SINGLE-PHASE, SINGLE-SWITCH, SENSORLESS SWITCHED RELUCTANCE MOTOR DRIVE UTILIZING A MINIMAL ARTIFICIAL NEURAL NET

Christopher A. Hudson

THESIS SUBMITTED TO THE FACULTY OF
VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE
IN
ELECTRICAL ENGINEERING

KRISHNAN RAMU, CHAIR
WILLIAM BAUMANN, MEMBER
PUSHKIN KACHROO, MEMBER

AUGUST 23, 2005
BLACKSBURG, VIRGINIA 24061

KEYWORDS: CONTROLS, SRM, MOTORS, MOTOR DRIVES, NEURAL NETWORK, FLUX LINKAGE, ESTIMATION

SINGLE-PHASE, SINGLE-SWITCH, SENSORLESS SWITCHED RELUCTANCE MOTOR DRIVE UTILIZING A MINIMAL ARTIFICIAL NEURAL NET

Christopher A. Hudson

(ABSTRACT)

Artificial Neural Networks (ANNs) have proved to be useful in approximating non-linear systems in many applications including motion control. ANNs advocated in switched reluctance motor (SRM) control typically have a large number of neurons and several layers which impedes their real time implementation in embedded systems. Real time estimation at high speeds using these ANNs is difficult due to the high number of operations required to process the ANN controller. An insufficient availability of time between two sampling intervals limits the available computation time for both processing the neural net and the other functions required for the motor drive. One ideal application of ANNs in SRM control is rotor position estimation. Due to reliability issues, elimination of the rotor position sensors is absolutely required for high volume, high speed and low cost applications of SRM's. ANNs provide a means by which drive designers can implement position sensorless drive technology that is both robust and easily implemented.

It is demonstrated that a new and novel ANN configuration can be implemented for accurate rotor position estimation in a sensorless SRM drive. Consisting of just 4 neurons, the neural estimator is the smallest of its kind for SRM rotor position estimation. The breakthrough that provided the reduction was the addition of a non-linear input. Typical input spaces for SRM position neural estimators consist of both current (i),and flux-linkage (λ). The neural network was trained off-line using these inputs and a third, non-linear input provided by a preprocessed product of the two typical inputs.

Acknowledgements

I would like to thank Krishnan Ramu for his guidance, patience, and instruction. I gained a great deal of understanding in the subjects of analysis, design, simulation, and controls. He demonstrated time and again his mastery in his knowledge of motor design, control, and simulation of drives. I hope to achieve as much in my career.

Professor William T. Baumann who provided me the opportunity to prove myself a worthy student by admitting me into the master's degree program at Virginia Tech. While giving me excellent instruction, he also provided sound advice.

I am thankful for Mr. Lon Whelchel, CEO and President, Panaphase Technologies, LLC., for their permission to use their proprietary converter and machine to develop the control algorithms presented in this thesis.

Of course, I am grateful to my wife for her patience. To my boys, who gave me comic relief from the work effort. Without them this work would have never come into existence.

Blacksburg, Virginia — Christopher A. Hudson
August 23, 2005

To my wife Robyn and my boys, Jack and Graham, without their support, laughter, and dedication, none of my achievements would have been possible.

Table of Contents

List of Figures

List of Tables

Introduction

In recent years, Artificial Neural Networks (ANN) have found a place in real-time control and system estimation applications such as motor control and rotor estimation. This is due to the increase in microprocessor speed and capabilities. Recently trends in control of Switched Reluctance Motors (SRM) have shown that neural nets have provided the ability to be used in the removal of position sensors. This is attractive to engineers as well as customers because position sensors increase the overall cost and reliability of the SRM drive. Many sensorless (position) algorithms have been proposed and validated for control of SRM's. However, implementing ANN's for speed controlled, sensorless SRM drives has been lacking partly due of the inherent non-linearity of the SRM and its drive. Development of a simple and accurate ANN's for sensorless control has been difficult. Highly non-linear systems, such as SRM motor drives, require a high degree of accuracy for proper control. Because SRM drive systems are inherently difficult to analytically model, ANN provide a useful means of system modelling and control implementation.

Previous to the current research, high-order ANN structures have been required to properly approximate SRM system dynamics accurately enough for design and control development. Techniques for rotor position estimation and rotor position sensorless torque control in SRM's have been proposed and implemented at low speeds [25]

using these ANN structures. This paper attempts to present a novel technique for sensorless control of a single switch based converter driven SRM using an extremely minimal order ANN.

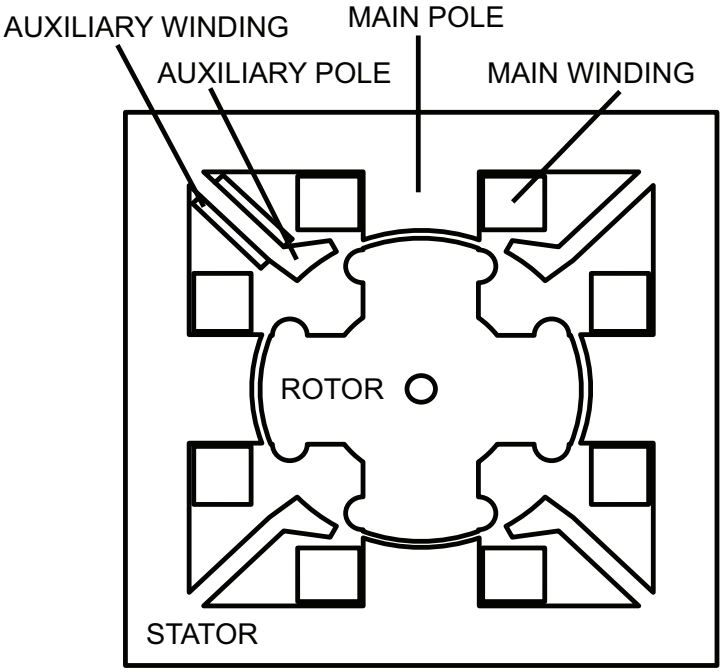

Figure 1: SRM Machine Cross-Section

The SRM drive system herein consists of the SRM shown in Figure1 which is discussed in [48], and the novel converter drive shown in Figure 2 which is also discussed in [48, 12, 26]. The converter uses only one controllable switch. It has been demonstrated that this converter driving a single phase SRM can provide four quadrant operation [13]. This system has no previous attempt to produce a position sensorless design.

It is believed that this SRM and single switch converter drive is the lowest cost SRM system. Therefore it is extremely attractive to demonstrate the proposed sensorless controller in order to achieve the absolute minimal system with the highest

reliability and control. However, the drive system is intended for low cost, low performance motor drives applications.

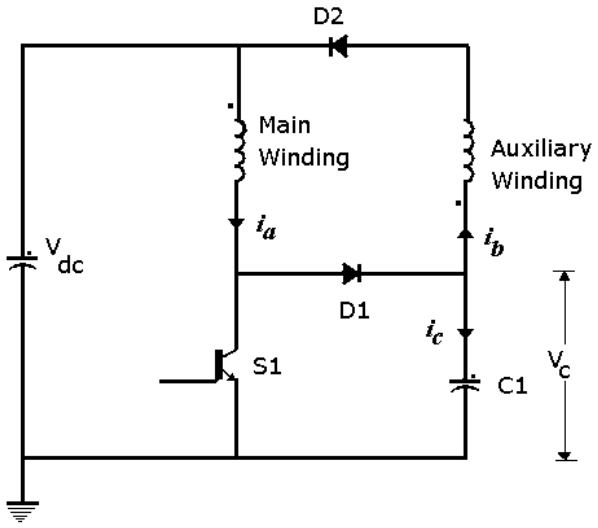

Figure 2: Simplified Converter Schematic

The proposed approach in this paper is to derive an absolutely minimal neural network topology that lends itself to sensorless switched reluctance rotor position estimation. A sensorless control algorithm is presented in conjunction with the ANN. This shows that the ANN is viable in real-time control of SRM drive with inexpensive Digital Signal Processors (DSP). Motor control is provided by the DSP, which provides the means to implement flux linkage estimation, linear current control, linear speed control, and said sensorless commutation algorithm. Finally, the experiment shows that closed-loop control with above system is possible for low performance motor drives.

Previous literature as in [15] and [35] suggests that the main stumbling block of SRM sensorless methods is the generation of the flux linkage estimate. Most researchers find it difficult to utilize flux linkage in sensorless algorithms. This is due

to a non-measurable quantity, flux linkage, that can only be derived from analytical techniques and other observation methods. Still, most previous sensorless techniques utilize flux linkage because of its fundamental magnetic relationship in the SRM. This relationship defines that if current and flux linkage in any phase is known, the rotor position can be derived. This fundamental relationship is the basis for sensorless position estimation methods.

With regards to ANN's, researchers have attempted to utilize the flux linkage and current as inputs while training the output to compute rotor position as in [34]. To achieve accurate results, the ANN's have contained several layers and many neurons. Processors have been weighed down by the computational effort required by the large number of neurons in hidden layers. It is not uncommon to encounter 15 to 20 neurons in each of the hidden layers. **The proposed approach in this paper is to significantly decrease the computational effort of the ANN by inserting a one step pre-processor in the input space of the ANN.** The ANN is intended to output rotor position for a given set of inputs comprised of flux linkage, current, and the pre-processed third input. This pre-process is a non-linear input such as the product of the phase current (i), and flux-linkage (λ). It is shown in later sections that the pre-processor provides the ability to reduce the number of total neurons in the ANN without sacrificing accuracy. Consequently, the computational burden for the ANN decreased while still achieving a sensorless SRM drive that fits the application intention.

Chapter 1

The Intent of the Research

This chapter introduces the concept of artificial neural nets and their use in control applications, specifically SRM rotor position estimation. The fundamental premise from which the research intent stems will be explored. A simplified proof that the intended method used to reduce the neural architecture works.

1.1 Introduction

Artificial Neural Nets (ANN) have been explored extensively in past decades for solving many complex problems including voice and face recognition, data mining, cellular traffic, and motor drives. Various implementations, ranging from torque ripple reduction to position sensorless estimation, have been used in SRM motor drives. The present research involves the use of ANN's to estimate rotor position from flux linkage and current. However, the main goal of the present research is not to develop an ordinary ANN sensorless implementation, but is to derive an extremely minimal neural structure by which said sensorless problem can still be solved. It will be shown that the use of pre-processing can provide a means to eliminate the need for hidden layers within the neural structure. By the layer removal method, the size

of the ANN and the required computational effort will be significantly reduced. Not only will the reduction prove to be readily implementable, but as accurate as other ANN implementations. The reduction in size provides the ability to implement more control functions by the embedded hardware. This is the result from the elimination of the neural calculation burden.

1.2 Research Proposal

Can a single neuron, with a simple pre-process, map an input space to an output that normally requires two neurons without losing accuracy? Further, can a greatly reduced neural net accomplish SRM sensorless rotor position estimation that would typically require a much larger network for control viability?

1.3 Neural Network Background

The fundamental design of an artificial neural network is to approximate the cerebral synaptic function of the human brain. Neurons are highly interconnected by synapses. The neural cell congregates all these synaptic signals and either fires an excitatory pulse or a inhibitory pulse in response to its aggregate input.[16]. In other words, the neuron's output activates to be on-or-off based on the "decision" it makes from the information it receives from all of it's inputs. The inputs, which are on or off electrical signals from other neurons, are delivered via the synaptic gap. In that gap, chemicals reside that provide either a low or a high impedance transmission path of the electrical pulses. This weighting of the inputs by the synaptic response are much more continuous than the on or off nature of the overall neural response. The model

of the basic neuron can be seen in Figure 1.1.

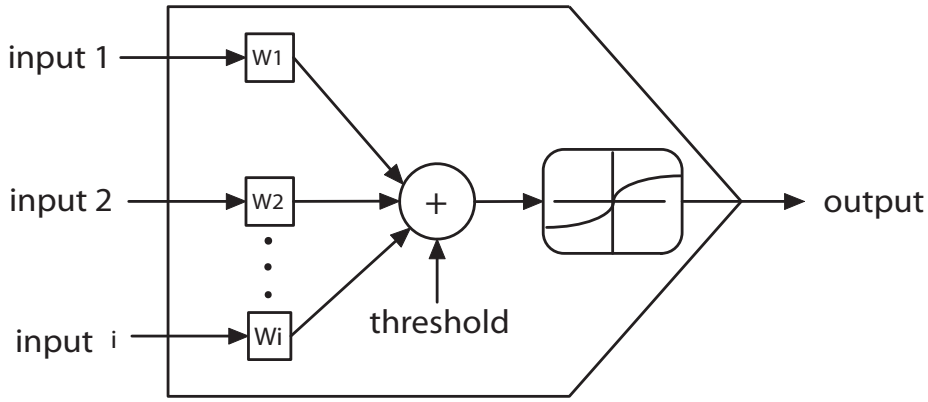

Figure 1.1: Simplified Neuron Block Diagram

This block diagram shows that the inputs are weighted by numerical values. This weight function is derived from the transmission impedance of the synapse. The neural cell collects the weighted signals into a complete sum, adjusts the collective with it's own chemical bias, and then processes the output. The output process is modelled after an activation function. Simple activation functions process the congregated inputs to a zero or a one (i.e. off or on). However for continuous realtime systems, the activation function must be a continuous one. The sigmoidal function is used most often to approximate a binary activation function.

In regards to the way this neuron behaves, a simple demonstration of its decision making ability may be of use. Referring to Figure 1.2, we see a single neuron decision reference plotted onto a simple input space. This decision 'surface' maps the input space to an output space to approximate a function; in this case an OR function.

A single neuron with two inputs, two input weights, a threshold, and an activation function approximates a straight line. If the input space equates to a point

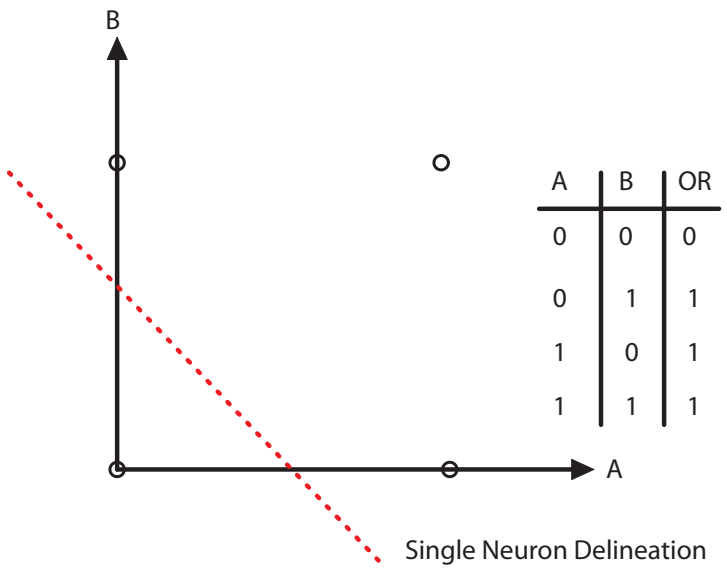

Figure 1.2: Single Neuron OR Function Map

greater than this separation 'surface', then the neuron maps an output to one and vice versus. The simple neuron can be calculated by multiplying the two inputs by their respective weights. Additionally, the sum of the weighted inputs can be offset by the threshold and then mapped to the output through the use of the activation function. The calculations required to process this neuron is 2 multiplies, 2 additions, and an activation function calculation; totalling 4 mathematical processes and one lookup process.

If we try to map the input space used in the previous OR case to an XOR case, the single neuron fails. This particular function requires two neurons to accomplish the function map as can be seen in Figure 1.3.

The neurons used in the XOR case are identical as in the OR case. excepting that their connection structure is cascaded. The cascaded network architecture uses the output of one neuron as an input to another. This, in effect, demonstrates that

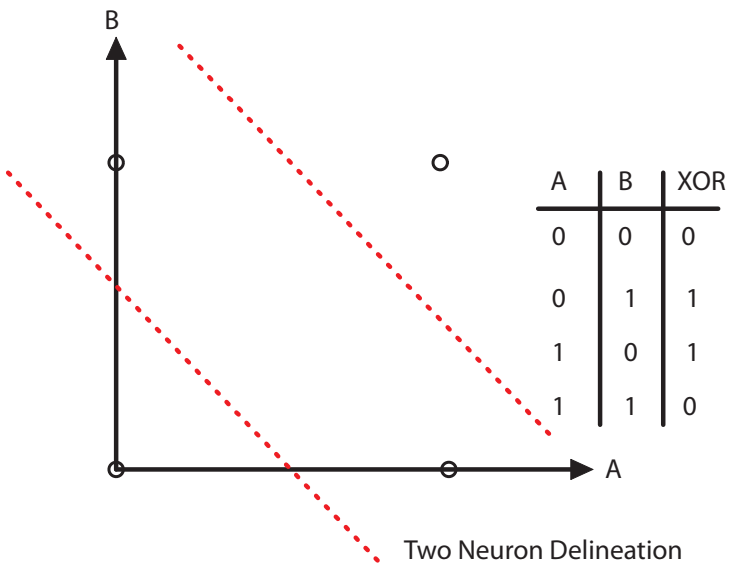

Figure 1.3: Two Neuron XOR Function Map

the reduction method intended to be shown in this research is possible. The output neuron essentially uses an additional non-linear input, defined by the other neuron, to approximate the input to output mapping. However, twice as many calculations exist in order to approximate a simple boolean function on the same input space as previously examined; specifically eight calculations versus 4 as in the OR function problem.

The neural research premise herein solves the problem by adding a simple pre-calculation instead of an entire neuron calculation, to map the input space to the output function. In particular, the two inputs are multiplied together to create the non-linear relationship used for a third input. The approximate function map can be seen in Figure 1.4. This Figure shows that indeed the XOR function is mapped by a single neuronal 'surface'. By doing so, we have reduced the neural calculations to $6 + 1 = 7$.

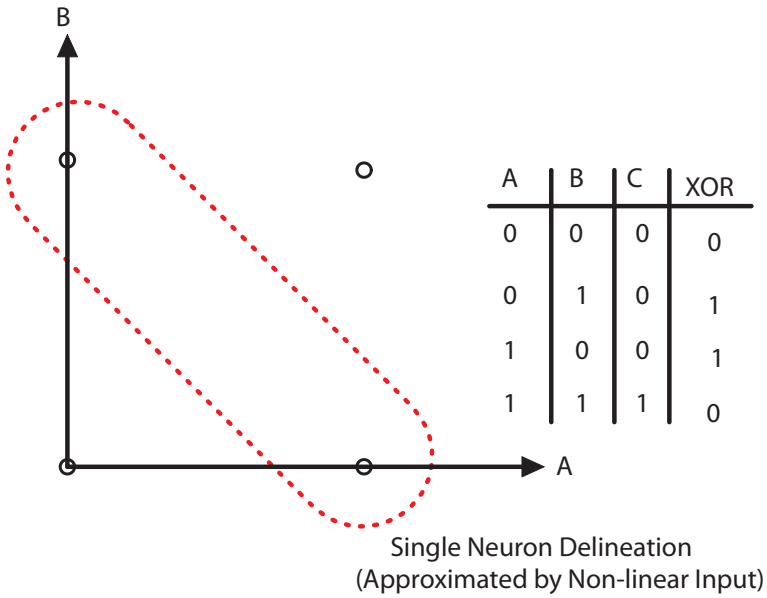

Figure 1.4: Single Neuron XOR Function Map

This may not seem impressive, for we only reduced the total number of calculations from the two neuron case by one. Most real neural nets, however, require many more neurons than just two. This simple example shows that the application of an additional non-linear input can greatly reduce the number of neurons required by typical neural applications. In our case we reduced the number of neurons by 50 percent.

Referring to Figure 1.5 (a), a typical design used for position estimation in SRM drives is shown. The net has two inputs, two input neurons, 6 hidden neurons, and a single output neuron. This network requires 53 calculations to process. Also shown in Figure 1.5 (b), the proposed neural network requires two inputs and a pre-processed non-linear input, three input neurons, and a single output neuron. This network requires $28 + 1 = 29$ calculations to process, about a 45 percent reduction. It will

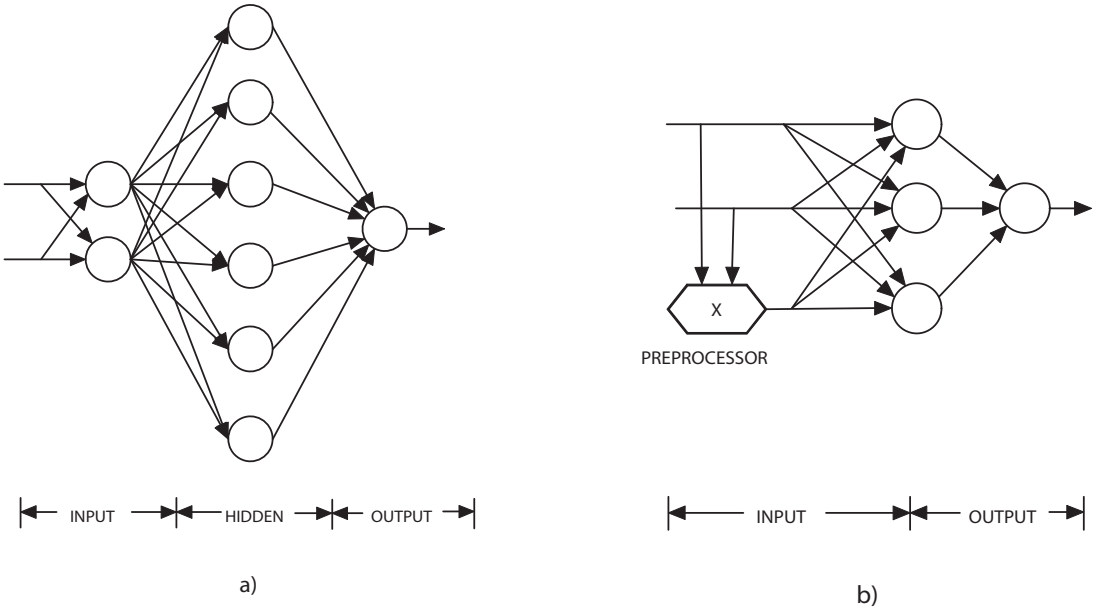

Figure 1.5: Neural Nets

be shown that accuracy will not be sacrificed to improve processing speed. As with the XOR demonstration, a single neuron provided the same function map as the two neuron case, thus no accuracy was sacrificed.

Chapter 2

Previous Research in SRM Sensorless Drives

2.1 Introduction

In this chapter, previous research in the field of sensorless drives is reviewed. Specifically, the methods, problems, resolutions, and implementations are discussed. As in the previous chapter, the focus will be Artificial Neural Networks for sensorless position estimation in SRM drives as well as the supporting functions required for fully implementing the estimator. Typical neural architectures will be examined as well as potential future trends in ANN's.

2.2 SRM Drive Reliability Issues

Typical SRM drive systems include a machine, drive power electronics, controllers, and position sensors. The power electronics deliver the necessary torque producing current which is closed-loop controlled by either analog electronics or a microprocessor. In recent decades, the trend has been that research is drifting away from using

analog equipment towards the microprocessors. This is largely due to new microprocessor developments. The improved features are very attractive for implementing control digitally.[9]

As SRM drives gain popularity, there is an increasing effort to reduce costs, improve performance, and increase reliability. As discussed in [9], one of the drawbacks of an SRM drive system is it's positions sensors. Rotor position sensors add hardware complexity, connectivity problems, and reliability issues that make the overall drive system prone to failure. Also, there are some harsh environmental conditions that require sealed motors. The required position sensors in SRM drives limit the viability in these applications.[9] Common SRM rotor position sensors include such devices as Hall Effect latches and encoders.

The removal of the SRM position sensors is a popular topic of research. Many researchers are trying to improve the robustness of the system by developing control algorithms that eliminate rotor position sensors. These systems are typically called sensorless drives, as they have no position sensors. "Sensorless" is a slight misnomer, as these systems have other sensors associated with measurement of electrical parameters, like voltage and current, used in the estimation of rotor position.

As processor technology increases, it becomes easier for design engineers to tackle problems that plague sensorless drives. If sensorless SRM drives are to become replacement technology for induction drive applications, sensorless schemes for Switched Reluctance Motor(SRM) drives must have the same robustness and reliability as induction drives. To achieve this goal, robustness must be designed into the algorithm because sensorless techniques are noise and error sensitive.[23] However, inherent reliability occurs just by the omission of the sensors as well as the reduction of hardware

therein. There exist many methods today by which engineers have accomplished sensorless implementations. Common methods developed include flux reference, inductance probing, fuzzy, neural, and model/observer based methods.

2.3 Sensorless Algorithms and Methods

The salient structure of an SRM is the main characteristic exploited by sensorless algorithm developers. The SRM structure provides a repetitive nature of the electrical states with respect to rotor position.[9] Whether the sensorless technique uses a lookup table for flux linkage, active current probing, or even neural-fuzzy methods, the flux linkage and current are the typical quantities used to derive rotor position.[38] This is because of a well know relationship in SRM's between current, flux linkage, and rotor position. If, at any given instant, the flux linkage (λ) and the current (i) of a particular phase are known, the rotor position can be derived.[38, 9] And by the saliency, the relationship repeats itself for every stator pole in the machine.

The various genre by which senorless SRM algorithms can be classified are [9]:

1. hardware-intensive methods like signal injection require extra external circuitry
2. lookup table methods can be memory intensive
3. model-based methods can be prone to errors especially if the model has hidden or undesirable characteristics
4. adaptive methods such as neural networks and fuzzy controllers are typically computationally intensive.

Active Probing

Although active probing has been used by other researchers it has inherent drawbacks. Some designers, like [2], use additional hardware and complicated software techniques to inject signals in unused phases. This is done so that the relative impedance can be measured during motor runtime. These methods require special care in order to handle the timing issues associated with probing inactive phases. Other problems that arise by using active probing methods are the additional noise injected into the system and the measurement of the transient probe signals. It is undesirable to perform active probing because of these issues. Non-probing methods include: flux-current methods, model-based estimator techniques, mutual voltage and back-emf sensing, as well as adaptive methods.[38]

Flux Reference

Most non-intrusive methods employ some sort of digital estimator for flux linkage by the use of Faraday's law,[8], as in flux reference methods like the one used in[36]. Flux linkage,inherently, cannot be measured directly, therefor it is very difficult to predict and control. Because flux linkage must be estimated in some way, many researchers are exploring better ways to estimate flux linkage for use in sensorless drives.

Algorithms that use flux linkage as a control quantity, typically employ a predetermined lookup table. Those methods utilize either simulated data [35], or modelled data [19, 35], to generate the lookup information. In either case, these commutation methods are implemented using a reference flux linkage from the lookup table, and by comparing a runtime estimate to the lookup table reference. By doing this comparison for the aligned condition, the commutation instant can be predicted.[8].

Thus when the estimate exceeds the reference derived from the lookup, the current is commutated from the active phase to the next inactive phase.

One major difficulty in using stored magnetization curves in a lookup table for sensorless control is the accurate modelling of the SRM.[35] Most magnetization data is derived from simulation instead of real-time experimental data.[35] By contrast, SRM magnetic characteristics are experimentally identified by a locked rotor test.[8]. In both cases, the quality of the control relies on the accuracy of the stored data.[15] Additional problems arise from injected noise or operating conditions not contained in the lookup table. Therefore, some researchers [29] attempted to control the system by analytical methods. Some analytical methods used observer dynamic models of the SRM and it's drive, while others only modelled the magnetic characteristics of the SRM. Difficulties arose out of the extreme non-linear nature of SRM's. Thus, the past two decades of research have tended towards adaptive, non-linear methods such as Artificial Neural Nets or Fuzzy Control.

Fuzzy and Neural

Artificial Intelligence is used for position estimation in SRM drives and is studied by many researchers.[35] Fuzzy linear interpolation can be used to construct real time magnetization curves, even for the intermediate positions.[35] One of the major contributions of interpolation methods is the reduction of data point acquisitions.[35] If experimental magnetization data is used, only the aligned and unaligned positions are derived from measurements.[35] A combination of fuzzy interpolation and quasi-accurate analytical models can alleviate large memory requirements usually needed in

lookup table methods.[35] Other fuzzy-reasoning based methods measure magnetization data for numerous rotor positions and then store that data in the form of fuzzy rule-based tables.[35] Some fuzzy logic based algorithms are used to eliminate the estimation error and provide the motor drive with more robust operation.[23] These estimators improve performance by adding features like fuzzy phase selection, fuzzy flux linkage estimation, and fuzzy angle predictors.[35]

In addition, rotor position estimation can be accomplished using Neural Networks. The evolution of storing magnetization curves has moved from the direct table implementation through the reduction of data sets by interpolation methods to the current methods by which data is stored intrinsically in the structure of the neural network. Neural networks behave like non-linear approximations. They efficiently store data in their weight structure to both model the dynamic system and interpolate the in-betweens. Some architectures can allow for the extrapolation of their performance for operating conditions to which the controller has not been introduced. These are the fundamental characteristics that have popularized the use of neural controllers for many types of control systems.

2.4 Artificial Neural Networks

Neural networks may have several advantages over traditional control methods.[6] Neural networks can learn to perform certain tasks based on experimental or real-time data. [50, 42] demonstrate that ANN's can organize information it receives during runtime to approximate functions. ANN's provide parallel computation through multiple output architectures. In addition, its extraordinary interpolation ability provides excellent fault tolerance.

ANN's have many important applications as denoted by [50]. They are used in system identification, adaptive control, modelling, optimization, and motion control.[49] In comparison, analytical models fall short due to the complexity required to appropriately model those systems.[3] ANN's are ideal for controlling discrete-time, non-linear systems with large non-linear variations in their states due to their extreme non-linearity.[49] ANN's have successfully modelled highly complex, non-linear systems, due to their unparalleled ability to recognize complex patterns in seemingly random data.[3]

ANN's architectures are primarily designed in two fashions. The first, Time-delay is more suitable for dynamic non-linear systems. [50] However, they are intrinsically more difficult to implement due to architectural complexity required to obtain estimation accuracy. In general, the feed-forward type of neural net is utilized most in control problems. [7]

The second type of ANN, a feed-forward neural network, processes information from input to output through one or more layers of neurons. Feed-forward networks demonstrate the greatest compatibility for the position estimation problem in SRM's.[3, 34] Feed-forward networks are either fully connected or partially connected. A fully connected neural net has all of the inputs connected to each neuron in the input layer. A partially connected network is a fully connected network with some of the synaptic weights set to zero.[34] Most research accomplished to date for SRM rotor position estimation utilizes some sort of feed-forward neural net.

A typical ANN for rotor position estimation contains 2 input neurons, some number between 5 and 30 hidden neurons, and 1 output neuron.[4, 3] Depending on the interconnection, anywhere from several dozen to several hundred individual weights

can be contained by these networks.[3] In addition to synaptic weights, an activation function is utilized in each of the neurons. Typically, the activation function is some variety of the sigmoid function.[4] The calculation of the sigmoid function can weigh heavily on microprocessors. Some of the burden can be alleviated through the use of additional lookup tables for activation function approximation.[4] An unusually small feed-forward neural network can be represented by some small plurality of inputs, a hidden layer with a small plurality of neurons, and an output layer with a single neuron. The neural network with a reduced amount of input neurons and a single output neuron is also referred to as a multi-layer perceptron.[34] This architecture represents the direction for the current research.

Certainly neural networks ease the control burdens, yet there are several outstanding issues that may present obstacles for a system designer. One is that ANN's maintain their accuracy through the force of large numbers of neurons.[4] If accuracy is critical, a successful neural net implementation may require a large amount of memory to contain all the synaptic weights. Second is the number of connections between layers can be problematic. Depending on the size of the network, the calculations required to process the net cannot be implemented with other control functions on most common microprocessors. Most ANN designers then tradeoff performance versus neural size, thus accepting more estimation error for the ability to capture some of the benefits ANN's provide. Estimation error is dependent upon both the number of neurons in the hidden layer and the number of samples in the training data set.[34] Training data is highly affective on the performance of the network. Estimation performance for small conduction angles is strongly affected by the chosen training set.[34] Methods of obtaining data and using that data to train the neural

networks is a topic of research for neural scientists.

2.5 ANN Design, Training, and Evaluation

ANN Design

At this time, no exact design rules exist for researchers wanting to implement intelligent control. However, there do exist some general steps to achieve a neural design implementation.

Steps towards designing ANN's include [27, 41]:

1. The designer should define the number of input, hidden, output layers.

2. The activation functions that are best suited for application should then be chosen. (i.e. match the function to data space dynamic range) It is generally accepted that non-linear activation functions should comprise the majority of the neurons while the output neurons can be linear.

3. The neural net weights should be initialized to random, small values to ensure that the network is not saturated by large values of weights. Also if the initial weights are identical, the neural net would not be trainable.

4. The designer should collect, obtain, or select the data set for training. The training data should include both the input and output spaces.

5. A training supervisory algorithm should be chosen based on the type of neural architecture that was chosen. Generally, a back-propagation type algorithm is sufficient for feed-forward networks.

6. The neural net should then be trained by adapting the weights using the supervisory algorithm. The algorithm should minimize the error between the neural output calculated from the input space and the associated target value.

7. The most important step of neural design is verification. The designer should verify the network with data pairs not from the original data set. Generally, the test data is an extraction from the original data set, yet the test data is never presented to the network during training.

Training Data

The most important step in designing a well performing neural net is choosing the proper data set. ANN performance is somewhat limited for use in estimating SRM rotor position. Studies show that an ANN trained to estimate flux linkage performs much better estimation than the one that is trained to estimate rotor position.[35] This should be taken into account when developing the data set to be used during training. To prevent poor position accuracy, the data set should be randomly distributed over the entire operating range.[34] Most neural training sets consist of data points of flux linkage, current, and the corresponding rotor position. However, several ambiguities exist in the complete data set which should be considered by designers creating ANN's for sensorless SRM's drives. As discussed in [34], the most important factor affecting the choice of an ANN data set is the salient nature of the machine. Saliency causes several values of current and rotor angle to exist for a single flux linkage value. To overcome this concurrency, quantities from at least two phases must be measured or **the data set must be limited to the interested angular interval that produces positive torque**.[34]

There are two possible ways to generate training data: model-based data generation and experiment-based data generation.[35]

Given a proper analytical model, flux linkage values can be computed for randomly generated phase current and rotor position values. In doing so, the resulting flux linkage, phase current and rotor position values will judiciously cover the intended operating region.[35]

The approximate analytical model for flux linkage is derived by using a function that is dependant on rotor angle and phase current[19, 35]:

$$\lambda(i,\theta) = \alpha_1(\theta)[1 - exp(\alpha_2(\theta)i) + \alpha_3(\theta)i \tag{2.1}$$

where the functions α can be described by a Fourier series:

$$\alpha_a = \sum A_a cos(B\theta) \tag{2.2}$$

Difficulties arise when using "pure" data generated via model based equations or even Finite Element Analysis simulation. The actual implementation may differ for several reasons including noise contained in the input space. However, the designer may overcome these obstacles by the previously mentioned method of data generation: experimental data collection.

In the experimental data approach, the motor can be run for certain operating points so that the derived magnetization characteristic is swept over the appropriate operating regions. A shaft encoder can be used to insure the angular intervals in which data is collected is consistent. This approach should alleviate data ambiguity.[35] A

better approach would be to run the motor from zero to full speed, or across the entire torque range, so that **every possible electric cycle of the flux linkage, phase current and rotor position can be sampled and recorded for use in training**.[35] The continuous run method provides a more judicious coverage of the magnetization characteristic.[35]

In either case, a sufficiently large training data set should be developed so that the ANN can build up a correlation between its output and the paired input set over the entire drive operating range.[34] After generating the training data set of current and flux, normalization is performed on the input space so that current and flux is scaled between zero and one.[35] It is up to the designer and the particular neural structure as to whether to normalize the output. If a linear activation function is used on the output neuron, the output data point need not be normalized. This may, however, cause implementation issues later on.

Training Method

A lot of research has been pursued to simplify and speed up training of neural nets. Hence, there are many training supervisory algorithms from which designers may choose. Most ANN designers train their networks using back-propagation supervisory algorithms.[3] This is due to ease of implementation and its compatibility with feed-forward network structures that are so common.

Some supervisory algorithms are better at training feed-forward networks than others. Performance of these algorithms is generally measured by the time it takes to minimize the neural estimate error and converging the neural weights to a steady value. The **Levenberg-Marquardt** back-propagation algorithm is considered a

method for supervised training based on an approximation of the Gauss-Newton optimization of the typical gradient descent method.

The typical back-propagation (BP) method utilizes a gradient descent update method that is derived from a performance index based on the Newtonian error. The performance index is given by:

$$V = \frac{1}{2}\, e^T\, e \tag{2.3}$$

This performance index is applied to the the weighting matrix for the neural net during the training process. The steepest descent method is an attempt to minimize the error as fast as possible through adjusting the weight matrix and recalculating the error each time. The weight update is given by:

$$\Delta\, W_i = -\alpha\, \frac{\partial\, V}{\partial W_i} \tag{2.4}$$

The weight matrix for the neural net is given by W and the learning constant is given by α. The partial derivative matrix describing the gradient can be denoted by $\nabla\ V$. The main drawback of this method is the inability to discern local minima from a absolute minimum. Thus the algorithm can converge but not achieve optimum neural estimation performance. An attempt was made to optimize the gradient descent through determining the optimal descent to the absolute minimum. The Gauss-Newton Back-propagation (GNBP) method boasts this optimization but is extremely difficult to calculate. Continuing with the BP performance index given in 2.3. The weight update is given by:

$$\Delta\, W_i = [\nabla^2\, V]^{-1}\, \nabla\, V \tag{2.5}$$

where the $\nabla^2 V$ is also called the Hessian and ∇V is also called the Jacobian [17]. The Hessian proves to be a complicated matrix to calculate and tends to slow down the convergence of the GNBP method. The Hessian calculation thus negates any improvement achieved through the optimization.

The Marquardt modification to BP training supervisory methods was first demonstrated in [17]. It attempted to approximate the GNBP method without the necessity of calculating the Hessian. If we assume the performance index is given by:

$$V = \Sigma\, e^2 \tag{2.6}$$

then the Jacobian can be re-written as:

$$\nabla V = J^T e \tag{2.7}$$

and the Hessian as:

$$\nabla^2 V = J^T J + S \tag{2.8}$$

where J is the first partial derivative of the error matrix which is much faster to compute that the Hessian. [17] continues to demonstrate that if 2.7 and 2.8 are substituted into the GNBP weight update method given by 2.5 we have:

$$\Delta W = [J^T J + S]^{-1} J^T e \tag{2.9}$$

For application of 2.9 to the training of neural networks, [17] concludes that S can be replaced by a μI matrix. Thus the Levenberg-Marquardt Back-Propagation (LMBP) training method is given by:

$$\Delta W = [J^T J + \mu I]^{-1} J^T e \tag{2.10}$$

If μ is large, the LMBP modification to the GNBP method approximates gradient descent BP. If, however, μ is small, the method approximates GNBP. The LMBP modification is also considered a trust-region modification. This method approximates the optimization gained from GNBP while retaining the speed of calculation of the normal gradient descent methods. Many designers lean towards this method because of its extremely fast convergence.[27, 37] It can be several orders of magnitude faster than typical BP methods.

To summarize the implementation of the LMBP training algorithm [17]:

1. Process the neural network, calculating both the outputs and errors.
2. Compute the Jacobian
3. Apply the neural weight matrix update method given by equation 2.10
4. Re-compute the errors
5. Adjust μ based on results of previous step. If new error is less than previous error, reduce μ; else, increase μ and repeat from step 3.
6. Repeat method until all weights have converged.

The performance of the training methods for ANN is inconsequential if the network doesn't accurately represent the system that was intended to be modelled. To properly evaluate the ANN performance, a data set should be constructed that is not

part of the original training values.[34] Performance analysis conducted on trained ANN's is typically evaluated via least mean squared error.[35]

In summary, the performance of the neural net is absolutely limited to the accuracy of the data with which it is trained. Factors such as training method and neural structure are overshadowed by inaccuracies that will be inserted due to invalid data. Much research has been performed in order to alleviate data inconsistencies. The majority of the research has been to remove measurement error from experimentally captured data sets including flux linkage estimation.

2.6 Flux Linkage Estimation

In [35], flux linkage is shown to be a quantity that can not be explicitly measured at the SRM electrical terminals. It must be estimated by measuring phase winding voltage in addition to the phase current using Faraday's Law. This estimation algorithm can be implemented either through a software algorithm or through an analog circuit. The software flux linkage estimation algorithm enjoys simplicity in implementation and tuning flexibility, refer to Equation 2.12.

However, flux linkage estimation is extremely sensitive to measurement error. Stringent protocol must be practiced to insure the accuracy in the hardware.[30] Because flux linkage is not directly measurable, we can deduce that the flux linkage will contain more error.[15] Flux linkage estimation requires numerical methods to process explicit measurements and is subject to many sources of error. According to [15], **the main disadvantage of sensorless algorithms is the prediction of flux linkage by equation 2.12**. To understand where the error is inserted, we must understand how flux linkage is derived.

The flux linkage is directly related to current and rotor position.[9] The flux linkage can be derived by measuring the winding voltage and phase current.[35, 9] The flux linkage for each phase is then predicted by applying Faraday's law:

$$\lambda = \int (v_{ph} - Ri_{ph})dt \tag{2.11}$$

where v_{ph} is the phase voltage and i_{ph} is the phase current.

The discrete flux linkage can be expressed by [5, 35, 9, 15]:

$$\lambda = (v_{ph} - Ri_{ph})T_s + \lambda_{old} \tag{2.12}$$

where T_s is the sampling time and λ_{old} is the previous calculated flux linkage. Equation 2.12 can be directly implemented on most microprocessors. By utilizing only electrically measurable quantities, flux linkage estimators and further position estimators have become an integral part of the discrete controller.[38] However, the accuracy of the software estimation implementation is limited by the sampling rate.[35] The **error increases during the current regulation mode** where relatively high frequency voltage pulses are needed to be sensed and integrated.[35] These fast transients become difficult to reproduce from digital samples. Aliasing becomes a dominant effect during this control mode. In order to achieve more precise flux estimation at slower speeds, analog estimators are typically built. Analog hardware introduced into the circuit for flux linkage estimation is subject to increased cost, decreased reliability, and application limitations.[35, 15] Thus many engineers prefer the software implementation and deal with limiting the sources of error introduced by analog methods.

The main reasons for flux linkage estimation errors are [15]:

1. the phase resistance is changing proportional to the motor temperature
2. uncertainties in the calculation of the phase voltage due to pulse durations and unknown external losses
3. long integration times
4. error in current measurements due to external noise

Flux linkage estimation methods are also sensitive to low levels of current. The main reason for this sensitivity occurs when the winding voltage and the resistive voltage drop due to current are of similar magnitudes[15]. Slight measurement errors or errors introduced by rounding effects in the microprocessor can lead to large errors in the integrand. Some researchers have overcome this by designing minimum current levels. Minimum current level designs are restricted to applications that are insensitive to such limitations like fan drives.[8]

The effect of **integration error is compounded at low speeds where the integration time is long**.[15] Discussed in [35], the elapsed time between integration resets is much longer at low speed than at higher speeds. The flux linkage estimation error grows until the integrator is reset. Longer integration times lead to higher flux measurement error. Accordingly, rotor position estimation error increases at lower speed.

The flux linkage integral, whether discrete or analog, must periodically reset. The **integrator requires a periodic reset** to offset an accumulation of errors. These

errors result from the DC offset in the feedback signal measurements, noise, and winding resistance variation from heat.[5] However, resetting the flux linkage to zero during commutation assumes that there is no active current in the phase prior to the start of estimation. Then, no mutual effects from other phases can be considered.[30] Most flux linkage estimation techniques make the assumption that current is carried inside the machine one phase at a time.[30] If no mutual effect are considered in the estimation process, the technique is only viable for lower speeds where the back-emf is negligible.[30]

The flux linkage software estimation implementation suggests that an initial condition may need to exist.[36] In most cases, the initial condition is zero, because the current in the active phase is always zero.[36] Some converter topologies prevent this actuality as in the converter used in the current research. **If the initial current in the active phase is greater than zero at the start of active conduction, then we may deduce that in fact the initial flux linkage is a positive, non-zero value**. This is significant if an accurate representation of flux linkage is to be derived for systems that do not allow for complete reset of flux linkage. A Linear Time Invariant (LTI) relationship between flux linkage, phase current, and rotor position must be maintained during drive operation in order for neural rotor position estimation to be successful.

One last consideration is the physical relationship of flux linkage to current and rotor position. At the extremes of rotor position (near un-alignment and near alignment), large errors in estimated angle can occur due to small errors in current measurement or small errors in flux linkage estimation.[30] Thus, the **viable range to use for accurate prediction of the rotor position should be restricted to**

angles between 20 and 40 degrees.[30]

2.7 Sensorless Implementation Limitations

Whether tackling complex system problems with linear or non-linear controllers, it is desirable to implement the simplest method that still achieves the performance required.[6] Most SRM sensorless methods require high speed processors that are capable of millions of instructions per second (MIPS). Digital Signal Processors (DSP's), then, are ideal for sensorless implementations.[9] Sensorless algorithms are limited at high speed by the speed of the DSP to cope with large amounts of mathematical operations.[15] Another limitation that can occur during implementation is memory. Limited amounts of memory in modern DSP's cause large lookup tables and large Neural Nets to be quite a burden. [3] Limited word size also poses problems for controller and neural implementations in fixed point DSP's due to the inability to maintain resolution when large numbers are required.

2.8 Survey of Current SRM Drive Technologies

2.8.1 Overview of Recent Technologies

Recent technological advances for SRM sensorless drive technology have focused on the optimization of SRM performance or the obsolescence of position sensors. Techniques for optimization include the use of Artificial Neural Networks, Fuzzy Logic, or Flux-linkage control methods, whereas rotor position estimation has primarily consisted of observer and current probe methods. The use of ANN's for position sensorless SRM drive has been researched extensively over the last 10 years, the majority of

which occurred around the mid-1990's. Recently, there has been limited application of ANN for use in the elimination of rotor position sensors in SRM drives.

All position sensorless SRM drive methods achieve control in one of two ways. One, forming a model of the motor, or two, by exploiting the salient nature of SRM's or some combination therein. Model methods fall primarily into two categories: dynamic equation or data oriented models. Equation-based methods include the use of observers and parametric dynamic equations while the data methods typically use stored tables. These tables usually contain fundamental non-linear, parametric relationships of SRM's, such as inductance vs. rotor angle or flux-linkage vs. rotor angle.

Fuzzy and neural methods are a sub-category that bridge the gap between dynamic modelling and stored parametric tables. Neural methods employ coefficient type dynamic equations which utilize parametric data, that is then encoded into the coefficients through training algorithms. These networks attempt to model the system in order to predict some intrinsic behavior that is typically unmeasurable. Neural methods can effectively map the parametric relationships of the motor. All model methods are software based algorithms implemented in either microcontrollers or DSP's.

Hardware sensorless and optimization methods are typically invasive to the machine, as demonstrated by current probing techniques. They exploit the saliency of SRM's by injecting certain types of signals and then measuring the dynamic response. The response is processed to deduce information about the motor, its drive, or its performance.

In this section, current technology is discussed and the various methods by which

sensorless SRM control and SRM drive optimization has been accomplished in the last 5 years. This section is divided into four parts: model-based methods, table and probing methods, which are followed by neural optimization applications, and then neural sensorless applications.

2.8.2 Model Based Methods

Model-based methods derive their control either from a traditional state observer implementation or from a dynamic equation perspective. There are several parameters associated with SRM's that require observation, as they are not directly measurable without either very expensive transducers or complicated hardware/software additions. Either course removes the advantages gleaned from the application of an SRM drive. Typical SRM quantities requiring estimation are torque and flux-linkage. Research attempts flux-linkage observation in various ways as demonstrated by [32, 47, 31].

In [32], an observer-based, mathematical model is used to optimize startup and runtime torque. [32] demonstrates the use of an parametric estimator that models the machine in a torque driven motor. This method mathematically models the electric and magnetic characteristics to derive torque vs. current profiles. By implementing these characteristics into the control, performance optimization can be achieved (such as torque smoothing and/or minimizing sensitivity to errors in rotor position measurement).

The control system includes an estimator that receives signals representing the phase winding voltage, current and rotor position. The estimator attempts to predict both the electrical and magnetic performance of the machine. The electrical model

describes electrical behavior of the machine as seen at the motor terminals; i.e. voltage and current. The magnetic model is developed from a transform of the electrical model which describes torque characteristics of the machine. In turn, simple P+I controllers utilize the predicted information to perform the required control.

In [47], Flux linkage estimation is performed by using a phase current measurement and an inductance lookup table. Specifically, the control method estimates the standing flux-linkage associated with the phase and in turn improves its' estimate of rotor position. The key note on this method, however, is that the algorithm corrects flux-linkage even if it is non-zero at the time voltage is applied to the windings for active current conduction. This algorithm improves over other methods by robustly working regardless of whether the current is continuous or discontinuous.

When the current is discontinuous the zero current value gives rise to a zero value of flux-linkage. When the current is continuous the value of current optionally is used to derive the non-zero flux-linkage.

An algorithm detects rotor position in a SRM by deriving a value for the current and flux linkage associated with at least one phase of the machine. The value of the flux linkage is determined at the moment when voltage is applied to the phase and is derived from the current at the moment. For example, the flux linkage at that moment is derived from the current and stored values of inductance for ordinates of current. Subsequently, the flux-linkage may be derived by integrating the phase voltage from the initial moment until active conduction is ceased. The rotor position may be derived from stored parameters having coordinates of phase current and flux-linkage.

Embodiments stated by [47] are particularly useful in the single-pulse mode of

operation of a switched reluctance machine which typically occurs at very high speeds.

Other flux observer methods, like [31], utilize sensing coils to derive voltages which can be used to estimate the SRM's flux magnitude during runtime. [31] utilizes a flux observer adapted to produce a signal indicative of flux-causing voltage across at least one phase winding. The control system as taken from [31] can be seen in Figure 2.1 below:

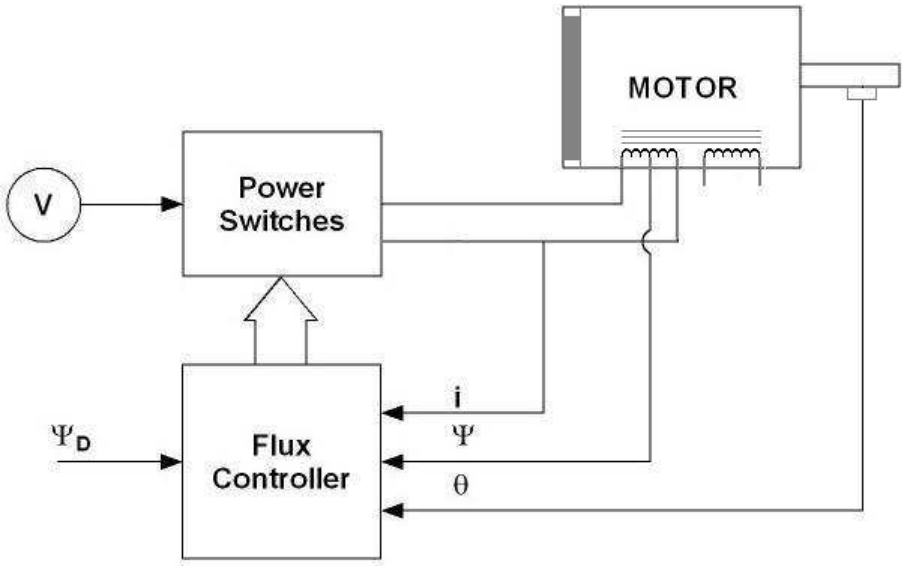

Figure 2.1: Flux Feedback Control System using Auxiliary Windings

The control system shown in Figure 2.1 includes a flux control loop adapted to receive a flux command as the input, and provides an energizing signal as an output. The system also includes an electromagnetic sensory system constructed to produce the feedback signal corresponding to the flux in the electromagnetic system. This flux observer can include a Hall-effect device, a Gauss meter, or an alternative winding that can be either separate or interspersed in the main phase windings. The observer

is arranged in a flux path of the machine and produces a voltage or current that is directly proportional to the flux. An equational estimator converts the voltage produced by the sensor into a meaningful, controllable parameter. The estimator includes a current model of the machine arranged to receive signals representing phase current and rotor position and is operable to produce a flux estimate for each phase winding. From there, a typical P+I+D control loop can be constructed.

Other researchers, like [11, 1], approach sensorless control using slightly different methods. In [11], a method for indirectly determining rotor position in a switched reluctance motor (SRM) is implemented based on instantaneous flux and phase current sensing. Phase flux and current measurements are made in a pair of predetermined sensing regions for each phase in a predetermined sequence. Rotor angle estimates are derived from the flux and phase current measurements made during the sensing region for each respective phase. The rotor angle estimate for each phase are normalized with respect to a common reference. These are used to generate the SRM rotor position estimate. The algorithm makes use of a generalized dynamic equation which specifies whether rotor poles of the SRM are approaching alignment or unalignment by incorporating a +1 or -1 into its rotor estimate. This becomes useful in determining the region of operation and translates well into 4-quadrant operation.

However, in [1], a sensorless rotor position measurement method for switched reluctance (SR) motors is presented based on measuring flux-linkage and current when a reference position estimation is reached. By comparing the measured flux-linkage with a pre-stored flux-linkage that corresponds to a particular position for the measured current, the angular difference between the estimated position and the particular position can be calculated. The sensorless correction algorithm can be seen

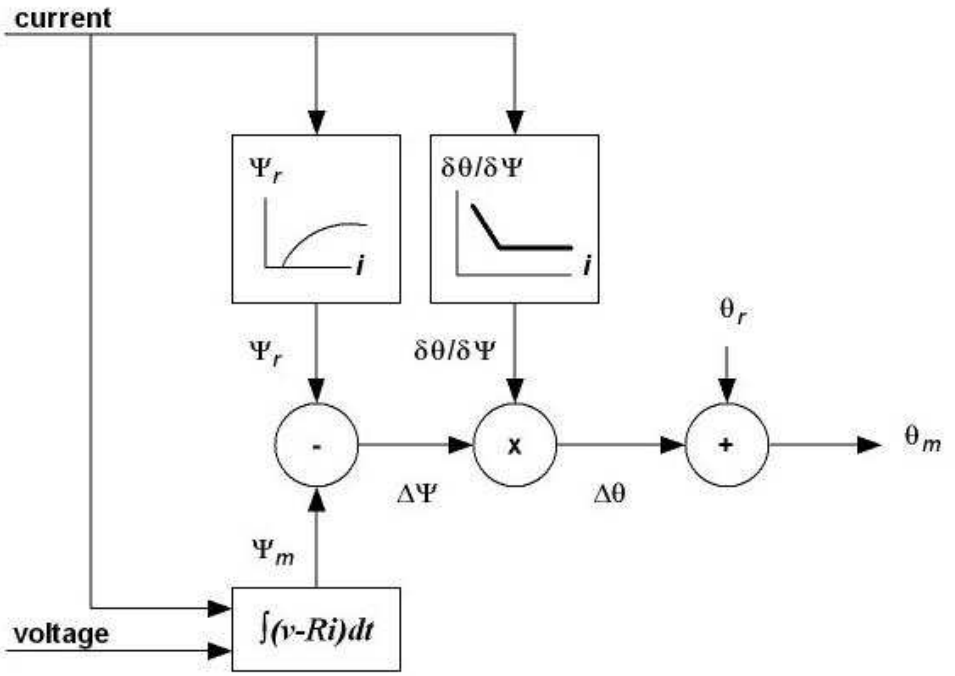

Figure 2.2: Estimated Rotor Angle Error Correction Algorithm

in Figure 2.2. The controller algorithm makes use of the error to predict and correct its angle estimate and commutation scheme.

These previous methods attempt to show how current researchers utilize combined dynamic equations and look-up tables to implement SRM performance optimization and sensorless control. In the next section, methods are discussed to show how hardware methods are combined with tabular models to predict SRM behavior.

2.8.3 Table and Active Probing Methods

Historically, SRM drives are specific to reluctance machine and are even more so for sensorless drives. Recently, researchers have implemented intricate algorithms and hardware in an effort to produce universal motor drives, like [52], or attempt to

simplify software methods by utilizing electrical probing, as in [18]. In either case, both utilize some form of rotor angle error correction as previously demonstrated by [1] . Others, like [22, 46], utilize hardware methods such as magnet insertion or current injection to create sensorless drives that do not utilize dynamic models.

[52]implements a universal drive automatically by forming a magnetization table containing information relating the magnetic flux, current, and rotor angle. The table is formed on-the-fly by first constructing the extreme boundaries of operation. The aligned and unaligned flux linkage information is populated for a wide range of currents. The drive hardware employs the use of transducers used to sense electrical parameters such as DC bus voltage and currents. These are used for estimating rotor position by the information contained in the magnetization tables. Additional control is utilized, in concurrence with the above estimation techniques, to provide the hardware with specific turn-on and turn-off times for the motor torque generation.

A three-dimensional rule table relates the torque of the machine to the turn-on and turn-off angles for different speeds is generated automatically during the initial system startup. The switch states of the SRM and the dc-bus voltage are input into a voltage state estimator and subsequently a flux calculator which obtains the estimated flux linkages for the respective phases. The system uses these estimated flux linkages as well as the phase currents to look up the estimated reference angle in the magnetization table . If the reference angle is not in the table, interpolation is used to estimate this angle. The reference angle is fed back to a differentiator which creates a signal representative of an estimated speed. The desired speed of the rotor is input into a PID regulator which integrates and differentiates the speed to obtain the torque. If this desired torque is not in the table, the turn-off angle is interpolated

and the turn-on angle from the lower torque value of the table is used in subsequent calculations.

[18] implements his method for sensorless control by computing an inductance based on the actual measured operating characteristics. He then correlates the inductance to a position to derive sensorless control.

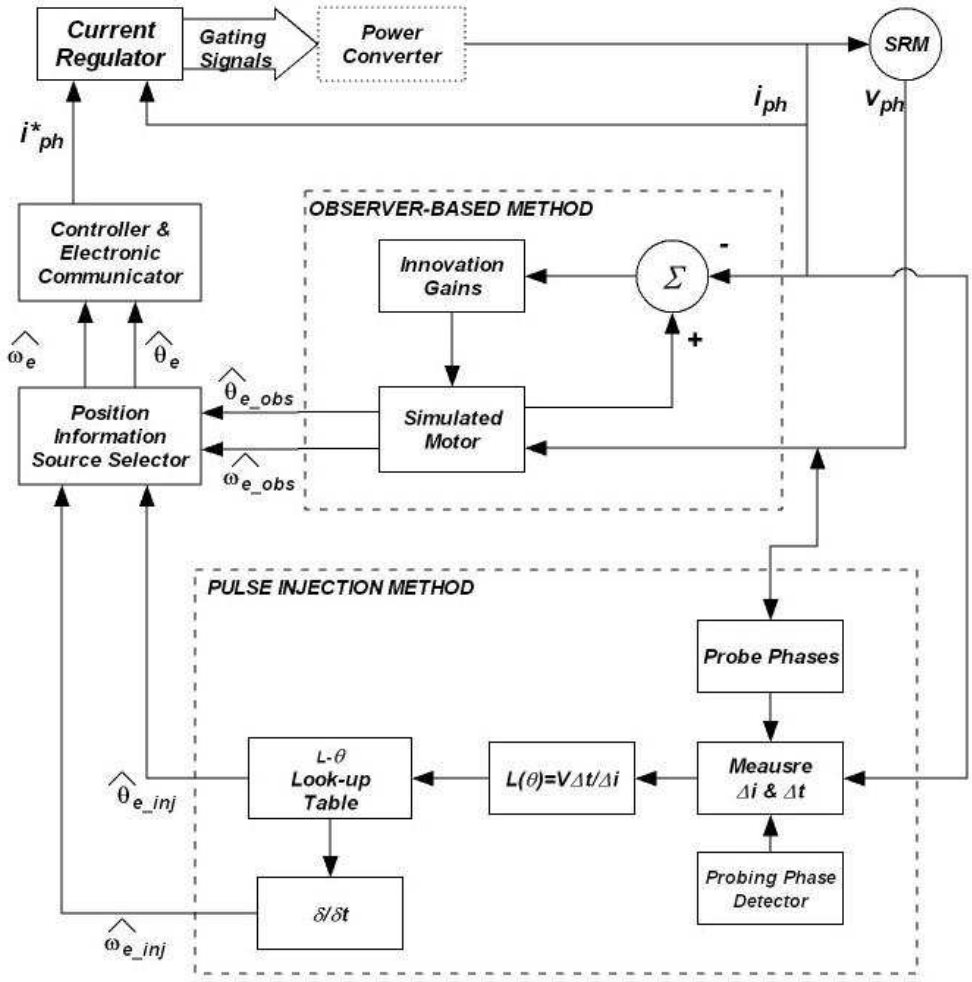

Figure 2.3: Inductance Look-Up Table SRM Sensorless SRM Controller

As demonstrated by Figure 2.3, [18] probes a selected phase of the machine with

pulse injections. The actual operating characteristic of the switched reluctance electric machine can be measured. The inductance is derived based on these actual operating characteristics, and then correlated with a reference position to formulate an estimated position.

These parameters are tracked and implemented into an observer-based model of the switched reluctance electric machine thus formulating a rotor position model based on actual, measurable operating parameters.

Both [18, 52] are hardware and software intensive. Both utilize a reference angle error to derive a dynamic model of the machine to create a drive controller. The following researchers derive their control by hardware methods, but do not utilize any equational model or observer.

In [22], the research uses magnets for a reference instants. Limit switches are also used as discrete angle indicators. In this invention, the controller has a position counter which acquires an estimated position of a magnetic pole from the position-and-speed estimation device. Simultaneously, an incremented value is reset when the reference position signal is input. This, in turn, re-starts the incremental operation. A speed command generator has a position controller. The operation is based on the error between the command, which pertains to the amount of estimated movement, and the value of reference derived from the incremented timer/counter. The sensorless controller can position the rotor at any predetermined location, synchronously, in the absence of position sensors. The generalized controller can be seen in Figure 2.4.

Figure 2.4 shows an external device. This is a transducer that determines a reference position signal serving as an origin for a positioning counting operation.

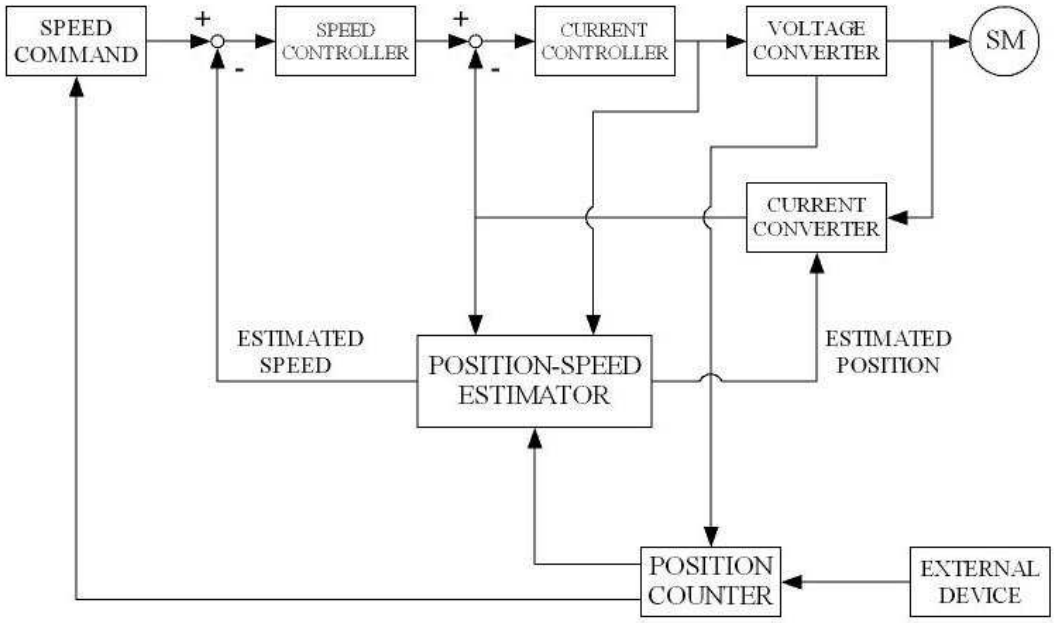

Figure 2.4: Sensorless SRM Servo Drive utilizing Observer-based model

The speed command generator is arranged to:

1. enable setting of a deceleration start position and a stop operation start position
2. start deceleration operation when a summated value output as a relative position from the relative position counter has reached the deceleration start position
3. start stop operation when the summated value has reached the stop operation start position.

Therefore, two limit switches are used to determine the start of deceleration mode and another for the stop mode.

Much differently, [46] utilizes hardware to sense the state of the machine rotor by injecting electrically measurable signals into the stator phases.

[46] determines inductance by the rise-times of current signals injected into unused phases. At the start of operation, the rotor position is determined by simultaneously injecting a current of known magnitude into two separate phases. The rise-times of the currents are measured and then used for comparison against a stored value of rise-time to derive rotor angle. This produces two possible rotor positions from each rise time. However, comparisons of angle pairs from the two separate phases yields the actual position. The control system as depicted by [46] is shown in Figure 2.5 below:

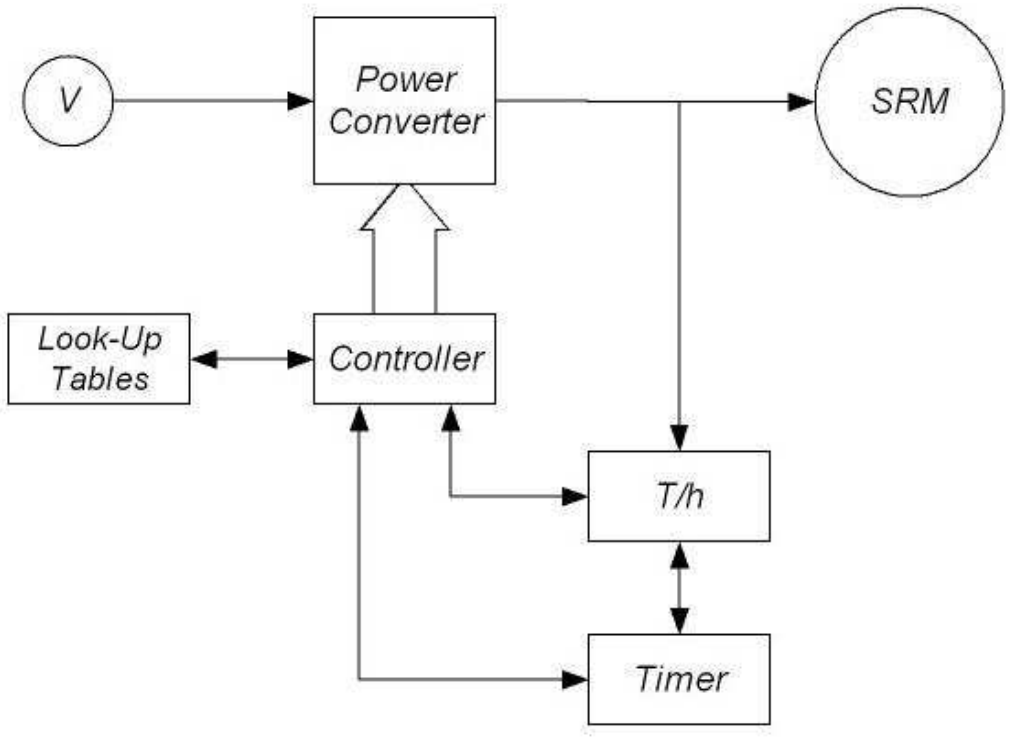

Figure 2.5: SRM Motor Drive based on Current Rise-Time and Look-up Tables

This method can determine rotor position at zero speed by timing the rise of current in phases to a predetermined level. A simple look-up table; comprised of

rotor angles for ordinates of rise-time, can be used to allow the rotor position to be determined. By comparing the possible rotor positions according to each reading for substantial agreement, a highly accurate angular determination can be made. [46] points out that this method does not require any stored flux-linkage data, that typically requires a great deal of memory space.

Up to now, only observer and other non-intelligent SRM control have been discussed. Most of the methods utilize some sort of look-up table or dynamic estimation equation to control the SRM. The problem is that look-up tables generally require large amounts of space or must be paired with an interpolation function, thus adding complexity. Neural methods improve on this by both providing a concise data mapping in addition to a tremendous interpolation capability.

2.8.4 Neural Optimization Methods

SRM performance optimization typically centers on reducing the torque ripple. In [33] and [43], neural networks are used in conjunction with rotor position sensors to optimize torque ripple. While in [40], two separate neural networks are used to optimize torque production at low or high speeds. Other sources of neural torques estimation and control include [28], [10], and [20].

Torque ripple arises when simple commutation algorithms fail to account for the non-linear torque generation of SRM's [33]. [43] utilizes a FPGA to implement an associative neural network to optimize the torque during runtime. The neural network used is a special type of associative neural memory network also know as a Cerebellar Model Articulation Controller (CMAC). CMAC networks are extremely interconnected neural constructs that have highly non-linear outputs. The neurons

are extremely simple consisting of a summation device and a binary decision function. The premise of operation is if the input presented to the neuron is in the trained recognition space the neuron outputs a "1", else it's output is zero.

This particular application construction has three inputs and two outputs. It learns the current profiles, on-line, to account for the magnetic interaction that occurs when phases conduct simultaneously. Thus, if the estimated torque being delivered differed from the commanded, the CMAC network would output the appropriate phase current or currents.

[33] presents another system for controlling the torque ripple in a switched-reluctance motor. In accordance with his implementation, a switched-reluctance motor drive system is explained that reduces or eliminates problems associated with the prior systems. Using both current and position feedback, a motor drive can optimize the runtime performance of a SRM in the presence of parametric variation.

This system employs the use of a ANN that periodically updates itself to optimize a specific parameter, torque. Most other neural methods typically utilize the off-line trained ANN for optimizing torque. As shown by Figure 2.6, an on-line, self-tuning control is preferable so as to optimize the performance of the SRM drive in the presence of variable variations. On-line tuning also provides adaptive control to maximize torque at low speeds. Unlike the other conventional NN based control schemes that use static test data for training, this NN based torque control system uses training data generated from a dynamic SRM model. Hence, the system and method control torque on an instantaneous basis rather than on an average basis, allowing an effective torque control. The effect of magnetic non-linearity in the dynamic SRM

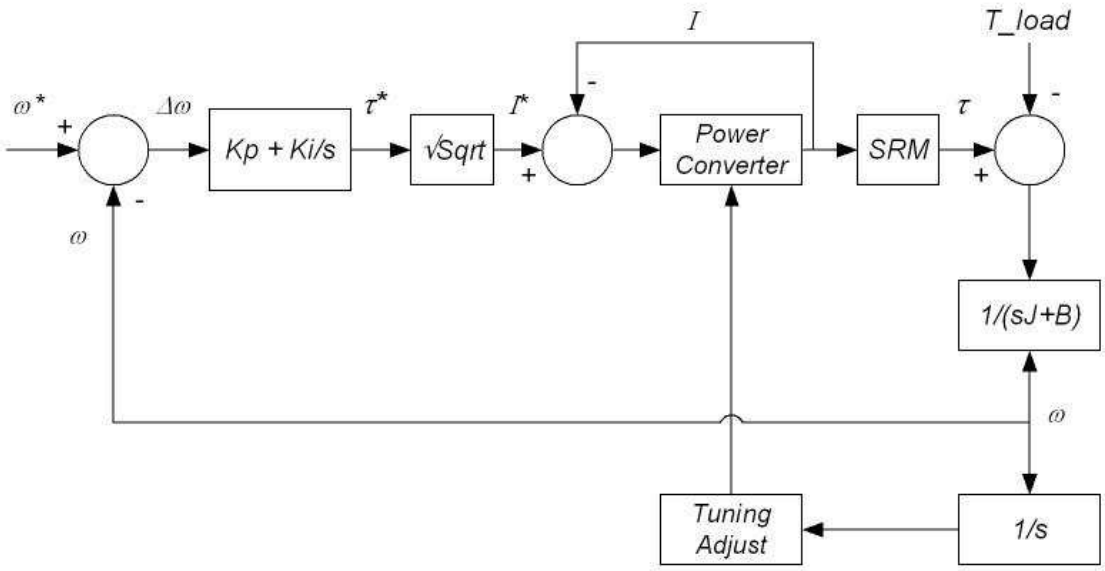

Figure 2.6: Self-Tuning Neural Net SRM Controller

model may be taken into consideration by using stored magnetization data.

Again as with most SRM controllers presented thus far, [33] combines the use of neural network control, dynamic modelling, and lookup tables to maximize the SRM performance. The next section presents neural methods which are used for sensorless motor drive operation.

2.8.5 Neural Sensorless Methods

The most prevalent body of work that exists in the field of neural sensorless methods is given by [44, 4, 34, 27]. This body of work was accomplished prior to the turn of the century. The use neural networks in sensorless motor drives have found applications ranging from ultrasonic motor drive such as in [45] to 4-Quadrant drives like that demonstrated by [14]. The focus of this survey is to overview the technology that has been developed since 2000.

Patented research that utilizes neural technologies was accomplished by [53] in 2002. A controller was implemented for a switched reluctance machine combining both a fuzzy controller and a feed-forward neural network to provide output control signals (e.g., turn-ON angle, turn-OFF angle and peak current) for controlling the energy in a switched reluctance machine.

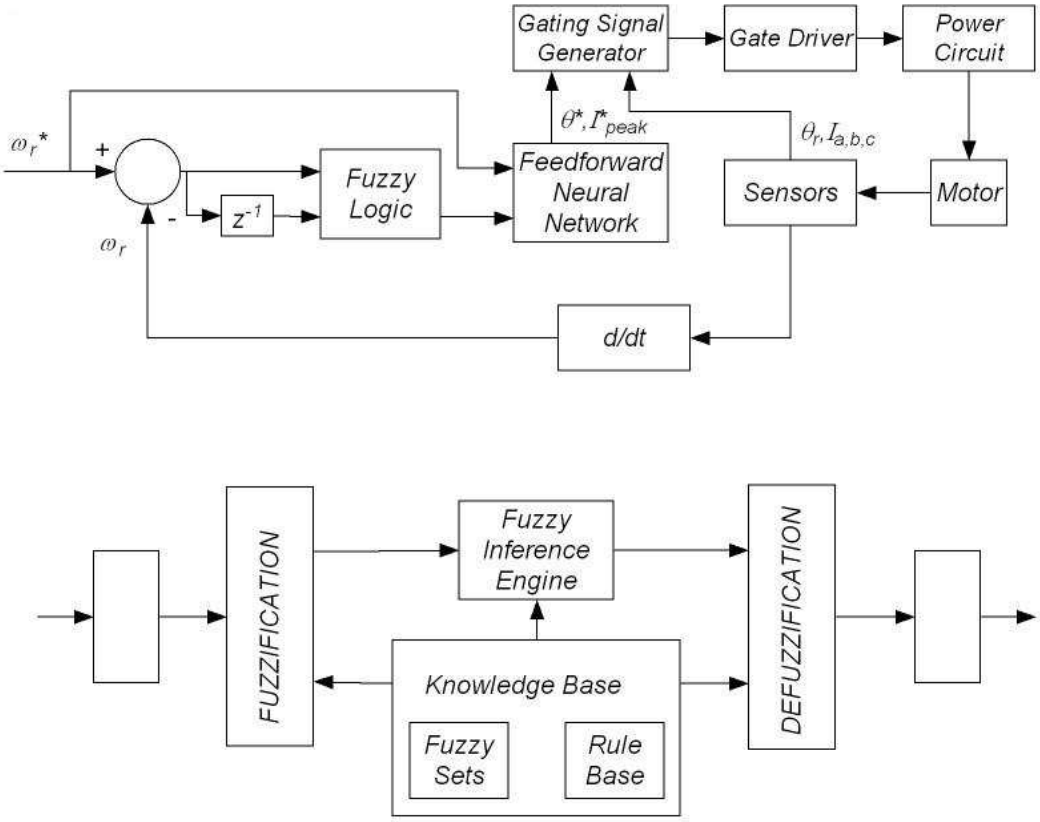

Figure 2.7: Fuzzy-Neural SRM Position Sensorless SRM Controller

The fuzzy-logic controller receives an error signal and then provides a torque control signal. The multi-layer neural network circuit receives the fuzzy-logic control circuit and a signal representing the rotational speed of the switched reluctance

machine and provides at its output control signals for controlling the energy of the switched reluctance machine.

This is a typical implementation of a torque drive, position sensorless SRM drive. The only modification to the system is the addition of a neuro-fuzzy controller in the control loop. The neural net can be seen in Figure 2.8.

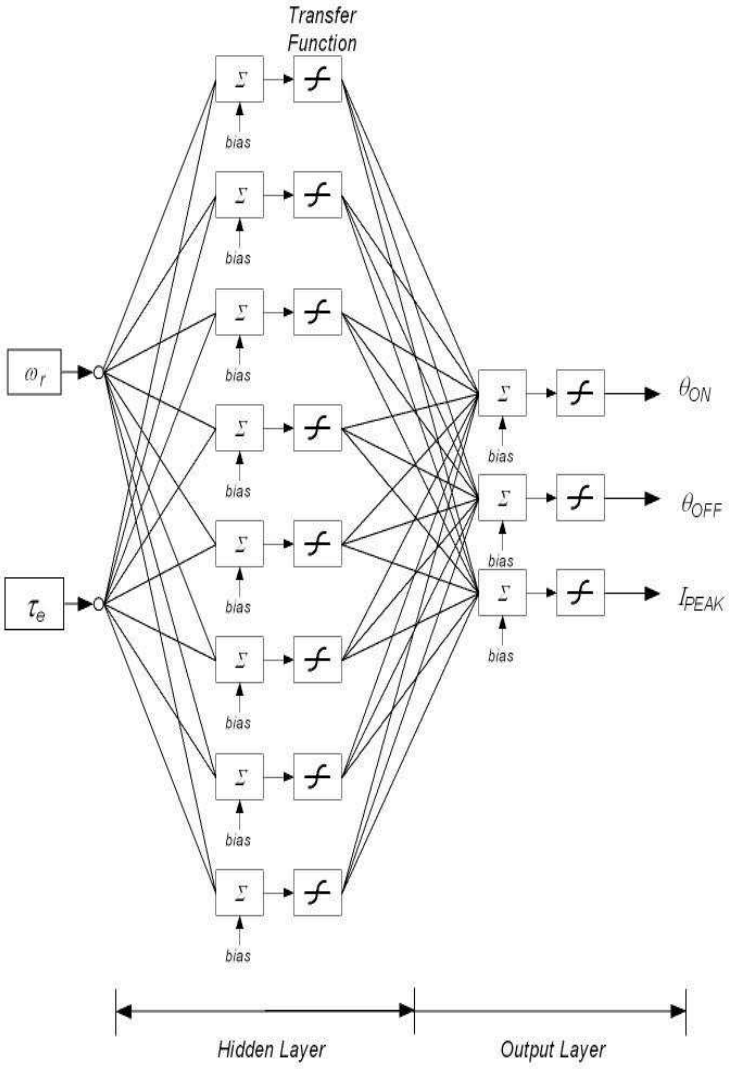

Figure 2.8: Neural Network Implementation

This network has a slightly different utilization than typical sensorless SRM neural controllers. The neural co-processing ability has been utilized to provide three outputs to the SRM drive. The outputs of the network provide commutation instances and peak current control commands. This provides control signals that can be decoded by simple logic gates, either ON or OFF. This method translates well to using more simplified neural arrangements, such as CMAC structures as discuss previously.

The most intriguing application was accomplished in 2004 by [51]. A space vector pulse width modulation (SVPWM) system was implemented using neural technology. Although an induction motor was used to demonstrate the technology, the primary goal presented was a neural speed controller. In the research, a very unique neural architecture was demonstrated that most closely resembles the current neural net development. In this application, the researcher combines 2 multi-layer neural networks together to determine the appropriate vector modulation. The neural architecture used is a 1-x-1 type. The smaller of the two nets demonstrates a 1-3-1 neuron arrangement that is used as a vector magnitude adjustment. A simple reduction in neuron count was accomplished for the purpose of function approximation. The initial neural can be thought of as non-linear preprocess that scales and offsets the input. After decomposing the input neuron to a preprocess, the reset of the magnitude network is a 3x1, which is exactly what the current research presents. Some differences should be noted.

The network shown in [51] maps a linear input to a linear output while having to preprocess 3 time versus once in the current research. The smaller network makes no attempt to model the induction motor; it simply maps the phase voltage magnitude to a space vector magnitude for control purposes. The current research presents a

very minimal neural estimator which models the magnetic relationship of a motor to its rotor angle.

As for the other, larger network, it is used to determine the pulse width for each phase in terms of turn-on and turn-off angles. The neural outputs are then multiplied together and used for the space-vector modulation. The neural controller can be seen in Figure 2.9.

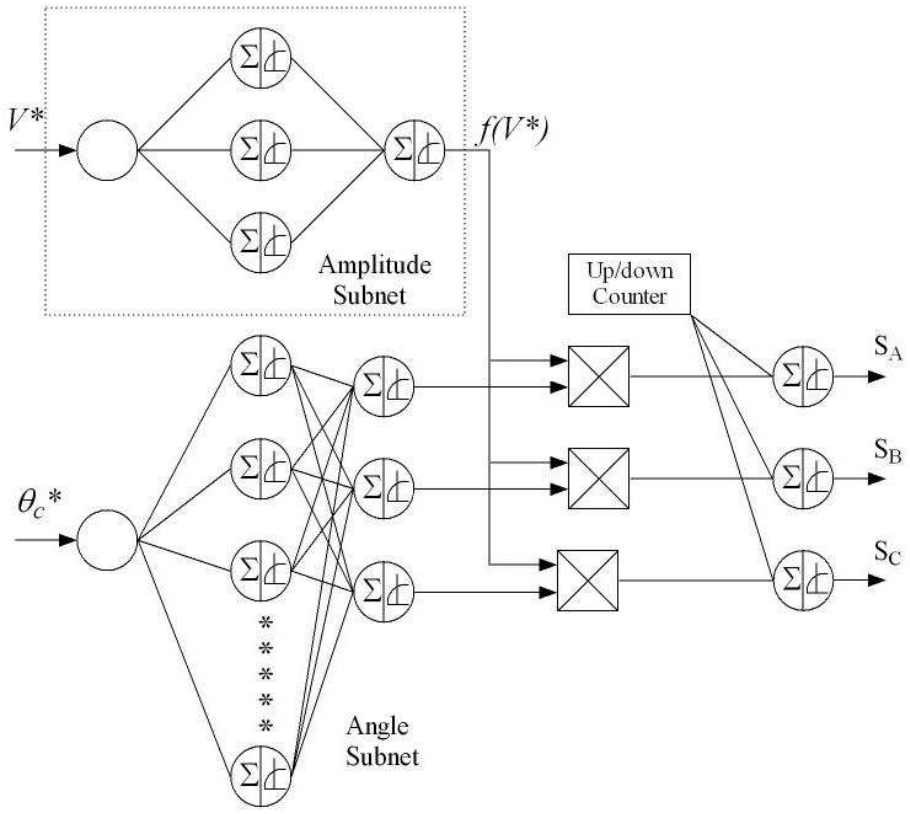

Figure 2.9: Neural Implementation of a Space Vector PWM

By keeping the neural nets to the minimum size, the controller gains the benefit of reducing the computational effort as compared to typical SVPWM schemes. The

neural PWM controllers are used in conjunction with typical PI loop controllers as seen in Figure 2.10.

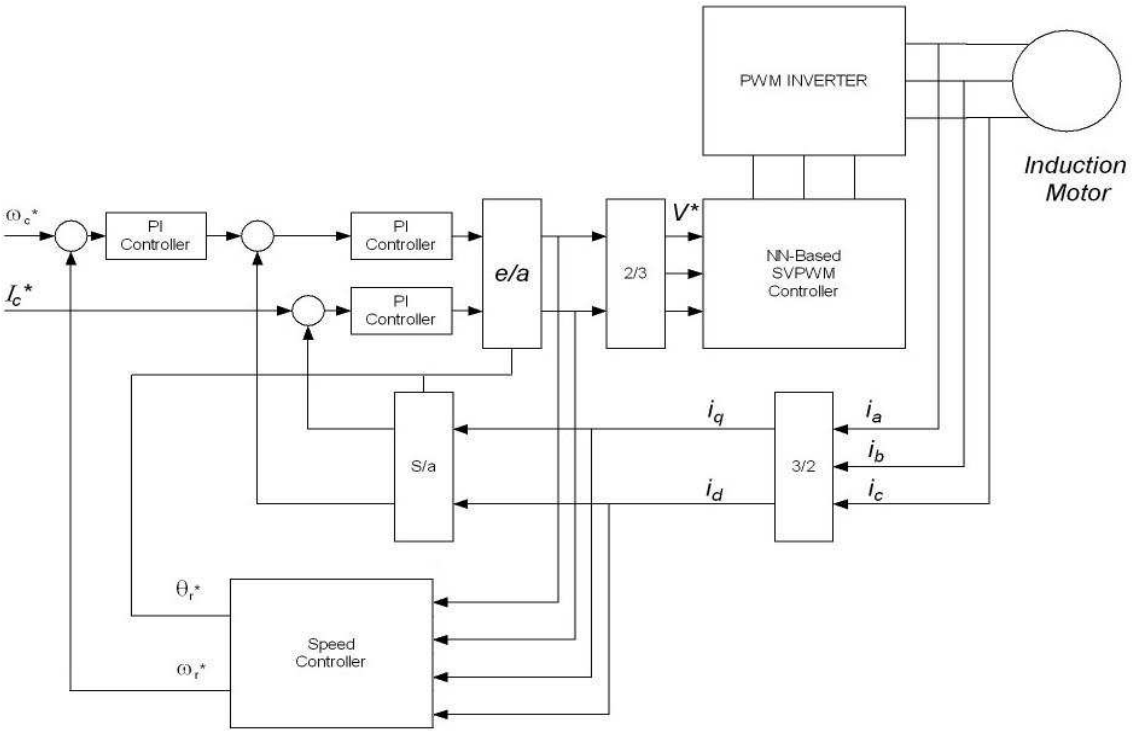

Figure 2.10: Closed-Loop Neural Controlled Induction Motor Drive System

In summary, SRM drive optimization and sensorless drive algorithms have focused specifically at solving particular problems that have arisen in previous research. SRMs are simple machines but require complex algorithms for efficient control. Most sensorless methods utilize a single algorithm to control a single parameter. Neural methods focus on torque optimization or rotor position estimation, but rarely on both. Table methods are inherently limited in use due to memory constraints and interpolation accuracy. With the advent of more powerful processors, table methods and dynamic observation are finding their implementation inaccuracies improved. Table methods,

however, will always be limited to the accuracy of the data.

The current research attempts to pull the benefits gleaned from various methodologies to achieve a robust sensorless drive control. The progress contained in this research utilizes probing methods, table methods, and neural estimation to produce a sensorless drive needing no special hardware requirements, interpolation, or complicated software observers. A size reduction for the neural network is also implemented as part of the body of work advancement thus giving an easy to implement intelligent controller that doesn't suffer from the burden of excessive calculations or the complexity of multiple layers. The reduced neural angle estimator runs the motor and is paired with a current probe algorithm to establish a start angle reference and a flux-linkage estimator for evaluation of the energy in the machine. This combination provides a completely position sensorless, closed-loop speed SRM drive implementation.

2.9 Summary

Several items from the body of research must be readdressed to stress their important nature to this thesis.

1. Artificial Neural Nets provide a streamlined method of SRM rotor position estimation
2. ANN's are only as good as the data set used to train them
3. Most ANN's that are used for rotor position estimation are multi-layer, feed-forward networks that have been trained using some back-propagation supervisory algorithm
4. Input spaces for these ANN's usually require current and flux linkage
5. Many issues with flux linkage estimation arise when using the software implementation of Faraday's Law as shown in A.4
6. Flux linkage estimation suffers from measurement, rounding, and integration error.
7. Flux linkage must be offset to corresponding values if no intrinsic method of mutual effects are considered.
8. Care must be taken during hardware and software implementation to reduce cost while increasing robustness, reliability, and accuracy.
9. Observer methods tend to be complicated and prone to error

10. Probing methods have typically used alternate, unused phases for there usefulness.

11. Table methods are subject to data accuracy issues, memory space limitations, and limitations in the interpolation accuracy.

12. Timers can be used to estimate speed and commutation instances.

13. Neural methods provide a means of online optimization and control that is not available to other traditional methods.

Chapter 3

Analysis and Development of the SRM Drive

This chapter presents an analysis of the motor and drive system in order to develop an appropriate hardware platform for the sensorless control development. This platform should include features by which experiments can be processed and the data can be retrieved. Following the example set by [48] and [39], a position sensor based drive system can quickly be implemented in such a way that the flux linkage estimator, neural net, and subsequent sensorless algorithms can be designed, developed, and implemented.

3.1 Introduction

In this chapter, a hardware platform is developed with the intent to design and implement a position sensorless SRM drive. In order to accomplish the sensorless implementation, the motor and drive system requires analysis. Specifically, the motor and converter system require modelling such that an appropriate commutation scheme is able to be developed. This scheme should provide, at a minimum, enough torque

to turn the rotor in the desired direction.

In addition to the basic drive operation, the motor and converter are to be analyzed dynamically such that an approximation of the drive operation can be developed. Further, those dynamic system equations should be linearized so that current and speed controllers can be developed. All of the system dynamics should be modelled and simulated prior to final hardware implementation. Please refer to Chapter 6 for a full explanation of the dynamic system simulation.

SRM electrical and mechanical dynamics are highly non-linear. However, linear approximations of the motor and converter should provide an acceptable means to develop linear feedback control. Current and speed control loops are typically implemented with a proportional + integral controller. The open loop system will be analyzed and the controllers will be designed and implemented on the DSP system.

Typically a digital SRM drive consists of a motor, converter drive, position sensors, interface electronics, and a digital controller. Referring to Figure 3.1, we can see that the SRM requires a converter drive. To properly control the converter, several support systems are required. First, interface electronics are needed to condition signals, such as sensed winding current. Secondly, the interface electronics should provide gate drive signals to the converter electronics. All other functions like the P+I controllers or the commutation scheme can be processed digitally by the microprocessor.

In sensor based systems, hall-effect sensor signals determine the position of the rotor. From that information, the DSP can determine the appropriate angular intervals by which a PWM drive signal can be applied, producing current in the phase

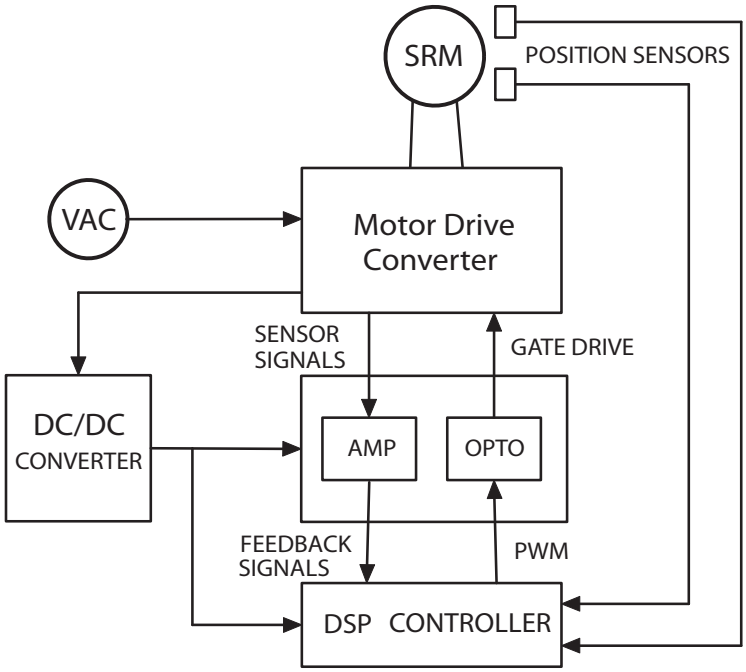

Figure 3.1: Sensored SRM Drive Block Diagram

winding of the SRM. A current control loop provides protection to prevent the current from running away during low winding impedance intervals. The speed loop provides the reference command for the current loop. This arrangement is referred to as a torque, or current, driven motor drive. From the analysis of said system, we can determine the modes of operation and, further, the appropriate conditions necessary for sensorless operation.

3.2 Theory of SRM Operation

Switched Reluctance Machines (SRM) are very simple to build, thus very attractive to manufacturers. Yet, due to their simplicity, they require advanced control methods to produce useable torque. As with all reluctance type machines, SRM's require current to be driven into their phases at particular angular intervals of the rotor. Usually

the Quadrant I torque producing interval occurs when the rotor is on approach to the stator pole, refer to 3.2. The inductance profile has a relationship to the sign of the torque produced in the motor, or vice versus. On approach from un-alignment to alignment, the inductance of the phase is increasing; this is the Quadrant I torque region if current is also positive.

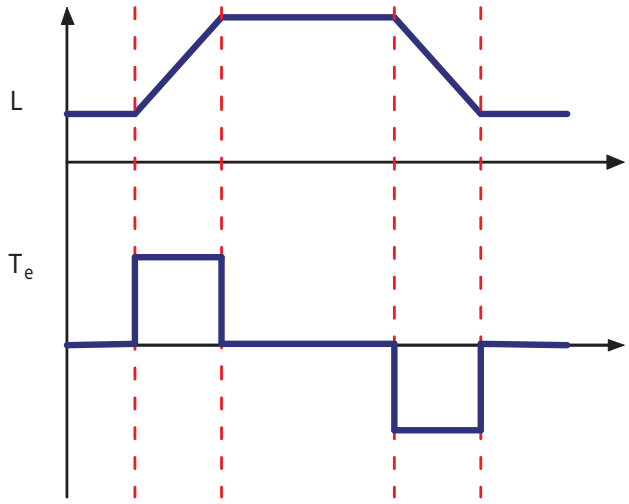

Figure 3.2: Torque vs. Inductance Profile
[39]

The reverse occurs on procession away from the poles, the negative torque region. In order to produce torque to turn the motor, current should either flow through the stator during the positive torque region or negative torque regions; not both. The motor will stall if current is allowed to flow in the stator during both regions (quadrants I and II). The dynamic relationship of the inductance profile can therefore be deduced in the form of:

$$T_e = \frac{1}{2} i^2 \frac{dL}{d\theta} \tag{3.1}$$

where the induced electromagnetic torque of the motor is proportional to the square of the winding current. And as said prior, the sign and magnitude is determined by the inductance profile slope. As an aside note, it is a non-linear relationship that is shown by 3.1, comparing current to rotational force of the motor.

3.3 Dynamic Motor Model

Construction of the included motor can be seen in 1. This construction is referred to as a 4x4x4 machine; 4 rotor, 4 stator, and 4 auxiliary poles. The 4 stator poles, as well are the auxiliary poles, are all connected in series. The arrangement of the converter drive can be seen in Figure 3.3. This drive is considered to be a single phase motor drive. However, the auxiliary poles are capable of producing torque and are subject to re-circulating current during drive operation. Thus, the system is, in fact, a quasi-single phase drive.

AC voltage feeds through a rectifier and filter to produce a DC bus voltage from which the drive operates. A gate drive signal is provided to turn on-off the main semiconductor switch. Thus two states occur during drive operation that can be used to derive the appropriate dynamics of the system switch-off and switch-on.

3.3.1 Switch On State

During this state the full Vdc bus voltage is applied to the main stator pole windings. This condition generates the most current through the winding. Due to the lack of back-emf, the current would quickly reach dangerous levels as the dynamic impedance

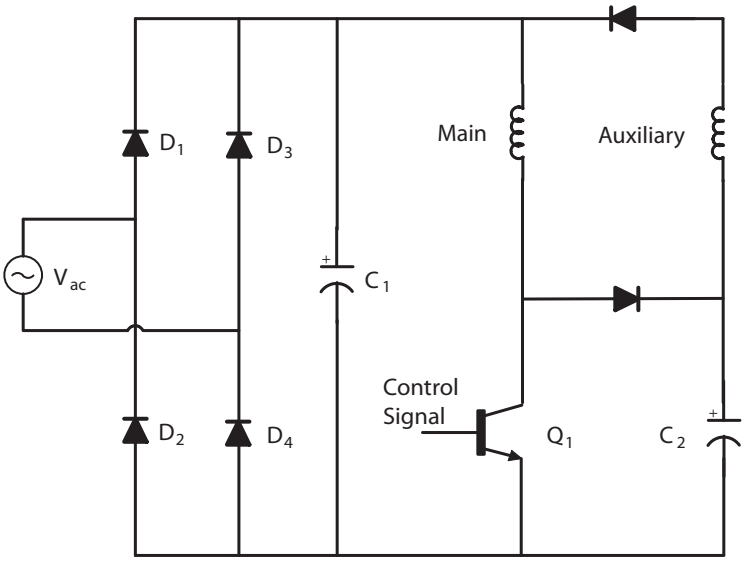

Figure 3.3: Converter System Diagram

of the stator winding is insufficient to sustain the full DC rail. Because of this a current controller is required. Diodes D5 and D6 insure that current flows only in the specified directions. In particular, the capacitor discharges any energy it has stored in previous cycles through the auxiliary winding and back to the source.

Equations describing the motor when the switch is turned on are:

$$\frac{d\lambda_a}{dt} = V_{dc} - R_a i_a \tag{3.2}$$

$$\frac{d\lambda_b}{dt} = V_c - V_{dc} - R_b i_b \tag{3.3}$$

$$\frac{dV_c}{dt} = \frac{i_c}{C} = -\frac{i_b}{C}, \text{ where } i_b \geq 0 \tag{3.4}$$

$$i_a + i_b + i_c = 0$$

$$J\frac{d\omega_m}{dt} + B\omega_m = T_e - T_l \tag{3.5}$$

where, J is the inertia of the rotor (Kg-m^2), B is the load constant, λ_a and λ_b

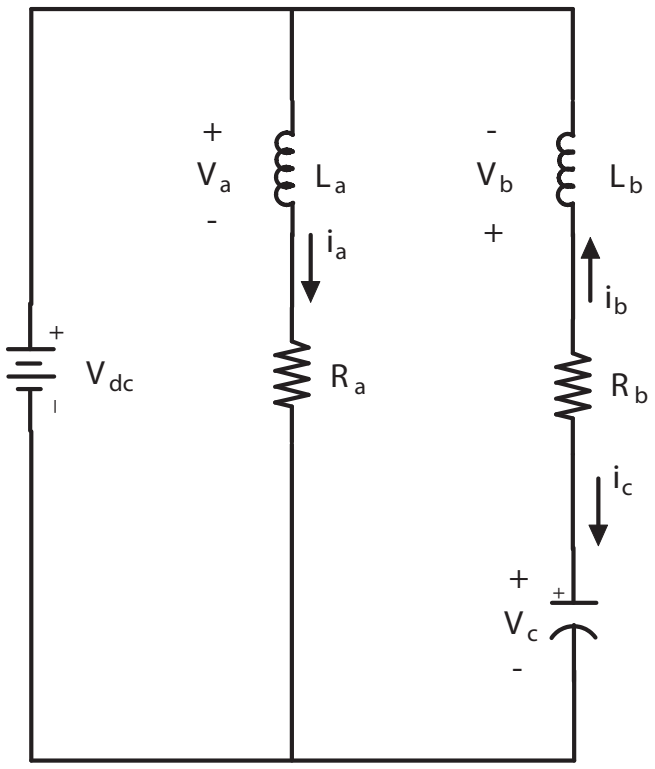

Figure 3.4: Converter Diagram with Main Switch On.

are the flux-linkages (V-s) of the main and auxiliary windings, respectively. i_a, i_b, i_c are the currents through the main windings, auxiliary windings and the capacitor, respectively. C is the capacitance of the energy storage capacitor. T_e and T_l are the electromagnetic torque (N-m) and load torque, respectively.

3.3.2 Switch Off State

Referring to Figure 3.5, we can see that the current that was flowing through the main switch continues to flow, but through D6 rather than the main switch. This current in turn charges the the capacitor while also recirculating through the auxiliary winding.

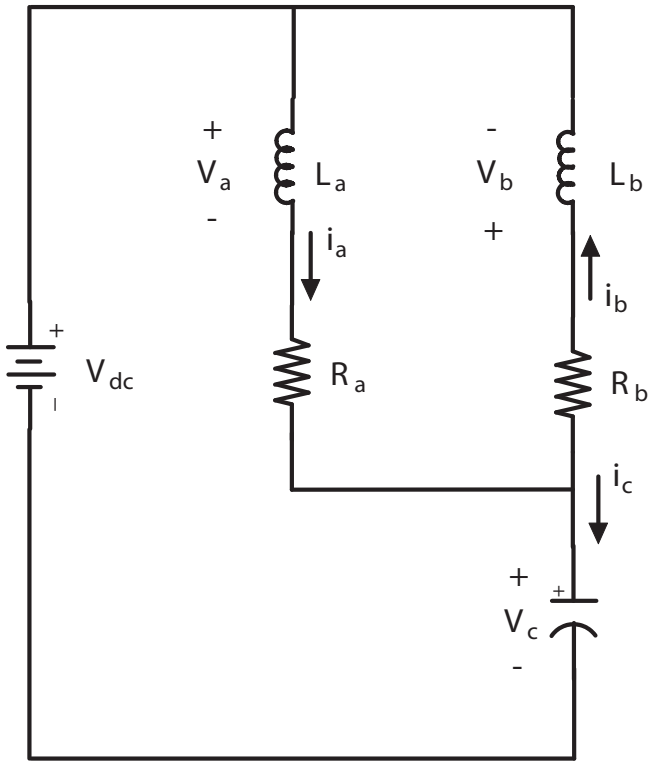

Figure 3.5: Converter Diagram with Main Switch Off

This recirculating current is the one major drawback of this particular topology. The discharge of current during non-conduction periods is passive. No negative voltage is applied to the winding to force the current to zero as fast as possible. At slow speeds the current decays to zero, and is set by the inductive time constant of the stator poles. The decay duration is insignificant to the total conduction time thus produces no significant negative torque. However as speeds increase, the time interval between conduction periods reduces, resulting in a conflict with the discharge time constant. The overall recirculating current starts to produce significant negative torque as compared to the positive torque producing current. The non-zero re-circulating current will prove to a formidable challenge when developing the flux linkage estimator in Chapter 4

The differential equations of the machine when the controllable switch is turned off are derived as:

$$\frac{d\lambda_a}{dt} = -R_a\, i_a + (V_{dc} - V_c) \tag{3.6}$$

$$\frac{d\lambda_b}{dt} = -R_b\, i_b - (V_{dc} - V_c) \tag{3.7}$$

$$\frac{dV_c}{dt} = \frac{(i_a - i_b)}{C} \tag{3.8}$$

$$i_a + i_b + i_c = 0$$

$$J\frac{d\omega_m}{dt} + B\,\omega_m = T_e - T_l \tag{3.9}$$

The the applied winding voltage during the off state is the difference between the DC bus voltage and the capacitor voltage. This condition is not present during the on state. This state is important to note when flux linkage estimation needs to be derived.

Further, the speed of the motor is proportional to the difference in the electromagnetic torque and the load torque. As stated earlier, the electromagnetic torque is proportional to the square of the winding current.

3.4 Linear Analysis and Controller Design

The dynamical equations contain non-linear elements such as the relationship between flux linkage and current. Most variables are dependant upon rotor position. The impedance of the stator winding changes with respect to rotor position and current level. It is not the scope of this paper to define the relative quantities of the

current, torque, flux linkage, and rotor position relationship. This research will, however, exploit the derived models and quantities therein to develop the control and simulation.

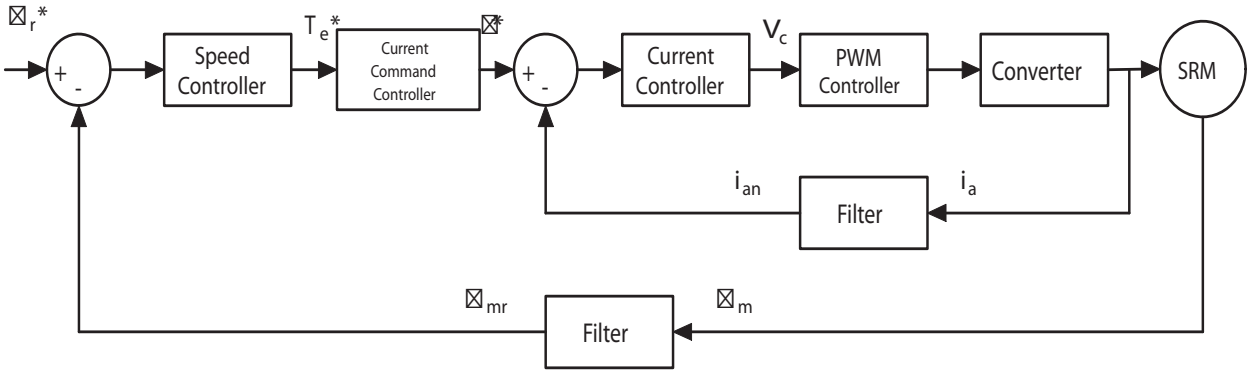

Figure 3.6: Closed-Loop Model of SRM Drive [25]

The overview of the linear model of the drive system can be seen in Figure 3.6. There are two main loops, current and speed. What is not obvious is the interdependency of the two loops. The results of the linearization can be seen in Figure 3.7. The small signal model of the SRM has been simplified to resemble that of a DC motor. The non-linear components have been replaced with average values. The model of a DC motor has interdependent loops. The mechanical dynamics affect the electrical and vices versus. The importance of the linearization is that the loop interdependence has been eliminated in order to provide a model by which a mutually exclusive design can be performed. The current loop controller can be designed independently from the speed loop controller.

The current loop is shown in Figure 3.7 (a), the speed in 3.7 (b). A full small signal model derivation is contained in the appendix and was derived from [25] and simplified by methods demonstrated in [24].

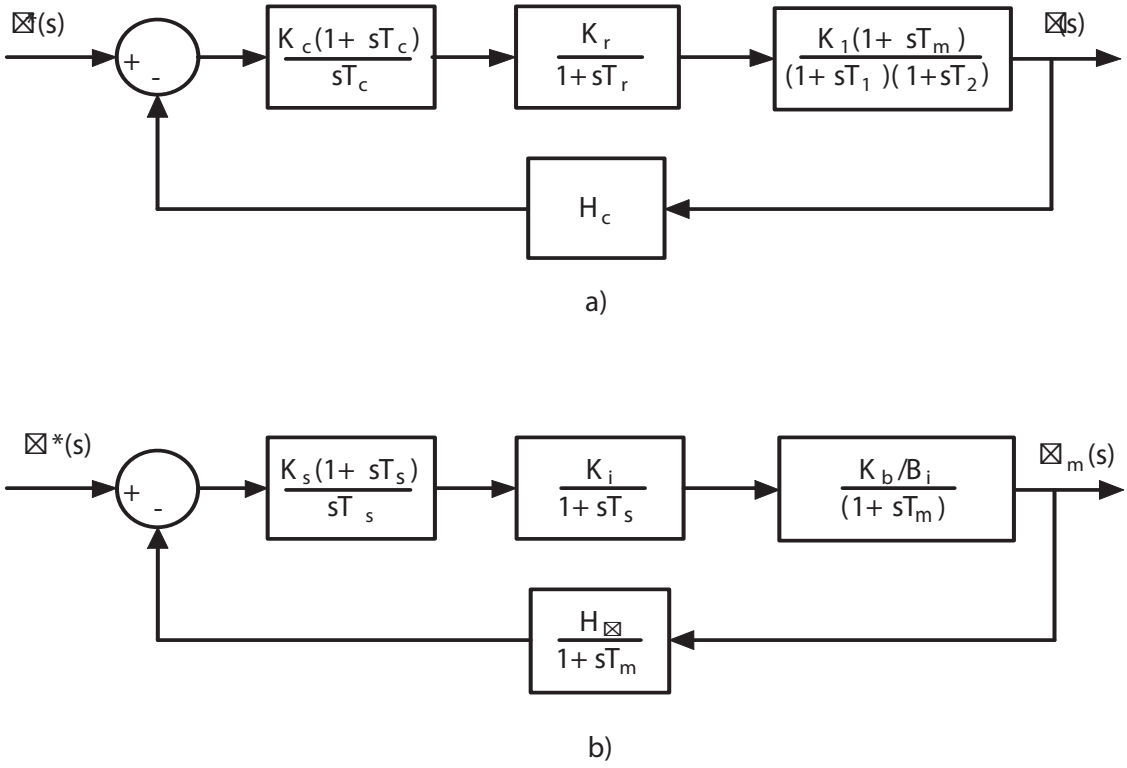

Figure 3.7: Simplified Linearized Models of Current and Speed Loops [25, 39]

3.4.1 Current Controller Design

The proportional + integral current controller loop is given by the block diagram in Figure 3.7 **(a)**. The purpose of this section is to determine the parameters K_c and T_c through analysis and/or root locus design. Please refer to the appendix section A.1 for a full derivation.

3.4.2 Speed Controller Design

The proportional + integral speed controller loop is given by the block diagram in Figure 3.7 **(b)**. The purpose of this section is to determine the parameters K_s and T_s through analysis and/or root locus design. Please refer to the appendix section A.1

Table 3.1: Comparison between Speed and Interrupt Cycle Count

RPM	**rad/sec**	**Int Cycles**	**Scale(r/s)**
10000	1047	15	15705
5000	524	30	15720
4000	419	37	15503
3000	314	50	15700
2000	209	75	15675
1000	105	150	15750
600	63	250	15750

for a full derivation.

3.4.3 Speed Estimate Function

The speed feedback signal developed in the prior subsection is actually an estimate. Initially, the speed estimation function was derived using the hall-effect rotor position sensor signals. There are two sensors that are placed 45 degrees apart on the motor frame in order to achieve discrete quadrature position information; i.e., the rotor position is know at discrete intervals. To produce positive torque, a boolean function can be used to discern the angular region to apply the PWM to the switch gate. For clockwise rotation, the discerning function is an XOR function. In order to estimate the speed from the XOR signals, a simple software counter can be setup to count the number of interrupt cycles between rising edges of conduction region. The speed is proportional to the inverse of this time. To convert back to radians per sec, the inverse of the timer must be scaled by an appropriate value.

Looking at Table 3.1, several speeds were chosen and then converted to radians per second. The approximate interrupt cycles can be derived by the time it takes the rotor to rotate 90 degrees divided by 50uS. The use of 50uS will become apparent in

the following sections, but for now this is the basis of the sampling interrupt interval. The results of the calculation can be found in column 3. We can find the scaling factor at each speed by multiplying the speed by the count cycle. Finally, we see that the scaling factor at each speed is roughly 15700. We can therefor derive the final speed algorithm for firmware implementation as follows:

$$Speed \approx \frac{15700}{InterruptCycleCount}, \frac{rad}{sec} \tag{3.10}$$

3.5 Hardware Development

The motor system consisted of a digital signal processor, interface analog electronics, and power electronics that serve as the motor drive. The interface electronics can be viewed as incoming and outgoing channels through which isolation can be maintained between the DSP and the power electronics. An outgoing channel consisted of gate isolation and was implemented via opto-coupler. A 15V drive level was maintained to insure robust operation of the main semiconductor switch. An on-board DC/DC converter was developed to insure that 15V, 5V, and 3.3V was available to the individual systems requiring the voltages. The hall-effect position sensors were buffered and translated from 15V to 3.3V through logic level transistors. Incoming channels consisted of amplifier electronics used for feedback. These circuits maintained signal voltage levels at or below 3.3V so as to be compatible with the DSP's analog-to-digital converter(ADC). The DC bus voltage and the voltage across the energy storage capacitor were sensed using voltage dividers. The signal was scaled 100:1 by

the divider and then buffered by operational amplifier circuits prior to connection to the DSP's analog-to-digital converters. The current, sensed via a hall-effect device, was also amplified and buffered via operational amplifier circuits. Minimal filtering was implemented on all of the feedback signals as to keep group phase delay low.

Note: For a complete description of the hardware system, please refer to the appendix section A.3

3.6 DSP Firmware Environment

A DSP timer interrupt was setup to give a duty-cycle controllable, 20-khz, PWM signal to drive the main semiconductor switch. The interrupt cycle provided a means to obtain a repetitive sampling interval. The digital linear controllers were a better fit for implementation in this system due to the consistent sampling interval. In our case, the sampling interval was 50uS. All calculations had to be completed within this time period to maintain proper system control.

Automatic DSP timer flags were setup so that the feedback signals were converted at either the midpoint of the PWM on-time or the midpoint of the off-time. To avoid post-processing, the feedback signals were sampled at the midpoint to insure the software could use the approximate average of these signals. Also, over-sampling was implemented wherever possible to reduce the effects from noise present in the drive system.

3.7 Data Collection

The data collection methods allowed for the storage of feedback variables in addition to any internal DSP variable necessary for debug and analysis. First and foremost,

it is necessary for debug purposes to have a rotor angle baseline by which captured runtime data, such as current or flux linkage, can be compared to or referenced in time. Accurate rotor position information is also important in order to develop and train a neural network for the estimate rotor position.

The DSP software development environment provided the means to save DSP memory locations to file. A function was written such that an array was used to store critical information as it was processed during runtime. The experimental data could be retrieved readily from the hardware platform, formatted, and used in external programs such as Matlab or Excel for analysis. The neural training data set was derived from experimentally captured results in the array.

3.8 Summary

The hardware and software were initially developed to run the SRM using position sensors. The initial software control algorithm was developed by commutating the phase windings with a flag variable set by a XOR function. The XOR function decodes the hall-effect sensor signals and derives that active conduction interval for stator windings to produce positive torque in the desired direction. It should be noted that the sensorless algorithm required that it's output emulate the same control over the commutation flag variable that was originally controlled by the XOR function. By controlling that flag variable, the speed estimate derived previously can still be used, only referenced to the sensorless controlled commutation period rather than the XOR decoded sensor signals. Please refer to Figure 3.8, for an outline of sensored firmware motor drive control.

Figure 3.8 demonstrates the basic sensored drive software algorithm that was implemented modularly so that future functions could be added without interfering with the original drive code. The flux linkage estimator, neural net, sensorless algorithms were all developed, implemented, and tested by insertion into this basic code format.

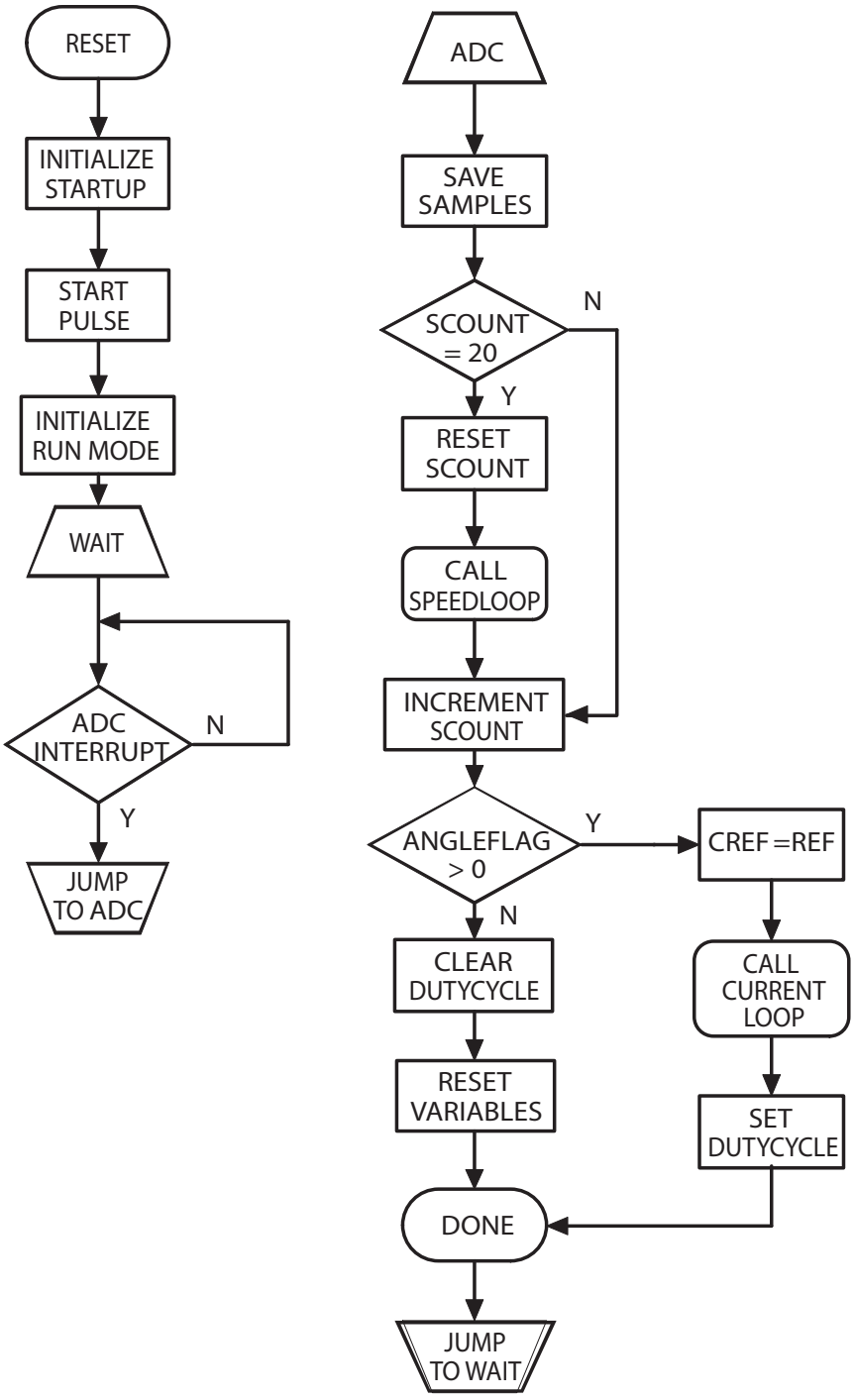

Figure 3.8: Basic Firmware Flow for Sensored SRM Control

Chapter 4

Analysis and Development of the Flux Linkage Estimator

This chapter introduces the concept of flux linkage estimation. Specifically, the analysis, derivation, and implementation of a flux linkage estimator specifically designed to be included in the motor drive firmware. This chapter presents the derivation of the algorithm from the sampling and scaling of the sensed signals to the dynamic equation implementation on the DSP controller.

4.1 Introduction

As discussed in previous chapters, flux linkage is a critical parameter used in the estimation of SRM rotor position. The current, flux linkage, and rotor position are intertwined in a non-linear relationship which is the foundation of most research involving SRM's. Also previously discussed, flux linkage cannot be measured directly. It must be derived by using either analog electronics or mathematically on microprocessors. As discussed previously, a primary goal of this research is to keep hardware costs to a minimum. Thus, a software approach using a DSP was chosen to implement

the flux linkage estimator. Most previous research has established marginal success in this area and it is the hope of this research to improve on the shortcomings of prior methods.

Flux linkage can be estimated mathematically using Faraday's law:

$$\lambda = \int v - R\, i\, dt \tag{4.1}$$

This equation implies that flux linkage is directly related to the instantaneous applied winding voltage minus the resistive drop due to the excitation current. The outcome of the integration is the applied volt-seconds to the motor phase. In this case, the converter only actively applies voltage to the main stator phase; hence the single-phase nomenclature for the system. The implementation herein pertains to estimating the flux linkage that is in the main phase of the machine. The auxiliary winding is assumed to have negligible contribution to the machine's overall flux linkage and therefore is totally ignored. At low speeds this was shown to be true [30], but with increasing speed the mutual effects grow. However, if the flux linkage estimates are consistent from runtime to runtime over the entire speed range across all operating points, then the mutual effects can still be ignored. Even though the flux linkage may deviate from actual values, it can still be used to train the neural net to accurately predict rotor position. However, it is still the intent of this research to accurately represent the flux linkage. It is just noted that the neural net is not sensitive to the accuracy, but the consistency, of the information used in training. In other words, the flux linkage must be extremely consistent, not accurate, during motor operation.

4.2 Converter Analysis for Estimation

Previous software methods attempted to measure the winding voltage directly as in [35]. The nature of winding voltage is directly comparable to the PWM pulses used to drive the switches. During current regulation, the applied winding voltage pulses can be extremely narrow. This drives the demand for a very fast analog to digital converter. To get around this issue, analog methods were introduced using external hardware. This negates any savings or ease of implementation that the software approach otherwise offers.

The problem related to measuring PWM controlled winding voltage is addressed in the current research. It will show that winding voltage can be derived from continuous signals that are readily measurable. This eliminates the aforementioned problem during current regulation, and provides a clean implementation using the software method.

As derived in the previous chapter, the flux linkage for the main winding can be expressed equationally for two different converter states, switch-on and switch-off. These are expressed as:

$$\frac{d\lambda_{on}}{dt} = V_{dc} - R_a\, i_a \tag{4.2}$$

$$\frac{d\lambda_{off}}{dt} = (V_{dc} - V_c) - R_a\, i_a \tag{4.3}$$

We can combine equations 4.2 and 4.3 by adding the variable of PWM duty. For

the switch-on period, the PWM duty can be represented by: **d**; conversely, the switch-off state PWM duty by: **1-d**. By substituting these into 4.2 and lamoff, respectively, we get the flux linkage for the on-time:

$$\frac{d\lambda}{dt} = d\ (V_{dc} - R_a\ i_a) \tag{4.4}$$

and the flux-linage for the off-time:

$$\frac{d\lambda}{dt} = (1-d)((V_{dc} - V_c) - R_a\ i_a) \tag{4.5}$$

And if we determine that the combination of the two demonstrates the total average flux-linkage for the entire switching period as given by:

$$\frac{d\lambda_t}{dt} = d\ (V_{dc} - R_a\ i_a) + (1-d)((V_{dc} - V_c) - R_a\ i_a) \tag{4.6}$$

We can then simplify....

$$\frac{d\lambda_t}{dt} = d\ V_{dc} - d\ R_a\ i_a + (1-d)V_{dc} - (1-d)V_c - (1-d)R_a\ i_a \tag{4.7}$$

$$\frac{d\lambda_t}{dt} = d\ V_{dc} + (1-d)V_{dc} - (1-d)V_c - d\ R_a\ i_a - (1-d)R_a\ i_a \tag{4.8}$$

$$\frac{d\lambda_t}{dt} = V_{dc} - (1-d)V_c - R_a\ i_a \tag{4.9}$$

Finally, we can integrate both sides to get the final algorithm function to implement:

$$\lambda_t = \int (V_{dc} - (1-d)V_c - R_a\, i_a)\, dt \tag{4.10}$$

Equation 4.10 shows us that the flux linkage estimate can be derived from sampling the DC bus voltage (V_{dc}), the energy storage capacitor voltage (V_c) during PWM off-time, and the main phase current (i_a). This is a significant result since all of the above signals are continuous and unipolar. The signals can be sensed using simple means such as resistive voltage dividers and hall-effect current senors. Scaled, unipolar signals can be presented directly to the ADC's present on the DSP controller.

What is not explicit in the derivation is the need for an offset of the flux linkage during operation. The converter topology lends itself to having non-zero current in the windings at the time the current control is required for positive torque generation. This insures the flux linkage is also non-zero. At the instant the PWM signal starts to actively control current through the main winding, the flux linkage needs an initial offset. The offset will provide the necessary adjustment for the flux linkage integration to track properly.

4.3 Estimator Implementation

The hardware to software interface is shown in Figure 4.1. The sensed signals are routed to an ADC and then reconditioned into their absolute values $x100$. The $x100$ scale was used to negate the effects of rounding error. Another factor that affects estimation accuracy is when the resistive voltage drop magnitude is comparable to the magnitude of the average winding voltage. This relatively low dynamic level can be negated with more decimal places. However, two significant digits behind

the decimal place was deemed acceptable level of accuracy tradeoff to the overall maximum dynamic range to be represented in the fixed point processor.

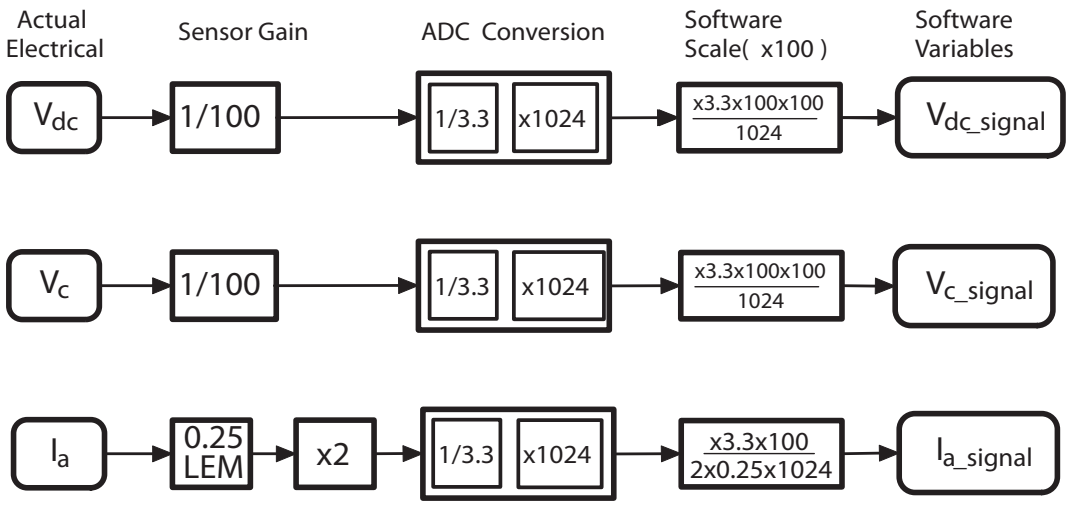

Figure 4.1: Signal Scaling used in Flux Linkage Estimation

The samples were synchronized to the PWM signal in order to assure an average signal could be represented easily. Two up-down timers were used to set the events to sample at the middle of the switch for both on-and-off time. Since no pre-filtering was used in the design, each of the signals were sampled twice at their respective time and then averaged to negate the effects of noise. The dc bus voltage and current were sampled at the midpoint of the on-time, while the capacitor voltage and dc bus voltage were sampled at the midpoint of the off-time. This midpoints were used to approximate the average of the signals. Since significant noise existed on the DC bus voltage, the samples from on and off times were used in the calculation of the average DC bus voltage. The over-sampling arrangement was not demonstrated in Figure 4.1 because it was deemed unimportant to the overall performance and implementation of the flux linkage algorithm shown in Figure 4.2.

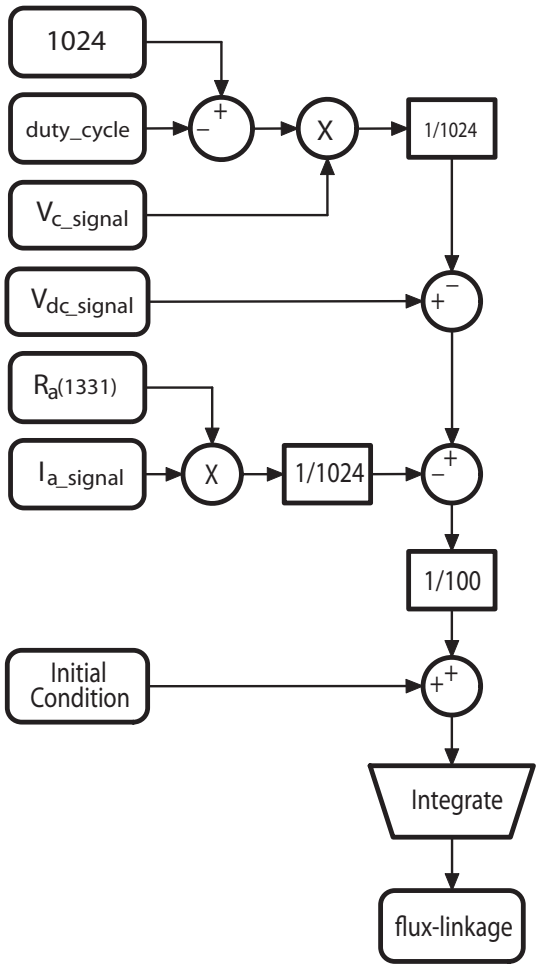

Figure 4.2: Implementation of Flux Linkage Estimator

Please note that all quantities used a $Q10$ format on the DSP and thus any multiplication inserted an extra 1024 scale factor that needed to be divided back out. The division was performed by a right shift of the variable by 10 places. Also just prior to integration, the x100 scale was divided out in order to reduce the dynamic range required for long term numerical integration that occurs at slow speeds as indicated by [15]. This division was carried out by essentially multiply by 1.6 and right shifting by 4 places (i.e. divide by 16); this procedure was carried out twice to achieve a

divide by 100.

The offset of the flux linkage at the initial current control instant was implemented by a lookup table. The table was compiled from FEA simulation values at an assumed angular reference of zero degrees. The active current control region should begin at full alignment which is the zero degree reference. Thus the table represents flux linkages at various current levels at a rotor position of zero degrees. If advanced angle current control is to be used, the flux linkage offset table should also include values for those angles.

4.4 Validation of Estimation vs. FEA Results

The results of the flux linkage estimation were collected at starting speeds and at steady state speed using a variable saving algorithm. Several current reference levels were chosen and the algorithm was implemented to collect current, flux linkage, and actual rotor position.

The algorithm attempts to scale the sampled current and voltages to represent the actual measured values as close as possible. A decimal shift of two places to the left just prior to integration serves to insure the flux linkage is also on an absolute scale. However, the DSP implementation is fixed point. Thus the time has been normalized out of the actual calculation. So the raw data had to be re-scaled by adding the integration time back into the value. Thus a $50uS$ was divided into the data.

The re-scaled flux linkage data was compared to FEA simulation data generated by [48]. The comparison is shown in Figure 4.3. Immediately one can see that the surface is relatively accurate at low values of rotor position and flux linkage but

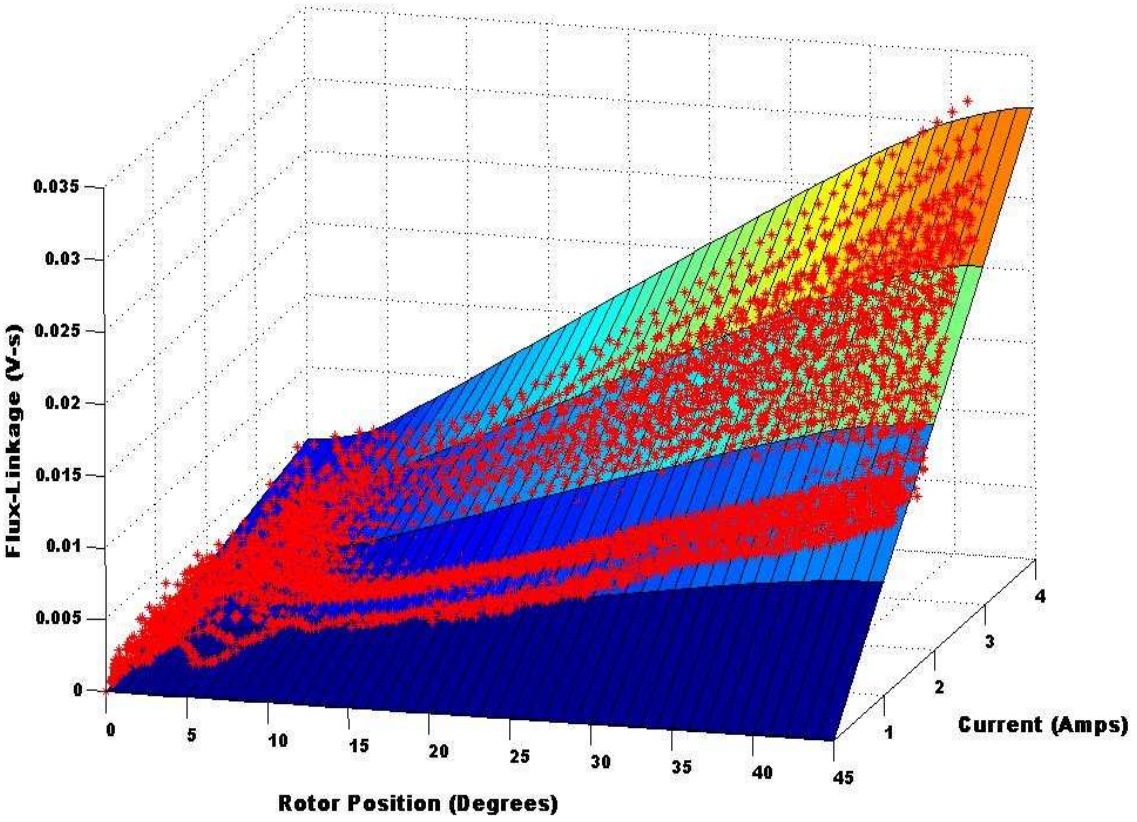

Figure 4.3: FEA Surface vs. Experimental Data

starts to deviate as rotor angle increases. This is partly due to measurement error and integration duration. The other outstanding feature is the error at high currents and low currents. At the low currents the error is due to the measurements while at high currents the error is due to increasing mutual effects. However, the most important realization is that the surface contains both low and high speed data. Thus little deviation due to long integration times occurs. **The non-variant nature of the flux linkage surface provides the ability to proceed to train a neural net, even in the presence of error, and still have accurate rotor angle prediction.**

The integral, however, was periodically reset to insure that the errors that did exist did not compound from conduction cycle to conduction cycle.

4.5 Summary

It was shown that a reasonable flux linkage estimator can be derived for software implementation. The estimator maintained consistency across the entire operating range of current and speed. Also, the estimator reproduced the same results at the same operating point. It can only be concluded that any mismatch that occurred in the results, as compared to the ideal FEA simulation results, are due to measurement error, round-off error, or mutual effects. The last being a potentially significant part of the deviation due to the recirculating current in the windings at higher currents and speeds. The main reason a consistent input is critical to the neural net is so that the estimate of the rotor angle is consistent for a given current and a given flux linkage. If there existed two entirely different flux linkage values for any given current and rotor position caused by speed changes, the surface would seem to be time variant, resultant in an unusable rotor position estimate. Thus, the time invariant nature of the resulting implementation allows us to continue to develop the sensorless controller.

Chapter 5

Development of the Minimal Artificial Neural Network

This chapter presents the main drive of the research, the artificial neural network. This chapter presents the development and evaluation of the neural net used in rotor position estimation. Its performance, as compared with other topologies, is applied to this particular system. The training and performance error analysis is also presented.

5.1 Introduction

As previously discussed, the intent of the research is to present a neural net that has a greatly reduced computational burden. This provides the ability to implement networks on less expensive micro-controllers. A result of the reduction is the ability for more complicated control algorithms to co-exist on the processor.

The concept of adding an additional non-linear input in order to reduce the network neuron count while not losing the function approximation accuracy was presented in chapter 1. In this chapter, we present the reduced neural net, the training set, and a comparison between typical neural implementations. It is further shown

that a non-linear input does indeed provide the ability to greatly reduce the number of neurons without losing rotor position estimation accuracy. Thus the computational burden is greatly reduced while maintaining drive robustness.

5.2 Data Adjustment for Neural Training

The hardware and DSP platform developed in Chapter 3 was used to collect training data during motor runtime. The data consisted of current, flux linkage, and rotor position. These were obtained at evenly spaced current levels at both the startup and steady state of the motor. For the purposes of neural training, the data set collected was pared down by 90 percent, thus 10 percent of the total collected data points were used to train the network. As with most data used for training neural nets, the experimental surface required normalization. The normalized training surface can be seen in Figure 5.1.

The normalization process was based on some assumptions of the motor and its maximum rated parameters. Derived from FEA simulation, the maximum values used to normalize the training data are contained in Table 5.1:

Table 5.1: Norm Reference Values

Variable	**Value**	**Units**
Current	8	Amps
Angle	45	Degrees
Flux Linkage	0.062	V-s

Please refer to [48] for a full analysis and derivation of these parameters. Although, the maximum rated current for the machine was 8 Amps, the operating range for

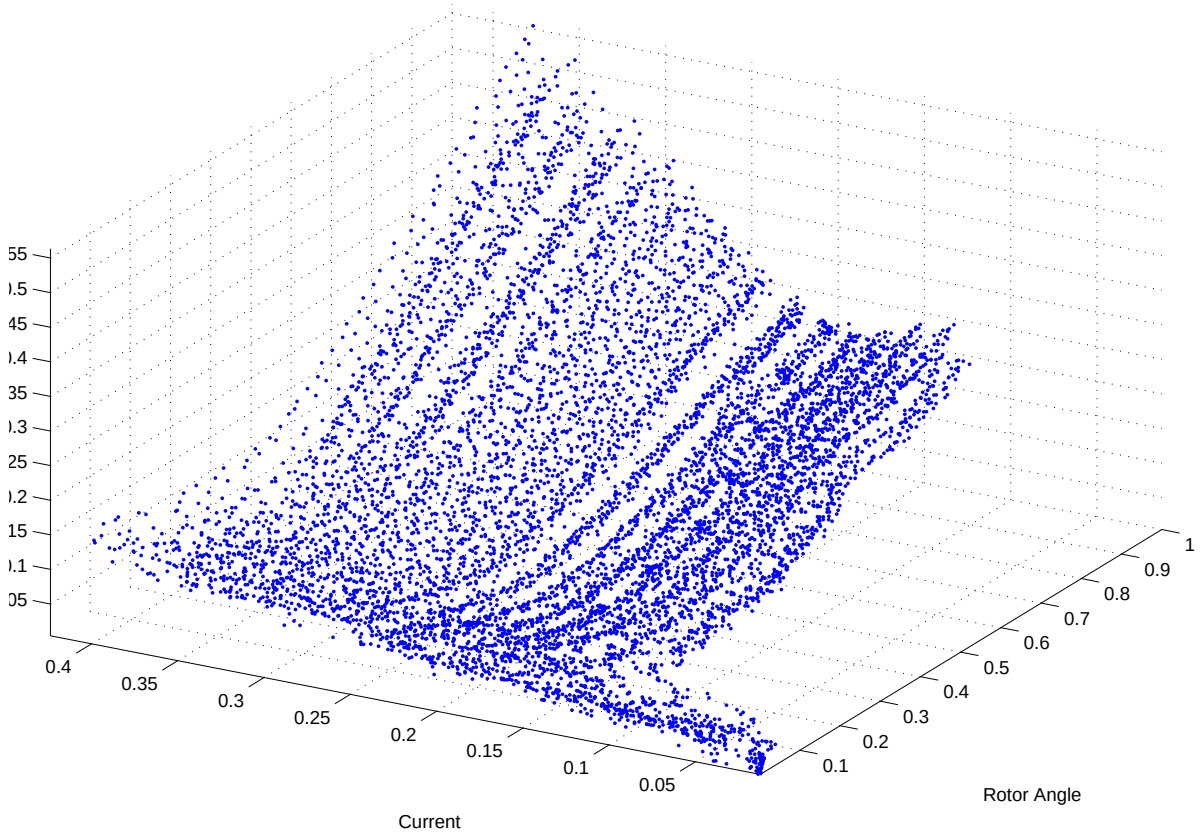

Figure 5.1: Normalized Current vs. Angle vs. Flux Linkage

current as recorded in the data experimentally was 1 to 3 Amps reference. This provided a control range of around 300 to 2000 RPM. Thus the normalized scale of the current barely exceeds 0.43. The flux linkage scale also exhibits less than full scale normalization. However, the rotor position scale represents the full range of the angular interval in which we are interested. To prevent the duplication of flux linkage values, the angular region from 45 to 90 degrees was eliminated. This will increase estimation resolution due to the full scale of 45 degrees is represented to 1. Future work can improve the neural surface by both increasing the current range and by

increasing the relative norm scale to 1 for both flux linkage and current even if either aren't exactly representative of their respective full scales.

5.3 Network Design

A typical neural implementation for SRM rotor position estimation and the reduced net are represented in Figure 5.2. Basically, the non-linear input eliminates the hidden layer from the typical design.

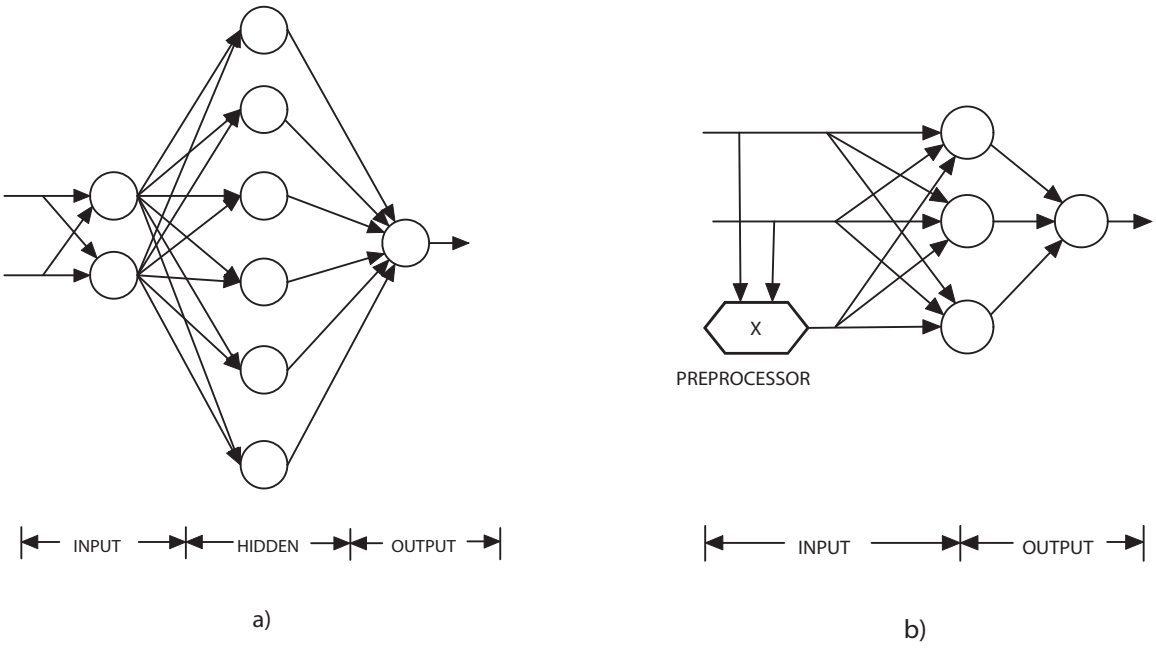

Figure 5.2: Neural Nets

The standard design rules for neural nets determines that one neuron shall exist on the input layer for every input.[27] Thus an extra neuron exists in the input layer for the reduced network as opposed to a typical one shown in Figure 5.2**(a)**.

The non-linear input was chosen as the product of the other two inputs so that it can be processed in real time, and to minimize the effects of the overall computational burden. Each of the neurons contains an activation function. Although continuity isn't an absolute necessity, a continuous function was chosen for this particular implementation. The chosen function for the input neurons is represented by the single-sided sigmoidal function as given by:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{5.1}$$

This particular function, although straight forward in its computation, is relatively difficult for a fixed point processor to calculate. A lookup table may prove to be the most efficient way to implement the function 5.3.

The output neuron is a straight linear combination of its inputs and weighting factors. In addition to the activation functions, each neuron contains an offset or threshold. This adjusts the responsiveness of the neuron.

If we count the number of operations required by a typical neuron with two inputs, it would require two multiplications, two additions (two inputs and threshold), and an activation function approximation. Continuing along these lines, we can compare the computational burden of a typical neural implementation versus the proposed neural architecture.

As seen in Table 5.2, the computational burden is reduced by 50 percent. It should be noted that the conventional neural net used in this example is among the

Table 5.2: Comparison of Typical ANN with Proposed Minimal ANN

	Conventional ANN 2:6:1	**Proposed ANN** 3:1
Multiplication	24	13
Summations	24	12
Activations	8	3
Total	56	28

smaller varieties used for SRM rotor position estimation. It has been demonstrated the hidden layers can exceed 50 neurons and hidden layer counts above 3.

5.4 Training and Performance Comparison

Originally a training program was developed to adjust the neural weights using a typical back-propagation method. A momentum term was used to speed up the training process and insure no local minima prematurely stopped error reduction. Initial network training runs were measured in tens of hours for an appropriate error magnitude. However, as the number of network evaluations increased, the required time to train them multiplied. Due to logistical time constraints, the training process needed to be compressed. Therefore, MATLAB's neural toolset was used to gain access to highly optimized training algorithm's like the Levenberg-Marquardt method. This allowed for a performance comparison between various topologies. All of the neural topologies were trained to a satisfactory level within 100 data epochs. During training, the time elapse was approximately 22 seconds. The self-generated code would take approximately 43K seconds to achieve the same level of neural error.

A new data set was developed from the original compilation of experimental data. This new data set was taken at random, so it was insured to be different than the one

used to train the network. The test was performed on a fully trained neural network to insure that the net would properly interpolate the rotor position for new input values not in the original training set. The performance of various network topologies including the two versions of the reduced network can be seen in Figure 5.3.

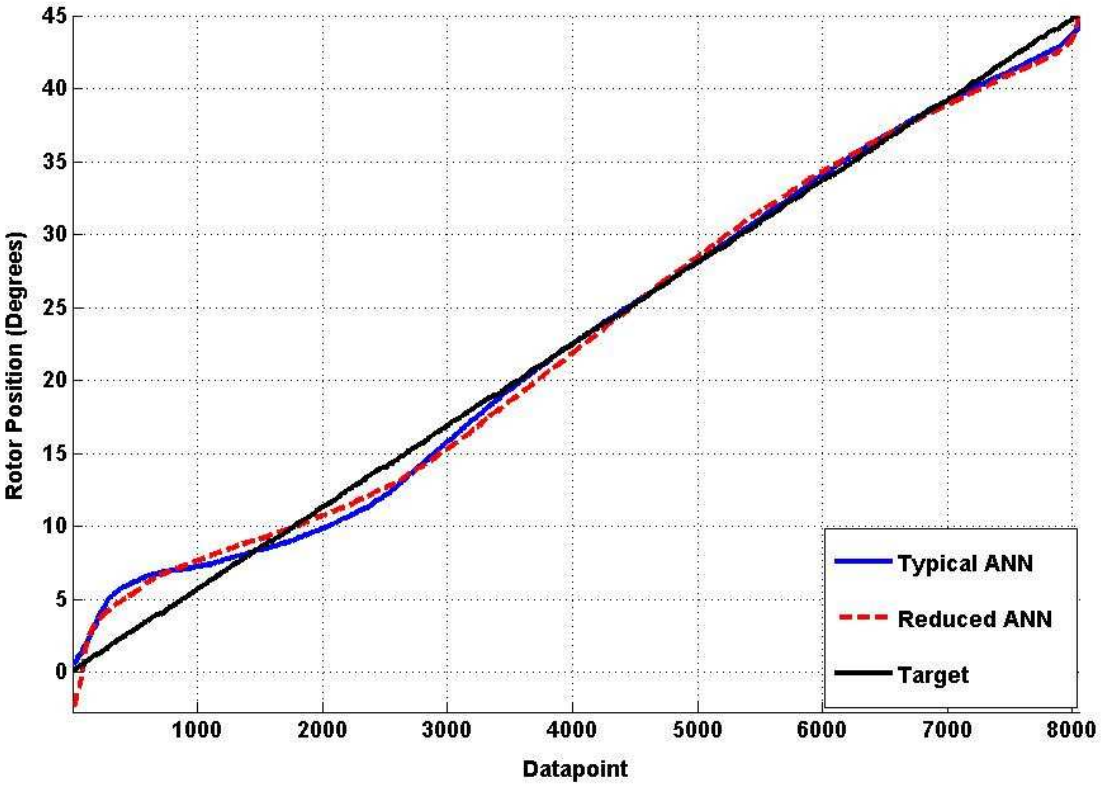

Figure 5.3: Neural Topology Performance Comparison on Random Data Set

The Typical ANN is a 2:6:1 topology with linear output neuron while the Reduced Network is a 3:1 topology with a pre-processor. As can be seen in Figure 5.3, both are very similar in their estimation performance. The extremes of all have significant error, while the middle portion is quite good.

5.5 Neural Estimation Error Analysis

The estimation performance error was evaluated on a squared average during training. A typical training error performance curve can be seen in Figure 5.4.

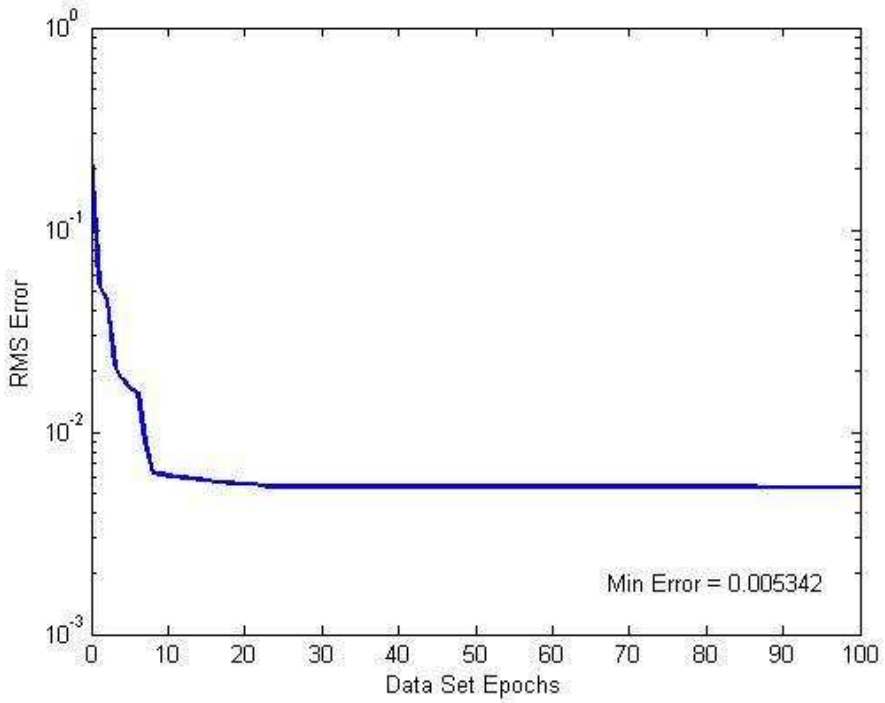

Figure 5.4: RMS Error Performance Using Levenberg-Marquardt Backpropagation

The overall RMS error for all the various topologies after training was about 0.005. This number is quite misleading. The sigmoidal activation functions used in the neurons have a distinctive s-shape symmetric about the 1:1 line. The tested output of the trained network also exhibits that shape to some degree. However, the error at the low end is negative as compared to the 1:1 line, while the error at the high end is positive. The average error across the entire input range will be the balance of both positive and negative error. In this case, a portion of the positive error cancels

out the negative error to produce a seemingly good, overall minimal error. We expect the extreme angular performance to suffer. The RMS error of 0.005 equates to about $\pm$ 3 degrees. The estimation error at 22.5 degrees, however, is less than 0.25 degrees for all topologies. The experimental performance of the linear reduced network can be seen in the next two Figures 5.5 and 5.6.

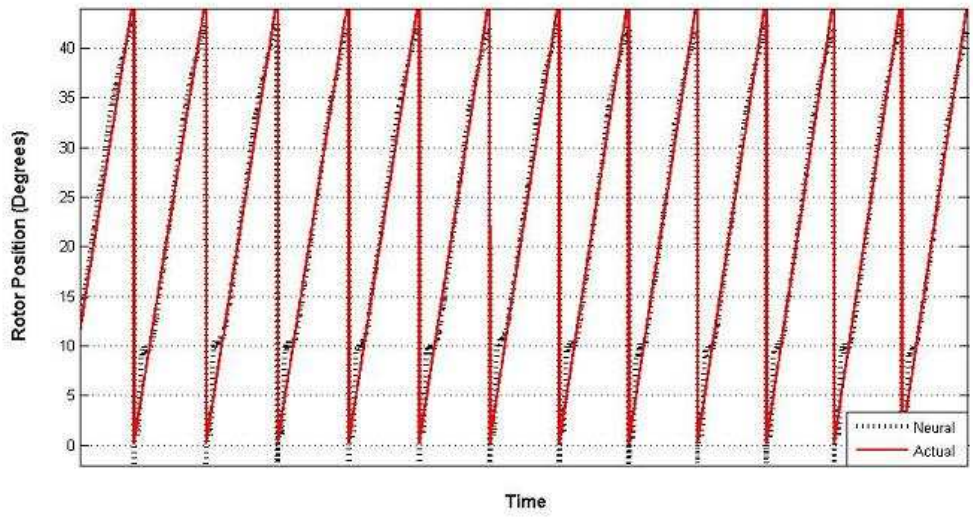

Figure 5.5: Actual Experimental Estimation Performance

The motor drive was operated using rotor sensors and the angular data from both the rotor sensors. Then the neural net was captured and subsequently compared. The motor architecture will accept a commutation error in excess of 10 degrees and still produce enough average torque to keep the motor from stalling. The calculated error percent is around 11 percent. The middle is less than $\frac{1}{2}$ percent. The accuracy of the middle points will prove to be useful in the development of the sensorless algorithm. A highly accurate reference point for angular position is all that is required to determine both speed and the commutation instant. The mid-range of the neural

estimate provides just the reference point we require. If an accurate speed estimate can be derived, then any error due to estimation inaccuracies can be negated through software. Error negation is explored in the sensorless algorithm development section in Chapter 6.

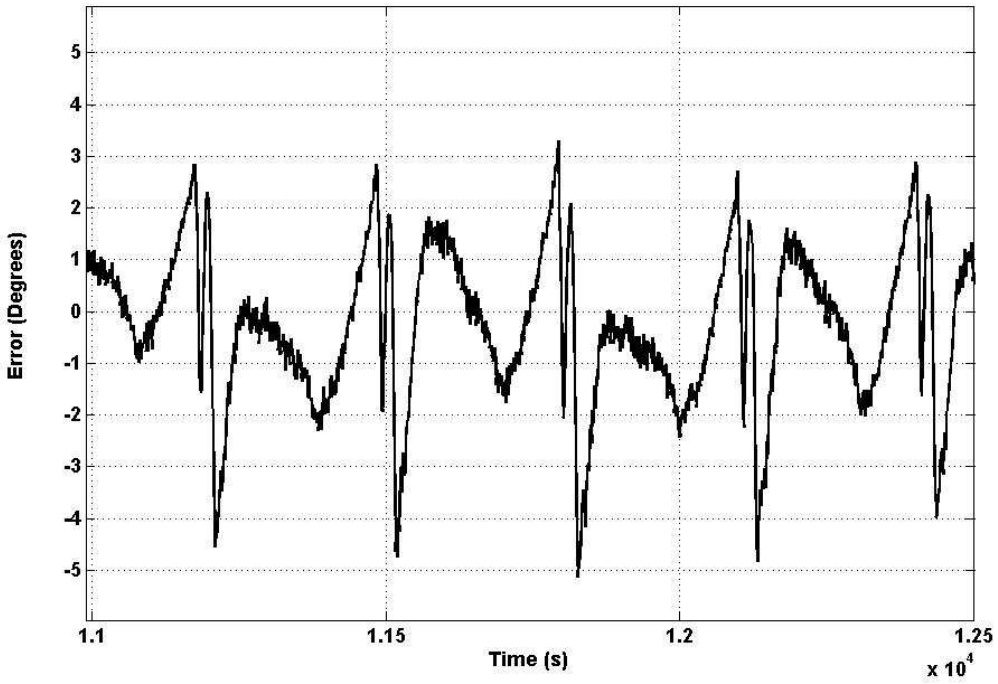

Figure 5.6: Absolute Estimation Error

In Figure 5.6, it can be seen that the majority of the estimation error occurs between ±2.5 degrees. This compares well with the training error RMS estimate of 0.005.

5.6 Neural Net Implementation

The software implementation of the neural net occurred in two stages: 1)the activation function representation and 2) the neural net with it's corresponding weight structure.

5.6.1 Sigmoid Activation Function

It should be noted that it is overly time consuming to use a fixed point process to precisely calculate a realtime sigmoid function. Therefore a lookup table was implemented to approximate the function.

The sigmoid function is symmetric about the $x = 0$ axis, and therefore only half of the function is represented in the lookup table. Any negative number given to the function algorithm, as denoted in Figure 5.7, is negated and used to look up the complement of the true value required. This complement number is subtracted from 1 to determine the actual sigmoid value required for negative numbers. Thus more resolution could be derived in the lookup table without using twice the space required by the entire spectrum of the function.

Numbers placed into the function that produced results exceeding the largest resolvable number were limited to 1. None the less, the function implementation proved to be accurate.

5.6.2 Network Implementation

The implementation of the neural net utilized the sigmoid approximation as a function call. Where needed, the network code would pass the weighted, summed inputs to the sigmoid function which would in turn pass out the function return value. The overall

efficiency of the code is acceptable. The optimization routines in the C compiler can interfere with the performance and should be monitored closely.

The algorithm, shown in Figure 5.8, demonstrates an overview of the implementation. The algorithm first calculates the non-linear input from the re-scaled current and flux linkage inputs. It then constructs an input vector array to feed the input matrix of the network. The input layer of neurons are processed and their outputs are formed into another vector. This vector is presented to the output neuron for processing. The network output is scaled to angular degrees (x10). The drive algorithm shown in Figure 3.8 calls the neural net as a modular function.

5.7 Summary

It was shown that a neural net size and subsequent computational burden could be greatly reduced by adding a non-linear input. Non-linear inputs other than the one used in this research could prove to be a more optimal choice. The optimal input space derivation for this neural net is outside the scope of this research. However, it was shown that the network performed as well as any conventional implementation. Thus the initial research proposal can be deemed to be successful.

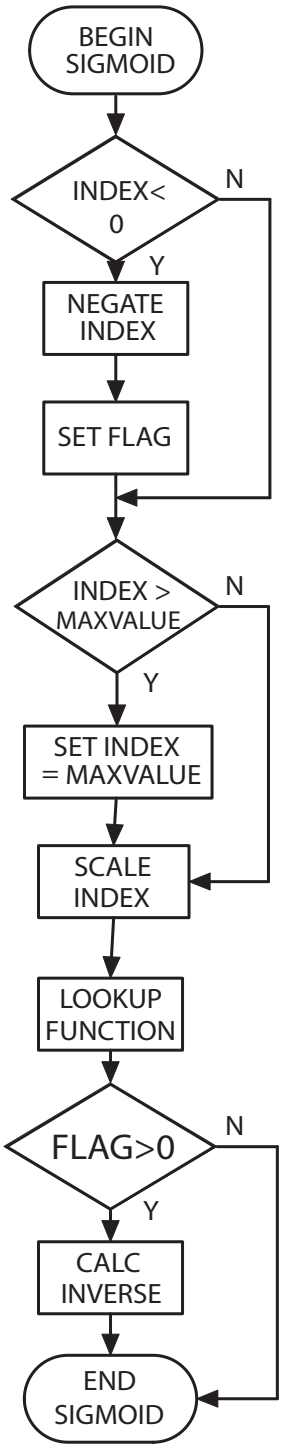

Figure 5.7: Sigmoidal Function Algorithm Implementation

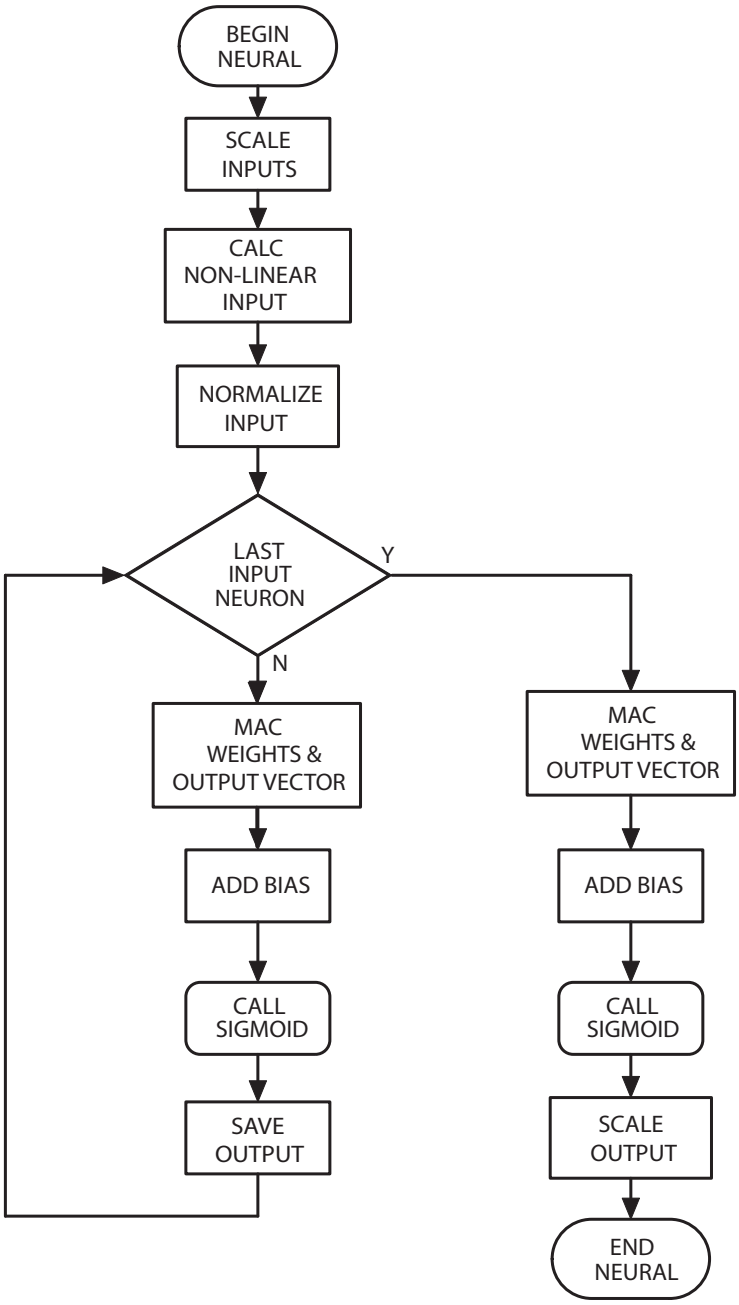

Figure 5.8: Neural Net Estimator Algorithm

Chapter 6

Dynamic Simulation of Sensorless Motor Drive

This chapter introduces the concept of motor drive simulation. We determine the appropriate method for modelling the drive and motor dynamics, as well as implementing the closed-loop control. The simulation algorithm for sensorless operation will be derived and implemented into the simulation. Finally, the results at various operating points will be presented and discussed.

6.1 Introduction

In the previous chapters, various functional requirements for sensorless operation were derived; two of which were the flux linkage estimator and the neural rotor position estimator. These were discussed prior to this section in detail because of the algorithms required to properly implement a simulation. The premise of simulation is to provide an approximation of drive operation and performance accurate enough to develop a sensorless algorithm for the machine. The simulation includes a motor model as well as a converter model. Current, speed, and commutation controllers

should co-exist with the models to provide a complete motor simulation system.

As with most sensorless systems, the sensorless functions and algorithms provide a drive for commutation. Supporting functions, such as the flux linkage estimation and the neural estimation, drive the current commutation algorithm for torque control. Typical proportional + integral controllers will control the current and speed in feedback loops around the commutation drive function.

The intent of the simulation is to completely model the entire system from motor to drive to DSP controller. This intent drives the premise that all controller functions that will be processed on the DSP should be processed in simulation in a similar fashion. It will be shown that an interrupt driven routine can be modelled inside the actual motor drive simulation and provide the necessary control to the system.

6.2 Basis of Motor Parameters

The motor model for simulation exist in the form of lookup tables. The tables were derived from FEA simulation and analysis derived in [48, 39]. The parameters included in the motor model were setup with respect to discrete current and angular rotor positions. The saliency of the motor occurred on 90 degree intervals, and thus the motor model parameters existed in $2-D$ tables that were $9x91$. The current was represented from 0 to 8 Amps, and the angles from 0 to 90.

The parameters contained in the table aside from current and rotor position are: torque, inductance, and flux linkage. They were represented for both the main stator winding and the the auxiliary winding A. To review the motor connections, the main stator windings ($x4$) were all connected in series. The auxiliary windings were

connected into two pairs, A and B. Only one set was used in both simulation and experimental implementation.

The simulation environment, MATLAB, provided interpolation functions that were used to derive values required by the simulation that were not present explicitly in the motor model tables.

6.3 Simulation Dynamics

Introduced in Chapter 3, the motor drive dynamic equations were derived for both the converter switch on and off states. These are to be used for the simulation of the motor. The simulation loop will use an external function that inputs flux linkage and rotor position (derived from dynamic equations) and returns winding current. Thus actual winding current is the basis for all other simulation parameters. The external function utilizes the motor lookup tables in its derivation of winding current. The dynamic equations provide the flux linkage in both main and auxiliary windings, speed, and actual rotor position. Rotor position is integrated from speed and thus must be reset to prevent an ever increasing variable. The saliency of the motor determined the reset of the rotor position variable to every 90 degrees.

Trapezoidal numerical integration was used in the derivation of the motor operation. For an explicit representation please refer to the Appendix for the code. Parameters designated by **a** are for main stator winding; by **b** for the auxiliary. The equations for both on and off states are shown below.

On-state Motor Parameter Integration

$$V_a = V_{dc} \tag{6.1}$$

$$V_c = \int -(\frac{i_b}{C})\ dt \tag{6.2}$$

$$\lambda_a = \int (V_a - R_a\ i_a)\ dt \tag{6.3}$$

$$\lambda_b = \int ((V_c - V_a) - R_b\ i_b)\ dt \tag{6.4}$$

Off-state Motor Parameter Integration

$$V_a = V_{dc} - V_c \tag{6.5}$$

$$V_c = \int \frac{(i_a - i_b)}{C}\ dt \tag{6.6}$$

$$\lambda_a = \int (V_a - R_a\ i_a)\ dt \tag{6.7}$$

$$\lambda_b = \int (-V_a - R_b\ i_b)\ dt \tag{6.8}$$

Addition Motor Parameter Integration

$$\omega_m = \int (\frac{1}{J}\ i_a - \frac{B}{J}\ \omega_m - \frac{1}{J}\ BL\ \omega_m)\ dt \tag{6.9}$$

$$\theta = \int \omega_m\ dt \tag{6.10}$$

All of the motor dynamic equations are processed on a incremental time step as opposed to other functions required by the simulation like DSP interrupts. This insures that, although incrementally discrete, the motor drive variables are relatively analog when compared to the DSP processed functions.

The overall simulation algorithm is processed on 1uS intervals where all the motor parameters are retrieved from either lookup tables or the dynamic equations. The DSP functions are updated on the 50uS carrier frequency. To produce a user friendly simulation output, all of the simulation parameters are tracked in order to generate time based plots. All simulation timers are incremented each cycle until a end of simulation match is made to a preset time limit which stops the simulation and then plots all tracked variables.

6.4 Explanation of DSP Functional Segments

The PWM gating signal is the interface to each motor dynamic equational set during the simulation. This signal was generated in much the same way the DSP will generate it. The method of PWM generation is called triangle carrier wave comparison. An up-down timer set to a period of 50uS will provide a means for a reference compare value to generate a set/reset signal. Every time the carrier wave matches the reference value the PWM signal is complemented from its previous state. The reference value can be adjusted by the P + I current controller for current control. All the DSP functions are processed at the end of every up/down timer period. Thus every DSP event is synchronized to the 20kHz carrier signal, in the same way as the current control.

Timing of the DSP functions with respect to the drive simulation is critical. If any

DSP integration is to occur, such as in flux linkage estimation or speed control, the integration time step must be exact. In addition to this, the variables used in DSP signal processing have a sample delay associated with them. As previously discussed, the sample period was synchronized to the 50uS triangle carrier wave period.

To minimize the sample delay effects or to properly model the digital delay effect, the sampled signals were used immediately at the end of the off-time sample period. If any algorithms used in the DSP process required previous samples, those samples were self-contained in their respective functions and overwritten with each successive function calls.

For example, a digital implementation of an IIR filter requires previous outputs and inputs in the difference equation. If the IIR filter function is used in the current control, then the current control function handles the store and recall of the variables needed from previous function calls. If the the IIR filter is used in the speed control as well, then that function handles it's own save and recall.

The functional requirements to be processed in the DSP are current P+I controller, flux linkages estimator, neural estimator, speed estimator, speed P+I controller, and of course the sensorless commutation algorithm. The simulation serves as a platform to derive algorithms that could be used in the final firmware/hardware implementation. With this as a goal, the code was written as close to what will actually be used in the DSP firmware. All functions written in C or C class type code are translatable between MATLAB and the DSP environment with only minor syntax changes.

6.4.1 Current Control Algorithm

The current control law implemented in this algorithm was determined from the digital conversion of the linear controller. A sample time of 50uS was utilized for a bilinear digital approximation of the linear P+I controller.

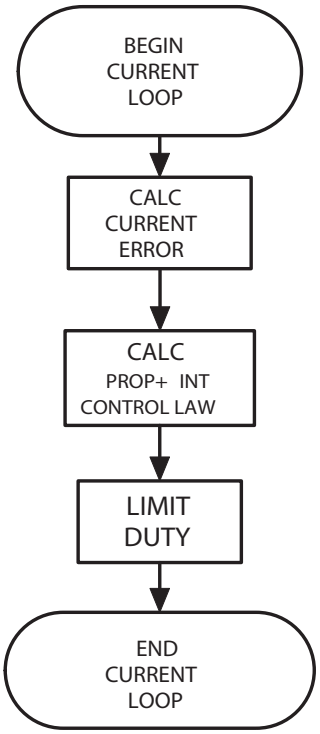

Figure 6.1: P + I Current Controller Flowchart Diagram

A difference equation was derived using the relationship that z^{-1} corresponds to a delay of 1 sample cycle. Thus if a particular variable in the z-transform is multiplied by z^{-1}, then this implies that the variable from the previous cycle should be used. Thus the control law is given by:

$$out(n+1) = A\ ierr(n) - B\ ierr(n-1) + out(n); \tag{6.11}$$

where the coefficients are determined by the z-transform from the linear controller gains, zeros, and poles. Looking at the control law, the error feed-through terms determine the zero placement, and the output feedback terms determine the poles. This type of controller is also known as a recursive IIR filter. More terms can added to produce additional poles and zeros.

The response of the electrical system is much faster than the mechanical system of the motor. A properly implemented current controller requires the IIR form, whereas the speed controller uses a simple P+I method. This method computes the error integral, gains it and then sums the result with the gained error. Due to the speed of the control loop, the required integral gain for the current controller would be much greater than what could be represented in the 16-bit word memory of the DSP if the simple method was used.

6.4.2 Flux Linkage Estimation Algorithm

Previously discussed in detail in Chapter 4. Please refer to that chapter for code flow algorithm.

6.4.3 Neural Net Algorithm

Previously discussed in detail in Chapter 5. Please refer to that chapter for code flow algorithm.

6.4.4 Speed Control Algorithm

In direct opposition of the current controller implementation is the speed controller implementation. The speed controller though processed in the DSP sample period is actually at a much slower interval. The controller is processed at 1kHz frequency, thus the implementation lent itself to the simple integration method. Please refer to Figure 6.2 for an overview.

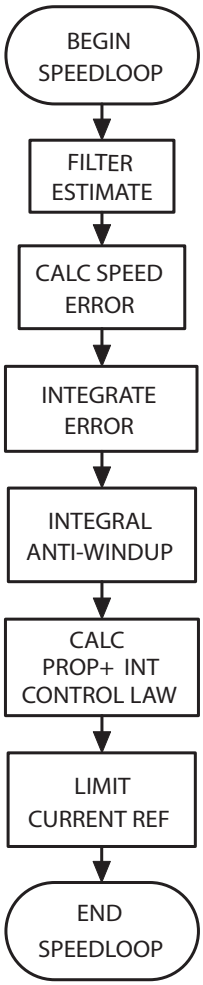

Figure 6.2: P + I Speed Controller Flowchart Diagram

The algorithm shows that the speed estimate is first filtered. The filter law is given by:

$$\omega_{sig}(n+1) = \beta\, \omega_{sig}(n) + (1-\beta)\, \omega_{est}(n); \tag{6.12}$$

This law is also a digital difference equation derived from a z-transform. The transfer function is given by:

$$H_{\omega}(z) = \frac{(1-\beta)\, z}{z-\beta} \tag{6.13}$$

This filter approximates a causal, first-order filter with the corner frequency set by the pole designated by β. For a sample period of $T_s = 0.001$ and β chosen at 0.9, the corner frequency is around 104Hz. The feed-through term can be completely eliminated should it cause problems with stability. The form of which can be expressed by:

$$H_{\omega}(z) = \frac{(1-\beta)}{z-\beta} \tag{6.14}$$

The actual speed control law used a simple backwards rectangular numerical integration method and can be demonstrated by:

$$\imath_{ref} = kps\ \omega_{err} + kis \int \omega_{err}\, dT_s; \tag{6.15}$$

Notice the output of the speed control law is i_{ref}. This is the current command required to process the current loop. The speed loop determines the drive requirement based on the speed error. If the error is large, then the law dictates that the current reference be high and vice versa. This arrangement is called a direct torque command drive.

6.5 Discussion of the Sensorless Algorithm

All functions discussed so far are control support functions for the motor drive. For proper torque generation, the current must be controlled in designated angular intervals based on the saliency of the motor. A commutation scheme for this motor drive is originally based on hall-effect sensors that gave discrete position indications. The current must be driven through the windings at 0 - 45 degree intervals and not driven from 45 - 90 degrees. This insures positive torque generation for clockwise rotation of the rotor.

The sensorless algorithm is developed on the premise that the error at the mid-range of angle estimates is extremely low. The use of simple timers was developed to estimate time durations of conduction, non-conduction, and speed. In fact three timers were used for variation measurements during motor operation:

1. a timer to measure the time between the neural mid-range angle references (22.5 degrees)

2. a timer to measure the time between mid-range and the end of the conduction period (commutation instant)

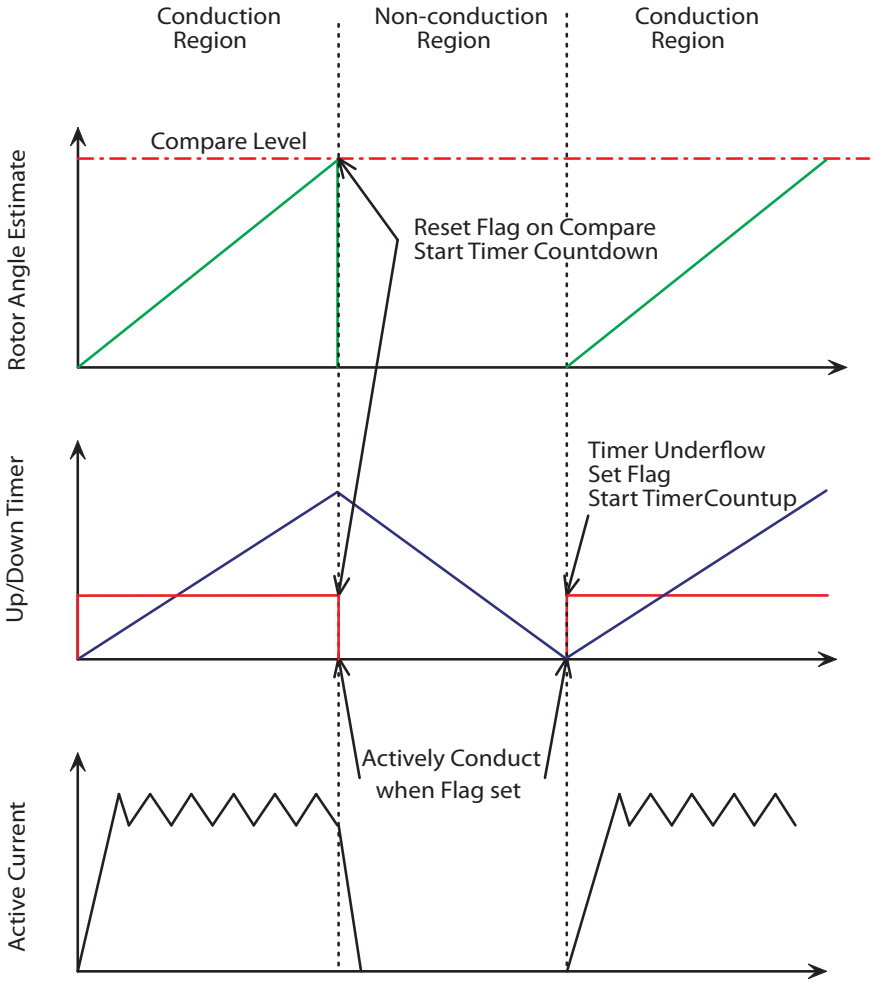

Figure 6.3: Basic Sensorless Timing

3. a timer to measure the duration of the non-conducting interval.

Referring to Figure 6.3, a timer is used to count up during conduction and then down during non-conduction. Its count direction is controlled by a flag variable. At turn-off, the countdown timer can be adjusted based on a speed estimate to better predict the time interval required to bypass until the next conduction period. This is the basis for the sensorless algorithm. Additional features exist to make the process more robust.

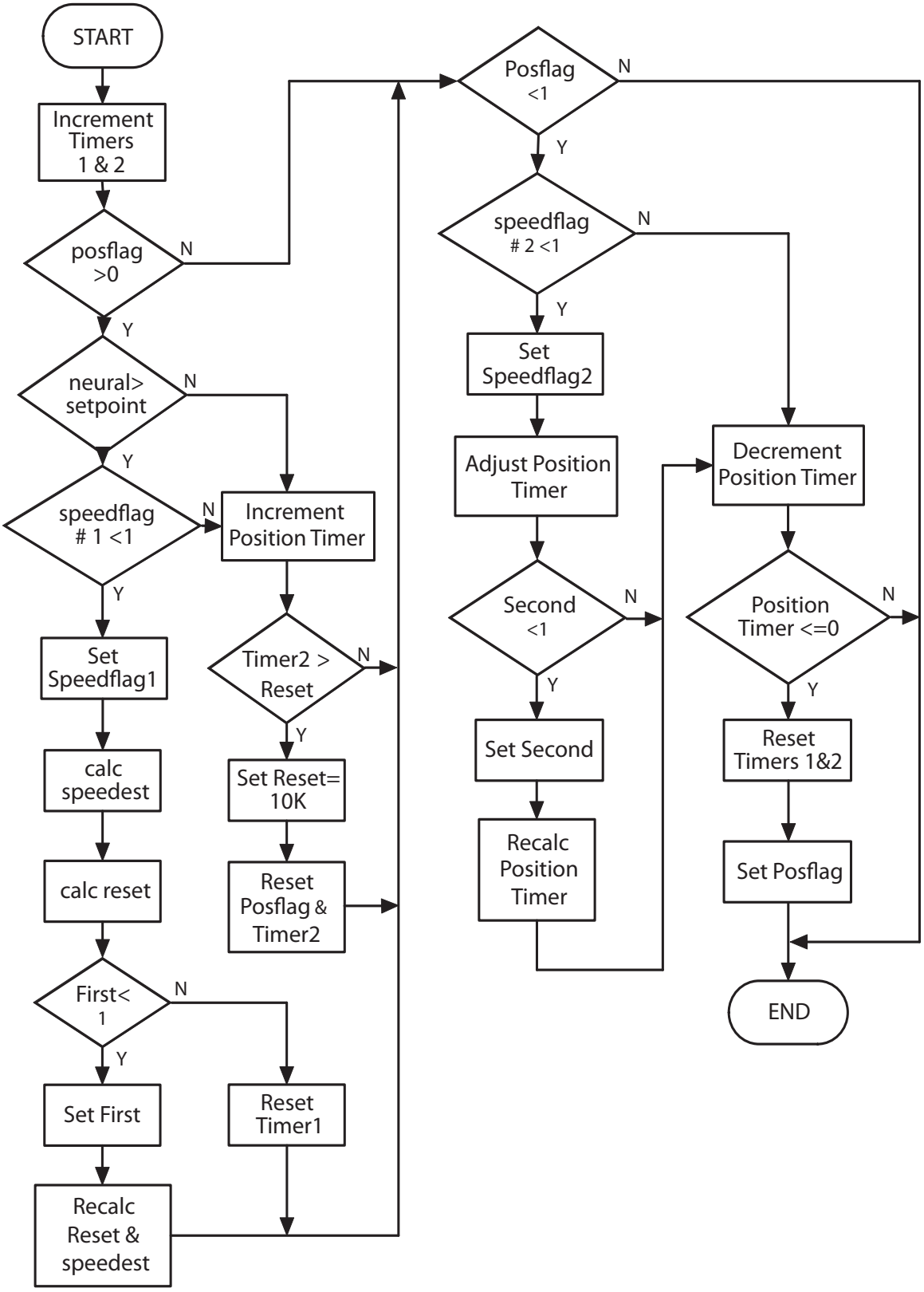

Figure 6.4: Sensorless Algorithm Flowchart Diagram

Referring to Figure 6.4, we see the entire final algorithm derived for sensorless commutation. The principle is the same, control current, calculate flux and neural estimate during the conduction interval and then no conduction during the countdown period. The PWM duty should be commanded to zero while the countdown progresses to insure no active conduction is commanded. The conduction and non-conduction intervals are controlled using a simple flag to drive the control functions. If the region is the positive torque, the flag is 1, else it is set to zero. The input to the algorithm is the rotor position estimate derived from the neural net and the output is the control flag variable. When the neural estimate reaches the midpoint of the commutation period, designated by 22.5 degrees, a projection based on previous speeds is made to predict when the conduction period should end. The projection is made from a speed estimate that is derived from the first timer. This timer counts up from each successive midpoint and then resets. The second timer is compared to the calculated project timer value and ends the conduction interval when a match occurs. The second timer is then reset. The second timer counts the cycles between commutations. The end value is exactly twice as long as the countdown should last. If this value is used as the start of the countdown timer, each non-conduction period can be inherently corrected.

When the end of the conduction period has been reached, the algorithm sets the non-conduction flag, adjusts the third countdown timer to half of the second timer value just prior to the reset. The third timer then begins to countdown. During this interval there is no PWM and all calculated variables are set to zero. When the third timer underflows(i.e. equals zero), everything is reset and the conduction flag is set to re-start the conduction region. Please note, the state of the conduction

flag determines whether a current command is present. The operation of the various timers can be seen in the simulation results depicted in Figure 6.7.

Although this algorithm adjusts itself based on the speed, care must still be taken to insure the robustness of the estimation. The algorithm is subject to aliasing. The algorithm can measure from one midpoint to another midpoint and skip one in between. Methods should be developed to account for this eventuality. One way to insure that no cycle is ever missed is to insure that the algorithm starts with the rotor very close to the zero reference. This can be controlled with a proper startup algorithm.

6.6 Results of Simulation

The simulations were run in both open-loop speed and closed-loop speed. Some explanation of the simulation plots may be required. The relevant simulation plots that are of importance contain variables designated by:

1. Capacitor Voltage: $(V_c \quad V_c^{\sim} \quad V_{dc})$
2. Winding Voltage: $(V_{an} \quad V_{an}^{\sim})$
3. Winding Current: $(i_{ref} \quad i_a \quad i_a^{\sim})$
4. Flux Linkage: $(\lambda_a \quad \lambda_a^{\sim})$
5. Rotor Position: $(\theta \quad \theta^{\sim})$
6. Torque: $(\tau_e \quad \tau_a \quad \tau_b)$
7. Speed:$(\omega \quad \omega^{\sim} \quad \omega_{avg})$

where the "real world" motor parameter is given in the form of X_x and the estimated or digitally sampled and/or calculated parameters are given in the form of $Y_y^{\sim}$. The "real world" or actual motor parameters are simulated via the dynamic equations as presented in Section 6.3. They are intended to represent the analog signal to be measured.

Description of the plots are as follow in order of their appearance:

1. Actual and sampled voltages on the energy storage capacitor.
2. Actual and sampled main winding voltages
3. Actual and sampled main winding current with current command
4. Actual and estimated flux linkage
5. Actual and estimated rotor position and 45 degree reference
6. Main and auxiliary winding torque with total electromagnetic torque
7. Actual, estimated, and filtered speed

Three sets of above plots are shown on the following pages. They demonstrate the simulation results for open-loop speed control at low speed and high speed. The last several plots are for the closed-loop speed, sensorless simulation.

6.6.1 Low Speed, Open-Loop Simulation Results

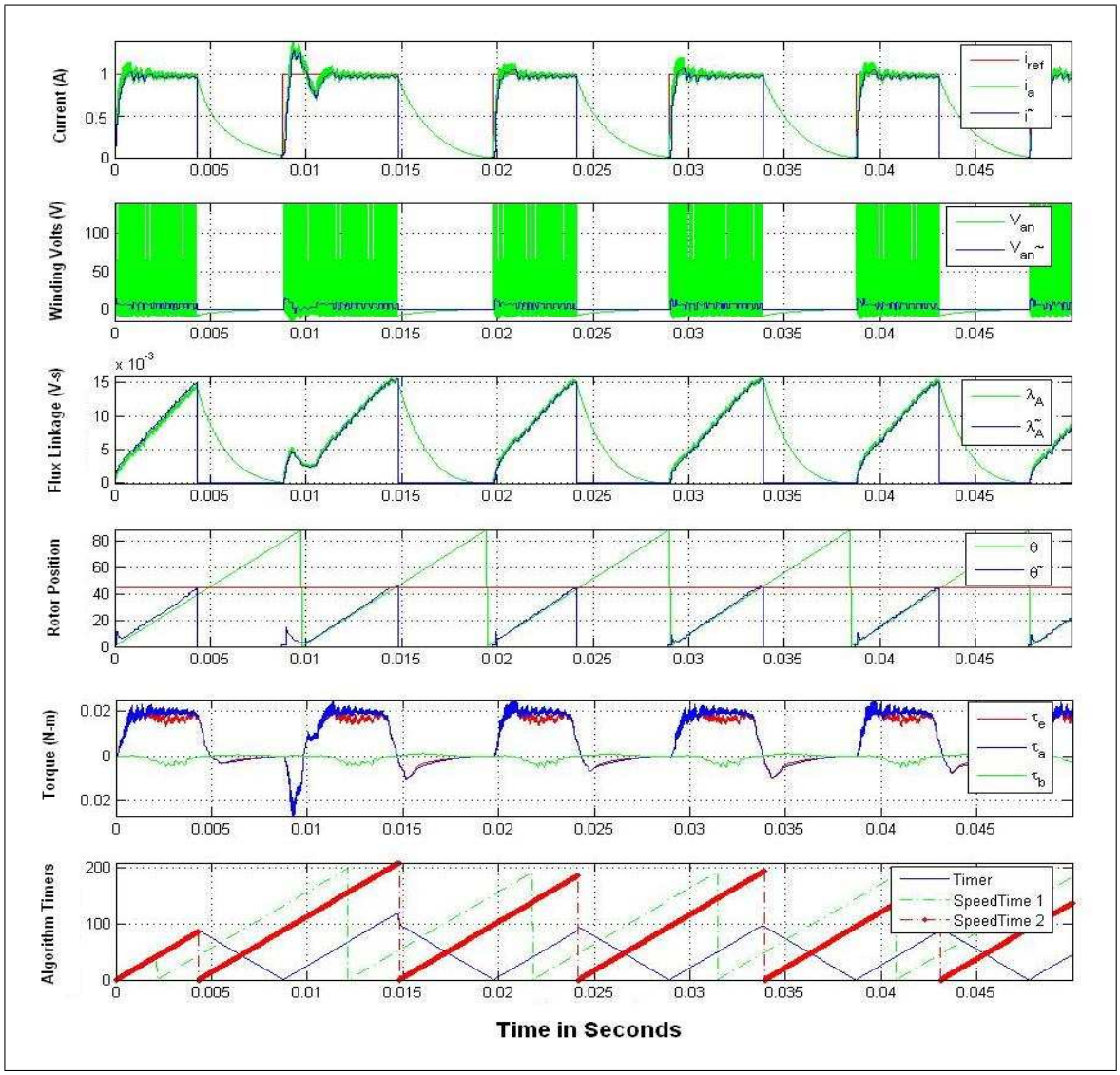

Figure 6.5: Low Speed Open-Loop Speed Simulation

6.6.2 High Speed, Open-Loop Simulation Results

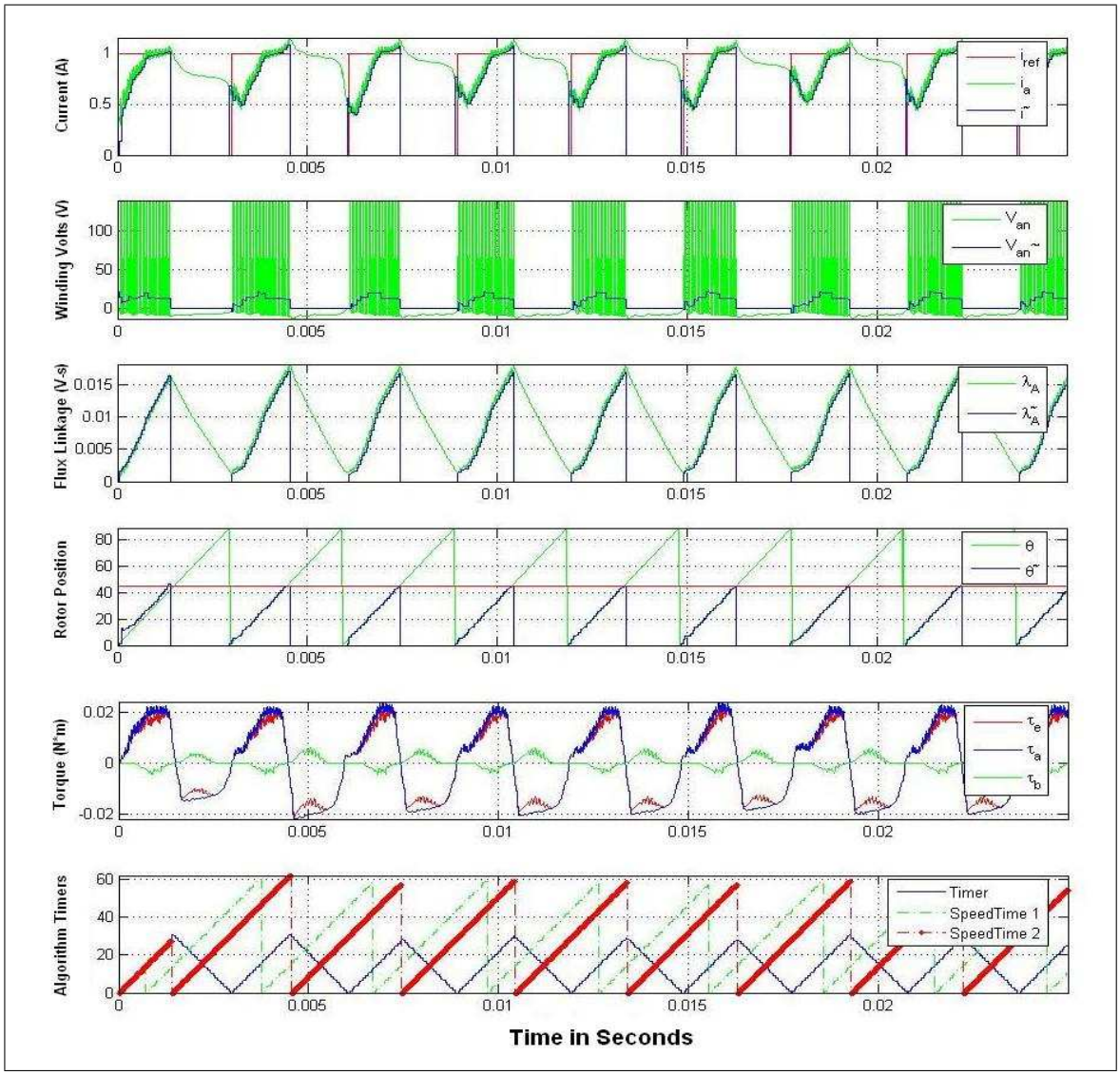

Figure 6.6: High Speed Open-Loop Speed Simulation

6.6.3 Closed-Loop Simulation Results

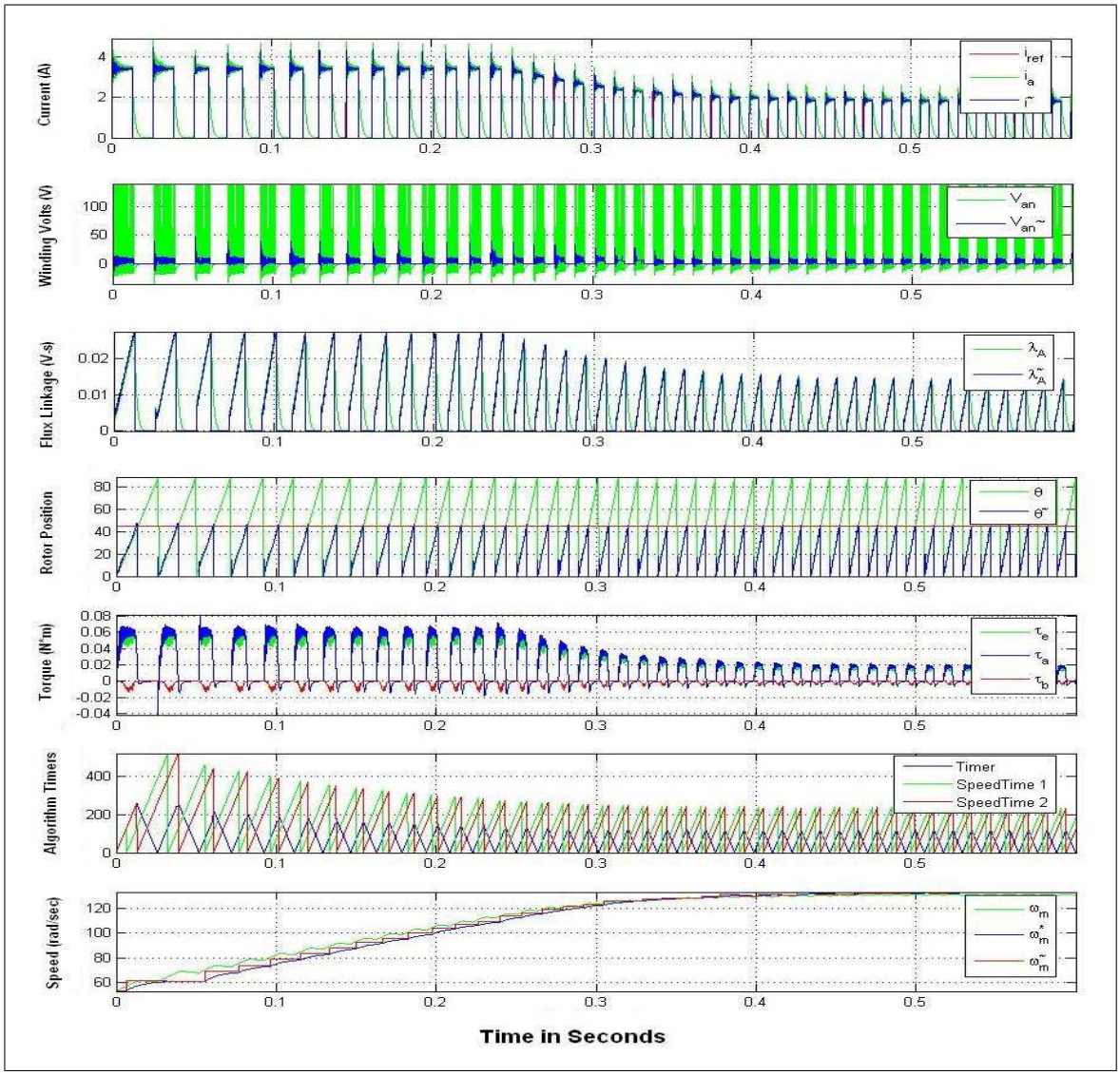

Figure 6.7: Closed-Loop Speed Simulation

6.7 Summary

In the previous sections, several simulation plots were given. Two of the plots were of open loop speed system at high and low speeds and the third plot was the closed loop speed plot. All pertinent motor parameters were shown. The dynamic simulation was deemed successful because it provided a platform by which the sensorless commutation algorithm could be developed and debugged at various operating conditions. In the next chapter, the code developed for the simulation will be implemented and verified experimentally. Code functions such as the flux linkage estimator, the neural estimator, and the sensorless commutation algorithm will be implemented into the actual DSP system.

Chapter 7

Experimental Implementation and Verification

This chapter demonstrates the final sensorless motor drive experimental implementation and results. First, the firmware implementation is discussed with reference to the original sensed version presented in Chapter 3. Second, the experimental results are reviewed along with a brief discussion of observations. Finally,the performance of the drive is evaluated. The results are presented in two stages:

Simulation Verification

The first results are intended to show the accuracy of the simulation versus the implementation. The plots include:

1. sampled capacitor voltage
2. estimated average winding voltage
3. sampled current versus current reference
4. flux linkage estimate
5. neural rotor position estimate vs. hall-effect sensors

6. sensorless algorithm timers

7. speed estimate

Closed-loop Speed Operation

These results are intended to shown the performance of the closed-loop speed control. The plots include:

1. speed

2. current

3. flux linkage

4. rotor position

7.1 Introduction

The final sensorless code DSP implementation is represented in Figure 7.1. The premise of the code flow has not changed since the presentation of the sensored code in Figure 3.8. However, the sensorless drive code utilizes the sensorless algorithm presented in Chapter 6 to drive commutation rather than using hall-effect position sensors. During conduction, the flux linkage estimation and the neural estimation are performed in addition to the current and speed control.

The normal operation of the motor in sensorless mode is considered to be the algorithm that occurs in the the ADC interrupt. This is the isolated branch on the

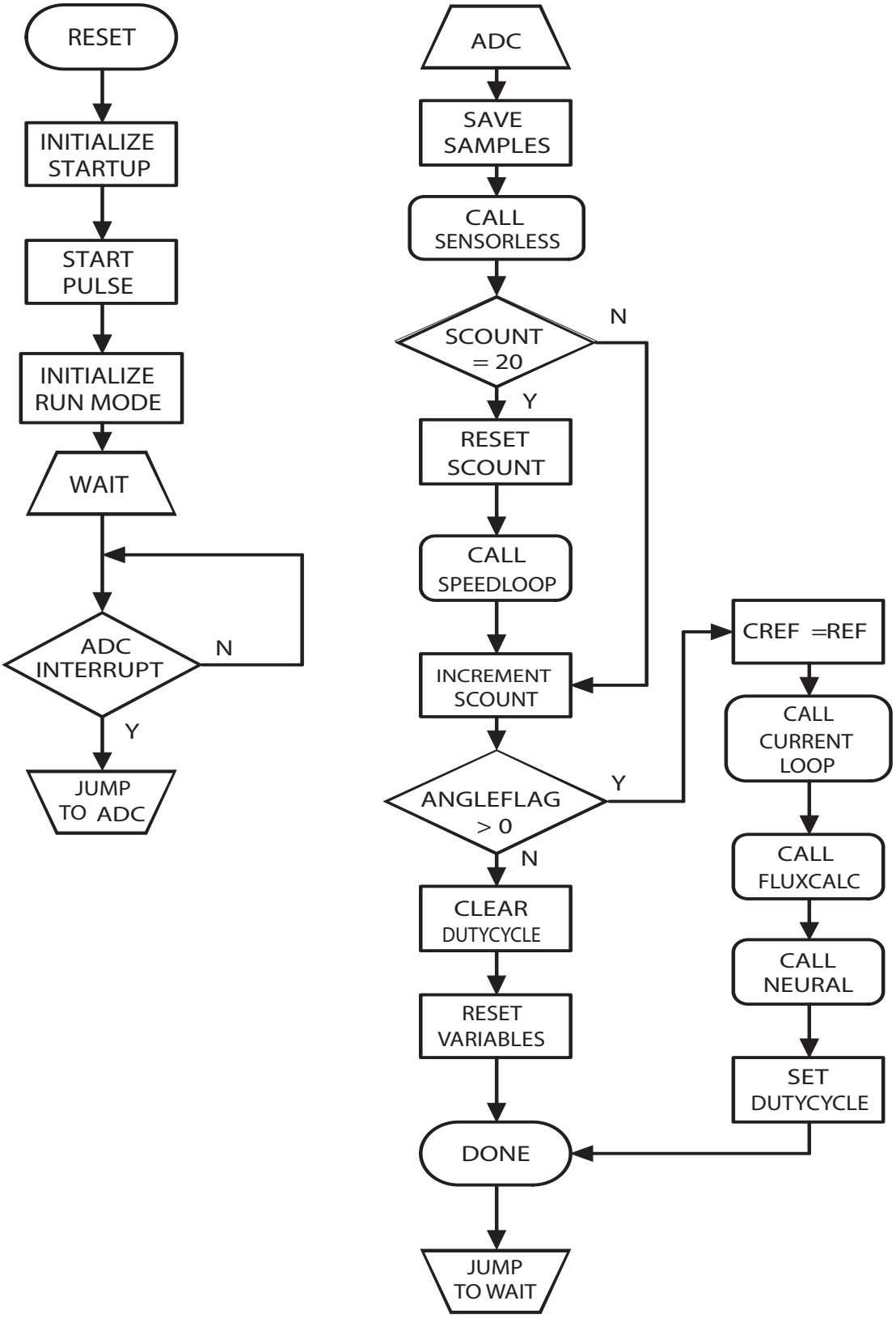

Figure 7.1: Basic Firmware Flow for Sensorless SRM Control

right of Figure 7.1. The jump from the main wait routine to the ADC interrupt occurs on 50uS intervals with all the feedback signals sampled just prior. The sensorless algorithm controls the variable 'angleflag' which, in turn, controls the conduction mode as previously discussed. The speed controller is called only every 20 interrupt cycles which equates to 1kHz sample frequency.

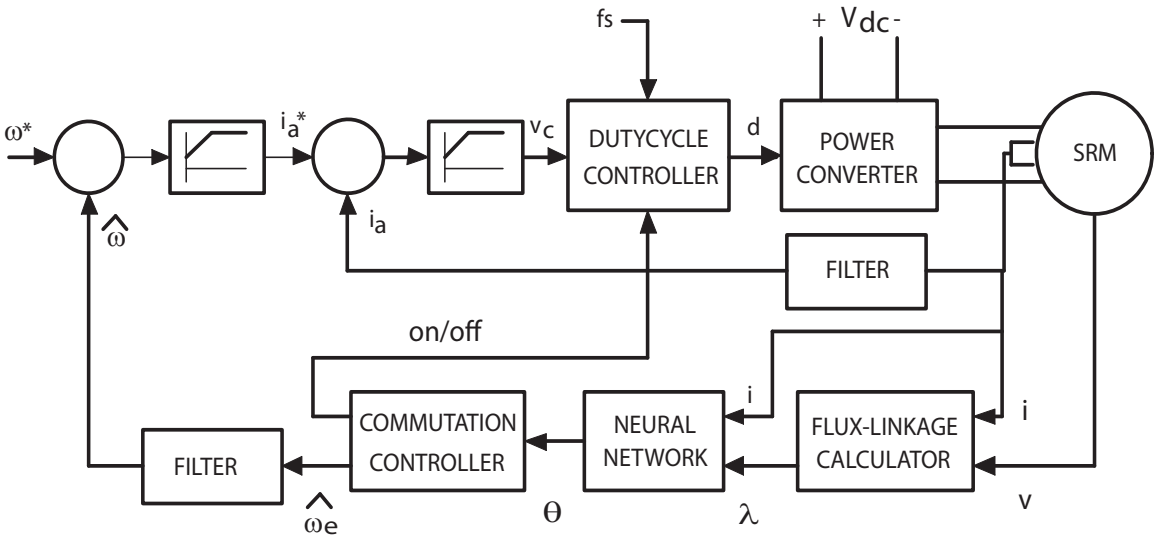

Figure 7.2: Final Closed-loop Control Implementation

The final hardware control loop is represented in Figure 7.2. The sensorless algorithm is represented by the commutation control block. The output of the algorithm is PWM interval control and a speed estimate. The speed estimate is filtered to provide a smoothed feedback signal to a P + I controller. The majority of the blocks represent firmware while the only hardware is the power converter, the motor, and the hardware associated with the voltage and current feedback signal.

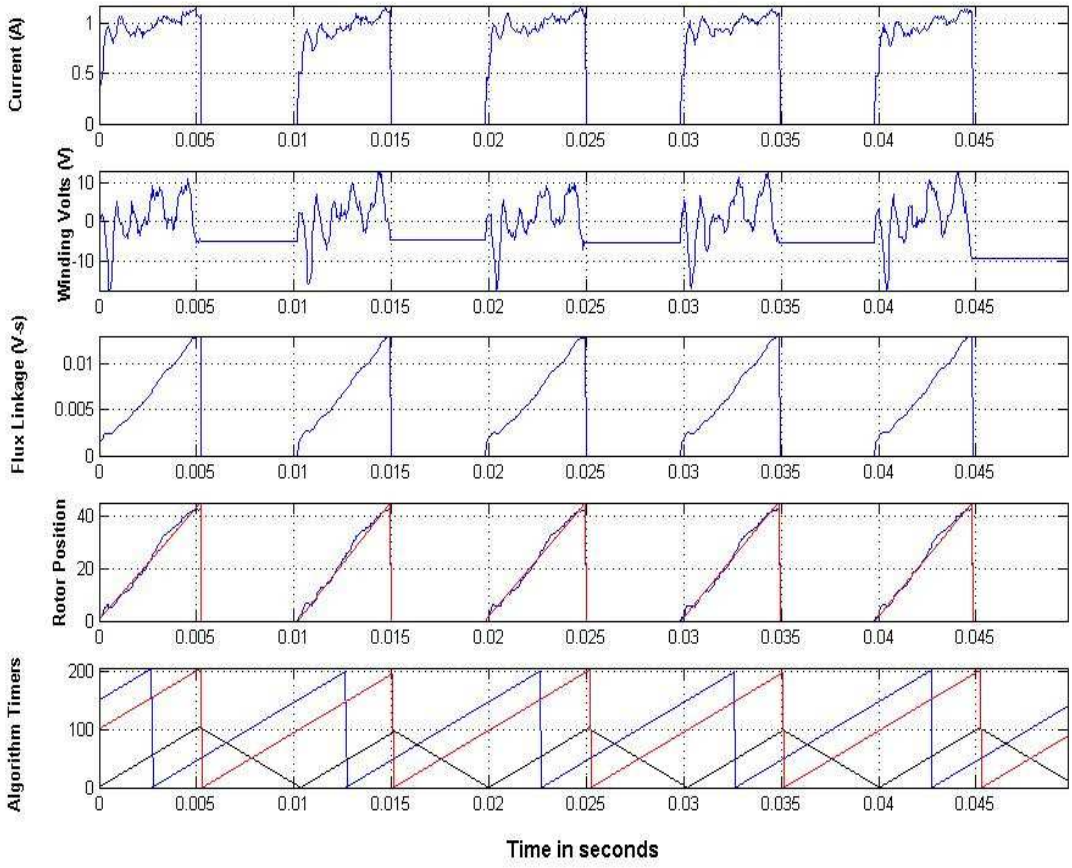

Figure 7.3: Open-Loop Speed Experimental Results

7.2 Open-loop Speed Results

The plots in Figure 7.3 show electrical parameters captured by the DSP controller during runtime. They represent current, applied main winding voltage, flux linkage estimate, neural estimate vs. actual rotor angle, and the sensorless algorithm timers, respectively. Due to the fact that the control and estimation processes are only calculated during active conduction, the experimental data only shows the signals

during that angular interval. The plots represented are the average sampled winding current, the average sampled winding voltage, the flux linkage estimate, the neural estimate (superimposed on the actual angle), and the associated sensorless timers as discussed in Section 6.5.

7.3 Closed-loop Speed Results

The plots in Figure 7.4 show electrical parameters captured by the DSP controller during runtime. They represent the closed-loop speed response of the sensorless system. The response is very similar to that represented in the simulation Figure 6.7. The important fact to note in the results is that the experimental results are captured from zero speed and the simulation results start at around 500 RPM, thus the time scales are slightly out of sync.

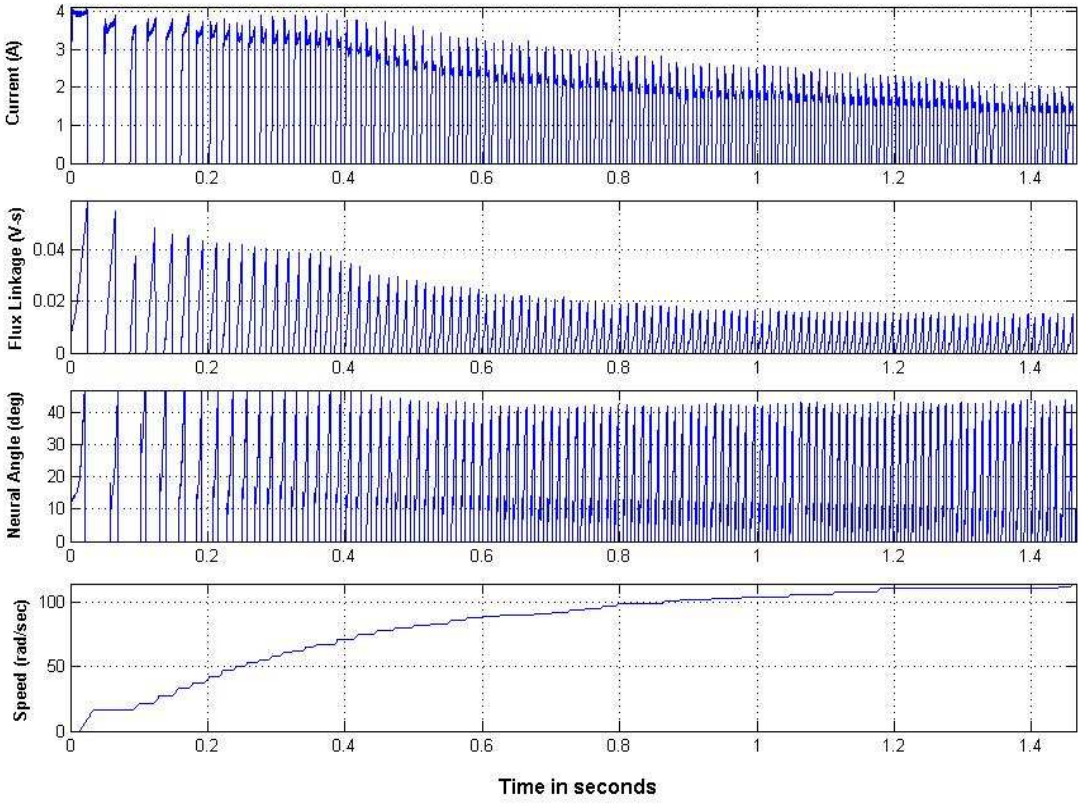

Figure 7.4: Closed-Loop Speed Experimental Results

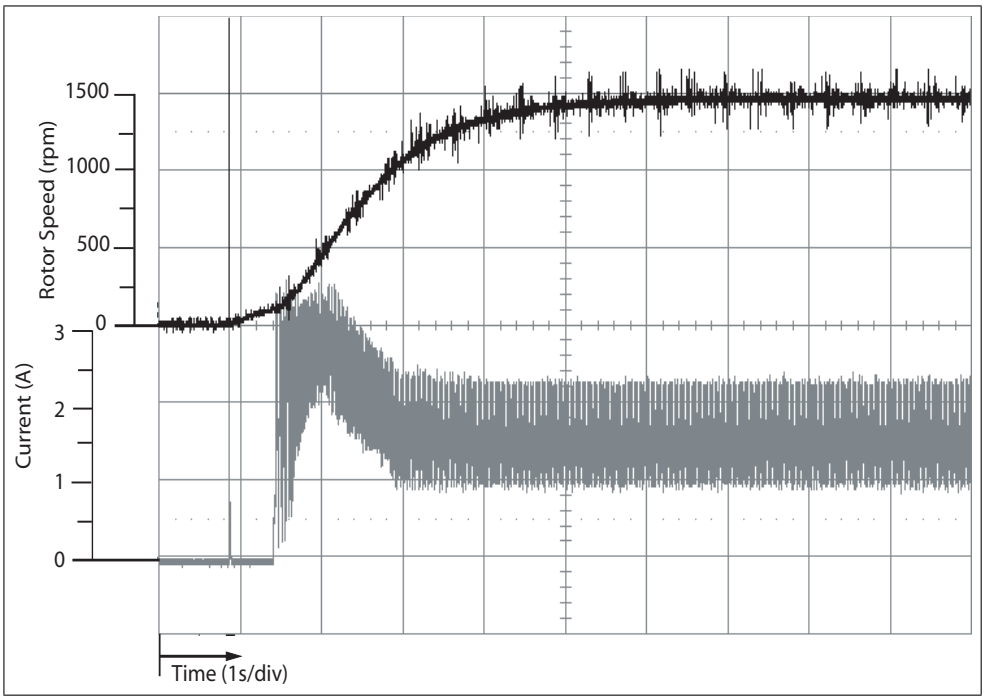

Figure 7.5: Speed and Current Response from Standstill to Steady-state

Overall, the experimental results closely match the simulation results. The actual neural response also mimics the response predicted by the neural training test in Figure 5.3.

7.4 Evaluation and Observations

The closed-loop speed sensorless drive was remarkably stable. Some limitations existed with minimum currents for both signal sensing and torque excitation, but the drive operated across a wide range of current references. One performance limitation existed in the current controller. The current loop dynamic performance is limited to a digital P + I zero location of $0.88 < z < 0.995$. This suggests that the flux linkage estimation accuracy is dependant on the response of the current loop. In turn, the accuracy of the flux linkage estimation affects the accuracy of the neural net and

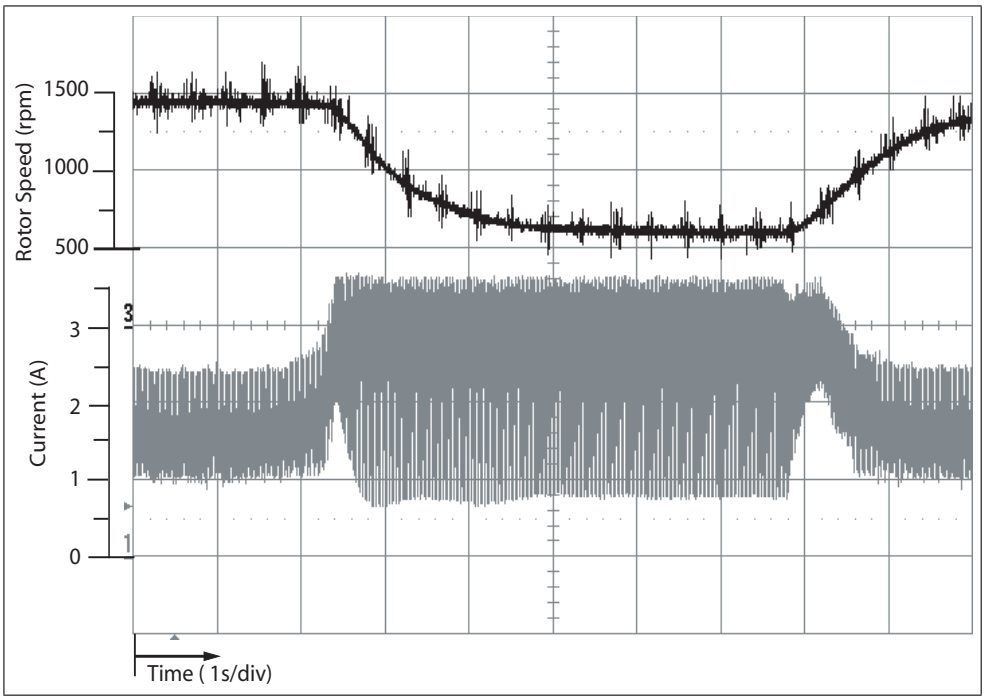

Figure 7.6: Dynamic Response to Heavy Load

thus the overall performance of the sensorless system. This limitation is non-critical due to the range of zero locations available. The current response is satisfactory to $z = 0.995$ and becomes sluggish past this. The sensorless drive performance is however very sensitive to the overall gain of the current loop. A better speed response occurs at lower current loop gains. A satisfactory level for the current P + I controller was tuned during closed-loop speed drive performance evaluation.

7.5 Summary

The startup of the sensorless algorithm is critical to proper tracking and performance. The sensorless algorithm should start all timers at or very near complete unalignment. For the initial sensorless algorithm debug, a routine was developed to use the hall-effect sensors to synchronize the algorithm start to rotor unalignment. This is not

demonstrated by the firmware code flow in Figure 7.1. However, the idea will be expounded in the next chapter when the sensorless starting algorithm is developed. It will be an extension of the code flow we have presented in this chapter, which is deemed the "normal operation" of the sensorless drive and directly replaces the routine using the hall-effect sensors.

Chapter 8

Sensorless Starting, Future Work, and Concluding Remarks

In this final chapter we present additional work developed to complete the sensorless SRM drive, the starting algorithm. Also, we investigate in some detail as to applications in which this research can be used as well as how the technology can be improved. A brief summary of the work done to date will be given to bring the research to close.

8.1 Introduction

We briefly discussed the necessity of starting the sensorless algorithm timers at or near full unalignment, zero degree reference, of the rotor with respect to the main stator windings. The normal operation of the sensorless motor drive requires this for the speed estimate to converge quickly. The speed estimate is derived from the sensorless algorithm timers. If this estimate doesn't converge within several commutation periods, the sensorless algorithm can alias to conduction at every 1.5 conduction intervals. This will still produce an average positive torque, but the response of the

drive will be sluggish and the speed error will be great. To overcome this we must detect the unalignment position without the use of flux linkage and the neural net as they are unavailable at startup.

An alternate method by which we can detect rotor position without using flux linkage was discussed briefly in Chapter 2. The method is called active probing. Reluctance variations occur during the rotor position transition with respect to the stator. The saliency of the motor design can be exploited to determine the zero degree reference. At this point, the impedance of the stator is at it's low point. If a rise or peak in current can be detected at this point, the motor can be switched to normal operation. This is the basis of the startup algorithm. This chapter will discuss in detail the implementation of the startup algorithm and what methods are used to switch over to normal sensorless operation.

8.2 Sensorless Starting

As demonstrated by [46, 18], an active probing method was used to detect the unaligned position of the rotor with respect to the stator. The variable inductance characteristic of the machine was exploited to determine a reference position of the rotor without requiring special additions either to the machine rotor or stator; i.e. notches, magnets, etc. A reference position is necessary to start the normal sensorless operation algorithm properly. To avoid aliasing of the sensorless timers, the normal conduction of the stator windings needs to begin with the rotor close to the unaligned position.

The sensorless starting method employs a series of large pulses to start the rotor turning from standstill. A small amount of current, around 150mA, is then injected

into the phase windings using a constant, small, PWM duty cycle and the main drive switch. By sampling the current, variations in the winding current are detected as the rotor turns. The constant winding voltage results in current oscillations that occur as the rotor turns through regions of minimum and maximum inductance. The inductance behavior of the SRM is described by Equation 8.1.

$$v = L(\theta)\,\frac{di}{dt} \tag{8.1}$$

If the winding voltage is held constant and inductance is known to vary with rotor position, the current varies inversely proportional to the inductance.

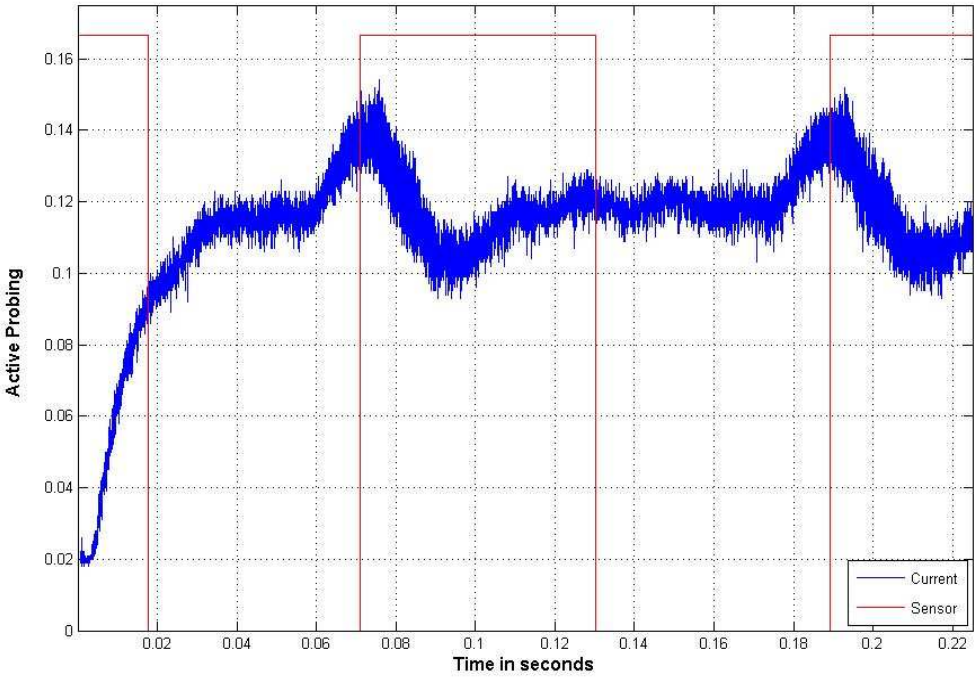

Figure 8.1: Current Probing with respect to Rotor Position

In Figure 8.1, this current variation can be seen with respect to the position sensor

signal. During rotation when the rotor turns to the unaligned position, the current in the stator rises. This rise in current is the basis for determining that the rotor is close to the unaligned position.

When the current during the active probe exceeds a preset reference, the algorithm switches to the normal sensorless drive algorithm. The startup algorithm utilizes a simple flag that is latched when the probe current exceeds a reference. The startup algorithm is fully explained in Figure 8.2.

The entire starting process of the motor drive, from the starting pulse to the normal sensorless operation, can be seen in Figure 8.3. The start pulses occur at a very specific intervals. The specific interval is necessary to both get the rotor turning at any angle and to insure that the rotor turns in one direction. A specific delay follows the starting pulses. In turn, the delay is followed by the active current probe. Depending on the level of excitation as the rotor passes the first unaligned position, the current peak level may or may not be sufficient to exceed the algorithm's preset reference. Thus, the algorithm may require the rotor to turn through a maximum of two unaligned positions before switching over to the normal sensorless mode. This method can cause some hesitation which the elimination therein is discussed by [21]

The normal operation block is explained fully by the interrupt branch of Figure 7.1. After the startup mode has ceased, all operations such as current control, flux linkage estimation, neural estimation, and sensorless commutation occur. A reset function can be derived to detect if a startup peak doesn't occur within a specific amount of time. If the peak doesn't occur, a second pulse sequence can be given, and the startup routine can be repeated.

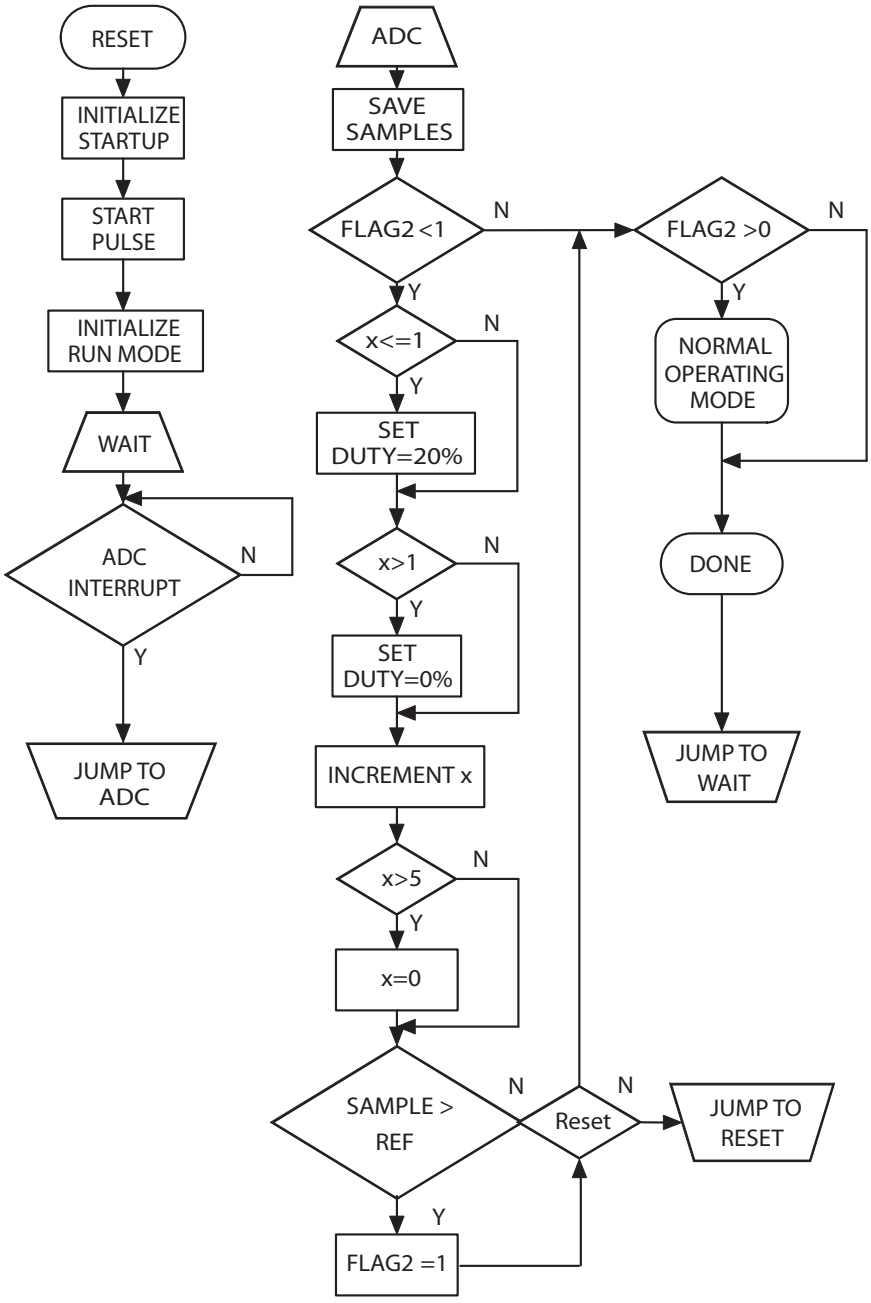

Figure 8.2: Sensorless Starting Algorithm

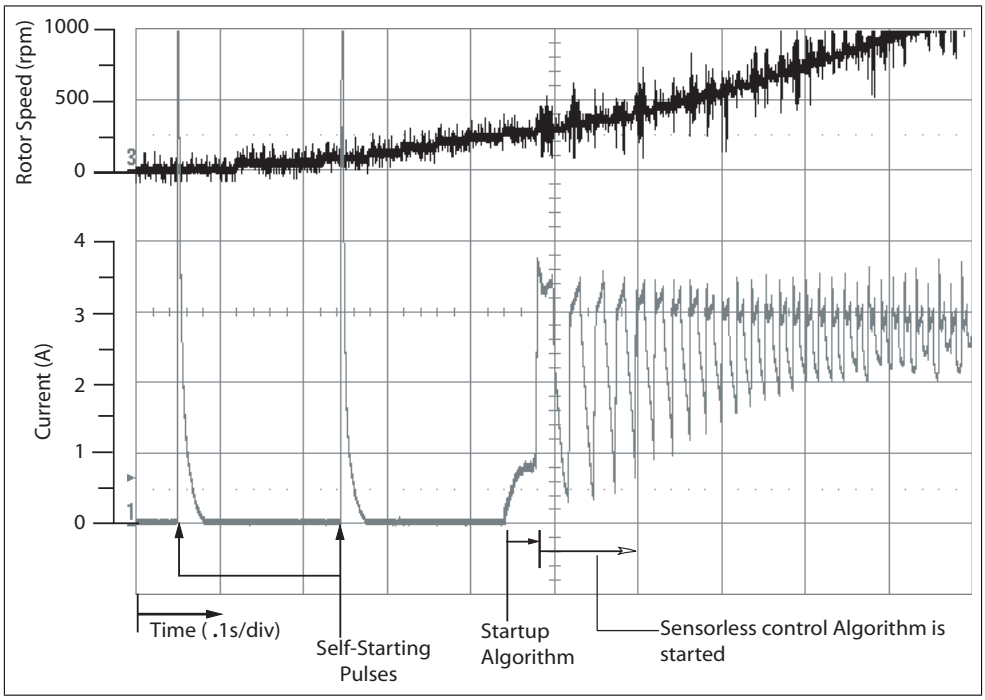

Figure 8.3: Sensorless Startup

8.3 Future Work

This research can be further improved by investigating several shortcomings encountered during this project. One, the flux linkage surface used to train the neural net only encompassed the active control range up to around 3 amps. This is only 40 percent of the rated range of the motor. Also, the current control was implemented with a simple P + I controller. The P+I proved to be both effective and easy to implement but became limited when the speed increased. Even while using the hall sensors for commutation, it became very difficult to achieve the current reference in the stator when the speeds were above 2000 RPM. Two, most SRM drives utilize some advanced angle calculations to commutate current into the phase starting prior to the positive torque generation interval. By providing this advanced angle commutation, the P + I

controller is provided the ability to achieve the current reference while the inductance is low and then maintaining the current. The torque is thus maintained at a high level throughout the positive torque generation region. The sensorless algorithm could also benefit from advanced angle calculation. However, the initial condition table for the flux linkage estimation would need to be expanded to include all of the angles required by the advanced angle commutation. This was not implemented due to two factors, memory and complexity. The table required to implement this exceeded the bounds of available flash memory on the DSP. The interpolation calculation required to use the advanced angle flux linkage correction added too much complexity to the code as to not be able to satisfy all the other control requirements necessary during one sample cycle. A much faster processor with more memory or much more efficient code compilation is needed to implement this.

A better way to overcome the inability to push current through the stator windings at high speeds would be to implement a sliding mode gain schedule for the current P+I controller. The sliding mode control could use the speed reference and the neural output to gauge it's control optimization. In fact, this may be the best method to implement to match the physical behavior of the SRM. However, some consideration will need to be made as to the effect of the current control on the flux linkage estimation.

If either method were to be implemented, the neural net would need to be retrained. By providing new features, it could possibly improve both the speed response and the top speed of the motor drive. Reset functions, stall recognition functions, as well as other features could be added to improve the commercial viability of the research. This research presented the basic technological development for a single

switch, single phase sensorless drive based on flux linkage - neural net estimation. As an aside note, an inductance estimator was developed but not chosen for implementation due to prior work and the novelty or the lack thereof.

8.4 Concluding Remarks

This research is both effective and reliable once the electronics and firmware were tuned to give precise current control and flux linkage estimation. Initially, a current controller was implemented using a resistive current sensor. This type of feedback proved to be not accurate enough to provide a consistent flux linkage estimate. The accuracy required for proper sensorless control limits the commercial viability due to the cost associated with the necessary hall-effect current sensor. Improvements in the supporting electronics were created to offset some of the cost. A high voltage DC/DC converter was constructed to provide single sided power supply to drive the feedback signal amplifiers. The cost associated with the power converter was estimated around $2.00. Aside from the converter, the necessary electronics consisted only of current and voltage sensors, amplifiers, DSP, and the power converter. The entire functionality of the SRM drive from current control, speed control, and sensorless commutation existed completely in firmware on the DSP controller. This is significant in the reduction of the entire system cost. The entire converter cost in low quantities (1KU/year) is estimated at $8 for the SRM drive, $2 for the DC/DC power supply, $2 for the signal conditioning electronics, and $4 for the DSP; for a total estimated system cost of $16. When the electronic cost is coupled with the low price of the motor and high volumes, this system should provide a very effective solution to compete with universal induction motor applications.

Appendix A

Appendix

A.1 Derivation of Small Signal Linear Model

Analysis method largely excerpted from: [25].

$$v = R_s\, i + \frac{d\lambda(\theta, i)}{dt} \tag{A.1}$$

$$T_e(\theta, i) - T_l = J\frac{d\omega_m}{dt} + B\omega_m \tag{A.2}$$

by using the following relationship $L\, i = \lambda$

we have:

$$v = R_s\, i + \frac{d}{dt}(L\, i) \tag{A.3}$$

expanding this:

$$v = R_s\, i + L\frac{di}{dt} + \frac{dL}{d\theta}\, \omega_m\, i \tag{A.4}$$

Also we know:

$$T_e = \frac{1}{2} i^2 \frac{dL}{d\theta} \tag{A.5}$$

By substitution of A.5 into A.2, we have:

$$\frac{1}{2} i^2 \frac{dL}{d\theta} - T_l = J \frac{d\omega_m}{dt} + B\omega_m \tag{A.6}$$

Definition A.1.1.

$$i = i_o + \delta i \tag{A.7}$$
$$\omega_m = \omega_m o + \delta\omega_m \tag{A.8}$$
$$v = v_o + \delta v \tag{A.9}$$
$$T_l = T_l o + \delta T_l \tag{A.10}$$
$$\tag{A.11}$$

By the substitution of the perturbed signals into A.4 and A.6, we have:

$$\frac{d(\delta i)}{dt} = (-\frac{R_s}{L} - \frac{1}{L}\frac{dL}{d\theta}\omega_{mo})\delta i - \frac{1}{L}\frac{dL}{d\theta}\, i_o\, \delta\omega_m + \frac{\delta\, v}{L} \tag{A.12}$$

and

$$\frac{d(\delta\omega_m)}{dt} = (-\frac{1}{J}\frac{dL}{d\theta} i_o)\delta i - \frac{B}{J}\delta\omega_m + \frac{\delta T_e}{J} \tag{A.13}$$

Definition A.1.2.

$$R_e q = R_s + \frac{dL}{d\theta}\omega_{mo} \tag{A.14}$$
$$K_b = \frac{dL}{d\theta} i_o \tag{A.15}$$
$$\delta e = \frac{dL}{d\theta} i_o \delta\omega_m = K_b \delta\omega_m \tag{A.16}$$

By substitution of the equivalent variables into A.12 and A.13 and transforming to the frequency domain, we can derive the linear small signal model.

Starting with the substitution and transformation of A.12, we have:

$$s(\delta i) = -\frac{1}{L}\ R_{eq}\delta i - \frac{1}{L}\delta e + \frac{\delta v}{L} \tag{A.17}$$

$$sL(\delta i) = -R_{eq}\delta i - \delta e + \delta v \tag{A.18}$$

$$(R_{eq} + sL)\delta i = \delta v - \delta e \tag{A.19}$$

$$\delta i = \frac{\delta v - \delta e}{R_{eq} + sL} \tag{A.20}$$

and

$$s(\delta\omega_m) = -\frac{1}{J}\ K_b\ \delta i - \frac{B}{J}\delta\omega_m + \frac{\delta T_e}{J} \tag{A.21}$$

$$sJ(\delta\omega_m) = -K_b\ \delta i - B\ \delta\omega_m + \delta T_e \tag{A.22}$$

$$(B + sJ)\delta\omega_m = -K_b\ \delta i + \delta T_e \tag{A.23}$$

$$\delta\omega_m = \frac{K_b\ \delta i - \delta T_e}{B + sJ} \tag{A.24}$$

The resulting system can now be shown in Figure A.1:

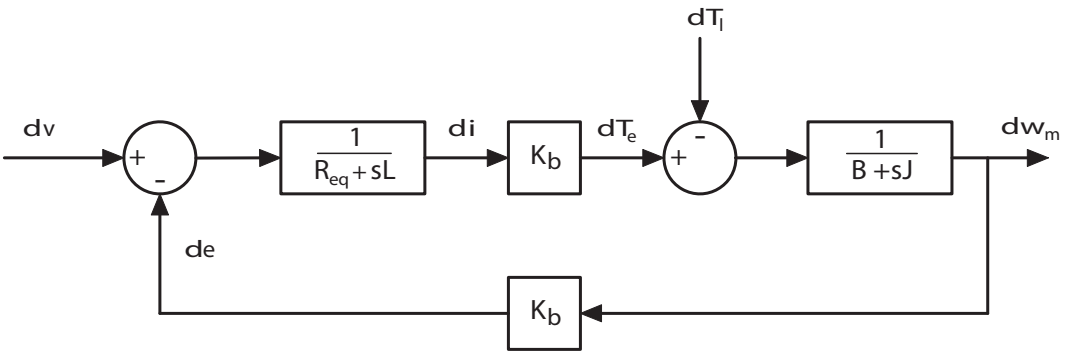

Figure A.1: Small Model Block Diagram

However, the drive converter is the input system δv. This causes a current loop from δi back. This loop is interdependent on the back-emf loop from $K_b \delta \omega_m$. Thus by simple block reduction we can arrive at the final decoupled model shown in Figure A.2

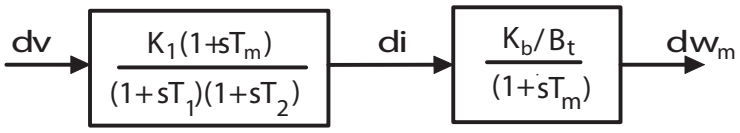

Figure A.2: Reduced Small Model Block Diagram

where the time constants and gains are combinations of the previous defined block parameters in A.1:

$$B_t = B + B_l \tag{A.25}$$

$$K_1 = \frac{B_t}{K_b^2 + R_{eq}\ B_t} \tag{A.26}$$

$$T_m = \frac{J}{B_t} \tag{A.27}$$

$$[-\frac{1}{T_1}, -\frac{1}{T_2}] = -\frac{1}{2}[\frac{B_t}{J} + \frac{R_{eq}}{L}] + \sqrt{\frac{1}{4}[\frac{B_t}{J} + \frac{R_{eq}}{L}]^2 - \frac{K_b^2 + R_{eq}\ B_t}{J\ L}} \tag{A.28}$$

Further, the current control loop can be approximated with a simple first order approximation. This is given in A.3 by:

$$\frac{I_a}{I_a^*} = \frac{K_i}{1 + s\ T_i} \tag{A.29}$$

where

$$Ti = \frac{T_1}{1 + K_{fi}} \tag{A.30}$$

$$Ki = \frac{K_{fi}}{H_c}\frac{1}{1 + K_{fi}} \tag{A.31}$$

$$K_{fi} = \frac{K_c\ K_r\ K_1\ T_m\ H_c}{T_c} \tag{A.32}$$

The final control loops are represented as:

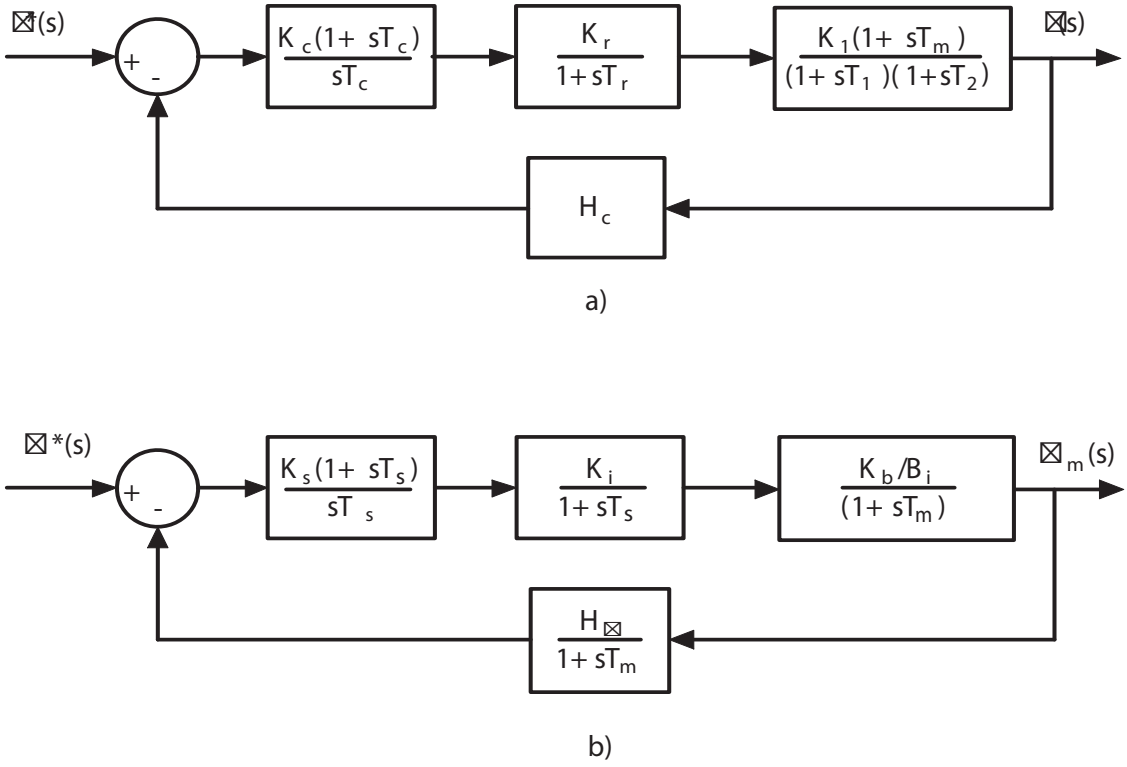

Figure A.3: Simplified Linearized Models of Current and Speed Loops [25, 39]

A.2 Analysis and Design of the Linear Controllers

A.2.1 Prop + Int Current Controller Analysis

Assuming that $T_r < T_2 < T_1$, we can safely state that:

$$T_c = T_2 \tag{A.33}$$

and that:

$$K_c = \frac{1}{2}\frac{T_1\ T_c}{T_r}\frac{1}{K_1\ K_r\ H_c\ T_m} \tag{A.34}$$

A.2.2 Prop + Int Speed Controller Analysis

Using the first order approximation defined in A.29, we can use the symmetric optimum design method as in [24] to design the speed controller. Thus by design method we define:

$$T_4 = T_i + T_\omega \tag{A.35}$$

$$K_2 = \frac{K_i\ K_b\ H_\omega}{B_t\ T_m} \tag{A.36}$$

we can clearly state the speed controller as:

$$K_s = \frac{1}{2\ K_2\ T_4} \tag{A.37}$$

$$T_s = 4\ T_4 \tag{A.38}$$

A.2.3 Controller Design and Implementation

We can now assign design values to the previous analytical equations to rough out a good starting point for the design.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Matlab Code for Analytical Controller Design
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc; close all; clear all; pi=3.14159; currentsamp=50e-06;
speedsamp=0.001; s=tf('s');z=tf('z',currentsamp);
Ra=1.3;                 %Ohms
La=0.008;               %Henries ...%Lamax=8mH or Lamin=1mH
Irated=8;               %Amps    Rated Current
dL=0.007/(pi/4);        %Henries/Radian   Profile Rate of Change
Lavg=0.004; Kb=dL*Irated;
J=0.0047; B1=0.001;
BL=0.001;               %Frictional load torque
Bt=B1+BL;
```

```
%Max. Machine Characteristics
hp=0.25; rpm=5000;
wm=(rpm*2*pi/60); %rad/sec

%Source Characteristics
Vdc=170; Vcm=5;

%Switching Frequency
f=20e03;

Hc = 1;                 %Current Feedback Gain
Hw = 1;                 %Speed Feedback Gain
Tw=0.0028;
S_Filter = Hw/(Tw*s+1); %Speed Feedback Filter

%Transfer function ws/Va  speed/volts
Req=Ra+dL*wm; La=Lavg;

%Electrical Ia/Va
G1=1/(Req+s*La);
%Mechanical wm/Ia
G2=1/(Bt+s*J);

%Separation of induced emf feedback

G1f=Kb^2*G2;

H1=feedback(G1,G1f); H2=Kb*G2;

%Calculated Machine
K1=Bt/(Kb^2+Req*Bt);

a=Bt/J+Req/La;
T1=-1/(-0.5*a+[0.25*a^2-(Bt/K1/J/La)]^0.5);
T2=-1/(-0.5*a-[0.25*a^2-(Bt/K1/J/La)]^0.5); Km=Kb/Bt; Tm=J/Bt;

H1=K1*((1+s*Tm)/((1+s*T1)*(1+s*T2)));
H2=Km*(1/(1+s*Tm));

%Calculated Converter Constants
Kr=Vdc/Vcm;
Tr=1/(f/2);         %average duty of chopper type converter
Gr=Kr/(1+s*Tr);

%Current Loop Controller
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Tc=T2;
Kc=T1*Tc/(2*Tr*K1*Hc*Tm);
Gc=(Kc*(1+s*Tc))/(Tc*s);        %Analytical Current PI controller



%Simplify Current Loop Controller to First Order Approximation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Kfi = Kc*Kr*K1*Tm*Hc/Tc;
Ki = (Kfi/Hc)*(1/(1+Kfi));
Ti =(T1+Tr)/(1+Kfi);
FOCL = Ki/(1+s*Ti);



%Speed Loop Controller
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
K2 = Ki*Kb*Hw/Bt/Tm;
T4 = Ti + Tw;

Ks = 1/2/K2/T4;
Ts = 4*T4;
Gs=(Ks*(1+s*Ts))/(Ts*s);        %Analytical Speed PI controller



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%END CODE
```

%%

The resulting controller gains are:

$$\text{Current:} \quad K_c = 20 \quad T_c = 0.00067$$
$$\text{Speed:} \quad K_s = 11.75 \quad T_s = 1047$$

The current controller was decided to be converted into a digital domain controller. Using a sample period of $50uS$ and using a bilinear transformation, we derive the controller as:

$$G_c = K \, \tfrac{(z-0.928)}{(z-1)}$$

The stable region for current controller gain was determined to be $K > 0$. Thus, the gain was set at $K = 1$ in implementation and fine tuned in software during runtime.

The speed controller however was implemented in a typical way. First the speed error was calculated and then integrated. The sample time was arbitrarily chosen to be $1mS$. The control law was implemented on the DSP as:

$$I_{ref} = K_p \, e_{speed} + K_i \, \int e_{speed}$$

where:

$$K_p = 11.75 \quad K_i = 0.01$$

After some fine tuning both controllers were altered slightly. The final controllers are:

$$G_c = 0.75 \, \tfrac{(z-0.995)}{(z-1)}$$
$$G_s = 10 \, e_{speed} + 0.01 \, \int e_{speed}$$

A.3 Hardware Schematics

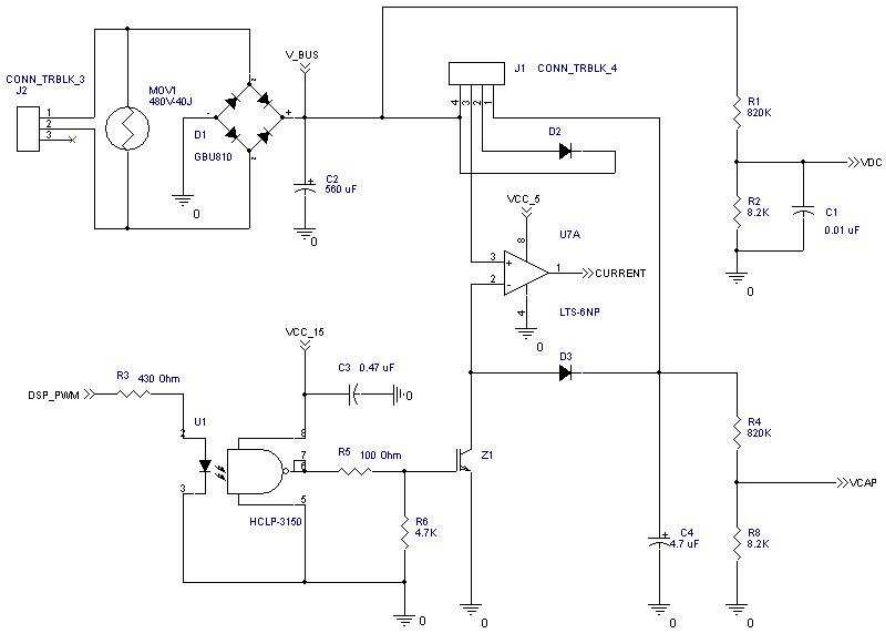

Figure A.4: Drive Converter Schematic

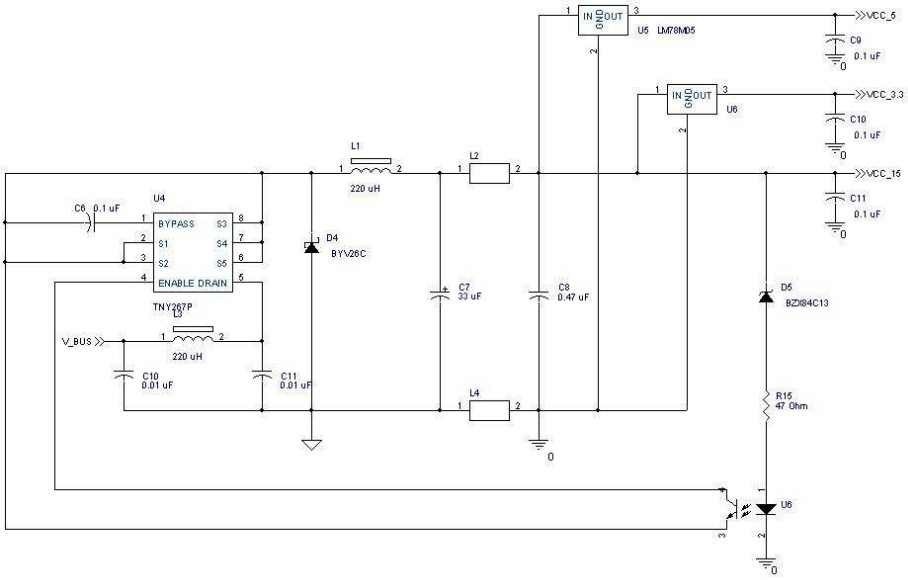

Figure A.5: DC/DC Power Converter Schematic

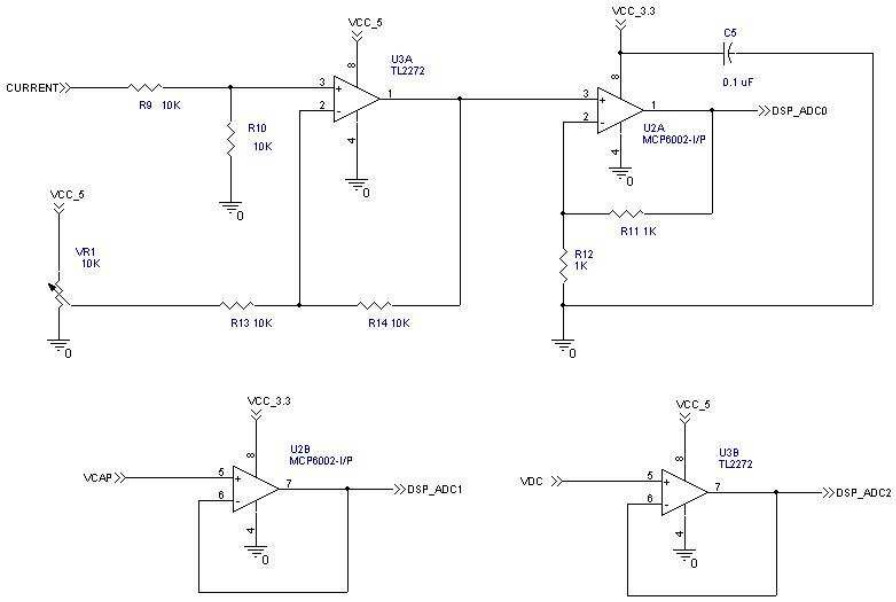

Figure A.6: Feedback Signal Conditioning Schematic

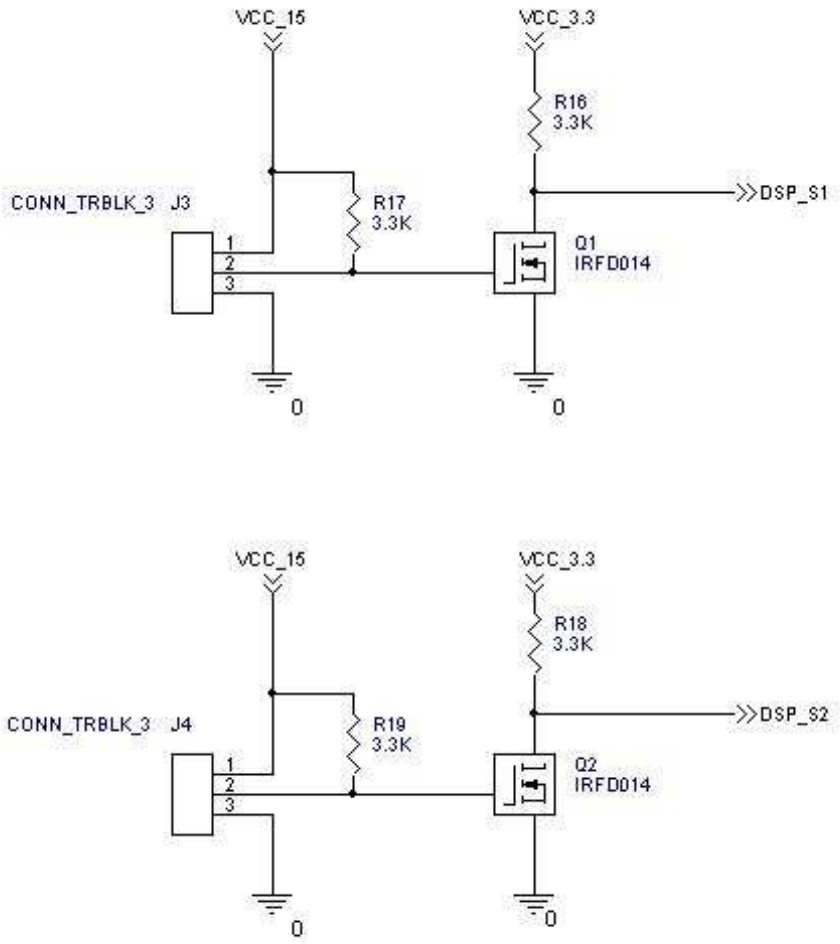

Figure A.7: Position Sensor Signal Feedback Schematic

Bibliography

[1] I.H. Al-Bahadly, *Analysis of position estimation method for switched reluctance drives*, Electronic Design, First IEEE International Workshop, January 2002, pp. 262 – 266.

[2] P. Arcarnley, R. J. Hill, and G. W. Hooper, IE, vol. 32, IEEE Transaction, 1985, pp. 215–222.

[3] P. Arpaia, P. Daponte, D. Grimaldi, and L. Michaeli, *Systematic error correction for experimentally modeled sensors by using anns*, Instrumentation and Measurement Technology Conference, vol. 3, Proceedings of the 16th IEEE, May 1999, pp. 1635 – 1640.

[4] Won-Sik Baik, Min-Huei Kim, Nam-Hun Kim, and Dong-Hee Kim, *Position sensorless control system of srm using neural network*, Power Electronics Specialists Conference, vol. 5, IEEE 35th Annual, June 2004.

[5] A. Cheok and N. Ertugrul, *High robustness and reliability of a fuzzy logic based angle estimation algorithm for practical switched reluctance motor drives*, PESC 98 Record. 29th Annual IEEE, vol. 2, Power Electronics Specialists Conference,, May 1998, pp. 1302 – 1308.

[6] F.N. Chowdhury, P. Wahi, R. Raina, and S. Kaminedi, *A survey of neural networks applications in automatic control*, 33rd, Southeastern Symposium on System Theory, March 2001, pp. 349 – 353.

[7] S. Cincotti, A. Fanni, M. Marchesi, and A. Serri, *An artificial neural network position estimator for a variable reluctance linear actuator*, Power Electronics Specialists Conference, vol. 1, 27th Annual IEEE, June 1996, pp. 695 – 700.

[8] M.T. DiRenzo and W. Khan, *Self-trained commutation algorithm for an sr motor drive system without position sensing*, Thirty-Second IAS Annual Meeting, IAS '97, vol. 1, Industry Applications Conference, October 1997, pp. 341 – 348.

[9] M. Ehsani and B. Fahimi, *Elimination of position sensors in switched reluctance motor drives: state of the art and future trends*, vol. 49 Issue 1, Industry Applications, no. 1, IEEE transactions, Feb. 2002, pp. 40 – 47.

[10] E. S. Abdin et. al., *Efficiency optimization of a vector controlled induction motor drive using an artificial neural network*, IEEE-IECON, 2003, pp. 2543–2548.

[11] Lyons et al, *Rotor position estimator for switched reluctance machine*, U.S. Patent 5097190, March 1992.

[12] R. Krishnan et al, *A novel single-phase switched reluctance motor drive system*, IEEE-IECON, 2001, pp. 1488–1493.

[13] ———, *Theory and operation of a four quadrant switched reluctance motor drive with a single controllable switch-the lowest cost four quadrant brushless motor drive*, IEEE-IAS, 2004.

[14] B. Fahimi and R. Sepe, *Development of 4-quadrant sensorless control of srm drives over the entire speed range*, IEEE, Industrial Electronics Confernce, 2002, pp. 1625–1632.

[15] G. Gallegos-Lopez, P.C. Kjaer, and T.J.E. Miller, *High-grade position estimation for srm drives using flux linkage/current correction model*, Industry Applications, vol. 35 Issue 4, IEEE Transactions, July-Aug 1999, pp. 859 – 869.

[16] Madan Gupta, Liang Jin, and Noriyasu Homma, *Static and dynamic neural networks*, Wiley Interscience, John Wiley and Sons, 2003.

[17] M. T. Hagan and M. Menhaj, *Training feedforward networks with the marquardt algorithm*, no. 6, vol. 5, IEEE TRANSACTIONS ON NEURAL NETWORKS, 1994, pp. 989–993.

[18] Mohammad Islam, Tomy Mir, Sayeed .and Sebastian, and Iqbal Husain, *Sensorless control of switched reluctance electric machines*, U.S. Patent 6,801,012, October 2004.

[19] F. Ismail, S. Wahsh, A. Mohamed, and H. Elsimary, *Optimal control of variable reluctance motor by neural network*, Industrial Electronics, Conference Proceedings, ISIE, IEEE International Symposium, June 1993, pp. 301 – 304.

[20] X. et al Jiang, *Neural network speed controller of induction motor drive based on direct mrac method*, IEEE-IECON, 2003, pp. 2531–2536.

[21] IEEE Jianrong Bu, Member and IEEE Longya Xu, Senior Member, *Eliminating starting hesitation for reliable sensorless control of switched reluctance motors*, IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS, 1, vol. 37, IEEE, February 2001, pp. 59–66.

[22] Toshiyuki Kaitani, *Sensorless control device for synchronous electric motor*, U.S. Patent 6,791,293, September 2004.

[23] Jinyoung Kim, Kabdong Kim, Youngseong Hwang, Sejin Seong, Jaedong Choi, and Hakju Lee, *Robustness analysis of the estimation of rotor position in srm drive*, Industrial Electronics, vol. 3, IEEE International Symposium, June 2001, pp. 1792 – 1797.

[24] R. Krishnan, *Electric motor drives-modeling, analysis, and control*, Prentice Hall, 2001.

[25] ———, *Switched reluctance motor drives*, edition 1 ed., CRC Press LLC, 2001.

[26] R Krishnan, *Single switch based power converter topologies*, U.S. Patent Pending, 2002, VTIP.

[27] T. Lachman, T.R. Mohamad, and S.P. Teo, *Sensorless position estimation of switched reluctance motors using artificial neural networks*, Robotics, Intelligent Systems and Signal Processing, vol. 1, IEEE International Conference, October 2003, pp. 220 – 225.

[28] Z. Lin, D. Reay, and Williams B., *On-line instant torque estimation of switched reluctance motor using adaptive b-spline neural networks*, IEEE-IECON, 2003, pp. 1033–1037.

[29] A. Lumsdaine, J.H. Lang, and M.J. Balas, *A state observer for variable reluctance motors*, Motion Control and System Design, Annual Symposium on Incremental Motion Control, 1986, pp. 267–273.

[30] S.R. Lyons, J.P. v MacMinn and M.A. Preston, *Flux-current methods for srm rotor position estimation*, Industry Applications Society Annual Meeting, vol. 1, Conference Record of the IEEE, October 1991, pp. 482 – 487.

[31] Joseph Marcinkiewicz, *Flux feedback control system*, U.S. Patent 6,731,083, May 2004.

[32] Joseph Marcinkiewicz and Michael Henderson, *Control system and method for a rotating electromagnetic machine*, U.S. Patent 6,756,757, June 2004.

[33] Ehsani; Mehrdad, *Self-tuning control of switched-reluctance motor drive system*, U.S Patent 6,472,842, October 2002.

[34] E. Mese and D.A. Torrey, *Sensorless position estimation for variable-reluctance machines using artificial neural networks*, Industry Applications Conference,

vol. 1, Thirty-Second IAS Annual Meeting, Conference Record of the 1997 IEEE, October 1997, pp. 540 – 547.

[35] ———, *An approach for sensorless position estimation for switched reluctance motors using artifical neural networks*, Power Electronics, vol. 17, IEEE Transactions, January 2002, pp. 66 – 75.

[36] N.H. Mvungi, M.A. Lahoud, and J.M. Stephenson, *A new sensorless position detector for sr drives*, European Power Electronics conference, 1990, pp. 249–252.

[37] H.S. Ooi and T.C. Green, *Simulation of neural networks to sensorless control of switched reluctance motor*, Power Electronics and Variable Speed Drives, Seventh International Conference IEEE, September 1998, pp. 281 – 286.

[38] D. Panda and V. Ramanarayanan, *Sensorless control of switched reluctance motor drive with self-measured flux-linkage characteristics*, Power Electronics Specialists Conference, vol. 3, IEEE 31st Annual, June 2000, pp. 1569 – 1574.

[39] Sung Yeul Park, *Design and implementation of four-quadrant operation in single-switch based switched reluctance motor drive system*, Masters thesis, Virginia Polytechnic Institute and State University, July 2004.

[40] K. M. et al Rahman, *Optimized torque control of switched reluctance motor at all operational regimes using neural network*, IEEE-IECON, 1998, pp. 701–708.

[41] Krishnan Ramu, *Ee 5724 neural and fuzzy systems*, VPI and SU Graduate Classwork, Spring 2004.

[42] D.S. Reay, T.C. Green, and B.W. Williams, *Application of associative memory neural networks to the control of a switched reluctance motor*, Industrial Electronics, Control, and Instrumentation, vol. 1, Proceedings of the IECON, November 1993, pp. 200 – 206.

[43] ———, *Minimisation of torque ripple in a switched reluctance motor using a neural network*, Artificial Neural Networks, Third International Conference, May 1993, pp. 224 – 228.

[44] D.S. Reay and B.W. Williams, *Sensorless position detection using neural networks for the control of switched reluctance motors*, Control Applications, vol. 2, Proceedings of the 1999 IEEE International Conference, August 1999, pp. 1073 – 1077.

[45] T. et al Senjyu, *Speed sensorless control of ultrasonic motors using neural network*, IEEE-IECON, 2003, pp. 335–340.

[46] Howard James Slater, *Method and system for determining rotor position in a switched reluctance machine*, U.S. Patent 6,608,462, August 2003.

[47] ———, *Rotor position detection of a switched reluctance drive*, U.S. Patent 6,853,163, February 2005.

[48] A.M. Staley, *Design and implementation of a novel single-phase switched reluctance motor drive system*, Masters thesis, Virginia Polytechnic Institute and State University, August 2001.

[49] P. Tai, H.A. Ryaciotaki-Boussalis, and D. Hollaway, *Neural network implementation to control systems: a survey of algorithms and techniques*, Signals, Systems and Computers, vol. 2, Twenty-Fifth Asilomar Conference, November 1991, pp. 1123 – 1127.

[50] Paul J. Werbos, *Neural networks for intelligent control*, U.S. Patent 6,882,992, April 2005.

[51] L.Y. et al Woo, *The design of motor speed controller using neural networks for disturbance rejection*, IEEE-IECON, 2004, pp. 1739–1742.

[52] Tang Yifan, *Low-cost universal drive for use with switched reluctance machines*, U.S. Patent 5,841,262, November 1998.

[53] ———, *Method and apparatus for implementing a low cost, intelligent controller for a switched reluctance machine*, U.S. Patent 6,442,535, August 2002.