepped-through three times as quickly for the third harmonic as for generation of the fundamental.

The control variable, ϕ_3[5], is applied to the table lookup argument during the ISR. Relative magnitudes of the fundamental and third harmonic are applied inside the ISR as well.

The fundamental phase voltages are phase-shifted with respect to each other to produce balanced three-phase waveform. Lastly, the third harmonic amplitude is added to each phase, and the results are stored in the three full compare registers, `CPMR1`, `CPMR2`, and `CPMR3`.

Between this and the next interrupt, a timer will count up to 300, or half of the interrupt service time, and then back down to zero, at which point, the ISR will be entered. The value of the counter is independently compared against the compare registers, generating the gating signals for the bridge IGBTs [8]. See Figure 6-4.

[4] to avoid switch shoot-through

[5] phase3 in the code of Listing F.1

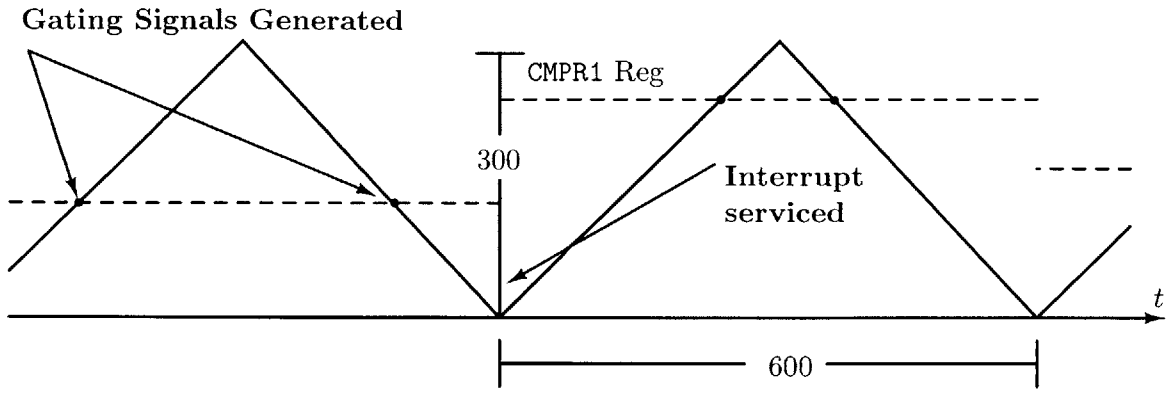


Figure 6-4: Carrier PWM Generation Scheme.

The V-Hz control is implemented outside the ISR. The step size through the table and the multiplicative amplitude are adjusted[6], and these values are used during the next ISR. In addition, ϕ_3 is adjusted outside the ISR as well, with new values of ϕ_3 being applied to the output waveforms during the next interrupt.

[6]ramped-up, in effect

Chapter 7

System Operation

IN THIS chapter, we demonstrate the operation of the voltage drive described in Chapter 6. After verifying the voltage inverter operation, we continue on to demonstrate a full system in which we vary a three-phase rectifier output voltage while maintaining drive on the induction machine.

7.1 Voltage Inverter Operation

Using the software in Appendix F, we demonstrate a constant V-Hz startup while driving an induction machine. Following the V-Hz ramp to steady state, third harmonic is injected into the machine. The phase of the third harmonic, ϕ_3, with respect to phase A voltage is varied.

7.1.1 Inverter Operating Procedure

The inverter, as pictured in Figure 7-1, is made up of the International Rectifier PIIPM, a capacitor bank, and a set of output chokes to attenuate high-frequency currents. All of these components, along with the serial converter and various indicators are located in a protective enclosure (Figure 7-1).

The procedure for programming the inverter is as enumerated below.

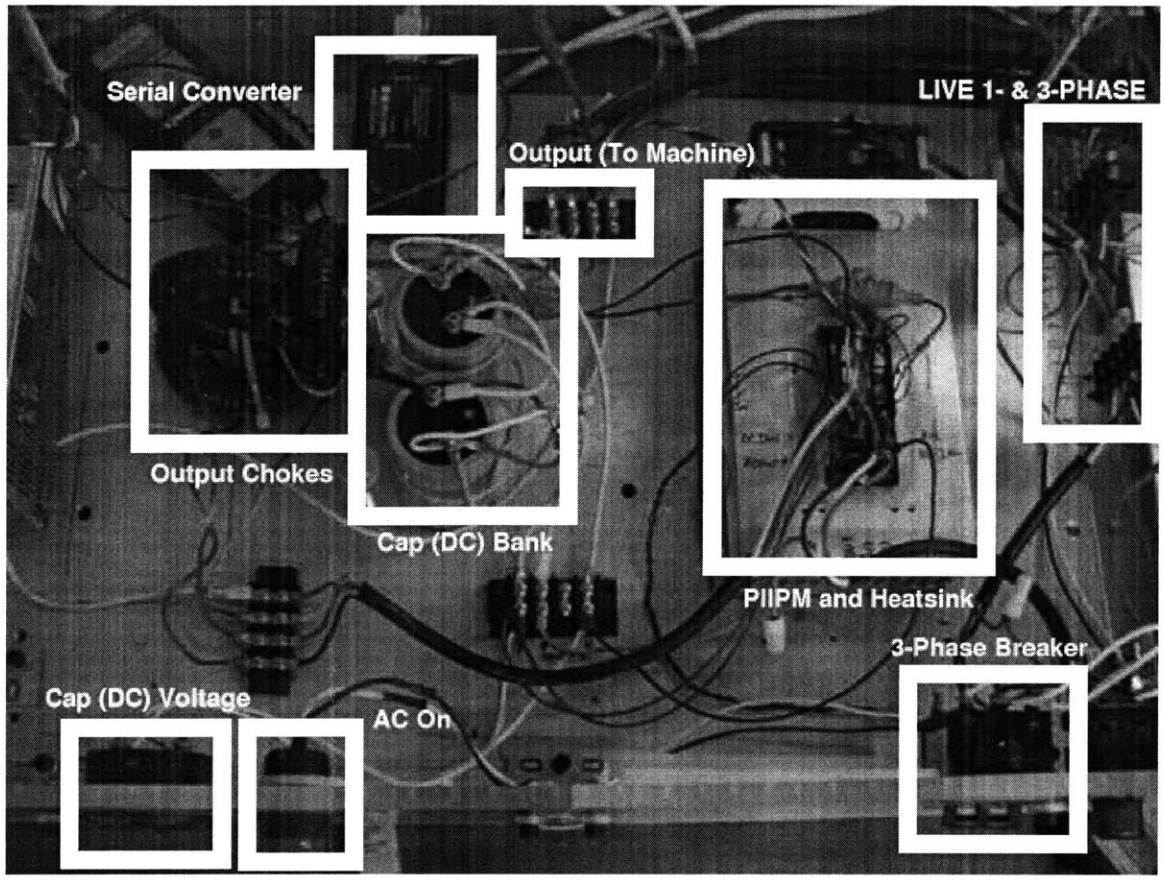


Figure 7-1: Inverter Enclosure.

1. Turn single-phase AC on. The red "AC on" light will glow.

2. Switch the DSP Mode switch, on the side of the enclosure, to "Program".

3. Turn the +15 VCD isolated lab supply on.

4. Begin programming the DSP by running[1] the Flash burning software on the connected PC.

5. Turn off the +15 VCD isolated lab supply when the software has completed the firmware transfer.

[1]First check the object file for unintentional Flash passwords using the software in Appendix F, Listing F.8.

Programming the Flash is complete. Now the inverter can run, driving the induction machine and rectifiers. Running the inverter is very similar to the programming procedure, except the DSP Mode Switch is set to select the "Run" mode. In addition, the three-phase breaker must be turned on before the isolated lab supply is turned on. When three-phase mains is connected to the inverter, the read light next to the three-phase breaker will illuminate. To stop the inverter, simply turn the isolated +15 VDC supply off.

7.1.2 V-Hz Ramp

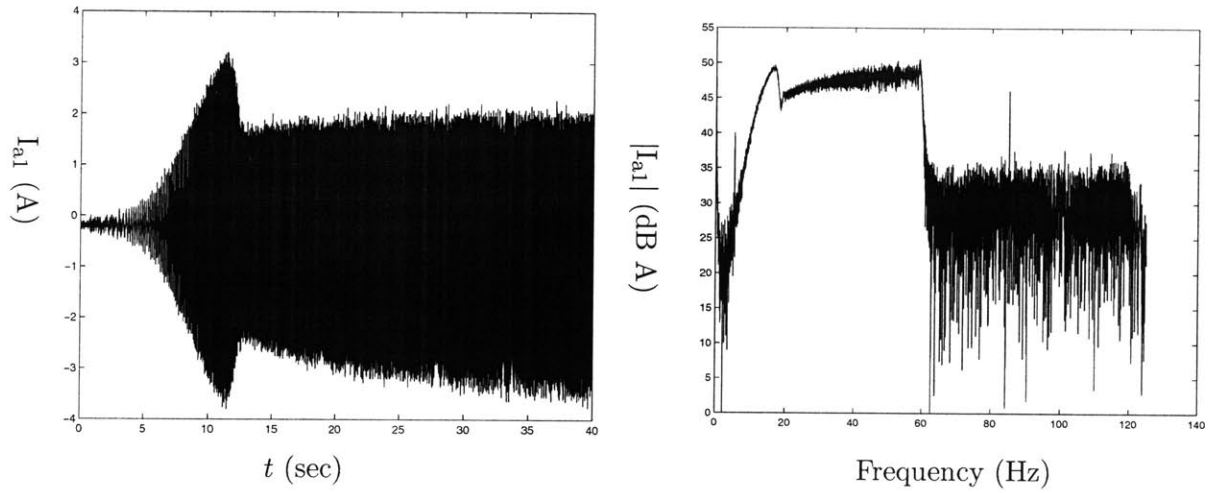
The 230 V_{RMS} induction machine was connected to the inverter by only one stator winding circuits. The remaining stator windings were allowed to float. That is, V_{a1}, V_{b1}, V_{c1}, and V_{n1} were connected to V_a, V_b, V_b, and V_n on the inverter, respectively. V_{a2}, V_{b2}, V_{c2}, V_{n2} on the second set of winding circuits and V_{a3}, V_{b3}, V_{c3}, and V_{n3} on the third set of windings were allowed to float.

The V-Hz ramping software in Appendix F was loaded, the DSP was started, and the connected induction machine was allowed to startup over several seconds. The phase current, I_{a1}, into the machine was recorded during the startup transient. Figure 7-2(a) shows the startup transient time-domain waveforms. Figure 7-2(b) shows the FFT of the collected data. We can see the amplitude ramping from the time-domain waveform. The frequency ramp can been seen in the band of frequencies from 0 to nearly 60 Hz in the FFT of the signal.

It is important to note that the initial large current ramp, from $t = 0$ to $t \approx 13$ s, and subsequent dip arise from the initial rotor friction. The dip occurs when the rotor actually starts to turn.

7.1.3 Third Harmonic Injection

The induction machine was again connected to the inverter by one of its three stator winding circuits. The other two windings were allowed to float. Following a V-Hz ramp, the inverter excites the machine with fundamental plus third harmonic.



(a) Time-domain startup

(b) Frequency-domain startup

Figure 7-2: Machine Phase A Input Current, I_{a1}, Startup Transient.

The steady-state line-to-neutral voltage waveforms for $\phi_3 \approx 0$ are shown in Figure 7-3. One can clearly see the $\frac{2\pi}{3}$ phase shift between waveforms. The PWM voltage waveforms contain large amount of higher-order harmonics due to the high switching frequency[2], but the reactance of the machine stator and the series chokes attenuates the high-frequency components in the current flowing into the machine.

Figure 7-4 shows the line current flowing from the voltage inverter into the machine on phase A with $\phi_3 \approx 0$. The data shows that the high-frequency components have been attenuated, producing relatively smooth waveforms. Figure 7-5 shows the spectral content of both the line-to-phase voltage, V_{an1}, and the phase A line current, I_{a1}.

We can see that the voltage waveform contains roughly an order of magnitude less, -20dB, third harmonic than fundamental. This is precisely the amount of third harmonic the software was written to produce.

The phase of the third harmonic, ϕ_3, was then moved from $\phi_3 \approx 0$ to $\phi_3 \approx \frac{\pi}{2}$. In the discussion of Section 4-5, we determined that $\phi_3 = 0$ corresponded to a *minimum time-domain* line-to-neutral voltage waveform amplitude, V_{an1}, for example. In that same

[2]approximately 65 kHz

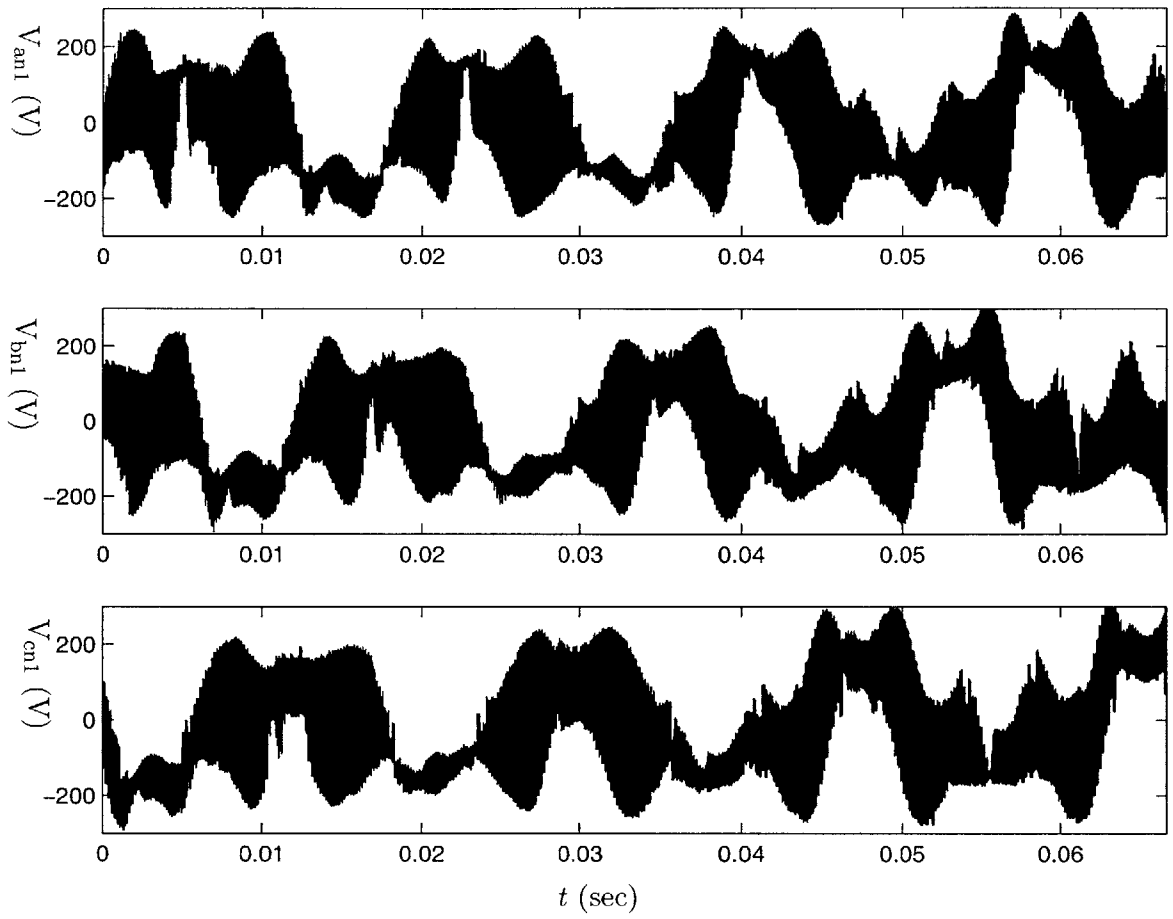


Figure 7-3: Phase-to-neutral Voltage Waveforms with 10% Third Harmonic at $\phi_3 \approx 0$.

discussion, we found that third harmonic injected at $\phi_3 = \frac{\pi}{2}$ was near the maximum time-domain line-to-neutral voltage waveform amplitude.

The steady-state line-to-neutral voltage waveforms for $\phi_3 \approx \frac{\pi}{2}$ are shown in Figure 7-6. Again, one can see the $\frac{2\pi}{3}$ phase shift between waveforms. It is difficult to make a comparison between the amplitudes of the voltage waveforms in Figures 7-3 and 7-6.

We can model the machine stator, in steady-state, with an equivalent circuit model. This circuit model is certainly linear, and as such, we can compare the time-domain line current waveforms at $\phi_3 \approx 0$ and $\phi_3 \approx \frac{\pi}{2}$. Figure 7-7 shows the line current into the machine on phase A given the drive voltage waveforms in Figure 7-6. Comparing the

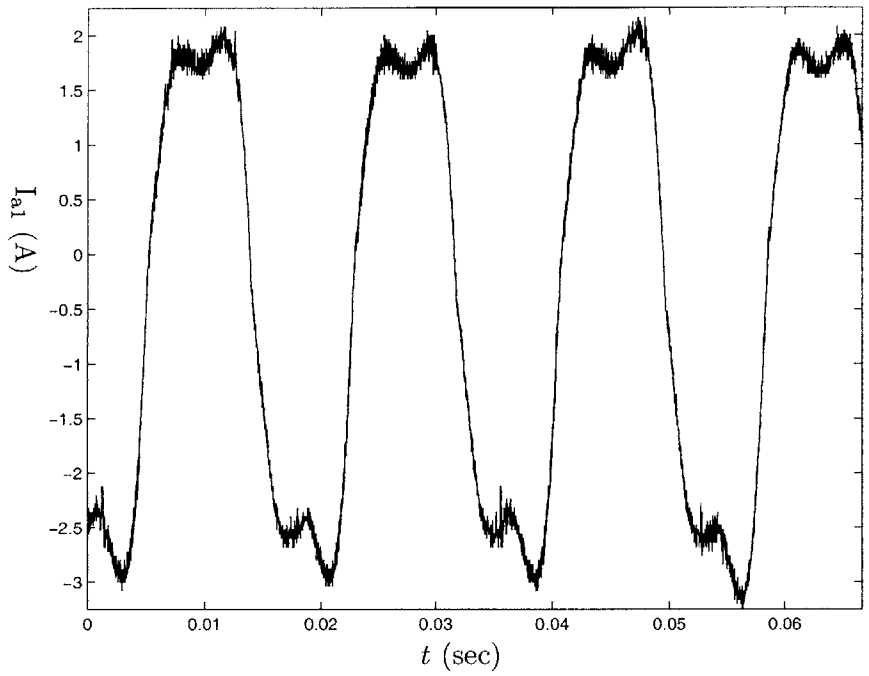


Figure 7-4: Phase A Line Current Waveform with 10% Third Harmonic at $\phi_3 \approx 0$.

relative magnitudes of the time-domain waveforms in Figures 7-4 and 7-7, we do find that the later case, $\phi_3 \approx \frac{\pi}{2}$, has a amplitude of approximately 3.5 A versus approximately 2.5 A with $\phi_3 \approx 0$. Again, as this system is roughly linear in steady state, the voltage waveform amplitudes would also demonstrate this difference.

The spectral contents of the line-to-neutral drive voltage, V_{an1}, and the corresponding line current, I_{a1}, with $\phi_3 \approx \frac{\pi}{2}$ are shown in Figure 7-8. Just as in the $\phi_3 \approx 0$ case illustrated in Figure 7-5, the voltage waveform contains approximately 10% third harmonic. The relative amplitudes of the voltage and current harmonics of Figures 7-5 and 7-8 are consistent. There is some distortion in the fundamental of the second case, $\phi_3 \approx \frac{\pi}{2}$. This can be seen as a "smearing" of the fundamental peak in Figure 7-8.

These experimental data show that the voltage inverter described in Chapter 6 can produce voltage and current waveforms of the appropriate spectral content to control the coupled induction machine-rectifiers system.

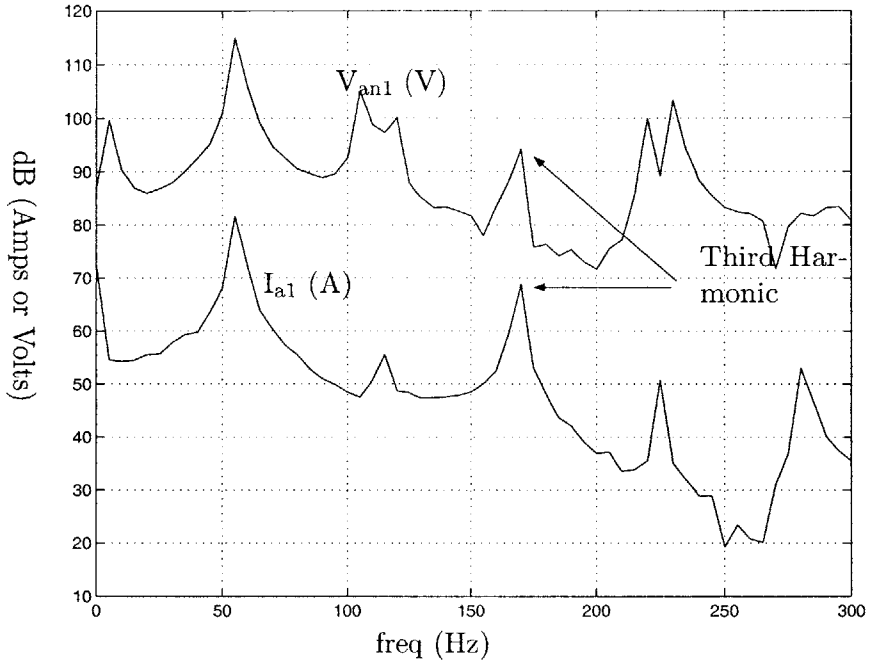


Figure 7-5: Input Waveform FFTs with 10% Third Harmonic at $\phi_3 \approx 0$.

7.2 Voltage Modulation Through Harmonic Excitation

The 230 V_{RMS} machine was connected to the voltage inverter and two three-phase rectifiers as shown in Figure 7-9. The inverter terminals V_a, V_b, V_c, and V_n are connected to the first set of machine stator winding circuits, V_{a1}, V_{b1}, V_{c1}, and V_{n1}, respectively.

A center-tapped three-phase rectifier is connected to the second set of machine stator winding circuits, V_{a2}, V_{b2}, V_{c2}. The rectifier DC bus center tap is shared with the second stator winding neutral, V_{n2}. We denote the rectifier load voltage with V_2.

A second center-tapped three-phase rectifier is connected to the third set of machine stator winding circuits, V_{a3}, V_{b3}, V_{c3}. The rectifier DC bus center tap is left floating, disconnected from any of the winding circuit neutrals. We denote this rectifier load voltage by V_3.

Given the theory, simulation, and experimental results from Chapters 4 and 5, we expect that by varying the phase of any third harmonic driven into the machine by the voltage

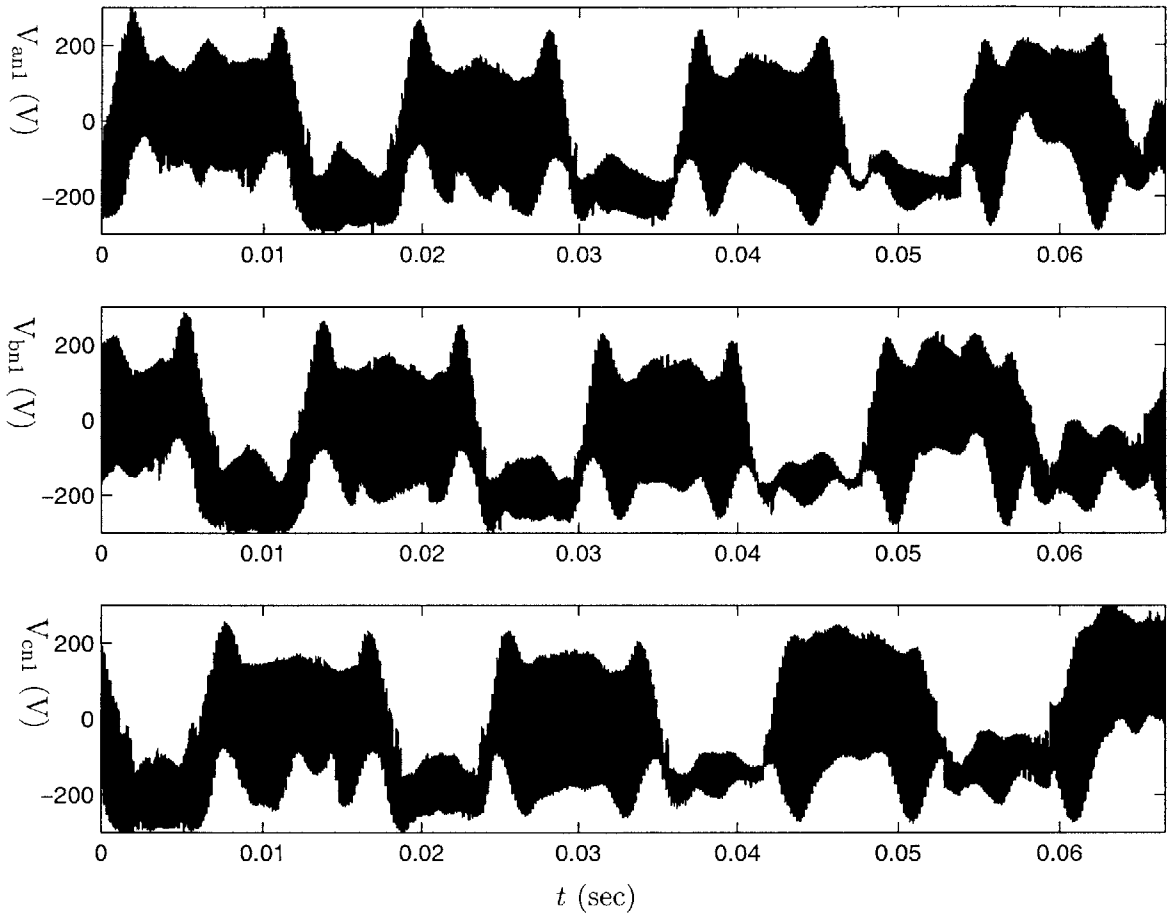


Figure 7-6: Phase-to-neutral Voltage Waveforms with 10% Third Harmonic at $\phi_3 \approx \frac{\pi}{2}$.

inverter, the load voltage V_2 will vary as a function of this third harmonic phase, ϕ_3. In addition, we should see no variation in the second rectifier load voltage, V_3, with respect to variations in ϕ_3. This functionality was demonstrated by means of an HP-6834B 3 Phase Power Source/Analyzer in Chapter 5.

The voltage inverter described in Chapter 6 was loaded with the firmware from Appendix F, Listing F.1. This firmware produces a V-Hz ramp on startup, motoring the machine to steady-state. Once steady-state has been reached, the inverter firmware begins producing third harmonic as described in Subsection 7.1.3. This harmonic component is created at a phase offset, ϕ_3, with respect to the phase A line-to-neutral voltage.

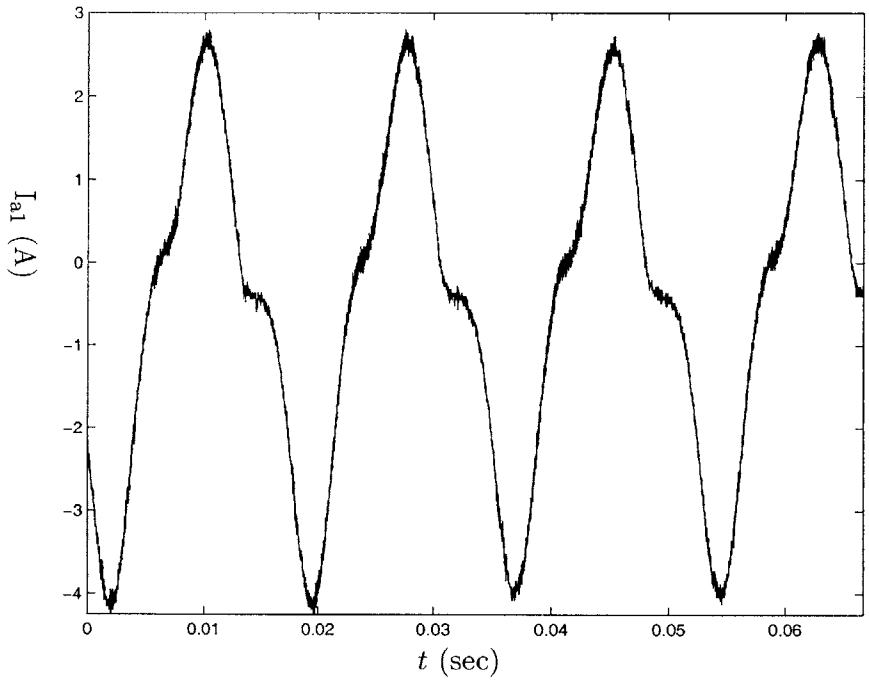


Figure 7-7: Phase-to-neutral Current Waveforms with 10% Third Harmonic at $\phi_3 \approx \frac{\pi}{2}$.

The variable ϕ_3 was slowly varied between 0 and π radians, and the rectifier load voltages, V_2 and V_3, were measured. The results are presented in Figure 7-10. We see less than 10 V of variation on V_3, and approximately 20 V of variation on V_2. The variation seen on V_2 is comparable to the simulated results presented in Chapter 4, specifically in Figure 4-7.

While the PWM inverter realized in this thesis does not produce 120 V_{RMS} line-to-neutral voltage, as seen in the voltage waveforms of Figures 7-3 and 7-6, and so Figure 7-10 cannot be directly compared to the curves of Figure 4-7, the system performance can be compared in terms of *relative* voltage variation with respect to the rectifier load voltage without third harmonic injection.

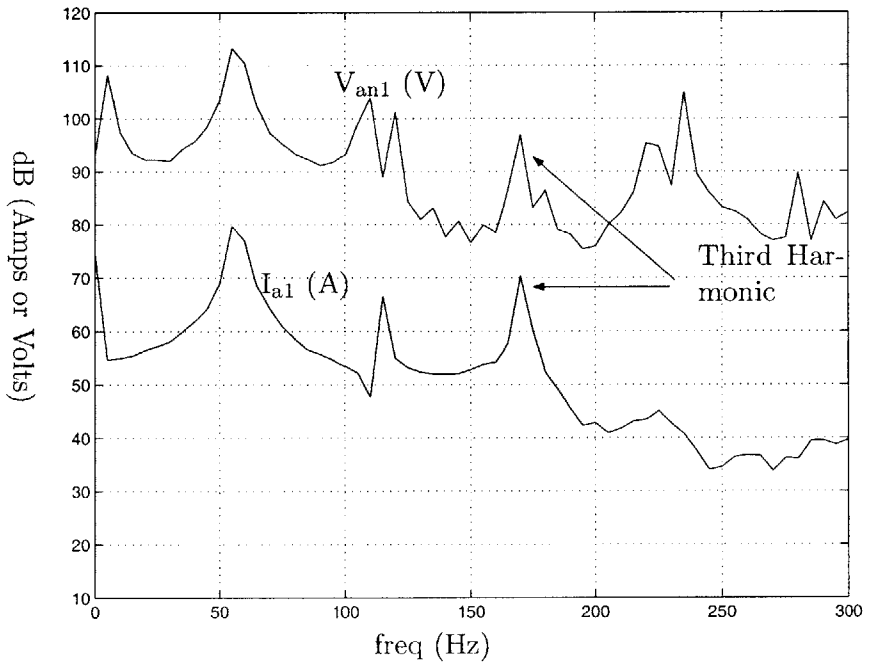


Figure 7-8: Input Waveform FFTs with 10% Third Harmonic at $\phi_3 \approx \frac{\pi}{2}$.

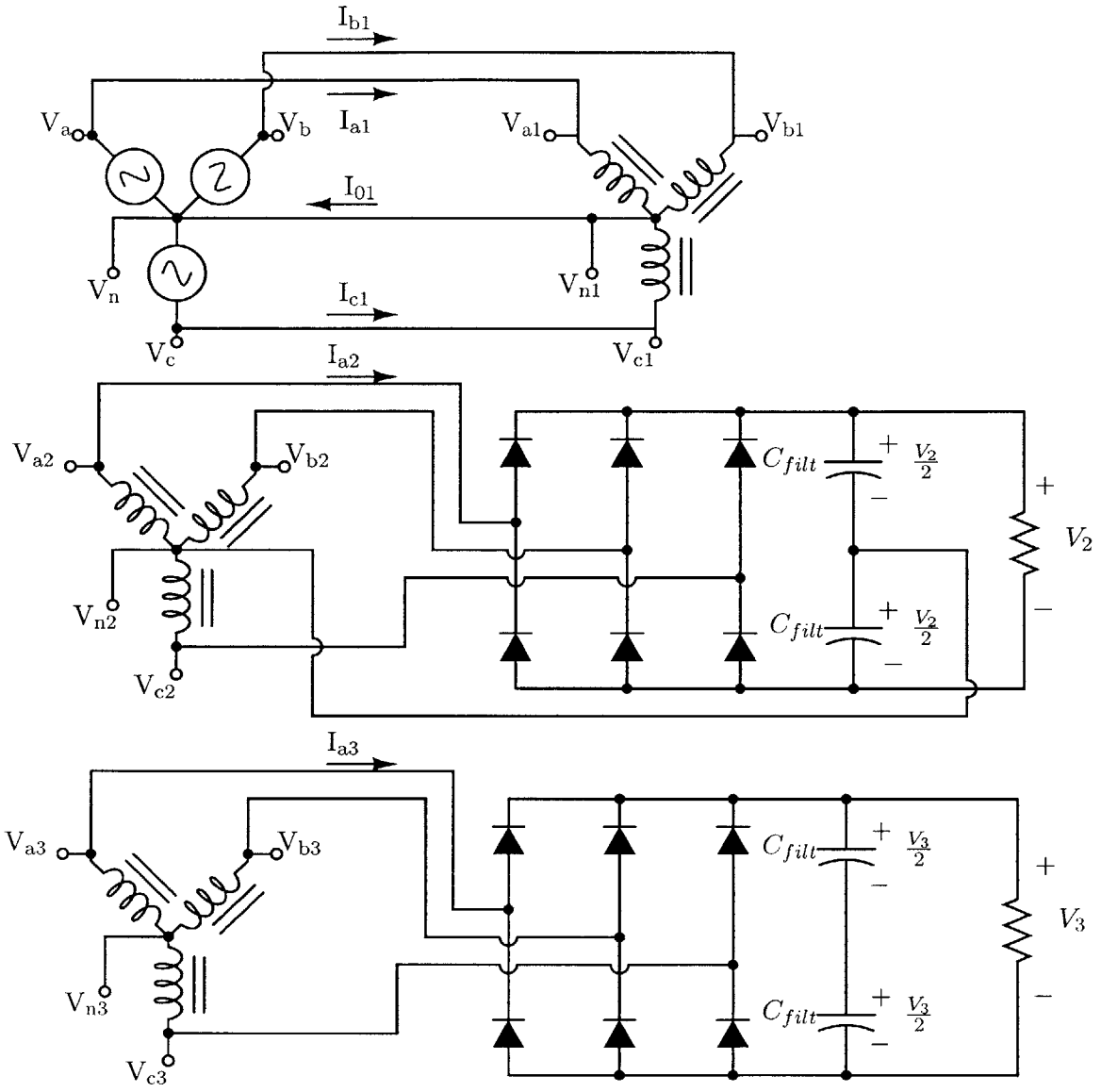


Figure 7-9: Experimental Setup: Multiple-stator Induction Machine Driving Three-Phase Rectifiers.

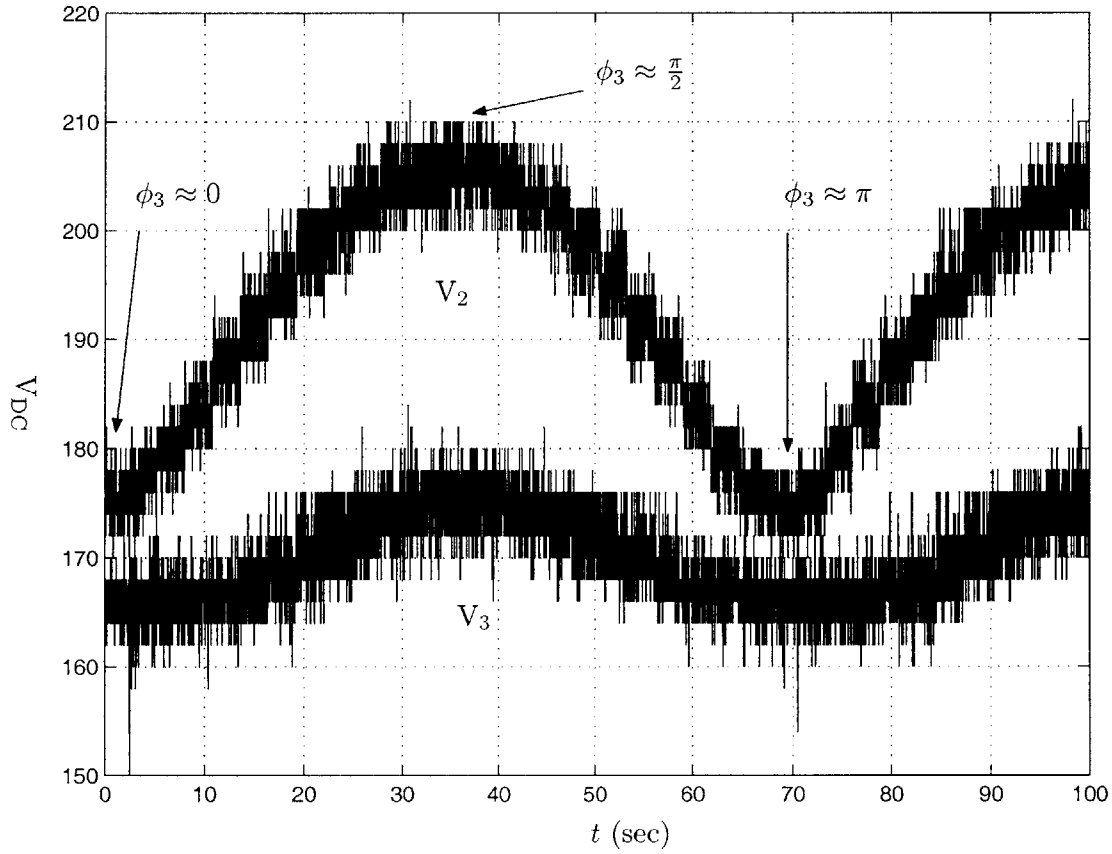


Figure 7-10: Experimental V_2 and V_3 Variation versus ϕ_3.

Chapter 8

Conclusion

GIVEN AN induction machine with multiple parallel stator winding circuits, exhibiting tight transformer coupling between the individual phases of each circuit, a system in which the machine can be operated in a traditional fashion and a tunable DC bus can be realized. Each of the machine stator winding circuits, that is, each of its three-phase field windings, must be electrically isolated from the remaining field windings. In addition, the stator circuits should be wye-connected, with physical access to the neutral points of each field circuit.

Imagine such a machine, in which the stator is made up of three parallel, three-phase, stator winding circuits. The first of these circuits is driven by a wye-connected voltage source in which the source neutral node and the stator star point are shorted. This connection allows triple-n, specifically third harmonic, line current components into the machine.

The voltage drive waveforms present at the first set of stator circuits will be present on the remaining machine stator circuits if these winding circuits are tightly-coupled, magnetically-speaking. These stator winding circuits can, in effect, be used as two three-phase transformers. The primary of both transformers is driven from the voltage source, and the secondary transformer winding represent the two remaining stator winding circuits.

A three-phase rectifier with a center-tapped output DC bus is connected to one of the remaining sets of winding circuits, and the rectifier DC bus center node is shared with the

star point. We denote this rectifier output voltage, V_2. Given this rectifier connection, third harmonic[1] current components can flow from the rectifier-connected stator windings into the rectifier. This thesis shows, through simulation and experiment, that the DC output voltage at this rectifier, V_2, varies as a function of the *phase* of the third harmonic, ϕ_3, from 0 to $\frac{\pi}{2}$. If ϕ_3 at these stator winding circuits can be controlled, V_2 can be tuned to accommodate for DC bus load changes and variations in the fundamental component of the drive voltage, for example.

For test purposes, an identical rectifier circuit is connected to the last set of winding circuits, however, the center tap of the rectifier DC bus is *not* shared with the neutral of the last stator winding circuits. The last rectifier output load voltage is referred to as V_3. Given this connection, triple-n current harmonics *cannot* flow into the rectifier. It is shown in this thesis that this last rectifier DC output voltage, V_3, is not tunable as a function of ϕ_3.

A half-bridge, PWM voltage inverter is developed to drive the first set of stator windings. This drive is capable of delivering balanced three-phase current, as well as varying amounts of third-harmonic current at controlled harmonic phase, ϕ_3. Using this inverter, a system, in which V_2 is modulated as a function of ϕ_3 while V_3 remains relatively constant, is demonstrated.

8.1 Further Work

8.1.1 Voltage Inverter

The voltage inverter presented in Chapter 6 is presently fairly unsophisticated. The PWM switching frequency on the half-bridge IGBTs is approximately 65 kHz, and the devices are being hard-switched. The device switching transients could be improved with the use of snubber circuits around each device [2,9]. While the snubbers would decrease the efficiency of the bridge, it is not crucial that this piece of power electronics be efficient in this proof-

[1] or any triple-n

of-concept stage.

In addition, the scheme used to generate the PWM duty cycle values is a simple sine wave lookup table. It has been shown in [2, 9] that there are a number of methods for generate PWM duty cycle values that result in better spectral-purity in the output AC waveforms. As was shown in Figures 7-5 and 7-8, the voltage waveforms produced by the inverter have a large amount of lower-order harmonics in addition to the intentional third harmonic. Eliminating these lower-order harmonics may eliminate any variations in the rectifier out voltage, V_3, not predicted in theory, simulation, or experimental verification with the HP-6834B 3 Phase Power Source/Analyzer.

8.1.2 Multiple-Stator Machine Characterization

A new steady-state model of the machine could be developed. Specifically, the equivalent circuit model for the transformer action happening between parallel stator circuits can be investigated. In addition, it is unclear what the effect of a load with reactive components on the non-driven stator winding circuits has on the electromechanical operation of the machine.

In addition, the injection of triple-n harmonic causes substantial zero-sequence current in the machine. This results in significant stator back-iron heating. The effect of heating on machine performance has not be investigated.

8.1.3 Rectifier Model

An analytic model for the variation of rectifier DC output voltage as a function of harmonic phase has not been developed. Currently, we only have simulated and qualitative models for this behavior. It may be possible to develop a fully analytic model of this behavior.

8.1.4 DC Voltage Control

The ϕ_3 to V_2 relationship shown in simulation and experimental results in Chapters 4, 5, and 7 is intriguing from a voltage control standpoint. Using straightforward feedback, an output

reference voltage, V_{ref} could be set and V_2 could be made to track V_{ref} by commanding the inverter to dynamically vary ϕ_3 to adjust for any disturbances in the system.

The system control could be realized through a lookup table, a simple linearization of the $V_2(\phi_3)$ characteristic, or an analytic model if an analytic $V_2(\phi_3)$ can be developed. Improved models of the machine, specifically the stator circuit transformer action, would aide in providing an accurate frequency-domain model for steady-state and transients through the driven stator winding circuits and into the secondary and tertiary winding circuits.

The Texas Instruments DSP used to realize the voltage inverter has a on-board multi-channel ADC. Through proper isolation and some analog front-end circuitry, the rectifier output voltages, in addition to any necessary phase current and line-to-neutral voltages, can be sampled directly and used in the inverter DSP firmware. This platform offers enough computation power and speed[2] to dynamically vary ϕ_3 in reaction to changing V_2.

[2] 40MIPS in its current incarnation

Appendix A

A derivation of the triple-n requirement in Wye-connected three-phase systems

Given a Wye-connected balanced load driven with a sinusoidal voltage three-phase system, as shown in Figure A-1, we show that all triple-n current harmonics must be zero. The current into the common node of the load must sum to zero,

$$I_a + I_b + I_c = 0. \tag{A.1}$$

and we define the phase currents as

$$I_a = \sum_{k=1}^{\infty} I_k \cdot \cos\left(k\omega t + \phi_k\right) \tag{A.2}$$

$$I_b = \sum_{k=1}^{\infty} I_k \cdot \cos\left(k\left(\omega t + \frac{2\pi}{3}\right) + \phi_k\right) \tag{A.3}$$

$$I_b = \sum_{k=1}^{\infty} I_k \cdot \cos\left(k\left(\omega t - \frac{2\pi}{3}\right) + \phi_k\right). \tag{A.4}$$

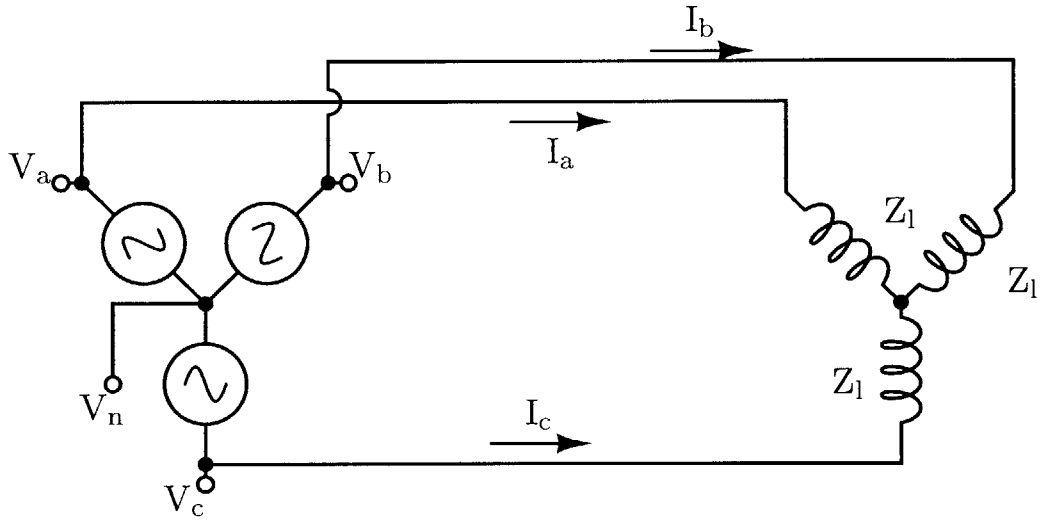


Figure A-1: Three-phase Wye-connected source and load with disconnected star-points.

Using the trig identity

$$\cos(a+b) = \cos a \cdot \cos b - \sin a \cdot \sin b$$

We can rewrite Equations A.3, A.4 as

$$I_b = \sum_{k=1}^{\infty} I_k \left\{ \cos(k\omega t + \phi_k) \cdot \cos\left(k\frac{2\pi}{3}\right) - \sin(k\omega t) \cdot \sin\left(k\frac{2\pi}{3}\right) \right\} \tag{A.5}$$

$$I_c = \sum_{k=1}^{\infty} I_k \left\{ \cos(k\omega t + \phi_k) \cdot \cos\left(k\frac{2\pi}{3}\right) + \sin(k\omega t) \cdot \sin\left(k\frac{2\pi}{3}\right) \right\}. \tag{A.6}$$

Using Equations A.1, A.2, A.5, and A.6 we sum the expressions for the phase currents

$$I_a + I_b + I_c = 0 = \sum_{k=1}^{\infty} I_k \left\{ \cos(k\omega t + \phi_k) \left[1 + 2\cos\left(k\frac{2\pi}{3}\right) \right] \right\}.$$

Solving for the summation argument, we find

$$I_k \left\{ 1 + 2\cos\left(k\frac{2\pi}{3}\right) \right\} = 0,$$

and thus, the requirement is satisfied for

$$\begin{cases} I_k \neq 0, & k = 3 \cdot n \pm 1, \quad n \in \mathbb{Z} \\ I_k = 0, & k = 3 \cdot n, \qquad n \in \mathbb{Z} \end{cases}$$

Thus, we see that all non-triple-n harmonics can exist in this system, but triple-n current harmonics cannot.

Appendix B

Fitting DQ-Axis Models to Machine Data

Using block-rotor and no-load tests, a set of rough machine parameters are determined. Using these initial parameter estimates and transient phase current and line-to-neutral voltage of the same machine during initial startup, the Matlab code in Listing B.1 performs a nonlinear least-squares fit over the machine parameters.

Listing B.1: fit_data.m

```
% Jack Holloway. fit_data.m
% This is the wrapper for the data-fitting routines.

%inital guesses
rs = 2.0;       % Stator resistance
rr = 1.5;       % Rotor resistance
Xm = 42.09;     % Magnetizing Impedance, in Ohms on a 60 Hz base
X1s = 2.80;     % Stator Side Leakage Impedance, in Ohms on a 60 Hz base
X1r = 2.80;     % Rotor Side Leakage Impedance, in Ohms on a 60 Hz base
J = .0168;      % Rotor Inertia
Bl = 0.0;       % Load Damping Coefficient
Tl = 0.0;       % Load torque.

%We have to introduce an arbitrary phasing component.
phi = 0.0;      % phase component

init_guess = [rs, rr, Xm, X1s, X1r, J, Bl, phi, Tl];

% Load current, voltage, and time data from the motor.
load vas.dat;
load ias.dat;
time = ([1:vas(1)]*vas(2))';
vas = vas(5:vas(1)+4);
```

```
% remember: scale the current as per the setting of the
% current probe. Here: 10A/div and the scope is set to 10mV/div
ias = 10/(10e-3)*ias(5:ias(1)+4);

% trim the data to the right domian (time). We just want
% the startup transient.
% time index 1100 (in the current/voltage data) puts us
% pretty close to the beginning of the startup transient...
time = time(1100:length(time)) - time(1100);
% start at time zero
vas = vas(1100:length(vas));
vas = vas - mean(vas);          % remove any DC offset
ias = ias(1100:length(ias));
ias = ias - mean(ias);          % remove the DC offset from the probes

exp_data = [time, vas, ias];

% lower/upper bound on the params:
LB = zeros(size(init_guess));
UB = Inf*ones(size(init_guess));
UB(length(8)) = 2*pi;

params = lsqcurvefit(@runind_param, init_guess, time, exp_data, LB, UB);

disp(['rs, rr, Xm, X1s, X1r, J, Bl, phi, Tl'])
disp(params)

out = runind_param(params, time);
hold off;
plot(out(:,1), out(:,3), '--')

hold on;
plot(time, ias);
grid on;
hold off;
ylabel('I_{as}(Amps)');
xlabel('t(secs)');
legend('Simulation', 'Experimental')
```

Listing B.2-B.4 are Matlab code for transient and steady-state simulation of an induction machine with the given parameters.

Listing B.2: runind_param.m

```
function out = runind_param(lump, time)
%  [time, vas, ias] = runind_param([rs, rr, Xm, X1s, X1r, phi], time)
%  This script runs the AC induction motor simulation during free acceleration
%
%  This software is distributed in the hope that it will be useful, but
%  WITHOUT ANY WARRANTY.  It is for educational use only.  Please do not
%  distribute or sell this software, or remove the copyright notice.
%
%  Copyright, 1995, 1998, 2000  Steven B. Leeb
%  Modified Jack Holloway Aug 2003

global P rs rr Xm X1s X1r we J Bl vds vqs vqr vdr Tl

%%%%%%%%%%%%%%%%%%%%
% A few things from indparam not specified by the function params
```

```
%%%%%%%%%%%%%%%%%%%
P = 4;          % Number of poles (*not* pole pairs)


% Use the below values  for initial guesses in lsqcurvefit
%rr             Rotor resistance
%Xm             Magnetizing Impedance, in Ohms on a 60 Hz base
%X1s            Stator Side Leakage Impedance, in Ohms on a 60 Hz base
%X1r            Rotor Side Leakage Impedance, in Ohms on a 60 Hz base
we = 377.;  % Base electrical frequency, rads per second (60 Hz)
%J              Rotor Inertia
%Bl             Load Damping Coefficient
vds = 0.0;  % D axis stator voltage
vqs = 120*sqrt(2);  % Q axis stator voltage (120V RMS).
vqr = 0.0;  % D axis rotor voltage
vdr = 0.0;  % Q axis rotor voltage
Tl = 0.0;   % Load torque

rs = lump(:,1);
rr = lump(:,2);
Xm = lump(:,3);
X1s = lump(:,4);
X1r = lump(:,5);
J = lump(:,6);
Bl = lump(:,7);
phi = lump(:,8);
Tl = lump(:,9);

disp(lump)

state0 = [0 0 0 0 0 0]';
[tout,yout] = ode45('ind',time, state0);

%setup the arbitrary phase angle.
yout(:,6) = yout(:,6) + phi;

[m, m2] = convind(tout, yout);
ias = m2(:,2);
vas = m2(:,5);

out = [time, vas, ias];
```

Listing B.3: ind.m

```
function [slopes] = ind(t,statev)
%  [slopes] = ind(t,statev)
%
%  This script computes the state variable derivatives for a fifth
%  order model of a balanced, three phase AC induction machine.
%  The state variables are the D and Q stator and rotor fluxes, and
%  the rotor speed (wr).
%  Use this script with ODE45 to simulate the performance of the
%  induction machine.
%
%  Run indparam.m to load sample machine parameters before simulating.
%
%  This software is distributed in the hope that it will be useful, but
%  WITHOUT ANY WARRANTY.  It is for educational use only.  Please do not
%  distribute or sell this software, or remove the copyright notice.
%
```

```
% Copyright, 1995, 1998, 2002 Steven B. Leeb
% The variable w determines the reference frame in which the simulation
% will be conducted. With w = 377, the simulation variables will be in
% a synchronously rotating reference frame.
global P rs rr Xm X1s X1r we J Bl vds vqs vqr vdr Tl
w = 377.;
Lm = Xm/we; % This is the magnetizing inductance
L1s = X1s/we; % Stator leakage
L1r = X1r/we; % Rotor leakage
Las = L1s + Lm;
Lar = L1r + Lm;

lamqs = statev(1);
lamds = statev(2);
lamqr = statev(3);
lamdr = statev(4);
wr    = statev(5);
th    = statev(6);

D = Lm*Lm - Las*Lar;
idr   = (Lm*lamds - Las*lamdr)/D;
iqr   = (Lm*lamqs - Las*lamqr)/D;
iqs   = (Lm*lamqr - Lar*lamqs)/D;
ids   = (Lm*lamdr - Lar*lamds)/D;

s1 = (vqs - w*lamds - rs*iqs);
s2 = (vds + w*lamqs - rs*ids);
s3 = (vqr - (w - wr)*lamdr - rr*iqr);
s4 = (vdr + (w - wr)*lamqr - rr*idr);
% P is the number of poles, *not* pole pairs
T  = (3/2)*(P/2)*(lamqr*idr - lamdr*iqr);
s5 = (P/2)*(T - Tl)/J;
s6 = 377;
slopes = [s1 s2 s3 s4 s5 s6]';
```

Listing B.4: convind.m

```
function [m,m2] = convind(t,y);
% [m,m2] = convind(t,statev)
%
% This script transforms the DQ stator and rotor fluxes computed using
% ind.m and ode45 back into laboratory frame stator currents and voltages,
% e.g., ias and vas for phase a.
%
% The variable w determines the reference frame in which the simulation
% will be conducted. With w = 377, the simulation variables will be in
% a synchronously rotating reference frame.
%
% On return, the output matrices m and m2 contain:
%       m = [ids iqs idr iqr];
%       m2 = [T ias ibs ics vas vbs vcs];
% This script also plots the simulated rotor torque versus speed on return.
%
% This software is distributed in the hope that it will be useful, but
% WITHOUT ANY WARRANTY. It is for educational use only. Please do not
% distribute or sell this software, or remove the copyright notice.
%
% Copyright, 1995, 1998, 2002 Steven B. Leeb
% Modified Jack Holloway, 2003
global P rs rr Xm X1s X1r we J Bl vds vqs vqr vdr Tl
```

```
% select reference frame (see ind.m)
 w = 377.;

 Lm  = Xm/we;  % This is the magnetizing inductance
 Lls = Xls/we; % Stator leakage
 Llr = Xlr/we; % Rotor leakage
 Las = Lls + Lm;
 Lar = Llr + Lm;

 lamqs = y(:,1);
 lamds = y(:,2);
 lamqr = y(:,3);
 lamdr = y(:,4);
 wr    = y(:,5);
 th    = y(:,6);

 D = Lm*Lm - Las*Lar;
 idr   = (Lm*lamds - Las*lamdr)/D;
 iqr   = (Lm*lamqs - Las*lamqr)/D;
 iqs   = (Lm*lamqr - Lar*lamqs)/D;
 ids   = (Lm*lamdr - Lar*lamds)/D;

 T     = (3/2)*(P/2)*(lamqr.*idr - lamdr.*iqr);

 ias = cos(th).*ids - sin(th).*iqs;
 ibs = cos(th - 2.*pi/3.).*ids - sin(th - 2.*pi/3.).*iqs;
 ics = cos(th + 2.*pi/3.).*ids - sin(th + 2.*pi/3.).*iqs;
 vas = cos(th).*vds - sin(th).*vqs;
 vbs = cos(th - 2.*pi/3.).*vds - sin(th - 2.*pi/3.).*vqs;
 vcs = cos(th + 2.*pi/3.).*vds - sin(th + 2.*pi/3.).*vqs;

 m = [ids iqs idr iqr];
 m2 = [T ias ibs ics vas vbs vcs];
```

Appendix C

Simulating Rectifier Ouput Voltage Versus Third Harmonic Phase

The Matlab code in Listing C.1 sweeps the phase of the third harmonic voltage component with respect to the fundamental of phase A of a three-phase system. A SPICE model of a three-phase rectifier, Listing C.2, is used for the circuit simulation.

The Matlab code in Listing C.1 makes use of Mike Perrott's HSpiceToolbox, available at the MIT MTL High-Speed Circuits and Systems group webpage.

Listing C.1: run_sim.m

```
% [phase, Vout, Vout_w_AC_L] = run_sim(third_amp)
% Jack W. Holloway
% Spring 2003
% Sweeps the phase of the third harmonic. V3 = amp*V1
% We assume a 120VRMS fundmental drive
function [phase_vec, outamp, outamp_comm] = run_sim(amp)

  phase_vec = [0:pi/96:2*pi];
  outamp = zeros(size(phase_vec));
  outamp_comm = zeros(size(phase_vec));

  % Machine commutation inductance
  for Xls = [0, 7.427e-3]
    k = 0;
    for phase = phase_vec
      k = k+1;
      disp(['Third harmonic (', num2str(100*amp),'%), ',...
            num2str(180/pi*phase),' degrees. ', ...
            num2str(100*k/length(phase_vec)),...
            '% done. Commutation L=',num2str(Xls*1e3),...
```

```
            'F.']);
        command = ['sed s/\<DELAY\>/',...
                   num2str(phase*180/pi),...
                   '/ rectifier.sp.templ | sed s/\<AMPIII\>/'...
                   num2str(amp*120*sqrt(2)),...
                   '/ | sed s/\<XLS\>/', num2str(Xls),...
                   '/ > rectifier.sp'];

        [s w] = system(command);
        [s w] = system('hspice rectifier.sp');

        datfile =  loadsig('rectifier.tr0');
        [s w] = system('rm ./*.tr0 ./*.ic ./*.st0');

        vout = evalsig(datfile, 'vp-vm');
        t = evalsig(datfile, 'TIME');

        %get rid of _ALL_ of the transient.
        N = length(t);
        if(Xls ~= 0)
          outamp_comm(k) = mean(vout(1000:N));
        else
          outamp(k) = mean(vout(1000:N));
        end;
      end;
    end;
```

Listing C.2: rectifier.sp.templ

```
3-PHASE RECTIFIER WITH Center-tapped DC bus
* Jack Holloway, Fall 2002.
.option temp=27

.param AMP_I=169V
.param AMP_III=<AMPIII>V
.param DELAY=<DELAY>
.param Xls=<XLS>
.MODEL diode D Rs=1

VP1_1 P1inL P1_3 sin(0 AMP_I 60Hz 0 0 0)
VP1_3 P1_3 GND  sin(0 AMP_III 180Hz 0 0 DELAY)
LphaseA P1inL P1in Xls
D1p P1in Vp diode
D1n Vm P1in diode

VP2_1 P2inL P2_3 sin(0 AMP_I 60Hz 0 0 120)
VP2_3 P2_3 GND  sin(0 AMP_III 180Hz 0 0 DELAY)
LphaseB P2inL P2in Xls
D2p P2in Vp diode
D2n Vm P2in diode

VP3_1 P3inL P3_3 sin(0 AMP_I 60Hz 0 0 240)
VP3_3 P3_3 GND  sin(0 AMP_III 180Hz 0 0 DELAY)
LphaseC P3inL P3in Xls
D3p P3in Vp diode
D3n Vm P3in diode

Cload_p Vp GND 750u
Cload_m GND Vm 750u
Rload_p Vp GND 150
```

```
Rload_m GND Vm 150

.options gmin=1e-9
.options ABSTOL=2e-9
.options VNTOL=2e-3
.options RELTOL=2e-3
.options post
.tran 0.025ms 2.0s 0ms
.op

.end
```

Appendix D

Phasor Representation of Phase-Control

The phasor description used in Subsection 4.3.1 can be applied to describe the voltage variation at the output of a polyphase center-tapped rectifier circuit driven with an arbitrary set of harmonics.

Given a k-phase center-tapped rectifier with excitation voltages

$$\begin{aligned}
V_{an} &= V_1 \sin(\omega t) + \sum_m^{\infty} V_m \sin(m\omega t + \phi_m) \\
V_{bn} &= V_1 \sin\left(\omega t + \frac{2\pi}{k}\right) + \sum_m^{\infty} V_m \sin\left(m\left(\omega t + \frac{2\pi}{k}\right) + \phi_m\right) \\
V_{cn} &= V_1 \sin\left(\omega t + 2\frac{2\pi}{k}\right) + \sum_m^{\infty} V_m \sin\left(m\left(\omega t + 2\frac{2\pi}{k}\right) + \phi_m\right) \\
&\vdots \\
V_{kn} &= V_1 \sin\left(\omega t + (k-1)\frac{2\pi}{k}\right) + \sum_m^{\infty} V_m \sin\left(m\left(\omega t + (k-1)\frac{2\pi}{k}\right) + \phi_m\right),
\end{aligned}$$

we can describe the input line-to-neutral time-domain waveform amplitudes for any harmonic phase shift, ϕ_m. Plotting the curves over all time and finding the imaginary axis

intercept with the resultant curve provides us with an AC-side time-domain amplitude. This, in turn, gives is a qualitative representation of the average output voltage amplitude. As ϕ_m is varied, we see how the average output voltage varies.

Mathmatically, we simply plot the angle and magnitude of the complex representations of the drive voltages. For example, the kth phase-to-neutral voltage in a k-phase system with injected lth harmonic becomes

$$\overline{V_{kn}} = V_1 e^{(j(\omega t+(k-1)\frac{2\pi}{k}))} + V_l e^{(jl(\omega t+(k-1)\frac{2\pi}{k})+j\phi_l)}$$

at a given ωt.

This phasor simplifies to the first line-to-neutral voltage for all systems and all ωt,

$$\overline{V_{kn}} = V_1 e^{(j\omega t)} + V_l e^{(j(l\omega t+\phi_l))}.$$

Ploting these phasors for varying k and l at $\phi_l = 0$ yields the curves in Figure D-1

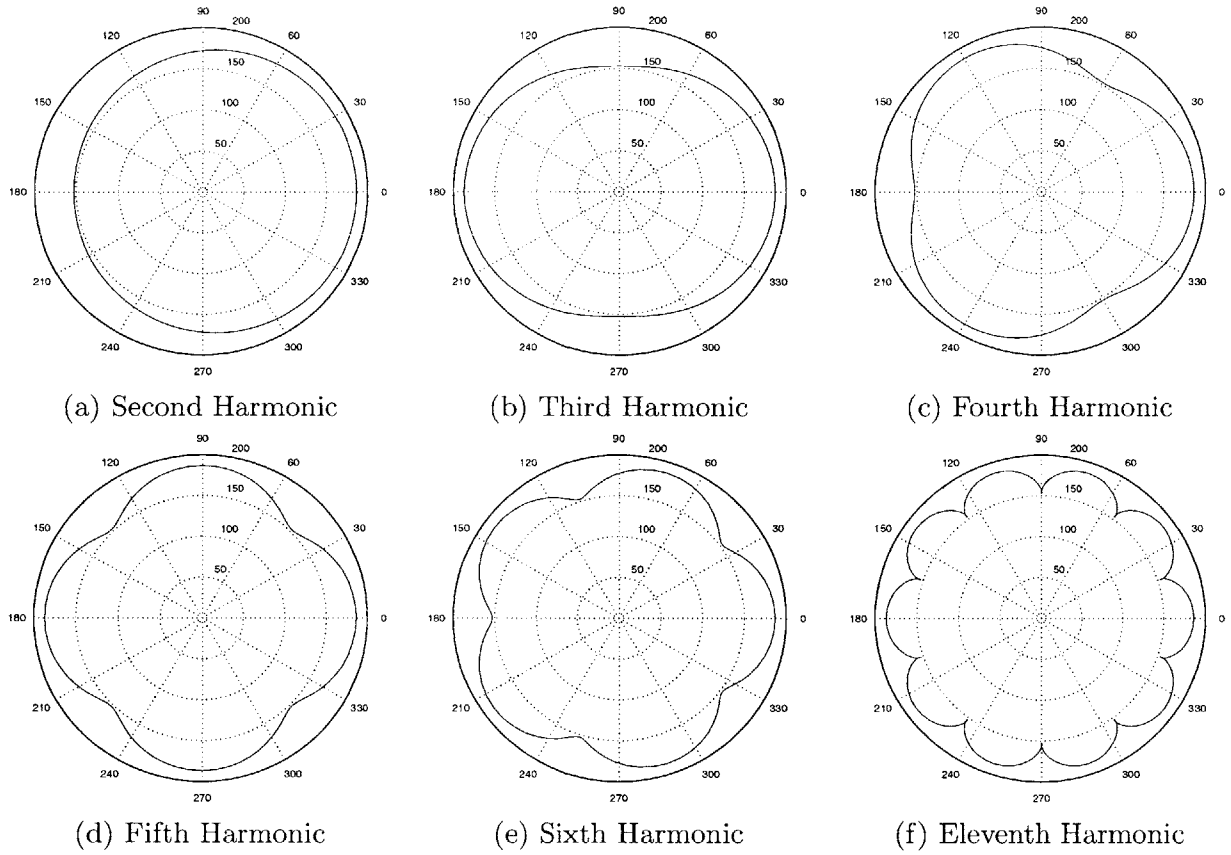
These plots all assume the excitating harmonic amplitude is 10% of the fundamental amplitude, $120V_{RMS}$ in this case. Of particular not is the 11th harmonic of Figure D-1(f), in which the curve has begun to intersect itself. Plotting systems with more phases results in the exact same curve shape. The curve shapes are strictly a function of the order of the harmonic excitation.

We can see that a rotation of $\phi_2 = -\frac{\pi}{2}$ in the second harmonic case (a) will maximize the time-domain waveform amplitude. For the fourth harmonic, we see $\phi_4 = +\frac{\pi}{2}$ will maximize the time-domain waveform amplitude.

For our sinewave drive, Figure D-1(d) shows us that for some fifth harmonic excitation, $\phi_5 = 0$ maximizes the time-domain waveform. In addition, this same phase, $\phi_5 = 0$ maximizes a cosine-fundamental[1] voltage waveform in the time-domain.

In the case of the sixth harmonic, Figure D-1(e), in order to maximize the time-domain waveform for a sinewave drive, the harmonic phase must be $\phi_6 = -\frac{\pi}{2}$. This procedure is

[1] observe the real-axis crossing



(a) Second Harmonic (b) Third Harmonic (c) Fourth Harmonic

(d) Fifth Harmonic (e) Sixth Harmonic (f) Eleventh Harmonic

Figure D-1: Generalized phasor harmonic plots for a three-phase systems.

very general and can be applied to any polyphase system for one harmonic excitation.

It is interesting to note that added different amounts of *more than one* harmonic allows one to affect multivariable control on the system. By independently varying the phase of, say, two harmonics, we would arrive at a spherical polar plot, in which we would be interested in one of the axis crossing[2].

[2]depending on whether or not the system is being driven by cosines or sines

Appendix E

Simulating Center-Tapped Rectifier Input Impedance

The Matlab code in Listing E.1 sweeps the phase of the third harmonic voltage component with respect to the fundamental of phase A of a three-phase system. A SPICE model of a three-phase rectifier, Listing E.2, is used for the circuit simulation.

The line current[1] fundamental Fourier component, parsed from the HSpice ouput using Listing E.3, magnitude and phase, are used to compute the input impedance of the rectifier.

The Matlab code in Listing E.1 makes use of Mike Perrott's HSpiceToolbox, available at the MIT MTL High-Speed Circuits and Systems group webpage. The code in Listing E.3 is derived from work in [26].

Listing E.1: run_impedance_tests.m

```
% Jack W. Holloway
% Spring 2004
% Sweeps the phase of the third harmonic. V3 = 0.1*V1
% We assume a 120VRMS fundmental drive

  phase_vec = [0:pi/256:2*pi];
  Zin_phase = zeros(size(phase_vec));
  Zin_mag = zeros(size(phase_vec));
  V = zeros(size(phase_vec));
  I = zeros(size(phase_vec));
  amp = 0.1;
  Xls = 0;%7.427e-3;
```

[1] current into the rectifier

```
k = 0;
j = sqrt(-1);
close all;
for phase = phase_vec
  k = k+1;
  disp(['Third harmonic (', num2str(100*amp),'%), ',...
        num2str(180/pi*phase),' degrees. ', ...
        num2str(100*k/length(phase_vec)),...
        '% done. Commutation L=',num2str(Xls*1e3),...
        'F.']);
  command = ['sed s/\<DELAY\>/',...
             num2str(phase*180/pi),...
             '/ rectifier.sp.templ | sed s/\<AMPIII\>/'...
             num2str(amp*120*sqrt(2)),...
             '/ | sed s/\<XLS\>/', num2str(Xls),...
             '/ > rectifier.sp'];

  [s w] = system(command);
  command = ['AVANTD_LICENSE_FILE=/usr/local/flexlm/',...
             'licenses/license.dat hspice rectifier.sp ',...
             '> out.txt'];
  [s w] = system(command);
  [s w] = system('./impextract.sh out.txt | grep -v HSPICE > four.dat');
  four = load('four.dat');
  % Voltage fundamental
  V1 = four(1);
  % Voltage phase, absolute
  PhiV = four(2)*pi/180;
  % Current Fundamental
  I1 = -1*four(3); % note the direction of the current in SPICE
  % Current phase, absolute
  PhiI = four(4)*pi/180;
  % four(5) = FREQ of fundamental
  datfile =  loadsig('rectifier.tr0');
  [s w] = system('rm ./*.tr0 ./*.ic ./*.st0 ./out.txt ./four.dat');
  V(k) = V1*exp(j*PhiV);
  I(k) = I1*exp(j*PhiI);
  Zin_phase(k) = PhiV - PhiI;
  Zin_mag(k) = V1/I1;
end;
Zin_phase = unwrap(Zin_phase);
phase = phase_vec;
```

Listing E.2: rectifier.sp.templ

```
3-PHASE RECTIFIER WITH Center-tapped DC bus
* Jack Holloway, Fall 2002.
.option temp=27

.param AMP_I=169V
.param AMP_III=<AMPIII>V
.param DELAY=<DELAY>
.param Xls=<XLS>
.MODEL diode D Rs=1

VP1_1 P1inL P1_3 sin(0 AMP_I 60Hz 0 0 0)
VP1_3 P1_3 GND  sin(0 AMP_III 180Hz 0 0 DELAY)
LphaseA P1inL P1in Xls
D1p P1in Vp diode
D1n Vm P1in diode
```

```
VP2_1 P2inL P2_3 sin(0 AMP_I 60Hz 0 0 120)
VP2_3 P2_3 GND  sin(0 AMP_III 180Hz 0 0 DELAY)
LphaseB P2inL P2in Xls
D2p P2in Vp diode
D2n Vm P2in diode

VP3_1 P3inL P3_3 sin(0 AMP_I 60Hz 0 0 240)
VP3_3 P3_3 GND  sin(0 AMP_III 180Hz 0 0 DELAY)
LphaseC P3inL P3in Xls
D3p P3in Vp diode
D3n Vm P3in diode

Cload_p Vp GND 750u
Cload_m GND Vm 750u
Rload_p Vp GND 150
Rload_m GND Vm 150

.options gmin=1e-9
.options ABSTOL=2e-9
.options VNTOL=2e-3
.options RELTOL=2e-3
.four 60 v(P1inL) i(VP1_1)
.options post
.tran 0.025ms 2.0s 0ms
.op

.end
```

Listing E.3: impextract.sh

```
#!/bin/bash

# extract impedance parameters from input spice file
# this script grabs the fundamentals of the fourier series
# resulting from '.four FREQ v(in) i(vinm)'
# Written by Riad Wahby, Spring 2004

grep -A7 'v(p1in1)' $1 | grep '^[[:space:]]*1[[:space:]]' | \
    awk 'BEGIN {OFS=",";} {print($3,$5);}' > $2.v
grep -A7 'i(vp1_1)' $1 | grep '^[[:space:]]*1[[:space:]]' | \
    awk 'BEGIN {OFS=",";} {print($3,$5,$2);}' > $2.i

paste $2.v $2.i | tr '\t' ,
rm $2.i $2.v
```

Appendix F

TI TMS320LF2406A DSP

F.1 Constant V-Hz Control with Third Harmonic Injection

This appendix contains the entirety of the code used to implement the V-Hz ramp and steady-state ϕ_3 sweeping used in Chapter 7.

Listing F.1 contains the C-code used to initialize the TI DSP, setup the PWM generation system, clear any PIIPM hardware faults, and then generate the appropriate duty cycle values for the inverter PWM scheme.

Listings F.2-F.5 are header files necessary for the code in Listing F.1 to compile and run.

Listing F.1: mot_cntl.c, top-level control software

```
/* Jack W. Holloway, code for a motor drive based on the LF240xa */

#include <stdlib.h>
#include "regs240x.h"
#include "pwm/include/F2407pwm.h"
#include "sysvecs.h"
#include "sine.h"
#include <math.h>

#define WAIT_STATES 0x40;

#define SET_LO(x,b) ((x)&=~(1<<(b)))
#define SET_HI(x,b) ((x)|=(1<<(b)))

#define PI 3.1415927
#define TWOPI 6.2831853
#define TWOPIBYTHREE 2.0943951
#define SEVENPI 21.9911485751286
```

```
#define ONEBYTWOPI 0.159154943091895

#define RAMP_TIMER 256000L
#define TIMER 300
#define OFFSET 150
// OFFSET is always half of TIMER

void interrupt periodic_interrupt_isr(void);
void interrupt phantom(void);
void trap(void);
void setup_PLL(void);
void setup_PWM(void);
void reset_PWM_fault(void);

#define disable_ints() asm("    setc    intm    ")
#define enable_ints()  asm("    clrc    intm    ")

/* pwm stuff */

unsigned long isr_count = 0;
int ramping = 0;
double gain = 0.0; // fund gain
double rad_advance = 0.0;
double rad = 0.0;

/* used in the ISR */
double phase3 = 0.0; // phase of third harmonic w/ respect to phase A
double gain3 = 0.0;

main()
{
        double rampval;
        const double const_rad_advance = 2.0*TWOPI/TAB_RES;
        const double const_gain = 0.5*TIMER/32768.0;
        double phase3_temp;
        disable_ints();

        WDCR = 0x68;

        setup_PLL();

        setup_PWM();

        IFR = 0xffff; /* Clear all interrupts. */
        IMR = 0x0002; /* Enable INT2. */

        EVAIFRA = 0xffff; /* Clear all EV1 group A EV interrupt flags. */
        EVAIMRA = 0x0080; /* Enable Timer 1 period interrupt interrupts */
        ramping = 1;

        enable_ints();

        while(ramping == 1)
        {
                /* If we are ramping voltage, we need to adjust
                the gain and table step size. Oh my.
                */
                rampval = (double)isr_count/RAMP_TIMER;
                //change the freq and gain multipliers.
                // normally, rad_advance = 2*PI/TAB_RES
                rad_advance = rampval*const_rad_advance;
```

```
80                  gain = rampval*const_gain;
                    gain3 = 0.0;
                    /* are we done ramping? If so, get out of this loop */
                    if(isr_count >= RAMP_TIMER)
                            ramping = 0;
            }

            gain3 = const_gain*0.1; //percentage of 3rd harmonic
            gain = const_gain - gain3;

90          while(1)
            {
                    /* we are now at steady state. Let's do some
                    control...
                    */

                    /* We'll flip the phase of the third harmonic between 0 and PI
                    every so many counts through the ISR...
                    */

100                 if(isr_count >= 16000L)
                    {
                            /* update the phase info of the f_3 */
                            /*
                            if(phase3 == 0.0)
                                    phase3 = PI/3;
                            else if(phase3 > 0.0)
                                    phase3 = 0.0;
                            */
                            phase3_temp = phase3 + PI/12;
110                         phase3 = fmod(phase3_temp, TWOPI);
                            isr_count = 0;
                    }

                    /* otherwise, wait.. */
            }

            /* end main() */
    }

120 /* Serviced at every timer interrupt ---
       we reload the compare registers here. */
    interrupt void periodic_interrupt_isr()
    {
            double third; //amount of third harmonic
            double rad_a, rad_b, rad_c;
            double rad3;

            disable_ints();

130         isr_count++;
            rad = rad + rad_advance;
            rad = fmod(rad, TWOPI);
            rad3 = 3.0*rad + phase3;
            rad3 = fmod(rad3, TWOPI);
            third = gain3*lookup_sin(rad3);

            rad_a = rad;
            rad_b = rad + TWOPIBYTHREE;
            rad_c = rad - TWOPIBYTHREE;
140
```

```
        /* Phase A duty cycle */
        CMPR1 = (int)(gain*lookup_sin(rad_a) + third) + OFFSET;
        /* Phase B duty cycle */
        CMPR2 = (int)(gain*lookup_sin(rad_b) + third) + OFFSET;
        /* Phase C duty cycle */
        CMPR3 = (int)(gain*lookup_sin(rad_c) + third) + OFFSET;

        /* Done with the ISR */

        IFR = 0xffff; /* Clear all interrupts. */
        IMR = 0x0002; /* Enable INT2. */
        EVAIFRA = 0xffff; /* Clear all EV1 group A EV interrupt flags. */
        EVAIMRA = 0x0080; /* Enable Timer 1 period interrupt interrupts */
        enable_ints();
}

/* a debugging tool */
void trap()
{
        //Square wave on the contactor pin (#13) on the serial connector.
        MCRA = MCRA&(~0x4000);
        //make that pin an output pin
        PBDATDIR = PBDATDIR|0x4000;
        while(1) {
                if(GP1CONA & (1<<13))
                        PBDATDIR |= 0x0040;
                else
                        PBDATDIR &= ~0x0040;
        }
}


void setup_PLL()
{
  /* setup the PLL module */
  /* 40MHz with a 10MHz on-board Xtal */
  SCSR1 = 0x0000;
}



void setup_PWM(void)
{
        /* setup the modules */

        //Setup pin (#13) on the serial connector as a digital IO pin.
        MCRA = MCRA&(~0x4000);
        PBDATDIR = PBDATDIR|0x4000;

        /* Set T3PWM low */
        SET_LO(MCRC,10);          // Select IOPF2
        SET_HI(PFDATDIR,10);// Output
        SET_LO(PFDATDIR,2);       // Pin low

        /* reset any faults */
        reset_PWM_fault();

        /* Set up PWM, the correct way: */
        SCSR1 |= 0x0004;
        T1PR = TIMER; /* Timer */
```

```
        DBTCONA = DBTCON_INIT_STATE;
        /* Setup the bridging IGBT gate driver polarities */
        ACTRA = COMPARE1_AL +
                        COMPARE2_AH +
                        COMPARE3_AL +
                        COMPARE4_AH +
                        COMPARE5_AL +
                        COMPARE6_AH;

        CMPR1 = 0; /* Phase A duty cycle */
        CMPR2 = 0; /* Phase B duty cycle */
        CMPR3 = 0; /* Phase C duty cycle */

        COMCONA=0xa200;
        T1CON = PWM_INIT_STATE;
        MCRA |= 0x0fc0;
        /* Setup done */
}

void reset_PWM_fault(void)
{
        int i;
        /* Clear faults on drivers */
        SET_LO(MCRC,3);          // Select IOPE3
        SET_HI(PEDATDIR,11);// Output
        SET_HI(PEDATDIR,3);      // Pin high
        SET_LO(PEDATDIR,3);      // Pin low

        /* Assert latch reset */
        SET_LO(MCRB,8);          // Select IOPD0
        SET_HI(PDDATDIR,8); // Output
        SET_LO(PDDATDIR,0);      // Pin low
        for(i=0;i<100;i++)
                asm("nop");              // Delay to allow the latch to reset
        SET_HI(PDDATDIR,0);      // Pin high
}

int lookup_sin(double arg)
{
        // TAB_RES entries for 2*PI radians.
        int index, normal_arg = 0;

        normal_arg = (int)((arg*ONEBYTWOPI)*TAB_RES);
        if(normal_arg >= TAB_RES)
                normal_arg = normal_arg - (TAB_RES - 1);
        else if(normal_arg < 0)
                normal_arg = TAB_RES + normal_arg;

        return sinetab[normal_arg];
}


void interrupt phantom()
{

}
```

Listing F.2: regs240x.h, TMS320LF2406A system registers

```
/* ========================================
```

```
//###########################################################################
//
// File name:        regs240x.h
//
// Originator:       Digital Control Systems Group
//                   Texas Instruments
//
// Description:  F240x register definitions.
//
// History:
//---------------------------------------------------------------------------
// 9-15-2000      Release Rev 1.0
//---------------------------------------------------------------------------*/

#ifndef __REGS240X_H__
#define __REGS240X_H__



#define IMR                 *((volatile int *)0x0004)
/*    Interrupt Mask Register                */
#define IFR                 *((volatile int *)0x0006)
/*    Interrupt Flag Register                */
#define SCSR1               *((volatile int *)0x7018)
/*    System Control &  Status Reg. 1        */
#define SCSR2               *((volatile int *)0x7019)
/*    System Control &  Status Reg. 2        */
#define DINR                *((volatile int *)0x701C)
/*    Device Identification Register.        */
#define PIVR                *((volatile int *)0x701E)
/*    Peripheral Interrupt Vector Reg.       */
#define PIRQR0              *((volatile int *)0x7010)
/*    Periph Interrupt Request Reg 0.        */
#define PIRQR1              *((volatile int *)0x7011)
/*    Periph Interrupt Request Reg 1.        */
#define PIRQR2              *((volatile int *)0x7012)
/*    Periph Interrupt Request Reg 2.        */
#define PIACKR0             *((volatile int *)0x7014)
/*    Periph Interrupt Acknowledge Reg 0. */
#define PIACKR1             *((volatile int *)0x7015)
/*    Periph Interrupt Acknowledge Reg 1. */
#define PIACKR2             *((volatile int *)0x7016)
/*    Periph Interrupt Acknowledge Reg 2. */
#define XINT1CR             *((volatile int *)0x7070)
/*    Ext. interrupt 1 config reg for X241*/
#define XINT2CR             *((volatile int *)0x7071)
/*    External interrupt 2 config. X241/2
/*/
#define MCRA                *((volatile int *)0x7090)
/*    Output Control Reg A                   */
#define OCRA                *((volatile int *)0x7090)
/*    Output Control Reg A                   */
#define MCRB                *((volatile int *)0x7092)
/*    Output Control Reg B                   */
#define OCRB                *((volatile int *)0x7092)
/*    Output Control Reg B                   */
#define MCRC                *((volatile int *)0x7094)
/*    Output Control Reg C                   */
#define ISRA                *((volatile int *)0x7094)
/*    Input Status Reg A x240x only          */
#define ISRB                *((volatile int *)0x7096)
/*    Input Status Reg B x240x only          */
#define PADATDIR            *((volatile int *)0x7098)
```

```
/*   I/O port A Data & Direction reg.    */
#define PBDATDIR            *((volatile int *)0x709A)
/*   I/O port B Data & Direction reg.    */
#define PCDATDIR            *((volatile int *)0x709C)
/*   I/O port C Data & Direction reg.    */
#define PDDATDIR            *((volatile int *)0x709E)
/*   I/O port D Data & Direction reg.    */
#define PEDATDIR            *((volatile int *)0x7095)
/*   I/O port E Data & Direction reg.    */
#define PFDATDIR            *((volatile int *)0x7096)
/*   I/O port F Data & Direction reg.    */
#define WDCNTR              *((volatile int *)0x7023)
/*   WD Counter reg                      */
#define WDKEY               *((volatile int *)0x7025)
/*   WD Key reg                          */
#define WDCR                *((volatile int *)0x7029)
/*   WD Control reg                      */
#define ADCTRL1             *((volatile int *)0x70A0)
/*   ADC Control Reg1                    */
#define ADCTRL2             *((volatile int *)0x70A1)
/*   ADC Control Reg2                    */
#define MAXCONV             *((volatile int *)0x70A2)
/*   Maximum conversion channels register*/
#define CHSELSEQ1           *((volatile int *)0x70A3)
/*   Channel select Sequencing control re*/
#define CHSELSEQ2           *((volatile int *)0x70A4)
/*   Channel select Sequencing control re*/
#define CHSELSEQ3           *((volatile int *)0x70A5)
/*   Channel select Sequencing control re*/
#define CHSELSEQ4           *((volatile int *)0x70A6)
/*   Channel select Sequencing control re*/
#define AUTO_SEQ_SR         *((volatile int *)0x70A7)
/*   Auto-sequence status register       */
#define RESULT0             *((volatile int *)0x70A8)
/*   Conversion result buffer register 0 */
#define RESULT1             *((volatile int *)0x70A9)
/*   Conversion result buffer register 1 */
#define RESULT2             *((volatile int *)0x70AA)
/*   Conversion result buffer register 2 */
#define RESULT3             *((volatile int *)0x70AB)
/*   Conversion result buffer register 3 */
#define RESULT4             *((volatile int *)0x70AC)
/*   Conversion result buffer register 4 */
#define RESULT5             *((volatile int *)0x70AD)
/*   Conversion result buffer register 5 */
#define RESULT6             *((volatile int *)0x70AE)
/*   Conversion result buffer register 6 */
#define RESULT7             *((volatile int *)0x70AF)
/*   Conversion result buffer register 7 */
#define RESULT8             *((volatile int *)0x70B0)
/*   Conversion result buffer register 8 */
#define RESULT9             *((volatile int *)0x70B1)
/*   Conversion result buffer register 9 */
#define RESULT10            *((volatile int *)0x70B2)
/*   Conversion result buffer register 10*/
#define RESULT11            *((volatile int *)0x70B3)
/*   Conversion result buffer register 11*/
#define RESULT12            *((volatile int *)0x70B4)
/*   Conversion result buffer register 12*/
#define RESULT13            *((volatile int *)0x70B5)
/*   Conversion result buffer register 13*/
```

```
#define RESULT14            *((volatile int *)0x70B6)
/*   Conversion result buffer register 14*/
#define RESULT15            *((volatile int *)0x70B7)
/*   Conversion result buffer register 15*/
#define CALIBRATION         *((volatile int *)0x70B8)
/*   Calib result, used to correct subseq*/
#define SPICCR              *((volatile int *)0x7040)
/*   SPI Config Control Reg                */
#define SPICTL              *((volatile int *)0x7041)
/*   SPI Operation Control Reg             */
#define SPISTS              *((volatile int *)0x7042)
/*   SPI Status Reg                        */
#define SPIBRR              *((volatile int *)0x7044)
/*   SPI Baud rate control reg             */
#define SPIRXEMU            *((volatile int *)0x7046)
/*   SPI Emulation buffer reg              */
#define SPIRXBUF            *((volatile int *)0x7047)
/*   SPI Serial receive buffer reg         */
#define SPITXBUF            *((volatile int *)0x7048)
/*   SPI Serial transmit buffer reg        */
#define SPIDAT              *((volatile int *)0x7049)
/*   SPI Serial data reg                   */
#define SPIPRI              *((volatile int *)0x704F)
/*   SPI Priority control reg              */
#define SCICCR              *((volatile int *)0x7050)
/*   SCI Communication control reg         */
#define SCICTL1             *((volatile int *)0x7051)
/*   SCI Control reg1                      */
#define SCIHBAUD            *((volatile int *)0x7052)
/*   SCI Baud Rate MSbyte reg              */
#define SCILBAUD            *((volatile int *)0x7053)
/*   SCI Baud Rate LSbyte reg              */
#define SCICTL2             *((volatile int *)0x7054)
/*   SCI Control reg2                      */
#define SCIRXST             *((volatile int *)0x7055)
/*   SCI Receiver Status reg               */
#define SCIRXEMU            *((volatile int *)0x7056)
/*   SCI Emulation Data Buffer reg         */
#define SCIRXBUF            *((volatile int *)0x7057)
/*   SCI Receiver Data buffer reg          */
#define SCITXBUF            *((volatile int *)0x7059)
/*   SCI Transmit Data buffer reg          */
#define SCIPRI              *((volatile int *)0x705F)
/*   SCI Priority control reg              */
#define GPTCONA             *((volatile int *)0x7400)
/*   GP Timer control register A .         */
#define T1CNT               *((volatile int *)0x7401)
/*   GP Timer 1 counter register.          */
#define T1CMPR              *((volatile int *)0x7402)
/*   GP Timer 1 compare register.          */
#define T1PR                *((volatile int *)0x7403)
/*   GP Timer 1 period register.           */
#define T1PER               *((volatile int *)0x7403)
/*   GP Timer 1 period register.           */
#define T1CON               *((volatile int *)0x7404)
/*   GP Timer 1 control register.          */
#define T2CNT               *((volatile int *)0x7405)
/*   GP Timer 2 counter register.          */
#define T2CMPR              *((volatile int *)0x7406)
/*   GP Timer 2 compare register.          */
#define T2PR                *((volatile int *)0x7407)
```

```
/*   GP Timer 2 period register.          */
#define T2PER                *((volatile int *)0x7407)
/*   GP Timer 2 period register.          */
#define T2CON                *((volatile int *)0x7408)
/*   GP Timer 2 control register.         */
#define COMCONA              *((volatile int *)0x7411)
/*   Compare control register A.          */
#define ACTRA                *((volatile int *)0x7413)
/*   Full compare action control register*/
#define DBTCONA              *((volatile int *)0x7415)
/*   Dead-band timer control register A. */
#define CMPR1                *((volatile int *)0x7417)
/*   Full compare unit compare register1 */
#define CMPR2                *((volatile int *)0x7418)
/*   Full compare unit compare register2 */
#define CMPR3                *((volatile int *)0x7419)
/*   Full compare unit compare register3 */
#define CAPCONA              *((volatile int *)0x7420)
/*   Capture control register A.          */
#define CAPFIFOA             *((volatile int *)0x7422)
/*   Capture FIFO status register A.      */
#define CAP1FIFO             *((volatile int *)0x7423)
/*   Capture Channel 1 FIFO Top           */
#define CAP2FIFO             *((volatile int *)0x7424)
/*   Capture Channel 2 FIFO Top           */
#define CAP3FIFO             *((volatile int *)0x7425)
/*   Capture Channel 3 FIFO Top           */
#define CAP1FBOT             *((volatile int *)0x7427)
/*   Bottom reg. pf capture FIFO stack 1 */
#define CAP2FBOT             *((volatile int *)0x7427)
/*   Bottom reg. pf capture FIFO stack 2 */
#define CAP3FBOT             *((volatile int *)0x7427)
/*   Bottom reg. pf capture FIFO stack 3 */
#define EVAIMRA              *((volatile int *)0x742C)
/*   Group A Interrupt Mask Register A    */
#define EVAIMRB              *((volatile int *)0x742D)
/*   Group B Interrupt Mask Register A    */
#define EVAIMRC              *((volatile int *)0x742E)
/*   Group C Interrupt Mask Register A    */
#define EVAIFRA              *((volatile int *)0x742F)
/*   Group A Interrupt Flag Register A    */
#define EVAIFRB              *((volatile int *)0x7430)
/*   Group B Interrupt Flag Register A    */
#define EVAIFRC              *((volatile int *)0x7431)
/*   Group C Interrupt Flag Register A    */
#define GPTCONB              *((volatile int *)0x7500)
/*   GP Timer control register B .        */
#define T3CNT                *((volatile int *)0x7501)
/*   GP Timer 3 counter register.         */
#define T3CMPR               *((volatile int *)0x7502)
/*   GP Timer 3 compare register.         */
#define T3PR                 *((volatile int *)0x7503)
/*   GP Timer 3 period register.          */
#define T3PER                *((volatile int *)0x7503)
/*   GP Timer 3 period register.          */
#define T3CON                *((volatile int *)0x7504)
/*   GP Timer 3 control register.         */
#define T4CNT                *((volatile int *)0x7505)
/*   GP Timer 4 counter register.         */
#define T4CMPR               *((volatile int *)0x7506)
/*   GP Timer 4 compare register.         */
```

```
#define T4PR                *((volatile int *)0x7507)
/*   GP Timer 4 period register.          */
#define T4PER               *((volatile int *)0x7507)
/*   GP Timer 4 period register.          */
#define T4CON               *((volatile int *)0x7508)
/*   GP Timer 4 control register.         */
#define COMCONB             *((volatile int *)0x7511)
/*   Compare control register B.          */
#define ACTRB               *((volatile int *)0x7513)
/*   Full compare action control register*/
#define DBTCONB             *((volatile int *)0x7515)
/*   Dead-band timer control register B. */
#define CMPR4               *((volatile int *)0x7517)
/*   Full compare unit compare register1 */
#define CMPR5               *((volatile int *)0x7518)
/*   Full compare unit compare register2 */
#define CMPR6               *((volatile int *)0x7519)
/*   Full compare unit compare register3 */
#define CAPCONB             *((volatile int *)0x7520)
/*   Capture control register B.          */
#define CAPFIFOB            *((volatile int *)0x7522)
/*   Capture FIFO status register B.      */
#define CAP4FIFO            *((volatile int *)0x7523)
/*   Capture Channel 1 FIFO Top B         */
#define CAP5FIFO            *((volatile int *)0x7524)
/*   Capture Channel 2 FIFO Top B         */
#define CAP6FIFO            *((volatile int *)0x7525)
/*   Capture Channel 3 FIFO Top B         */
#define CAP4FBOT            *((volatile int *)0x7527)
/*   Bottom reg. pf capture FIFO stack 1 */
#define CAP5FBOT            *((volatile int *)0x7527)
/*   Bottom reg. pf capture FIFO stack 2 */
#define CAP6FBOT            *((volatile int *)0x7527)
/*   Bottom reg. pf capture FIFO stack 3 */
#define EVBIMRA             *((volatile int *)0x752C)
/*   Group A Interrupt Mask Register B    */
#define EVBIMRB             *((volatile int *)0x752D)
/*   Group B Interrupt Mask Register B    */
#define EVBIMRC             *((volatile int *)0x752E)
/*   Group C Interrupt Mask Register B    */
#define EVBIFRA             *((volatile int *)0x752F)
/*   Group A Interrupt Flag Register B    */
#define EVBIFRB             *((volatile int *)0x7530)
/*   Group B Interrupt Flag Register B    */
#define EVBIFRC             *((volatile int *)0x7531)
/*   Group C Interrupt Flag Register B    */
#define CANMDER             *((volatile int *)0x7100)
/*   CAN Mailbox Direction/Enable reg     */
#define CANTCR              *((volatile int *)0x7101)
/*   CAN Transmission Control Reg         */
#define CANRCR              *((volatile int *)0x7102)
/*   CAN Recieve COntrol Reg              */
#define CANMCR              *((volatile int *)0x7103)
/*   CAN Master Control Reg               */
#define CANBCR2             *((volatile int *)0x7104)
/*   CAN Bit COnfig Reg 2                 */
#define CANBCR1             *((volatile int *)0x7105)
/*   CAN Bit Config Reg 1                 */
#define CANESR              *((volatile int *)0x7106)
/*   CAN Error Status Reg                 */
#define CANGSR              *((volatile int *)0x7107)
```

```
/*   CAN Global Status Reg                  */
#define CANCEC              *((volatile int *)0x7108)
/*   CAN Trans and Rcv Err counters         */
#define CANIFR              *((volatile int *)0x7109)
/*   CAN Interrupt Flag Registers           */
#define CANIMR              *((volatile int *)0x710a)
/*   CAN Interrupt Mask Registers           */
#define CANLAM0H            *((volatile int *)0x710b)
/*   CAN Local Acceptance Mask MBx0/1       */
#define CANLAM0L            *((volatile int *)0x710c)
/*   CAN Local Acceptance Mask MBx0/1       */
#define CANLAM1H            *((volatile int *)0x710d)
/*   CAN Local Acceptance Mask MBx2/3       */
#define CANLAM1L            *((volatile int *)0x710e)
/*   CAN Local Acceptance Mask MBx2/3       */
#define CANMSGID0L          *((volatile int *)0x7200)
/*   CAN Message ID for mailbox 0 (lower */
#define CANMSGID0H          *((volatile int *)0x7201)
/*   CAN Message ID for mailbox 0 (upper */
#define CANMSGCTRL0         *((volatile int *)0x7202)
/*   CAN RTR and DLC                        */
#define CANMBX0A            *((volatile int *)0x7204)
/*   CAN 2 of 8 bytes of Mailbox 0          */
#define CANMBX0B            *((volatile int *)0x7205)
/*   CAN 2 of 8 bytes of Mailbox 0          */
#define CANMBX0C            *((volatile int *)0x7206)
/*   CAN 2 of 8 bytes of Mailbox 0          */
#define CANMBX0D            *((volatile int *)0x7207)
/*   CAN 2 of 8 bytes of Mailbox 0          */
#define CANMSGID1L          *((volatile int *)0x7208)
/*   CAN Message ID for mailbox 1 (lower */
#define CANMSGID1H          *((volatile int *)0x7209)
/*   CAN Message ID for mailbox 1 (upper */
#define CANMSGCTRL1         *((volatile int *)0x720A)
/*   CAN RTR and DLC                        */
#define CANMBX1A            *((volatile int *)0x720C)
/*   CAN 2 of 8 bytes of Mailbox 1          */
#define CANMBX1B            *((volatile int *)0x720D)
/*   CAN 2 of 8 bytes of Mailbox 1          */
#define CANMBX1C            *((volatile int *)0x720E)
/*   CAN 2 of 8 bytes of Mailbox 1          */
#define CANMBX1D            *((volatile int *)0x720F)
/*   CAN 2 of 8 bytes of Mailbox 1          */
#define CANMSGID2L          *((volatile int *)0x7210)
/*   CAN Message ID for mailbox 2 (lower */
#define CANMSGID2H          *((volatile int *)0x7211)
/*   CAN Message ID for mailbox 2 (upper */
#define CANMSGCTRL2         *((volatile int *)0x7212)
/*   CAN RTR and DLC                        */
#define CANMBX2A            *((volatile int *)0x7214)
/*   CAN 2 of 8 bytes of Mailbox 2          */
#define CANMBX2B            *((volatile int *)0x7215)
/*   CAN 2 of 8 bytes of Mailbox 2          */
#define CANMBX2C            *((volatile int *)0x7216)
/*   CAN 2 of 8 bytes of Mailbox 2          */
#define CANMBX2D            *((volatile int *)0x7217)
/*   CAN 2 of 8 bytes of Mailbox 2          */
#define CANMSGID3L          *((volatile int *)0x7218)
/*   CAN Message ID for mailbox 3 (lower */
#define CANMSGID3H          *((volatile int *)0x7219)
/*   CAN Message ID for mailbox 3 (upper */
```

```
#define CANMSGCTRL3         *((volatile int *)0x721A)
/*   CAN RTR and DLC                          */
#define CANMBX3A            *((volatile int *)0x721C)
/*   CAN 2 of 8 bytes of Mailbox 3            */
#define CANMBX3B            *((volatile int *)0x721D)
/*   CAN 2 of 8 bytes of Mailbox 3            */
#define CANMBX3C            *((volatile int *)0x721E)
/*   CAN 2 of 8 bytes of Mailbox 3            */
#define CANMBX3D            *((volatile int *)0x721F)
/*   CAN 2 of 8 bytes of Mailbox 3            */
#define CANMSGID4L          *((volatile int *)0x7220)
/*   CAN Message ID for mailbox 4 (lower */
#define CANMSGID4H          *((volatile int *)0x7221)
/*   CAN Message ID for mailbox 4 (upper */
#define CANMSGCTRL4         *((volatile int *)0x7222)
/*   CAN RTR and DLC                          */
#define CANMBX4A            *((volatile int *)0x7224)
/*   CAN 2 of 8 bytes of Mailbox 4            */
#define CANMBX4B            *((volatile int *)0x7225)
/*   CAN 2 of 8 bytes of Mailbox 4            */
#define CANMBX4C            *((volatile int *)0x7226)
/*   CAN 2 of 8 bytes of Mailbox 4            */
#define CANMBX4D            *((volatile int *)0x7227)
/*   CAN 2 of 8 bytes of Mailbox 4            */
#define CANMSGID5L          *((volatile int *)0x7228)
/*   CAN Message ID for mailbox 5 (lower */
#define CANMSGID5H          *((volatile int *)0x7229)
/*   CAN Message ID for mailbox 5 (upper */
#define CANMSGCTRL5         *((volatile int *)0x722A)
/*   CAN RTR and DLC                          */
#define CANMBX5A            *((volatile int *)0x722C)
/*   CAN 2 of 8 bytes of Mailbox 5            */
#define CANMBX5B            *((volatile int *)0x722D)
/*   CAN 2 of 8 bytes of Mailbox 5            */
#define CANMBX5C            *((volatile int *)0x722E)
/*   CAN 2 of 8 bytes of Mailbox 5            */
#define CANMBX5D            *((volatile int *)0x722F)
/*   CAN 2 of 8 bytes of Mailbox 5            */
/*-------------------------------------------------*/
/* I/O space mapped registers                      */
/*-------------------------------------------------*/
#define WSGR    portfffff
ioport unsigned portfffff;      /* Wait-State Generator Control Reg */

#define FCMR    portff0f        /* Flash mode control register */
ioport unsigned portff0f;
ioport unsigned port0;
ioport unsigned port1;
ioport unsigned port2;
ioport unsigned port3;
ioport unsigned port4;

#define DAC0    port0
#define DAC1    port1
#define DAC2    port2
#define DAC3    port3
#define DACL    port4

#endif /*__REGS240X_H__  */
```

Listing F.3: pwm/include/F2407pwm.h, PWM initialization structures

```
/* ============================================================
File name:          F2407PWM.H

Originator:       Digital Control Systems Group
                        Texas Instruments
Description:
Header file containing data type and object definitions and
initializers. Also contains prototypes for the functions in F2407_PWM?.C.
============================================================
History:
------------------------------------------------------------
9-15-2000       Release Rev 1.0
------------------------------------------------------------ */

#ifndef __F2407_PWM_H__
#define __F2407_PWM_H__
#include "../include/F2407BMSK.h"

/*-------------------------------------------------------------
Initialization constant for the F2407 Timer TxCON for PWM Generation.
Sets up the timer to run free upon emulation suspend, continuous up-down mode
prescaler 1, timer enabled.
-------------------------------------------------------------*/
#define PWM_INIT_STATE   (FREE_RUN_FLAG +          \
                          TIMER_CONT_UPDN +        \
                          TIMER_CLK_PRESCALE_X_1 + \
                          TIMER_ENABLE_BY_OWN    + \
                          TIMER_ENABLE)
/*-------------------------------------------------------------
Initialization constant for the F2407 ACTRx register for PWM Generation.
Sets up PWM polarities.
-------------------------------------------------------------*/
#define ACTR_INIT_STATE ( COMPARE1_AH + \
                          COMPARE2_AL + \
                          COMPARE3_AH + \
                          COMPARE4_AL + \
                          COMPARE5_AH + \
                          COMPARE6_AL )

/*-------------------------------------------------------------
Initialization constant for the F2407 DBTCONx register for PWM Generation.
Sets up the dead band for PWM and sets up dead band values.
-------------------------------------------------------------*/
#define DBTCON_INIT_STATE ( DBT_VAL_10 +  \
                            EDBT3_EN   +  \
                            EDBT2_EN   +  \
                            EDBT1_EN   +  \
                            DBTPS_X4 )
/*-------------------------------------------------------------
Define the structure of the PWM Driver Object
-------------------------------------------------------------*/
typedef struct {

        int period_max;         /* PWM Period in CPU clock cycles.  Q0-Input  */
        int mfunc_p;            /* Period scaler. Q15 - Input                 */
        int mfunc_c1;           /* PWM 1&2 Duty cycle ratio. Q15, Input       */
        int mfunc_c2;           /* PWM 3&4 Duty cycle ratio. Q15, Input       */
        int mfunc_c3;           /* PWM 5&6 Duty cycle ratio. Q15, Input       */
        int (*init)();          /* Pointer to the init function              */
```

```
        int (*update)();        /* Pointer to the update function          */
        } PWMGEN ;

/*-----------------------------------------------------------------------
Define a PWMGEN_handle
-----------------------------------------------------------------------*/
typedef PWMGEN *PWMGEN_handle;



/*-----------------------------------------------------------------------
Default Initializers for the F2407 PWMGEN Object
-----------------------------------------------------------------------*/
#define F2407_EV1_FC_PWM_GEN {1000,    \
                              0x7fff,  \
                              0x4000,  \
                              0x4000,  \
                              0x4000,  \
                             (int (*)(int))F2407_EV1_PWM_Init,   \
                             (int (*)(int))F2407_EV1_PWM_Update \
                             }

#define F2407_EV2_FC_PWM_GEN {1000,    \
                              0x7fff,  \
                              0x4000,  \
                              0x4000,  \
                              0x4000,  \
                             (int (*)(int))F2407_EV2_PWM_Init,   \
                             (int (*)(int))F2407_EV2_PWM_Update \
                             }

#define PWMGEN_DEFAULTS F2407_EV1_FC_PWM_GEN

/*-----------------------------------------------------------------------
 Prototypes for the functions in F2407_PWM1.C , F2407_PWM2.ASM
-----------------------------------------------------------------------*/
int F2407_EV1_PWM_Init(PWMGEN *);
int F2407_EV1_PWM_Update(PWMGEN *);
int F2407_EV2_PWM_Init(PWMGEN *);
int F2407_EV2_PWM_Update(PWMGEN *);

#endif  /*__F2407_PWM_H__*/
```

Listing F.4: sysvecs.h, DSP interrupt vectors and ISR calls

```
/******** RESET AND INTERRUPT VECTORS ****************/
/* NOTE: Though this file will support multiple inclusion in the same
         source file, it must NOT be included in multiple separately linked source
         files. This will cause the linker to generate an error.
         This file is intended for only the framework file to include.
*/


/*-----------------------------------------------------------------------*/
#ifndef __SYSVECS_H__
#define __SYSVECS_H__

/*
```

```
RSVECT B START ; Reset Vector");
INT1 B PHANTOM ; Interrupt Level 1");
INT2 B SINE ; Interrupt Level 2");
INT3 B PHANTOM ; Interrupt Level 3");
INT4 B PHANTOM ; Interrupt Level 4");
INT5 B PHANTOM ; Interrupt Level 5");
INT6 B PHANTOM ; Interrupt Level 6");
RESERVED B PHANTOM ; Reserved");
SW_INT8 B PHANTOM ; User S/W Interrupt");
SW_INT9 B PHANTOM ; User S/W Interrupt");
SW_INT10 B PHANTOM ; User S/W Interrupt");
SW_INT11 B PHANTOM ; User S/W Interrupt");
SW_INT12 B PHANTOM ; User S/W Interrupt");
SW_INT13 B PHANTOM ; User S/W Interrupt");
SW_INT14 B PHANTOM ; User S/W Interrupt");
SW_INT15 B PHANTOM ; User S/W Interrupt");
SW_INT16 B PHANTOM ; User S/W Interrupt");
TRAP B PHANTOM ; Trap vector");
NMINT B PHANTOM ; Non-maskable Interrupt");
EMU_TRAP B PHANTOM ; Emulator Trap");
SW_INT20 B PHANTOM ; User S/W Interrupt");
SW_INT21 B PHANTOM ; User S/W Interrupt");
SW_INT22 B PHANTOM ; User S/W Interrupt");
SW_INT23 B PHANTOM ; User S/W Interrupt"); */


asm("              .sect    \"vectors\"                          ");
asm("              .ref     _c_int0                              ");
asm("              B        _c_int0          ;00h reset          ");
asm("              B        _phantom         ;02h INT1           ");
asm("              B        _periodic_interrupt_isr       ;04h INT2          ");
asm("              B        _phantom         ;06h INT3           ");
asm("              B        _phantom         ;08h INT4           ");
asm("              B        _phantom         ;0Ah INT5           ");
asm("              B        _phantom         ;0Ch INT6           ");
asm("              B        _phantom          ; 0E               ");
asm("              B        _phantom          ; 10               ");
asm("HUNG12        B        _phantom          ; 12               ");
asm("HUNG14        B        _phantom          ; 14               ");
asm("HUNG16        B        _phantom          ; 16               ");
asm("HUNG18        B        _phantom          ; 18               ");
asm("HUNG1A        B        _phantom          ; 1A               ");
asm("HUNG1C        B        _phantom          ; 1C               ");
asm("HUNG1E        B        _phantom          ; 1E               ");
asm("HUNG20        B        _phantom          ; 20               ");
asm("TRAP          B        _phantom          ; 22               ");
asm("NMI           B        _phantom          ; 24               ");
asm("MON_ETRAP_B   B        _phantom          ; 26               ");
asm("HUNG28        B        _phantom          ; 28               ");
asm("HUNG2A        B        _phantom          ; 2A               ");
asm("HUNG2C        B        _phantom          ; 2C               ");
asm("HUNG2E        B        _phantom          ; 2E               ");
asm("HUNG30        B      _phantom          ; 30                 ");
asm("HUNG32        B      _phantom          ; 32                 ");
asm("HUNG34        B      _phantom          ; 34                 ");
asm("HUNG36        B      _phantom          ; 36                 ");
asm("HUNG38        B      _phantom          ; 38                 ");
asm("HUNG3A        B      _phantom          ; 3A                 ");
asm("HUNG3C        B      _phantom          ; 3C                 ");
asm("HUNG3E        B      _phantom          ; 3E                 ");
```

```
#endif  /* __SYSVECS_H__ */
```

Listing F.5: sine.h, sine table lookup routine header

```
// Jack Holloway, Sine table

#define TAB_RES 200

int lookup_sin(double);

int sinetab[]={
0, 1029, 2057, 3083, 4106, 5126,
6140, 7148, 8149, 9141, 10125, 11099,
12062, 13013, 13951, 14876, 15786, 16680,
17557, 18418, 19260, 20083, 20887, 21669,
22431, 23170, 23886, 24579, 25248, 25891,
26509, 27101, 27666, 28204, 28714, 29196,
29649, 30072, 30466, 30830, 31164, 31466,
31738, 31978, 32187, 32364, 32509, 32622,
32703, 32751, 32767, 32751, 32703, 32622,
32509, 32364, 32187, 31978, 31738, 31466,
31164, 30830, 30466, 30072, 29649, 29196,
28714, 28204, 27666, 27101, 26509, 25891,
25248, 24579, 23886, 23170, 22431, 21669,
20887, 20083, 19260, 18418, 17557, 16680,
15786, 14876, 13951, 13013, 12062, 11099,
10125, 9141, 8149, 7148, 6140, 5126,
4106, 3083, 2057, 1029, 0, -1029,
-2057, -3083, -4106, -5126, -6140, -7148,
-8149, -9141, -10125, -11099, -12062, -13013,
-13951, -14876, -15786, -16680, -17557, -18418,
-19260, -20083, -20887, -21669, -22431, -23170,
-23886, -24579, -25248, -25891, -26509, -27101,
-27666, -28204, -28714, -29196, -29649, -30072,
-30466, -30830, -31164, -31466, -31738, -31978,
-32187, -32364, -32509, -32622, -32703, -32751,
-32767, -32751, -32703, -32622, -32509, -32364,
-32187, -31978, -31738, -31466, -31164, -30830,
-30466, -30072, -29649, -29196, -28714, -28204,
-27666, -27101, -26509, -25891, -25248, -24579,
-23886, -23170, -22431, -21669, -20887, -20083,
-19260, -18418, -17557, -16680, -15786, -14876,
-13951, -13013, -12062, -11099, -10125, -9141,
-8149, -7148, -6140, -5126, -4106, -3083,
-2057, -1029};
```

Listing F.6 contains a set of commands instructing the TI linker on where to place various portion of memory on the DSP physical memory space. For more information on the TI TMS320LF2406A DSP memory layout and architecture, please see [7,8].

Listing F.6: 2406A linker commands

```
/*************************************************************************/
/* linker command file to place user code (vectors & .text)              */
/* sections beginning at 0000h of external program memory (in MP mode).  */
/* This file should be modified if it is desired to load code in B0 memory or */
```

```
/* if on-chip SARAM is to be used.; This example file is applicable for LF2406A.*/
/* It needs to be modified to make it suitable for other devices.              */
/* Jack Holloway                                                               */
/*****************************************************************************/

MEMORY
{

PAGE 0:                              /* PROGRAM MEMORY                  */

PM1       :ORIGIN=0000h    ,  LENGTH=0040h  /* PM up to passwords */
PM2       :ORIGIN=0044H    ,  LENGTH=07FBCH /* 32K On-chip flash memory     */
SARAM_P   :ORIGIN=08000H   ,  LENGTH=0800H  /* 2K SARAM in program space    */
B0_PM     :ORIGIN=0FF00h   ,  LENGTH=0100h  /* On-chip DARAM if CNF=1, else */

PAGE 1:                                        /* DATA MEMORY                 */
REGS      :ORIGIN=0h       ,  LENGTH=60h    /* Memory mapped regs & reserved address */
BLK_B2    :ORIGIN=60h      ,  LENGTH=20h    /* Block B2                    */
BLK_B0    :ORIGIN=200h     ,  LENGTH=100h   /* Block B0, On-chip DARAM if CNF=0 */
BLK_B1    :ORIGIN=300h     ,  LENGTH=100h   /* Block B1                    */
SARAM_D   :ORIGIN=0800h    ,  LENGTH=0800h  /* 2K SARAM in data space      */
PERIPH    :ORIGIN=7000h    ,  LENGTH=1000h  /* Peripheral register space   */

PAGE 2:                              /* I/O MEMORY                      */

IO_IN     :ORIGIN=0FF0Fh   ,  LENGTH=01h    /* On-chip I/O mapped peripherals */
/* this is just one register for Flash control */
}

SECTIONS

{

        vectors :{}     >       PM1             PAGE 0
        .cinit  :{}     >       PM2             PAGE 0
        .text   :{}     >       PM2             PAGE 0
        .switch :{}     >       PM2             PAGE 0
        .data   :{}     >       BLK_B1          PAGE 1
        .sysmem :{}     >       SARAM_D         PAGE 1
        .bss    :{}     >       BLK_B0          PAGE 1
        .stack  :{}     >       SARAM_D         PAGE 1
        .const  :{}     >       BLK_B2          PAGE 1
        .const  :{}     >       PM2             PAGE 0

}
```

The header file in Listing F.5 was created using the C-code of Listing F.7.

Listing F.7: sine.c, C-code to generate the sine table

```
// this guy produces the sine table for use on the TI DSP.
// Jack Holloway, February 2004.


#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#define PI 3.1415927
#define RESOLUTION 200
int main(void)
```

```
{
  int j, i;
  int sinevalue;
  FILE *out;
  double arg, A;

  out = fopen("./sine.h", "w");
  j = 0;
  fprintf(out, "// Jack Holloway, Sine table\n\n");
  fprintf(out, "#define TAB_RES %i\n\n",RESOLUTION);
  fprintf(out, "int sin(double);\n\nint sinetab[]={\n");
  for(i = 0; i< RESOLUTION; i++)
    {
      arg = (double)(2.0*PI*i/RESOLUTION);
      A = 1.0;
      sinevalue = (int)((32768)*sin(arg));
      fprintf(out, "%i, ", sinevalue);
      if(j++>4)
        {
          fprintf(out, "\n");
          j = 0;
        }

    }
  fprintf(out, "};\n");

  return 0;
}
```

F.2 Object File Password Parsing

The TMS320LF2406A DSP has the ability to password-protect a piece of firmware after it has been programmed onto the DSP Flash memory. Once this password has been set, it can only be reset using the current password. There is no way to reset the code protection except with the use of this password. [7]

Listing F.8 contains C-code for parsing the object file to be written into DSP Flash. It is essential that these files be checked to avoid making the DSP[1] virtually useless.

If the code in Listing F.8 has been compiled in a binary, `hexpw`, the `.HEX` to be programmed into the DSP flash memory should be checked by executing `hexpw FILENAME.HEX`. If no password is found in the `.HEX` file, `hexpw` will print `No password`. After checking the file for passwords, the file can be uploaded using the TI tools. If an unintentional password *is* found by `hexpw`, *do not* program the DSP using this `.HEX` file.

The code listied in F.8 was written by Jim Paris.

[1] and thus the entire PIIPM

Listing F.8: hexpw.c, checks object files for Flash memory passwords

```
#include <stdio.h>
#include <stdint.h>

int main(int argc, char *argv[])
{
        char s[64];
        uint16_t b[100000];
        int i, c, n;
        int len, offset, entry, prog, data, addr;
10      uint16_t p[4]={0,0,0,0};
        FILE *in;

        if(argc!=2) {
                fprintf(stderr,"usage: %s file.hex\n",*argv);
                return 1;
        }
        if((in=fopen(argv[1],"r"))==NULL) {
                fprintf(stderr,"can't open %s\n",argv[1]);
                return 1;
20      }

        n=0;
        while(fgets(s,64,in)!=NULL) {
                if(strlen(s)!=5 && strlen(s)!=6) continue;
                s[4]=0;
                b[n++] = strtoul(s,NULL,16);
        }

        c=0;
30      entry = b[c++];
        while(c<n) {
                len = b[c++];
                if(len==0) {
                        if(c!=n) printf("Hit early EOF?\n");
                        continue;
                }
                offset = b[c++];
                prog = b[c++];

40              for(i=0;i<len;i++) {
                        data = b[c++];
                        addr = offset + i;
                        if(prog==0 && addr>=0x40 && addr <=0x43)
                                p[addr-0x40] = data;
                }
        }
        if((p[0]==0 && p[1]==0 && p[2]==0 && p[3]==0) ||
           (p[0]==0xffff && p[1]==0xffff && p[2]==0xffff && p[3]==0xffff)) {
                printf("No password\n");
50              return 0;
        } else {
                printf("Password is set: %04x %04x %04x %04x\n",
                       p[0],p[1],p[2],p[3]);
                return 1;
        }
}
```

Bibliography

[1] M.I.T. E.E. Staff. *Applied Electronics.* John Wiley & Sons, Inc., New York, New York, 1944.

[2] Robert W. Erickson and Dragan Maksimović. *Fundamentals of Power ELectronics.* Kluwer Academic Publishers, Norwell, Massachusetts, second edition, 2001.

[3] Arthur E. Fitzgerald, Charles Kingsley, Jr., and Stephen D. Umans. *Electric Machinery.* MaGraw-Hill Series in Electrical Engineering. McGraw-Hill, New York, New York, fifth edition, 1990.

[4] Davide Giacomini, Enrico Bianconi, Luca Martino, and Marco Palma. *A New Fully Integrated Power Module for Three Phase Servo Motor Drive applications.* International Rectifier Corporation, Venaria, TO Italy, 2003.

[5] Ernst A Guillemin. *Introductory Circuit Theory.* John Wiley & Sons, Inc., New York, New York, 1953.

[6] Ernst A Guillemin. *Synthesis of Passive Networks.* John Wiley & Sons, Inc., New York, New York, 1957.

[7] Texas Instruments. *TMS320LF2406A DSP Controllers Datasheet.* TI, Houston, Texas, 2000.

[8] Texas Instruments. *TMS320LF/LC240xA DSP Controllers Reference Guide.* TI, Houston, Texas, December 2001.

[9] John G. Kassakian, Martin F. Schlect, and George C. Verghese. *Principles of Power Electronics.* Addison-Wesley, Reading, Massachusetts, 1992.

[10] James L. Kirtley, Jr. Massachusetts instiute of technology, course 6.685 electric machines, lectures. electronic documents, lectures, 2003.

[11] James L. Kirtley, Jr. and H. W. Beaty. *Electric Motor Handbook.* MaGraw-Hill Handbooks. McGraw-Hill, New York, New York, 1998.

[12] Ramu Krishnan. *Electric Motor Drives, Modeling, Analysis, and Control.* Prentice Hall, Upper Saddle River, New Jersey, 2001.

[13] Alexander S. Langsdorf. *Theory of Alternating-Current Machinery.* Electrical Engineering Texts. McGraw-Hill Book Company, Inc., New York, New York, first edition, 1937.

[14] Kwangduk D. Lee, Steven B. Leeb, Leslie K. Norford, Peter Armstrong, Jack Holloway, and Steven R. Shaw. Estimation of variable speed drive power consumption from harmonic content. *IEEE Transactions of Energy Conversion*, 2004. in review.

[15] Thomas H. Lee. *The Design of CMOS Radio-Frequency Integrated Circuits.* Cambridge University Press, New York, New York, first edition, 1998.

[16] Michael Liwschitz-Garik. *Electric Machinery*, volume II of *A-C Machines.* D. Van Nostrand Company, Inc., New York, New York, fifth edition, April 1950.

[17] B&B Electronics Ltd. *9-Pin RS-232 to RS-485 Optically Isolated Converter Model 485OT9L datasheet.* B&B Electronics Ltd, Ottawa, Illinois, November 2000.

[18] Carver Mead. *Analog VLSI and Neural Systems.* Addison-Wesley, New York, New York, 1989.

[19] J. H. Morecroft. *Continuous and Alternating Current Machinery.* The Wiley Technical Series. John Wiley & Sons, Inc., New York, New York, first edition, 1915.

[20] International Rectifier. *PIIPM15P12D007 Programmable Isolated IPM.* IRF, El Segundo, California, September 2003.

[21] Herbert J. Reich. *Thoery and Applications of Electron Tubes.* McGraw-Hill Book Company, Inc., New York, New York, 1939.

[22] James K. Roberge. *Operational Amplifiers: Theory and Practice.* John Wiley and Sons, New York, New York, 1975.

[23] William McC. Siebert. *Circuits, Signals, and Systems.* MIT Press, McGraw-Hill, Cambridge, Massacusetts, 1986.

[24] David H. Staelin, Ann W. Morgenthaler, and Jin A. Kong. *Electromagnetic Waves.* Prentice Hall, Upper Saddle River, New Jersey, 1998.

[25] George J. Thaler and Marvin P. Pastel. *Analysis and Design of Nonlinear Feedback Control Systems.* McGraw-Hill, New York, New York, 1962.

[26] Riad S. Wahby. Radio frequency rectifiers for dc-dc power conversion. Master of enginerring, Massachusetts Institute of Technology, Electrical Engineering and Computer Science, 2004.

[27] Markus Zahn. *Electromagnetic Field Theory, a problem solving approach.* Krieger Publishing Company, Malabar, Florida, 2003.