ON-LINE PARAMETER ESTIMATION AND ADAPTIVE CONTROL OF PERMANENT MAGNET SYNCHRONOUS MACHINES

A Dissertation

Presented to

The Graduate Faculty of the University of Akron

In Partial Fulfillment

Of the Requirements for the Degree

Doctor of Philosophy

Samuel J. Underwood

May, 2006

ON-LINE PARAMETER ESTIMATION AND ADAPTIVE CONTROL OF PERMANENT MAGNET SYNCHRONOUS MACHINES

Samuel J. Underwood

Dissertation

Approved:	Accepted:
Advisor Dr. Iqbal Husain	Department Chair Dr. Alexis De Abreu-Garcia
Committee Member Dr. Robert Veillette	Dean of the College Dr. George K. Haritos
Committee Member Dr. Joan Carletta	Dean of the Graduate School Dr. George R. Newkome
Committee Member Dr. Graham Kelly	Date
Committee Member Dr. Kevin Kreider	

ABSTRACT

High performance control of permanent magnet machines (PMSM) requires accurate knowledge of the parameters that describe their mathematical models. This parameter information enables the controller to optimize the drive performance and efficiency, and to react to possible changes in the machine model. Several methods have been tested in order to get motor parameter estimates. Most of them are based on off-line measurements or estimates, which are then stored in the controller. Other methods published are compatible with on-line implementation, but these are usually restricted to a subset of the machine parameters.

This dissertation proposes a solution to the problem of on-line estimation of PMSM stator resistance, torque constant and *d*-*q* inductances. An analysis of the machine parameters and their effects on motor drive performance that motivates the development of a new parameter estimation algorithm is presented. This algorithm combines two instances of the recursive least squares method, which interact in order to account for different machine parameter dynamics. As a consequence, the presented method is able to provide the controller with parameter estimates even when sudden changes in operation take place.

The effectiveness of the proposed parameter estimation algorithm is validated using both a computer simulation and an experimental motor drive. This simulation model was developed specifically for this project and includes accurate inverter and controller modeling, in addition to a machine model that features parameter variation. The simulation model is used in both the algorithm development stages and its validation. The experimental setup provides additional verification of the effectiveness of the proposed algorithm. It is based on the use of a digital signal processor for the controller algorithm implementation, and includes motor drive classical algorithms as well as the proposed parameter estimation program. Both simulation and experimental results demonstrate the performance of the parameter estimation algorithm, both in transient and steady state operations.

DEDICATION

To my fiancée, Coral, and my family.

ACKNOWLEDGEMENTS

First, I would like to thank Dr. I. Husain, who offered me the opportunity to come to Akron and gave me the motivation and assistance I needed to accomplish this. I am deeply grateful for his help and support.

I would also like to thank the members in my Ph.D. committee, for their advice and their help during the project. I did not get the chance to take classes with all of them, but I would like to add that the classes I took with Dr. Husain and Dr. Veillette have improved my skills and understanding of motor drives and control systems greatly.

My thanks also go to Dr. Mir and Dr. Islam, with whom I worked for two summers at Delphi Saginaw Steering Systems. This experience was extremely beneficial for me.

Finally, I would like to express my gratitude to the staff of the department of electrical and computer engineering, and particularly to Mrs. Boden.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The development of digital electronics and the recent technological advancements in the field of power electronics have caused major changes in the industry related to electrical machines. The range of possible applications for such devices has expanded tremendously because of their ease of use and their excellent efficiency. Those technological changes have also led to the increased use of new types of electrical machines. For instance, the very popular DC motor is now being challenged for servo applications by Permanent Magnet (PM) motors, Switched Reluctance (SR) motors, and even induction motors. The development of vector control theory has also allowed improvements in terms of control for existing motor technologies, such as induction machines.

The introduction of Digital Signal Processors (DSP) in motor control applications has allowed electrical machines to reach their full potential, in terms of speed range and dynamic behavior. Complex control algorithms can now be implemented and the motor drive can perform a wider range of operations with optimization of algorithms with regard to efficiency, robustness or dynamic response. For example, in permanent magnet machines the controller can optimize the machine output torque in order to minimize the required current, the required voltage, or the power losses.

The performance of electric motor drives now relies as much on software as on hardware configuration. Numerous algorithms have been developed that can now substitute estimations for measurements, reducing the drive cost and increasing its robustness. The most popular of these indirect parameter estimations are related to rotor position estimation because of the price and bulkiness of position sensors. These estimators mostly use measured machine currents and electrical parameters to extract position information. On the other hand, efforts have been made in order to reduce the number of current sensors by reconstructing the three phase currents from DC bus current rather than from measurement of phase currents.

A property that most advanced control algorithms have in common is their need for accurate knowledge of the machine analytical model. A control system designed for a plant that is different from the one it was intended for is likely to have poor performance. This is why the focus of this dissertation will be the estimation of plant parameters.

1.1 Synchronous Machines

Permanent magnet (PM) machines are electromechanical energy conversion devices that mainly use the interaction of the stator electromagnetic and rotor magnetic fields to produce torque. Most of these machines are non-salient, but depending on the mounting of the rotor magnets, they can also present magnetic saliency that can be used for torque production.

In their operation and even construction, the PM machines are very similar to the wound rotor (WR) AC synchronous machines. The difference resides in the fact that the rotor excitation is fixed and provided by permanent magnets instead of coming from an external circuit through slip rings and brushes. The stator construction can be the same for both types of machines.

In the past, AC synchronous machines were used mostly for generator applications. Their use as a motor was limited due to the difficulty of controlling the frequency of their supply voltages. The introduction of power electronics PWM inverters has allowed the motor drive to have complete control over the magnitude and frequency of machine phase to phase voltages.

Another factor that helped the development of PM synchronous machines is the expansion of industrial production of permanent magnets. The first magnet type to be produced on an industrial scale was the Alnico in the early twentieth century. As a result, S. Evershed [1][2] in 1920 made some important contributions to principles of PM torque production. At first, PM machines received severe criticism because of the large tolerance they have in terms of control parameters. Permanent magnet materials exhibit important nonlinearities and are sensitive to temperature and operating point. In 1946, W. Kober first mentioned using PM synchronous machines for alternator applications [3]; in 1951, R. M. Saunders and R. H. Weakley significantly contributed to their design considerations [4]. Most of these first approaches to PM machine design considered only Alnico type magnets. Rare earth magnets, which are significantly superior to the Alnico type, appeared in the 1970s. While at first very expensive, these materials have found an increasing interest in the last few years and are now commonly used in PM machines.

PM synchronous machines present several advantages when compared to the WR type machines. First, the PM machines present a much larger energy density and can therefore be of smaller size for a given power. They also have much lower rotor inertia, which is an important advantage for applications where a fast response is needed. Finally, the absence of brushes to supply the rotor circuit makes them much more mechanically robust. On the other hand, the price of PM materials is quite high and PM machines are not economically interesting above a certain power rating (about 20 kW). WR machines are consequently still used, typically for electrical energy production.

1.2 Types of PM Synchronous Machines

At this point it is necessary to mention the existence of two families of PM machines, depending on their stator construction. The first one, which will be referred to as PM synchronous machines and which will be the focus of this research, involves sinusoidally distributed windings on the stator side. It is, in that regard, essentially equivalent to a WR synchronous machine with no damper windings. The second family corresponds to a case where the stator windings are concentrated, so that the electromotive force generated by rotor movement is generally trapezoidal. These machines are usually called brushless DC machines, because their operation is very similar to that of standard DC machines. The focus of this research is not directly applicable to this type of machine.

PM synchronous machines can be further decomposed into two main categories, depending on their rotor construction. While the stator remains essentially the same, the machine rotor can present varying magnetic properties depending on how the permanent magnets are attached to the rotor. Here one needs to familiarize oneself with the general structure of a PM machine.

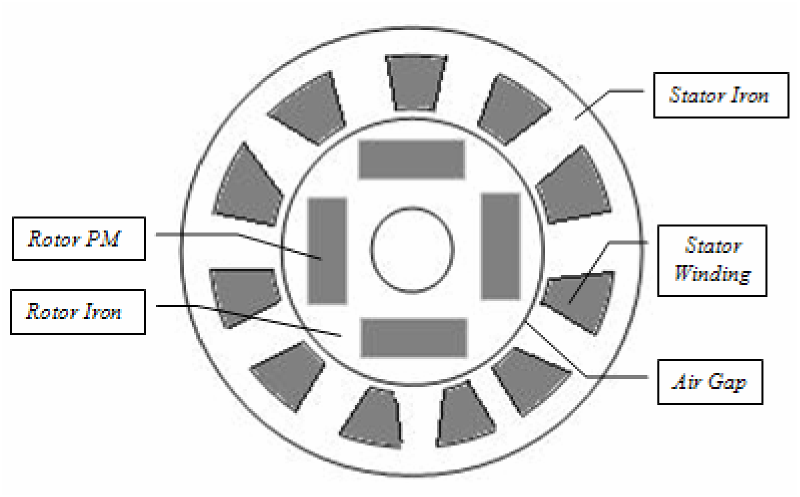

Fig. 1.1: PM machine construction example.

Figure 1.1 shows a PM machine as an entity composed of two main parts, a stator and a rotor. The stator is a part that is mechanically fixed and connected to external circuitry. It can be broken down into an iron part, which is “magnetically conductive”, and winding slots, which contain electrical windings that generate the stator magnetic flux. The rotor, on the other hand, is the part that is mechanically free to rotate and is attached to the stator only with bearings (mechanical, sometimes magnetic). The rotor is also made of two parts: iron that conducts the magnetic flux, and permanent magnets that produce the rotor magnetic flux. The interaction between stator and rotor fluxes is what generates the main part of the machine electromagnetic torque.

The distinction between the different types of PM machines is made essentially from the arrangement and location of the rotor permanent magnets. One important fact is that the permeability of the permanent magnets, which can be seen as the magnetic equivalent of the electrical conductivity, is almost the same as that of air. Depending on how the magnets are mounted on the rotor, there can be a large difference between a magnetic path which includes magnets and one that does not. Figure 1.2 shows three examples of PMSM construction.

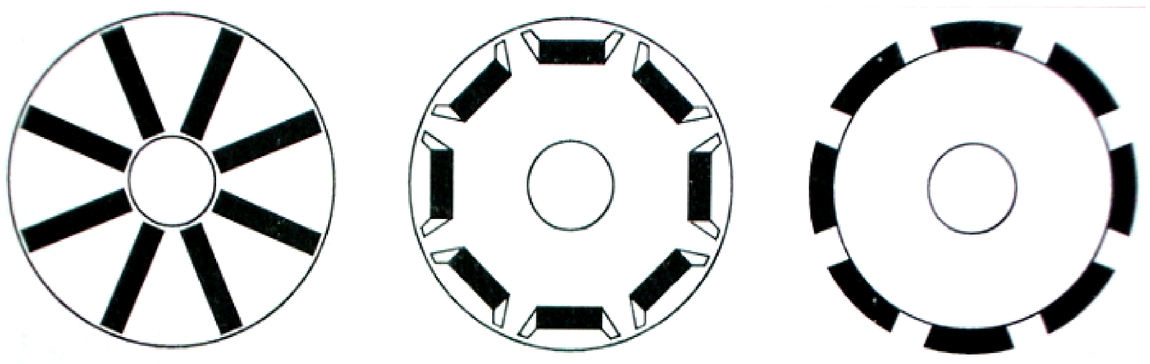

Fig. 1.2: Different rotor configurations for PMSM.

These three rotor configurations are the three most commonly found in the industry. The rotor on the left has internal permanent magnets, and these are magnetized tangentially, with alternating directions. The middle example also has internal PM, but is magnetized radially. The rotor on the right is different from the two others in that the magnets are mounted on the external surface of the rotor iron. The interesting point about this third one is that the magnets are completely transparent to the stator magnetic flux, because from a magnetic standpoint they are equivalent to air.

This third example in Figure 1.2 is also the most popular configuration because it is the easiest one to manufacture. This design is called a Surface Mounted PMSM. The magnets are glued to the iron, and a magnetically neutral wrap is also typically placed around them. It is interesting to note that this configuration does not show any magnetic saliency, and therefore cannot exploit any form of reluctance torque.

On the other hand, the left two configurations do present magnetic saliency, as a difference in magnetic paths between including the magnets and not doing so. This particularity allows them to offer both magnetic interaction and reluctance torque production capabilities to the user. Usually these machines are used for higher speed operation than surface mounted PMSM because they can still produce reluctance torque in field weakening operation. These machines are usually called Interior or Internal PMSM, or IPM machines. They are naturally more expensive to manufacture, and slightly more complex to control, because of the need to optimize the combination of two methods of torque production.

1.3 Research Objective

The main objective of this research is to address the issue of on-line parameter estimation for IPM machines. The analysis presented should also be compatible with a surface mounted configuration and should provide performance improvements for that motor drive as well. This research will focus on the machine parameters that are the most relevant to a control system design for a PMSM. On-line parameter estimation for a PMSM controller is particularly relevant because of the time varying nature of these parameters. The most common perturbation factor to consider is the change in temperature, which cannot be measured in most controllers because of a lack of temperature feedback. Another factor which is often omitted in controller design is magnetic saturation, which has a very noticeable effect in most IPM machines.

Several attempts have been made to solve this problem in the past. However, most only focused on a subset of the parameters, relying on the assumption that the rest of the parameters were sufficiently known. Even for such a case, only a few methods were suitable for on-line parameter estimation. Other algorithms relied on the use of offline measurements that are introduced in the control algorithm using look-up tables or interpolating functions. These methods present the important drawback of being unable to deal with conditions that cannot be easily accounted for, such as machine aging. The main problem that all these methods avoid or try to overcome is the poor mathematical conditioning of on-line parameter estimation for PM machines. One paper [5] presented a procedure for on-line parameter estimation of a WR synchronous machine, which has a more complex model than PM machines, but also relied on fixing a small subset of machine parameters and used a non linear method.

The research presented here aims at surmounting the numerical difficulties associated with the problem of IPM parameter estimation by using specific properties of parameter subsets to ease computation. Even though the focus of this research is on IPM machines, it will be shown that some surface mounted PMSM can also have significant saturation-related parameter dependency, and could potentially take advantage of the presented algorithm.

1.4 Dissertation Organization

This dissertation began with an introduction to the focus of this research. A brief history of permanent magnet machines was presented, followed by a presentation of the different types of such machines, and then the current research objectives were explained.

Chapter II will go further into details with a more thorough presentation of PM machines modeling and control. A literature review will follow to show the prior research efforts in the area of PM machines parameter estimation.

Chapter III will focus on the parameter set that is of interest for this research. It will analyze how they can be modified as the machine operates, and will then study the possible consequences of these changes on controller performance.

Chapter IV will introduce the reader with the solution to the problem of on-line parameter estimation for PM machines that is the subject of this research. The recursive least squares algorithm, which is the basis for this project, will be presented first, and then the proposed algorithm structure used in this research will be given.

Chapters V and VI will be dedicated to the simulation model that was developed for this research in order to design and first validate the proposed algorithm. Chapter V will focus on the simulation model itself, describing how it was designed, whereas Chapter VI will present the simulation results relevant to this research.

Chapters VII and VIII will have a similar structure, but they will be dedicated to the experimental setup. Chapter VII will present the different elements of the experimental setup and analyze features that are relevant to this research, while Chapter VIII will show experimental results obtained with this setup.

Finally, Chapter IX will conclude this dissertation and will present possible future research topics related to this control area of PM machine drives.

CHAPTER II

PM SYNCHRONOUS MACHINES

2.1 PMSM Drive Structure

A PM synchronous machine drive includes several elements in addition to the machine itself. The complete motor drive is a structure that includes the machine, its associated sensors, a power electronics converter, and the controller. The latter processes sensor feedbacks and controls the converter for the desired operation. Figure 2.1 shows the motor drive structure that is used in most PMSM applications, and will also be used in this research.

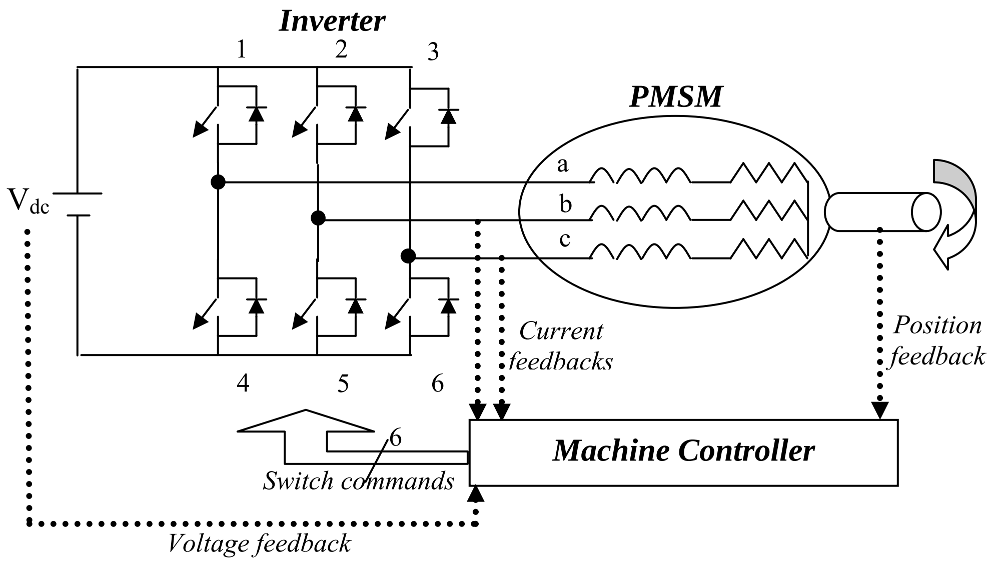

Fig. 2.1: PMSM motor drive.

In the drive structure shown, the position feedback is used to synchronize the stator flux with the rotor one. Position feedback is also used for speed estimation, and for speed or position control. The two current sensors allow the controller to reconstruct the three phase currents and to perform current control, which in turn allow torque control. The DC bus voltage feedback is used in the PWM controller to translate machine voltage commands into switch duty cycles. These feedbacks will also be used to achieve the goals of this research, and more emphasis will be placed on this aspect in later chapters.

2.2 PMSM Modeling

PM synchronous machines are three phase AC machines that involve the interaction of the stator flux, which is controlled by the motor drive, and both the rotor PM flux and the reluctance flux path. The rotor has no windings or electrical connections to the stator. In order to operate the machine properly, the rotor position has to be known, either from the feedback given by a position sensor, or from a position estimation algorithm.

The following assumptions will be made for this research:

- Saturation will be taken into account through parameter changes
- The machine induced Electro-Motive Force (EMF) is sinusoidal
- Eddy currents and hysteresis losses are negligible
- There are no field current dynamics.

The approach that is commonly used in order to model three phase machines is the one based on the Park transformation [6]. This method transforms a three phase balanced system into a two-dimensional one. The transformation changes a complex non linear model into a much simpler one, where machine variables are referenced to a rotating reference frame attached to the rotor magnetic axis. Figure 2.2 illustrates this for a two-pole machine.

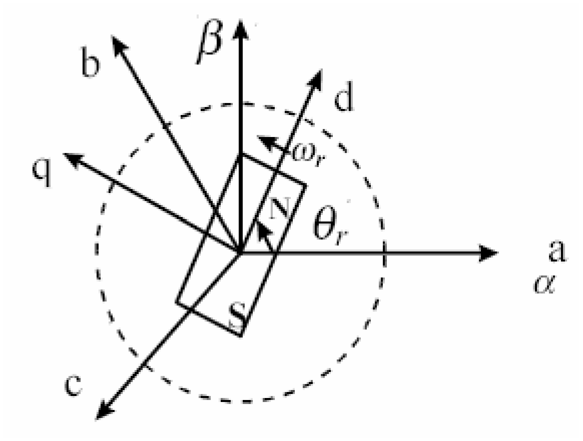

Fig. 2.2: PM machine reference frames.

In Figure 2.2, the magnetic axes of the three stator windings are labeled a, b and c. From this three dimensional coordinate based system two possible results for the Park transformation are commonly used. The first one is labeled α-β and is attached to the "a" phase axis. It is consequently called the fixed reference frame. The corresponding machine model is rather complex, but can be useful in applications where the rotor angle is unknown, such as position estimators. The other case, which will be used in the analysis of this research, is attached to the rotating magnetic axis of the rotor, and is usually called the *d-q* rotor flux reference frame. With this base, the machine model becomes quite simple, and this makes it easier to develop control algorithms. The machine modeling starts in the "abc" reference frame with the following set of equations:

$$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} = \begin{bmatrix} R & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & R \end{bmatrix} \cdot \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + \frac{d}{dt} \begin{bmatrix} \varphi_a \\ \varphi_b \\ \varphi_c \end{bmatrix} \tag{2.1}$$

where V_{abc} are the machine voltages referenced to the ground, i_{abc} are the machine phase currents, R is the machine phase resistance, and φ_{abc} are the magnetic fluxes associated with each phase.

The Park transformation is a matrix transformation which converts the three-phase abc system to the *d-q* reference frame. A third component called "0" is also present in order to have a bijective transformation. This "0" or homopolar component is equal to zero in balanced three-phase systems and will be omitted later in the chapter. The matrices for the magnitude invariant Park transformation and its inverse are:

$$\begin{bmatrix} f_q \\ f_d \\ f_0 \end{bmatrix} = \frac{2}{3} \cdot \begin{bmatrix} -\sin(\theta_r) & -\sin(\theta_r - 2\cdot\pi/3) & -\sin(\theta_r + 2\cdot\pi/3) \\ \cos(\theta_r) & \cos(\theta_r - 2\cdot\pi/3) & \cos(\theta_r + 2\cdot\pi/3) \\ 1/2 & 1/2 & 1/2 \end{bmatrix} \cdot \begin{bmatrix} f_a \\ f_b \\ f_c \end{bmatrix} \tag{2.2}$$

and

$$\begin{bmatrix} f_a \\ f_b \\ f_c \end{bmatrix} = \begin{bmatrix} -\sin(\theta_r) & \cos(\theta_r) & 1 \\ -\sin(\theta_r - 2\cdot\pi/3) & \cos(\theta_r - 2\cdot\pi/3) & 1 \\ -\sin(\theta_r + 2\cdot\pi/3) & \cos(\theta_r + 2\cdot\pi/3) & 1 \end{bmatrix} \cdot \begin{bmatrix} f_q \\ f_d \\ f_0 \end{bmatrix} \tag{2.3}$$

When the Park transformation is applied to equation (2.1) with θ_r being the rotor position and taking into account the previous assumptions, we obtain

$$\begin{cases} V_q = R\cdot i_q + p\cdot\varphi_q + \omega_e\cdot\varphi_d \\ V_d = R\cdot i_d + p\cdot\varphi_d + \omega_e\cdot\varphi_q \end{cases} \tag{2.4}$$

where

$$\begin{cases} \varphi_q = L_q\cdot i_q \\ \varphi_d = L_d\cdot i_d + \varphi_{mag} \end{cases}$$

In the above equations, the *d*-axis variables are the ones that are aligned with the permanent magnet position, whereas the *q*-axis corresponds to an axis 90 degrees ahead.

In these equations, p is the Laplace differential operatot. It can be noted that the permanent magnet flux φ_{mag} only appears on the d-axis. In terms of notations, V_{qd} are the q- and d-axes stator voltages, which are the results of the Park transformation applied to V_{abc}, and the same conclusion applies to i_{qd} and φ_{qd}. L_{qd} are the inductances associated with the q- and d-axes, ω_e is the electrical speed of the motor, which is equal to the number of machine pole pairs times the mechanical speed ω_r, and p is the Laplace differential operator. The machine torque is obtained from the derivative of the magnetic energy with respect to the rotor position and is given as

$$T_e = \frac{3 \cdot P}{2} \cdot \left[\varphi_{mag} \cdot i_q + \left(L_d - L_q\right) \cdot i_d \cdot i_q\right] \tag{2.5}$$

In equation (2.5), P is the number of rotor pole pairs in the machine. There are two torque producing terms present in equation (2.5). The first one involves the interaction of the magnet flux and the q-axis current and is the main machine torque. The second is based on the difference between the d- and q-axes inductances, and is therefore called the reluctance torque. This latter component is almost non-existent in surface mount PM synchronous machines but is a particularly interesting feature of IPM machines, giving them extended speed range capabilities. For convenience, a state space representation of the machine model can be obtained as

$$\begin{cases} p \cdot i_q = \left(V_q - R \cdot i_q - \omega_e \cdot L_d \cdot i_d - K_T \cdot \omega_r\right)/L_q \\ p \cdot i_d = \left(V_d - R \cdot i_d + \omega_e \cdot L_q \cdot i_q\right)/L_d \\ p \cdot \omega_r = \left(T_e - T_{load}\right)/J \end{cases} \tag{2.6}$$

In the previous set of equations, K_T is the torque or back-emf constant of the machine and J is the moment of inertia of the rotor and its load. These equations can be used to build a model of the machine when coupled with equations (2.2) and (2.3).

It is necessary at this point to emphasize the fact that the model developed from equations (2.6) gives a somewhat simplified model for a PMSM. The exclusion of iron losses (eddy current and hysteresis losses) has a small impact on the accuracy of simulated results. The reason why these are not included here is that the controller complexity required for them to be taken into account generally is not justified by the small error introduced by neglecting them. The model described here is suitable for control oriented problems. However, in a machine design problem the emphasis would certainly be different and iron losses should be included. A simple way to visualize the impact of including iron losses in the model is to draw the machine equivalent circuit shown in Figure 2.3.

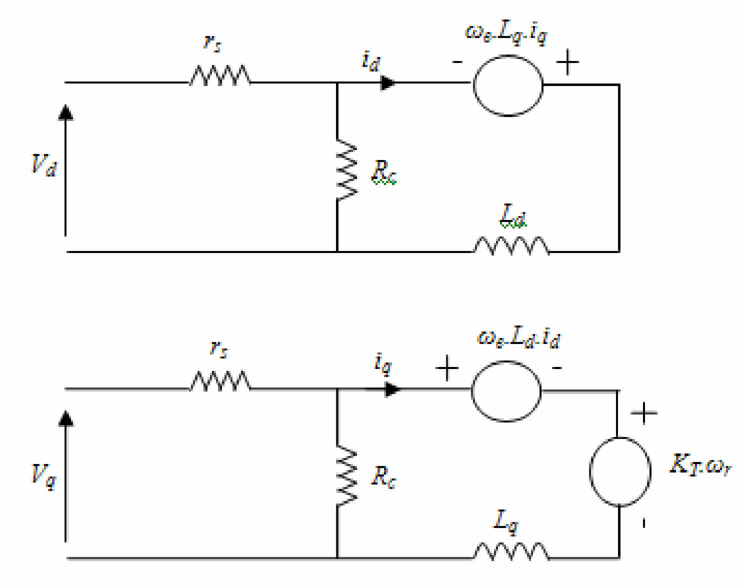

Fig. 2.3: PMSM equivalent circuits including iron losses.

Usually, the core losses resistance R_c is not a constant, but a function of the operating frequency given as

$$\frac{1}{R_c} = \frac{1}{R_{c0}} + \frac{1}{R_{c1} \cdot \omega_r} \tag{2.7}$$

In this research, however, this resistance will be considered infinite, and consequently the core losses will be neglected.

2.3 PMSM Control

The machine model that has been presented can be used for controller design. Since the focus of this research is on the electrical parameters of PM synchronous machines, our emphasis will be mostly on the current and torque controllers. If an outer loop were to be implemented for speed or position control, the dynamics involved would mostly rely on the mechanical parameters.

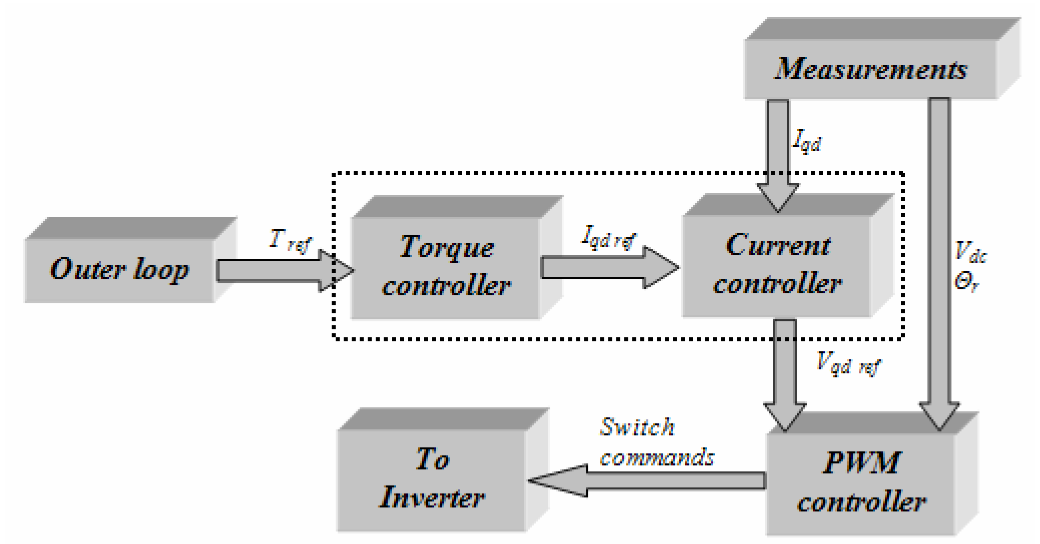

Fig. 2.4: PMSM controller data flow.

Figure 2.4 shows the different blocks that are commonly found in PMSM controllers. In this section, the operation of the torque and current controllers will be discussed, and the other blocks will be described in detail later in a hardware related part. The PWM controller does not require motor parameters because the operation it performs scales and shifts the voltage references in order to convert them into duty cycles for the inverter switches.

In the following sections, the equations relating the machine torque to its currents will be discussed along with the description of how to control current by acting on the voltages. This will then allow us to visualize more effectively the importance of machine parameters in such controllers.

2.3.1 PMSM Torque Controller

This section will present the various algorithms that can be used with a PMSM in order to relate its machine currents with its electromagnetic torque. The motivation behind the choice of one algorithm over another is usually a function of complexity and the operating point of the machine at a given time. Each of the presented algorithms relies on equation (2.5) which is repeated here for convenience:

$$T_e = \frac{3 \cdot P}{2} \cdot \left[\varphi_{mag} \cdot i_q + \left(L_d - L_q\right) \cdot i_d \cdot i_q\right] . \qquad (2.5)$$

2.3.1.1 Zero *d*-axis Current Control

For a surface mounted PMSM ($L_d = L_q$) or if the *d*-axis current is set to zero, equation (2.5) becomes:

$$T_e = \frac{3 \cdot P}{2} \cdot \varphi_{mag} \cdot i_q \tag{2.8}$$

Equation (2.8) shows that the machine electromagnetic torque is completely independent of the *d*-axis current and is proportional to the *q*-axis current. In such a case, the *d*-axis current is usually controlled to remain at zero, so that the current vector magnitude is minimized. This operation is typically called "i_d = 0 control" [6] and the reference currents are obtained from

$$i_q = \frac{2 \cdot T_e}{3 \cdot P \cdot \varphi_{mag}} \tag{2.9}$$

The algorithm is also compatible with IPM machines and is attractive due to its simplicity, but it completely nullifies the possible contribution of the reluctance torque. It is equivalent to using an IPM as a SM PMSM, which is not desirable due to the considerable price difference between the two types of machines. Equation (2.9) shows that this simple algorithm also has the attractive feature of using only one machine parameter φ_{mag}, since *P* can be regarded as a known constant.

2.3.1.2 Maximum Torque per Ampere Control

Another possible algorithm in the case of IPM machines is the one that minimizes the input current to the machine for a given output torque. This type of operation is usually referred to as "maximum torque per ampere (MTPA)". The current to minimize is

$$I_s = \sqrt{i_d^2 + i_q^2} \tag{2.10}$$

The algorithm is based on finding the point where $dT_e/dI_s = 0$. Equation (2.10) substituted into equation (2.5) gives [8]:

$$T_e = \frac{3 \cdot P}{2} \cdot \left[\varphi_{mag} + \left(L_d - L_q\right) \cdot i_d\right] \cdot \sqrt{I_s^2 - i_d^2} \, .$$

And $dT_e/dI_s = 0$ gives

$$\beta = \arcsin\left(\frac{-\varphi_{mag} + \sqrt{\varphi_{mag}^2 + 8 \cdot \left(L_q - L_d\right)^2 \cdot I_s^2}}{4 \cdot \left(L_q - L_d\right) \cdot I_s}\right) \tag{2.11}$$

where β is the angle between I_s and i_q. From this relation the controller can obtain the *d*- and *q*-axes currents that give the maximum torque for a given current magnitude. The equations associated with this algorithm are nonlinear and require much more computational power than in the case of equation (2.9). However, this operation is very interesting because it maximizes the motor drive's torque capability when the machine operates below its rated speed. It also minimizes copper losses, which are proportional to the square of the stator currents. In this case, the *d*-axis current is likely to be different from zero, which implies that the controller takes advantage of both the main machine torque and the reluctance torque capabilities of the machine to minimize the current used.

With the MTPA algorithm, however, a good knowledge of three machine parameters is required: both the *d*- and *q*-axes inductances in addition to the permanent magnet flux. Unlike in the "$i_d = 0$ control," the relation between the desired torque and the machine currents is not linear and an error in machine parameters can have significant consequences.

2.3.1.3 Maximum Torque per Voltage Control

The maximum torque per voltage (MTPV) control for the PM machine, involves the optimization of the current vector to minimize the required input voltage to the machine. This is equivalent to minimizing the machine flux linkage between stator and rotor, and that is why this algorithm is also called maximum torque per flux (MTPF) control. The motor drive operates with both current and voltage constraints and it may happen in an application that the voltage constraint is the harder one to satisfy. This occurs typically at higher speeds, when the motor back-emf becomes so dominating that it leaves only small freedom in terms of voltage available. At low speeds, the MTPA algorithm is usually preferred because it yields a higher efficiency. In the case of MTPV, the variable to minimize in the torque equation (2.5) is the flux linkage, which is given in equation (2.12)

$$\varphi_0 = \sqrt{\left(L_d \cdot i_d + \varphi_{mag}\right)^2 + \left(L_q \cdot i_q\right)^2} \ . \tag{2.12}$$

When this term is combined with equation (2.5) and the resulting equation is differentiated with respect to the flux linkage, the following result is obtained [8]

$$\begin{cases} i_d = -\dfrac{\varphi_{mag} + \Delta\varphi}{L_d} \\ i_q = \dfrac{\sqrt{\varphi_0^2 - \Delta\varphi^2}}{L_q} \end{cases} \tag{2.13}$$

where

$$\Delta\varphi = \frac{-L_q \cdot \varphi_{mag} + \sqrt{\left(L_q \cdot \varphi_{mag}\right)^2 + 8 \cdot \left(L_q - L_d\right)^2 \varphi_0^2}}{4 \cdot \left(L_q - L_d\right)}$$

This last term can be regarded as the amount of flux weakening necessary for the algorithm to achieve optimum operation. The MTPV algorithm requires the same parameters as the MTPA one. The MTPV is also an interesting control method since it minimizes the iron losses through the minimization of stator flux. These losses can be significant at high speeds and are directly related to the flux linkage of the machine.

2.3.1.4 Loss Minimization Control

It has been mentioned in the two previous sections that the maximum torque per ampere minimizes the copper losses in the machine and the maximum torque per flux minimizes the iron losses. If the controller objective is to maximize the efficiency of the machine, then the resulting optimal control will be a combination of these algorithms. This combined control known as loss minimization control. In this case the current vector will be both a function of the machine speed and its torque. For a given torque, at low speeds the current vector loci will be close to the maximum torque per Ampere curve, and will gradually shift towards the maximum torque per flux trajectory as the speed increases. It is difficult to obtain an analytical function giving the current vector as a function of the torque and speed, and one way of doing it involves using the circuit of Figure 2.4 with a model for R_c. Common implementations of this algorithm involve a two-dimensional look-up table of results that are calculated offline. The problem with this method is that it fixes the algorithm for the set of machine parameters it was calculated for, which can vary, as will be shown.

2.3.1.5 Flux Weakening Control

The flux linkage due to the permanent magnet cannot be controlled by the user, but equation (2.12) shows that its contribution can be minimized by injecting a negative *d*-axis current. This feature becomes interesting as machine speed increases because it can allow the drive to reach speeds that would otherwise cause the drive to exceed its voltage capability limit. The flux or field weakening method of control is one that will follow the voltage limit trajectory. This method is attractive because it is simple to implement and quite robust with regards to parameter changes since it relies on the drive electrical limits. With $V_{\max}^{'} = V_{\max} - r_s \cdot I_{\max}$ we get

$$i_d = -\frac{\varphi_{mag}}{L_d} \pm \frac{1}{L_d} \cdot \sqrt{\left(\frac{V_{\max}^{'}}{\omega_r}\right)^2 - \left(L_q \cdot i_q\right)^2} \tag{2.14}$$

Even though this method requires the same three machine parameters, its goal is not to achieve optimum performance, but rather to achieve operation at a desired operating point. In this regard, its operation at a point that would not be the optimum but would give the desired torque and speed could be deemed acceptable.

2.3.1.6 Summary of Current Vector Control Schemes

The algorithms presented up to this point are summed up in Figure 2.5 for a 0.7 kW IPM machine [9].

For a given torque, there is a multitude of possibilities of current vectors. The one that has the smallest magnitude corresponds to the maximum torque per ampere algorithm (point A). The other algorithms require larger current vectors, but can minimize losses (point D), or the voltage required from the inverter (point B). The flux weakening trajectory lies on a constant voltage ellipse, and the zero *d*-axis current control lies on the *q*-axis (point E).

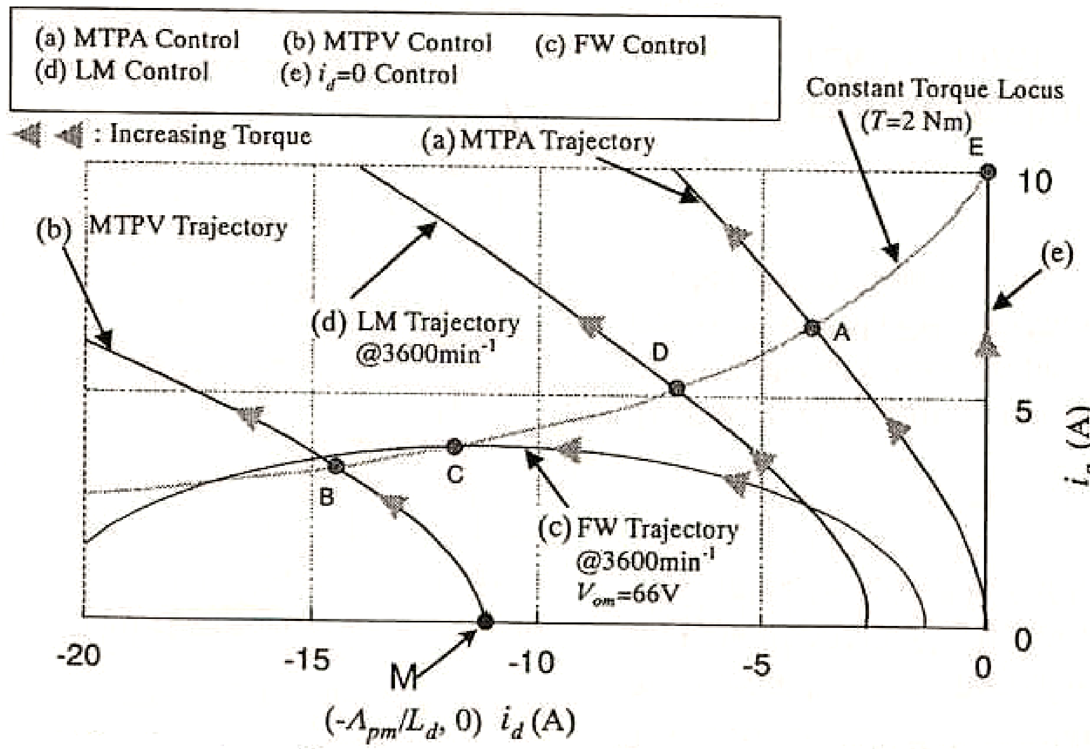

Fig. 2.5: Summary of current vector control schemes.

2.3.2 PMSM Current Controller

In the block diagram of Figure 2.4, the purpose of the current controller block is to find the *d-q* axis voltages required from the inverter in order to establish the desired currents in the machine. Two methods can be used for that purpose. The first one is called hysteresis control [7]; it forces the machine phase currents to remain within a predefined range of their reference by switching inverter configurations whenever the error gets too large. This technique has the important drawback of forcing a variable switching frequency, which can get very large; it also requires external circuitry and makes it harder for the controller to track the voltage commands sent to the inverter.

The second method is the one that will be used in this research; it uses Pulse Width Modulation (PWM) at a fixed frequency to control the voltage vectors sent to the machine. This technique requires the calculation of the voltage vector to be sent to the inverter, and is therefore made readily accessible by the controller [6][7]. Let us start from the machine electrical equations

$$\begin{cases} V_q = r_s \cdot i_q + p \cdot L_q \cdot i_q + \omega_e \cdot L_d \cdot i_d + \omega_r \cdot K_T \\ V_d = r_s \cdot i_d + p \cdot L_d \cdot i_d - \omega_e \cdot L_q \cdot i_q \end{cases} \tag{2.15}$$

A common method used to make the controller simpler is to use a feed forward compensator, which decouples the *d*- and *q*-axes, and cancels the back-emf term. The previous equations then become:

$$\begin{cases} V_q^{'} = V_q - \omega_e \cdot L_d \cdot i_d - \omega_r \cdot K_T = r_s \cdot i_q + p \cdot L_q \cdot i_q \\ V_d^{'} = V_d + \omega_e \cdot L_q \cdot i_q = r_s \cdot i_d + p \cdot L_d \cdot i_d \end{cases} \tag{2.16}$$

This in turn gives

$$\begin{cases} \dfrac{i_q}{V_q^{'}} = \dfrac{1}{r_s + p \cdot L_q} \\ \dfrac{i_d}{V_d^{'}} = \dfrac{1}{r_s + p \cdot L_d} \end{cases}$$

The above two transfer functions are simple and linear, and a PI controller is usually sufficient to achieve high performance current control. However, in order to reach this stage, a good knowledge of the machine parameters K_T, L_q and L_d is required. If the values used in the controller differ significantly from the actual machine parameters, the system will present cross coupling and the perturbation of the back-emf, which would normally require a more complex type of control.

2.4 Existing Achievements for Parameter Estimation of PMSM

Several methods exist in the literature that try to compensate for parameter variation in PM machines. These methods can be classified into offline ones and on-line ones. Offline algorithms use measurements taken at a certain point in time to build look-up tables or interpolating functions to get parameter estimates as a function of external factors. These techniques are easy to implement, but are difficult to implement with parameters that can be functions of several variables. They will also fail to account for any change in the machine parameters due to factors unknown at the time of measurement, such as ageing factors. On the other hand, on-line methods use measurements provided by the controller while the machine is in operation in order to get their parameter estimates. The great advantage of these methods is that they can track parameter variations almost independently of their sources. Whether the stator resistance changes due to temperature or mechanical damage will not matter from the point of view of an on-line method. Nevertheless, these methods require complex numerical algorithms, which may sometimes become unstable and lead to drive malfunction. Particular care must be given to their design.

2.4.1 Stator Resistance and Torque Constant Estimation

These two parameters need to be treated somewhat separately from the inductances, because most controllers for surface mounted PMSM will not even consider inductance variations. As a consequence, the problem of estimating r_s and K_T is common to surface mount and IPM machines. Offline identification of these two parameters as a separate subset is difficult to implement even if the assumption is made that they are only temperature dependent. The reason is that temperature feedback is generally unavailable in common PMSM drives. As a consequence, the only viable approach to this problem is on-line parameter estimation. Papers on this subject usually take the approach of considering only the steady state model of the machine [10]. By setting the differential terms in equations (2.15) to zero, one obtains

$$\begin{cases} V_q = R \cdot i_q + \omega_e \cdot L_d \cdot i_d + \omega_r \cdot K_T \\ V_d = R \cdot i_d - \omega_e \cdot L_q \cdot i_q \end{cases} \tag{2.17}$$

The above equations will be referred to as steady state equations. Identification of the resistance and torque constant is usually done by considering the inductances to be constant; then the problem becomes well conditioned for either an observer structure or an algorithm like the least squares method. This approach is valid because the temperature change dynamics are much slower than the electrical ones. This technique is quite simple and is commonly used for surface mounted PMSM that do not have much saturation sensitivity. On the other hand, if saturation occurs, the errors in inductance will have a large impact on the estimation error. This approach was used in [10] with a position sensorless algorithm for a surface mount PMSM.

2.4.2 Inductance Estimation

The problem of estimating the inductance for the controller is quite different from the previous one. In this case, it is possible to measure these two functions offline since each inductance can be considered as a function of the current in its axis, if cross-saturation is neglected. The inductance in the controller is then obtained from the measured currents in the machine as it runs.

2.4.2.1 Offline Inductance Estimation

Several methods of parameter estimation in PMSM have focused on offline inductance measurement. The methods vary since the *q*- and *d*- axes inductances are not readily available for measurement, but are somewhat fictitious parameters obtained through the Park transformation.

A first method consists of getting inductance estimates from a finite element analysis. This technique requires the user to know precisely the geometric and material properties of the machine, and then to run a series of computationally intensive simulations. However, the data necessary to have accurate simulation results is not usually available from the manufacturer, and these simulation results often need to be double checked with experimental data for possible numerical problems. Another problem is that this method doesn't solve the issue of finding estimates for the resistance and torque constant. An example of this technique has been presented in [11] and [12]. In both cases the results were verified experimentally: in [12], the inductances were obtained from locke*d*-rotor measurements. Other methods for offline measurements are possible. Some use the decay time of a phase current following a voltage pulse for different locked rotor positions and initial current levels [13]. Offline measurements are usually more convenient than on-line ones because they can force situations that are inaccessible to a non-intrusive on-line algorithm. In the case of a locked rotor situation, the motor *d*-*q* equations become much simpler:

$$\begin{cases} V_q = R \cdot i_q + p \cdot L_q \cdot i_q \\ V_d = R \cdot i_d + p \cdot L_d \cdot i_d \end{cases}$$

Another technique to extract the machine inductances consists in running the machine in a succession of steady states and calculating inductance estimates from voltage, current and speed measurements. To get to this point, some methods directly work with the Park transformation of the measurement, like in [14], while others work with the Fourier transform of motor phase variables [15]. When it comes to using measurements obtained offline, a common method was given in [16] and [17], that used piecewise linear functions to approximate the inductances as a function of their respective currents. This approach of course neglected the impact of cross saturation.

2.4.2.2 On-line Inductance Estimation

On-line parameter estimation methods have also been developed to identify the machine inductances. One approach found in [18] and [19] is to consider the stator resistance and torque constant to be fixed, and to run an observer based on the machine model. Additional information from offline inductance measurements was also injected in order to improve dynamic behavior and numerical stability. Figure 2.6 shows the implementation structure for this method. Three main drawbacks can be found for this technique. The first one is that errors in resistance and torque constant will have a negative impact on the estimation. Another one is that compensation for cross saturation can hardly be made. Finally, the method requires offline measurements, which take away some of the flexibility that could be expected from an on-line algorithm.

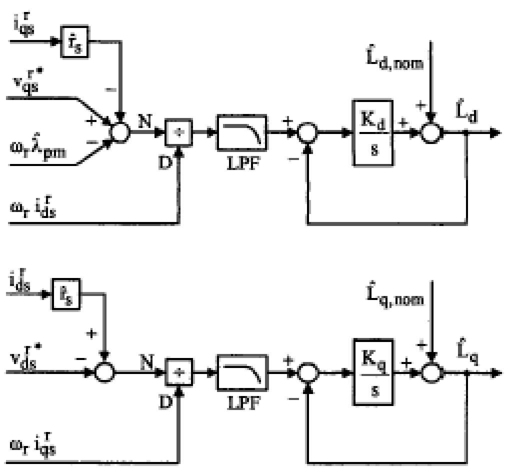

Fig. 2.6: Observer based on-line inductance estimation.

A second approach was found in [5] where the parameters of a wound rotor synchronous generator are estimated using a non-linear version of least squares estimation. The model used in this paper is different from the one used in this research (different machine structure), and the method considered a small subset of the machine parameters to be constant in order to obtain decent numerical stability. This paper is mentioned here because it can be considered to be the closest one available in the literature to provide a solution to the problem of on-line parameter estimation of electrical machines.

2.5 Shortcomings in Existing Research

The previous section has shown the various methods that have been tried to overcome the problem of parameter variation in PMSM. Off-line methods have been implemented based on both simulation results (FEA) or experimental measurements, but prove to be limited when it comes to having estimates for all four parameters. These methods also make it difficult for the controller to compensate for the effects that were not foreseen at the time results were taken. These methods also lack compensation of temperature effects, since temperature sensors are usually not present in motor drives.

On the other hand there have been attempts to develop on-line parameter estimation, but the examples found limited themselves to a subset of the parameters and/or used a combination of off-line and on-line results to operate properly.

The advantages of having an on-line parameter estimation algorithm for PMSM are numerous. These include the fact that parameter estimates track machine parameters regardless of the causes of parameter changes. From the point of view of the algorithm, it is equivalent to estimate parameter variations due to cross saturation or temperature changes, as opposed to an off-line method. If the controller is constantly supplied with accurate parameter estimates, it can operate in an optimal way, depending on the control algorithm it uses. This allows the motor drive to reach its full potential, in terms of efficiency, speed range or dynamic response.

CHAPTER III

PARAMETER VARIATION PROBLEM ANALYSIS

Based on the literature review done in the previous chapter, the need for the following have been identified:

- An analysis of how and why the machine parameters change, the factors that affect them and their properties.
- A complete study of the effects of electrical parameter variation on the performance of the torque and current controllers of PMSM.

These elements will serve as a basis for this research project and will show how on-line parameter estimation can improve controller performance.

3.1 Machine Parameter Sensitivities

It has been shown in the previous chapter how machine parameters play an important role in determining the location of the desired current vector for a given torque level, machine speed and method of optimization. In this section we will focus on the different factors which can affect these parameters. The set of machine parameters that will be the subject of this research is defined in equation (3.1).

$$\theta = \begin{bmatrix} L_q \\ L_d \\ r_s \\ K_T \end{bmatrix} \tag{3.1}$$

These four parameters are the ones found in the equivalent circuit of Figure 2.3, once core losses have been neglected.

- L_q is the inductance associated with a magnetic path that links stator and rotor going between magnet poles
- L_d is the inductance of a magnetic path linking stator and rotor that goes through the permanent magnets
- r_s is the electrical resistance associated with one stator phase
- K_T is the torque constant of the machine, and is directly proportional to φ_{mag}

The torque constant K_T is given by

$$K_T = \frac{P}{2} \cdot \varphi_{mag} \tag{3.2}$$

The objective of this research is to estimate these parameters and to track their possible variations on-line during machine operation. The following sections focus on the factors that affect them, and how they affect them.

3.1.1 Parameter Sensitivities to Temperature

The main external factor that can cause the machine model to vary is the machine temperature since it affects both electrical and magnetic material properties. Most motor drives do not include temperature sensors and it is usually difficult to compensate for temperature variations. These variations can be caused by either the machine's external environment or the machine itself. Both copper and iron losses contribute to a rise in machine's internal temperature, and mechanical losses due to friction in the bearings may add to this effect. It is difficult to quantify the way in which these temperature changes affect the different machine parameters, because these are essentially related to the types of material used in the motor construction. It is necessary to keep in mind that in PM machines, temperature changes are considered to have extremely slow dynamics when compared with the electrical or mechanical dynamics.

3.1.1.1 Stator Resistance Sensitivity to Temperature

In the case of the stator resistance r_s, the problem is not too complex because most electrical machines have copper windings. The general equation that relates resistance changes in conductors as a function of temperature is

$$R_{(T)} = R_0 \cdot \left(1 + \alpha \cdot \left(T - T_0\right)\right) \tag{3.3}$$

where R_0 is the resistance of the conductor measured at the reference temperature T_0, and α is the temperature coefficient of the conductor material. This relation is of course an approximation, but is suitable for most applications. Most of the time α is given for a temperature T_0 of 20˚C or sometimes 0˚C. For copper conductors, $\alpha = 0.004041$˚C^{-1} and for aluminum conductors $\alpha = 0.004308$˚C^{-1}, both measured at 20˚C. This means that for a copper conductor, a change of 20˚C from the initial point leads to a variation of 8 % in its resistance value. PM machines usually have an operating temperature range of about 100˚C, and the associated resistance range (about 40%) should not be neglected in controller design equations.

3.1.1.2 Torque Constant Sensitivity to Temperature

This machine parameter is directly associated with the intensity of the magnetic flux induced by the machine permanent magnets. This flux is temperature dependent, but the way it is affected is a function of the permanent magnet material, its shape, and the magnetic circuit attached to it. It consequently is very difficult to derive a simple model for the effects that temperature changes will have on the magnetic flux.

A rough estimate for a temperature coefficient can however be given by permanent magnet manufacturers and was also found in [20]. For example, the N3571 Neodimium Iron Boron (NdFeB) will have a coefficient of -0.11%/˚C for its residual magnetism B_r. As a consequence, an equation similar to equation (3.3) could be derived for this parameter. A decrease in B_r is also likely to affect the magnet's intrinsic coercive force, which is its resistance to demagnetization. Both these effects are likely to adversely affect the magnetic flux induced by the material in a given circuit.

In addition to the thermal coefficient, most permanent magnet materials will have an approximate Curie temperature [20], which is also function of magnet material, shape and circuit. This parameter is the temperature at which the permanent magnet becomes permanently demagnetized, and should consequently be avoided at all costs in permanent magnet machines. One of the few disadvantages of NdFeB when compared to other types of permanent magnet material such as Samarium Cobalt (SmCo) is its much lower Curie temperature and its higher sensitivity to temperature changes. For example, the temperature coefficient for the B_r of a SmCo magnet is around -0.03%/˚C. However, SmCo magnets are generally more expensive than NdFeB and have a lower $(B.H)_{max}$ energy product. Both these magnet types are used in high performance PM machines.

3.1.1.3 Inductance Sensitivity to Temperature

Temperature changes also affect the *q*- and *d*- axes inductances of the machine. Most of the magnetic paths associated with these inductances take place in a ferromagnetic or ferrimagnetic material. Such types of materials also present certain sensitivity to temperature changes. For example, an increase in temperature in a typical ferrite of 80˚C can result in a permeability drop of about 25%. This, in turn, results in a decrease of about 25% in its inductance.

3.1.2 Parameter Sensitivities to Magnetic Saturation

This section only concerns the changes in the machine *q*- and *d*- axes inductances due to iron saturation. Magnetic saturation is a phenomenon that occurs in the magnetic iron parts of PM machines. The electrical resistance is not affected by this effect, and the torque constant can be affected, but in a negligible way. On the other hand, machine inductances will directly be affected. Figure 3.1 provides a better understanding of the problem.

Figure 3.1 shows a typical flux response in an inductor with an iron core when subjected to an increasing current. One can see that past a certain point (P1), the curve ceases to be linear and magnetic flux tends to increase in a slower way. This is what is called magnetic saturation. The inductance is the slope of the curve at a given point.

The inductance will remain constant at its maximum value for the linear portion of the curve in Figure 3.1; then, its value drops as saturation comes into play. Electrical machines are much more complicated to model, but this example can serve as a basis for understanding the phenomenon. Another thing to notice here is that inductance changes in response to current changes in the machine have much faster time scales than temperature related parameter variations.

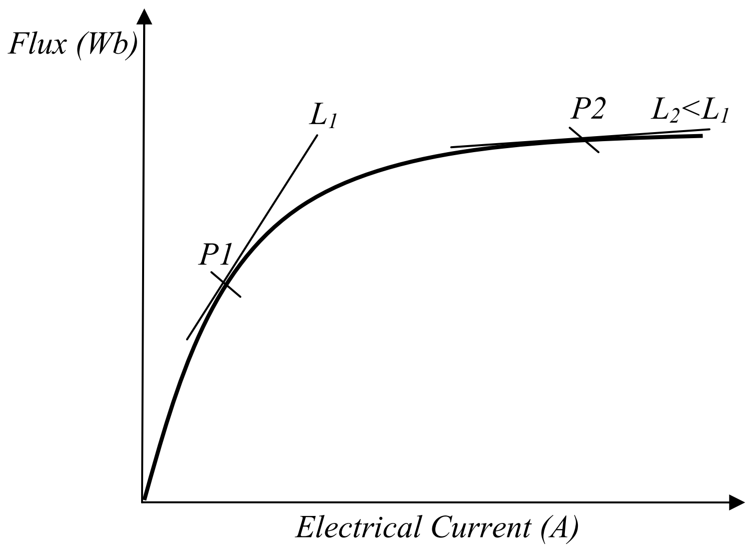

Fig. 3.1: Saturation in iron material.

When it comes to PM synchronous machines, both the *q*- and *d*- axis inductances are affected by saturation. However, it is necessary to consider important differences between the behaviors of these two parameters:

- The *d*-axis of the machine includes the permanent magnet, and the corresponding iron path is subjected to the magnetic flux. This important magnetic flux can be the source of a certain level of saturation even when the machine is not excited (as in point P2 of Figure 3.1). The electrical currents associated with this axis are usually oriented to oppose the magnetic flux, either to exploit reluctance torque, or to achieve flux weakening. Depending on the intensity of these currents, a noticeable change may occur in the machine inductance. However, the artificially increased air gap thickness due to the magnets also reduces the sensitivity of L_d towards i_d.

- The q-axis inductance does not include the permanent magnets, and is not excited when the machine is at rest (except for remanent flux). For small q-axis currents, one can consider that the q-axis magnetic circuit operates in the linear region on the iron material. However, for large currents (i.e. large torques), the magnetic path may become saturated and the q-axis inductance will consequently drop.

Surface mounted machines differ from IPM machines because when the machine is at rest, their q- and d- axes inductances are almost the same. A consequence of their construction is also that they generally have larger air gaps than IPM, in order to accommodate the permanent magnets. However, when a surface mounted machine operates at heavy load, saturation effects can still appear in the q-axis and these effects will be discussed later. An assumption is usually made that the q- and d- axes inductances in PM machines are decoupled, which implies that i_q will not affect L_d and vice versa.

Figure 3.2 shows the inductances of an IPM machine [9] with low saliency ratio, which is the kind of machine that was used in the experiments for this research project.

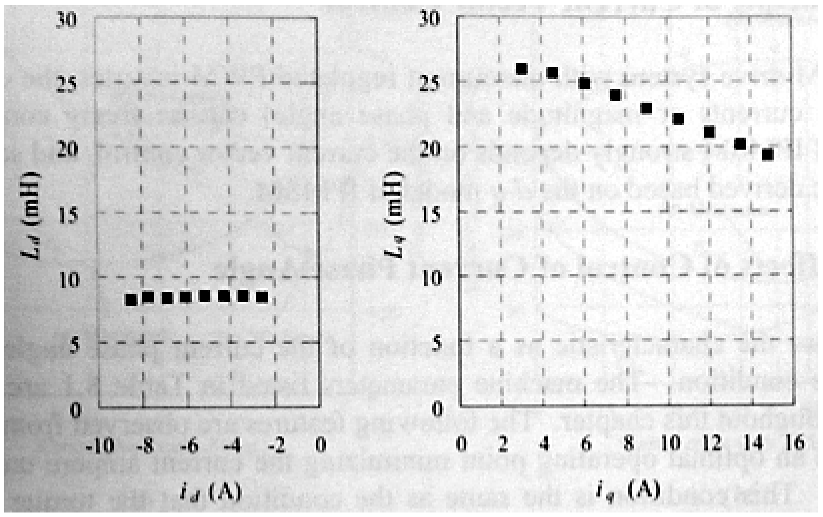

Fig. 3.2: Inductance variation in IPM due to saturation [9].

This IPM machine has a small sensitivity towards saturation, and its saliency ratio L_q/L_d is not significantly affected by it. It is also possible to see that the *d*-axis inductance does not seem to be affected much by the *d*-axis current. Similar studies have also been conducted for surface mounted PMSM [21], as shown in Figure 3.3.

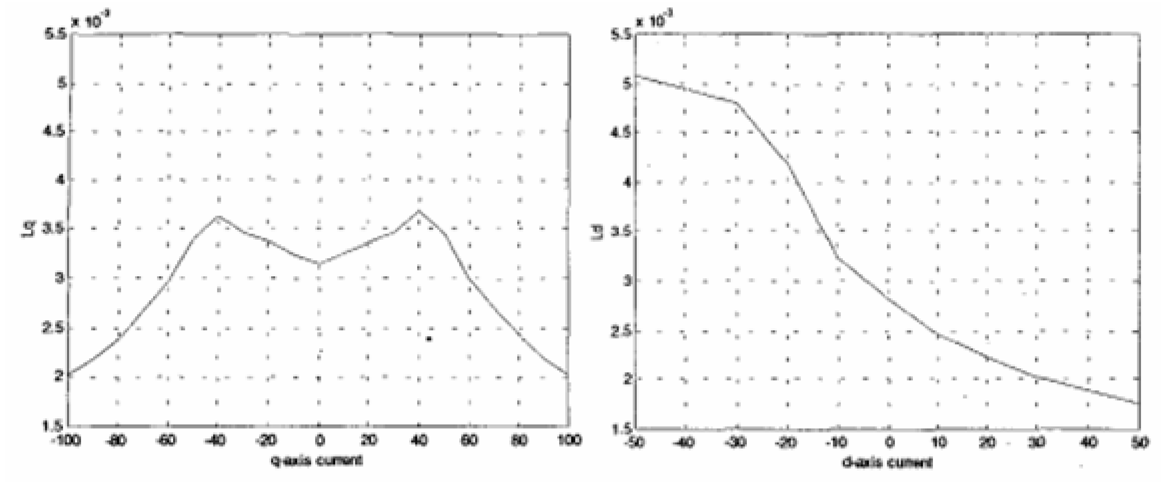

Fig. 3.3: Saturation effects in surface mount PMSM.

In Figure 3.3, the currents are measured in Amperes and the inductances in Henries. One can notice large changes in inductances, with L_q decreasing with increasing *q*-axis currents, and L_d increasing with large demagnetizing currents (negative *d*-axis). Saturation effects in surface mounted machines are often neglected in controller design but the results of this research show that a high performance controller could probably be improved from considering these changes.

3.2 Study on the Effects of Parameter Variation on Controller Performance

The previous chapter has shown that the issue of on-line parameter estimation for a permanent magnet synchronous machine has not been adequately addressed in previous research. The solutions that have been found to compensate for parameter variation usually fail to correct what was not accounted for at the time of controller design. Temperature effects, cross-saturation and ageing effects, for example, will in most cases have an impact on controller performance. This section focuses on the impact of parameter variation on the controller, which served as the primary motivation for this research. The parameters of interest for this research have an impact on both the torque and current controller of PM synchronous machines, and the study on the effect of parameter variation is presented in the following sections. The study was restricted to the operation of a maximum torque per ampere controller in order to show the results. However, a similar analysis could be conducted based on another algorithm, which would lead to similar conclusions.

3.2.1 Impact of Parameter Variation on Torque Controller

It has been seen in section 2.3.1 that for all torque controller designs that were presented, only the torque constant and *d-q* inductances were important. The stator resistance did not have an impact on the design equations of torque controllers.

The results presented in the following sections were obtained with the Matlab / Simulink® model developed for this project. They aim at simulating results that are achievable with a machine referred to as machine "B", whose parameters are given in Table 3.1.

Since detailed information was not available about the machine construction materials, it was assumed that the stator windings were made of copper and the permanent magnets were NdFeB for the temperature effects.

Table 3.1: Machine “B” parameters

Parameter name	**Machine “B”**
Stator resistance, r_s (Ω)	1.45
Torque constant, K_T (V.s)	0.172
L_q (mH) no saturation	18
L_d (mH) no saturation	6
Rotor inertia, J (N.m.s^2)	99.6e-6
Number of pole pairs, P	2
Rated current, I_{max} (A)	25

3.2.1.1 Torque Constant Variation

The results shown in this section are the (i_d, i_q) loci for a torque reference that changes linearly from zero to almost the machine rated torque. Two sets of results were obtained for this section corresponding to two values of the torque constant or temperature. In this type of problem it is necessary to consider both the machine and controller sides to study the effect of temperature. Most controllers are designed by considering K_T to be constant and use the value measured at room temperature. In these tests, first, the controller was assumed to be running in ideal conditions (room temperature), and then the drive was simulated to be operating at a higher temperature, inducing a 30% change in K_T, both for the same controller, and for one that would have a feedback in K_T.

These results were extracted from a speed control test at low speed, with a ramp load torque. The curve on the left side, labeled “ML-CL” corresponds to an initial test at low temperature, where controller and machine have matching parameters. On the right-hand side, the two loci were taken from simulations where the machine was at high temperature (lower K_T), one where the controller and machine parameter matched (MH-CH) and one where they did not (MH-CL).

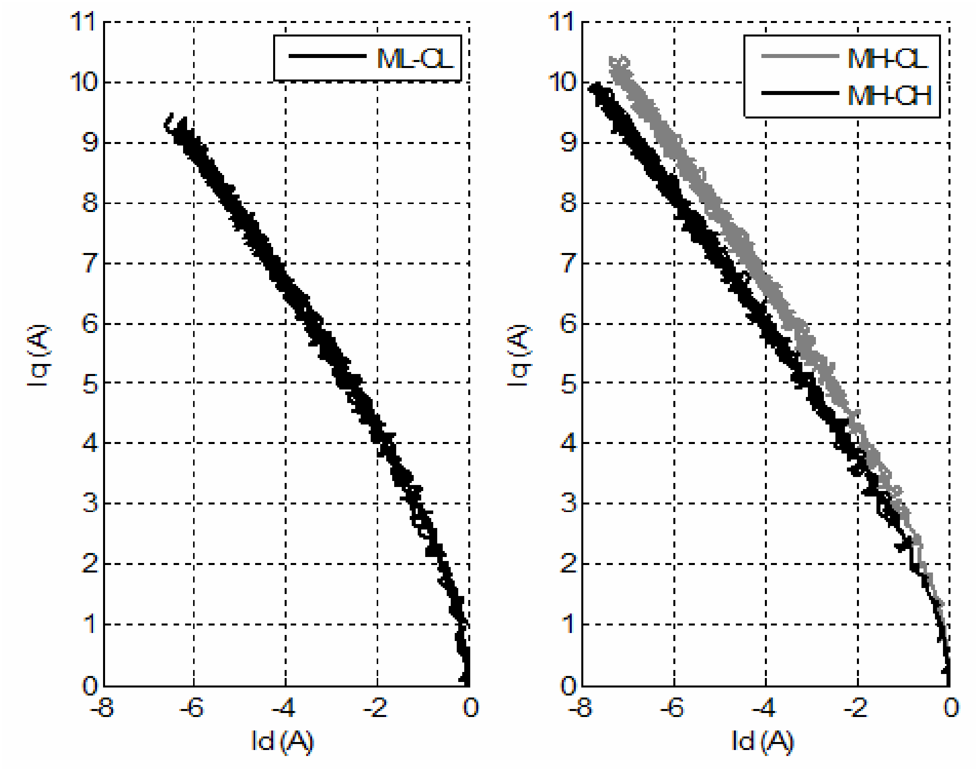

Fig. 3.4: Torque constant change in max Torque / Ampere controller.

A conclusion that can be drawn from this test is that the machine will require larger currents to operate at the same torque level for operation at higher temperature. As an example, the points that are the farthest from origin correspond to the same torque level. However, the direct correspondence between torque level and current magnitude was omitted here for clarity. This is due to the fact that φ_{mag} drops with rising temperatures and so does the torque production capability of the machine for the interaction between stator and rotor magnetic fluxes. For a given torque level, the machine will require larger currents at higher temperature to compensate for this drop. One can notice though, that the curve on the left (Figure 3.4 (a)), can almost be superimposed to the MH-CL one on the right (Figure 3.4 (b)), which is why it was left alone. The reason for this is that in both cases the controller operates on the same parameters, giving the same current loci.

For high temperature operation, the two curves in Figure 3.4(b) show that whether the controller has perfect knowledge of the torque constant or not will not make a large difference in the maximum Torque per Ampere locus. The sensitivity of this type of controller is therefore low to changes in temperature. This behavior is nonetheless a function of the machine, and a larger change could be seen on a machine with lower saliency.

There is however one interesting comment one can make about the two curves in Figure 3.4(b). The one that has the parameter mismatch (K_T larger than it is in the machine) uses larger q-axis currents than the other one. In this particular case, the controller that uses the accurate parameter values appears to be relying more on the saliency component of the torque equation (2.5). The reason for this is that as the torque constant drops (or as the temperature increases), it becomes more and more efficient to use the saliency component rather than the permanent magnet one for a given current magnitude to maximize the output torque. The following equations may help visualize the problem.

$$\begin{aligned} T_e &= A \cdot \left[K_{Thigh} \cdot i_q + B \cdot i_d \cdot i_q \right] \\ T_e &= A \cdot \left[K_{Tlow} \cdot i_q^{'} + B \cdot i_d^{'} \cdot i_q^{'} \right] \\ K_{Tlow} &> K_{Thigh} \end{aligned} \tag{3.4}$$

The same torque T_e is obtained with different *q*- and *d*-axes currents at two different temperatures. The controller that has the larger torque constant will try to use a larger *q*-axis current because it “thinks” that the machine’s permanent magnet torque capability is higher than it actually is.

3.2.1.2 Effects of Saturation

The simulations realized for this section are based on the machine “B” saturation characteristics shown in Figure 3.5.

The load torque of the machine is varied linearly from zero to a value close to the rated one, and the machine is operated under speed control mode. As the load torque increases, the machine torque also increases (to maintain constant speed), and the *q*- and *d*- axis currents describe the maximum torque per ampere trajectory. However, as the currents increase, saturation starts to play its role as described in Figure 3.5, affecting machine performance.

The left side of Figure 3.6 (Fig. 3.6(a)) shows the maximum torque per Ampere loci in two different cases in terms of controller parameters, and the right side shows the corresponding stator current magnitude as a function of machine torque, which is supposed to be minimized for a given torque.

Relating Figure 3.6 with Figure 3.5, it can be observed that the *d*-axis inductance remained constant for the values of currents that were used in this test. On the other hand, the *q*-axis current went as high as about 18 A, which represents a decrease of about 70% of the value of *q*-axis inductance. The large increase in *q*-axis current affected the saliency ratio of the machine in a significant way.

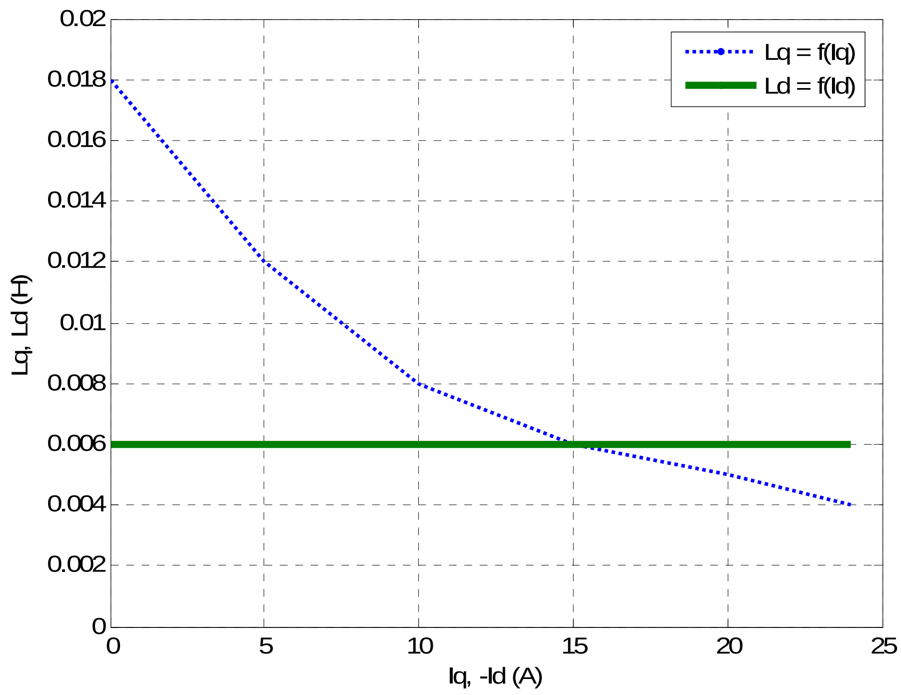

Fig. 3.5: Inductance waveforms of machine “B”.

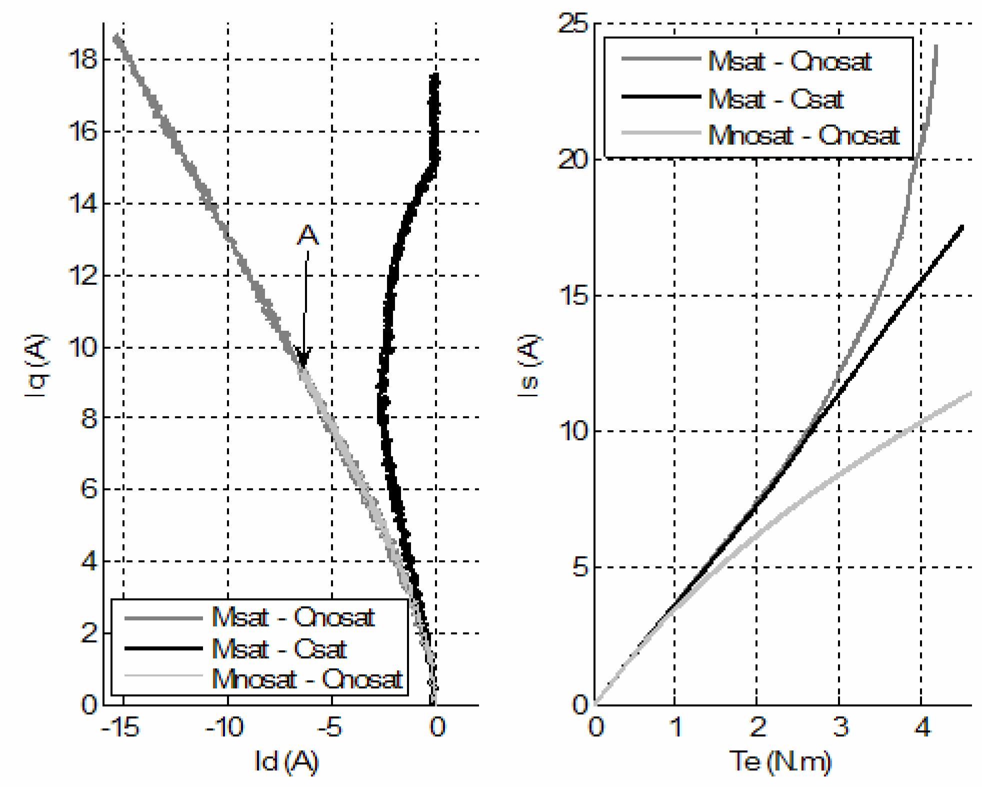

Fig. 3.6: Saturation effect on max Torque / Ampere.

Three tests were conducted to study the effects of saturation. The first one corresponds to the ideal case where there is no saturation (Mnosat-Cnosat). In that case the machine has constant *q*- and *d*-axes inductances, and the controller values match them. The second test (Msat-Cnosat) was essentially the same, except that the machine used the inductance waveforms of Figure 3.5, and the controller remained unchanged. In this case the current vector locus is almost the same as in the ideal case, although larger in magnitude ("Mnosat-Cnosat" stops at point "A" in Figure 3.6). The reason for this is that the controller operates on the same characteristic, even though it does not correspond to the optimum trajectory. The difference between these two cases shows on the right plot, where one can see that for the rated torque there is a 50% difference in motor current magnitude.

The third test is what is more important and really shows the importance of having accurate parameter feedback in the controller. In this last case, the machine presented saturation, and the controller used the saturated values of inductance to calculate the maximum Torque per Ampere trajectory. The difference between this locus and the others is very large. As the machine torque and currents increase, the saliency ratio diminishes and the machine begins to operate somewhat like a surface-mount machine. The contribution of the saliency torque becomes small and the torque controller finds it more efficient to use more of the magnetic flux interaction for the same torque level.

$$T_e = A \cdot \left[K_T \cdot i_q + B \cdot i_d \cdot i_q\right]$$
$$T_e = A \cdot \left[K_T \cdot i_q^{'} + B^{'} \cdot i_d^{'} \cdot i_q^{'}\right]$$
$$B^{'} << B << K_T$$

From the above equations it is clear that the required optimal *q*-axis current will have to be larger for the controller that takes saturation into account ($i_q^{'}, i_d^{'}$), for the same level of torque as in the initial one. However, the operating point that would be given by the maximum Torque per Ampere algorithm with parameter feedback is not the optimal operating point. The equations used to derive this algorithm in section 2.3.1.2 make the assumption that the *d-q* inductances in the machine are constant. To get the optimal point, it would be necessary to repeat the derivation and consider the inductances to be functions of the machine currents. This approach is however incompatible with a basic on-line parameter estimation algorithm. The reason for this is that optimum operation would require algorithm derivation with complete knowledge of machine parameters, and the parameter estimation algorithm only provides the controller with the inductance parameters that correspond to the current operating point.

These results also show that for an interior PM synchronous machine with high saturation it is not efficient to operate at high levels of saturation. When these machines are operated under flux weakening or maximum torque per flux, the controller takes advantage of the saliency torque and reduces the *q*-axis current, thereby reducing the effect of saturation.

3.2.2 Impact of Parameter Variation on Current Controller

It has been shown in section 2.3.2 how machine parameters play a certain role in the performance and design of the current controller. The feedforward compensator which rejects the disturbances created by the cross coupling between the *d*- and *q*- axes and the back-electromotive force takes advantage of parameter feedback. The terms introduced by that compensator are all proportional to the motor speed, and do not make a difference at standstill. The results to be presented in this section show the effect of speed as a disturbance in the complete current controller, i.e. one that includes feedforward. These results were obtained with a machine model that matches the one used in the experimental setup (machine "A"). The simulated machine was run at a speed of about 1500 RPM with a sinusoidal variation of 500 RPM around that. The *q*- and *d*-axes currents were to be maintained constant by the current controller.

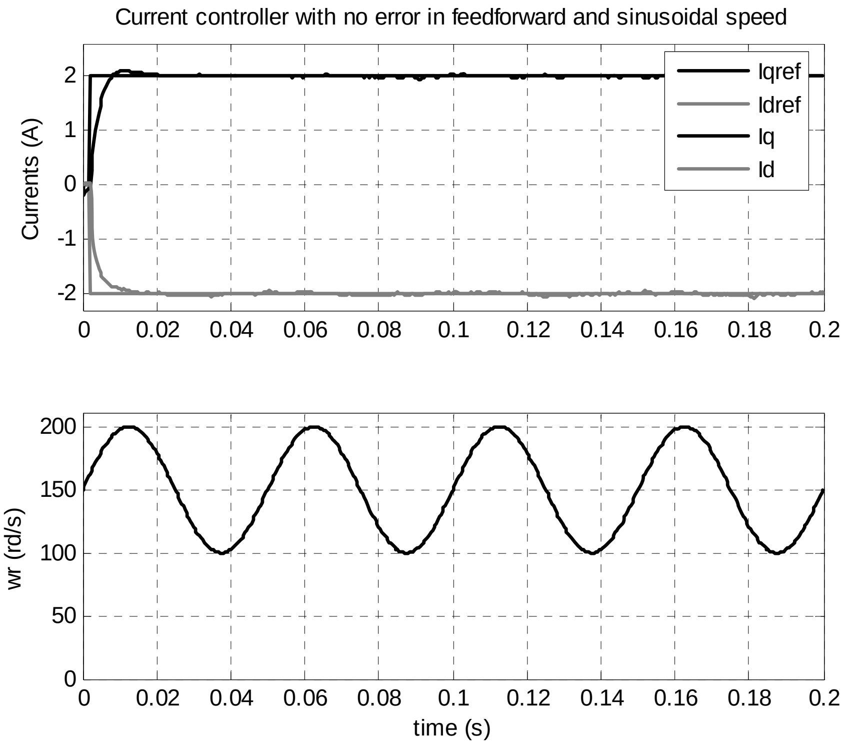

Fig. 3.7: Response of current controller to sinusoidal speed perturbation.

In Figure 3.7, the response of the current controller to a step in the *q*- and *d*-axes reference currents as the machine speed varies sinusoidally is shown. The current responses track the references very effectively, and the speed perturbation doesn't seem to affect the performance.

Figure 3.8 on the other hand shows the response of the current controller to the same speed perturbation, but with errors in its torque constant and *q*-axis inductance. Just as in section 3.1.1, a variation of 30% in torque constant and 40% in *q*-axis inductance were used.

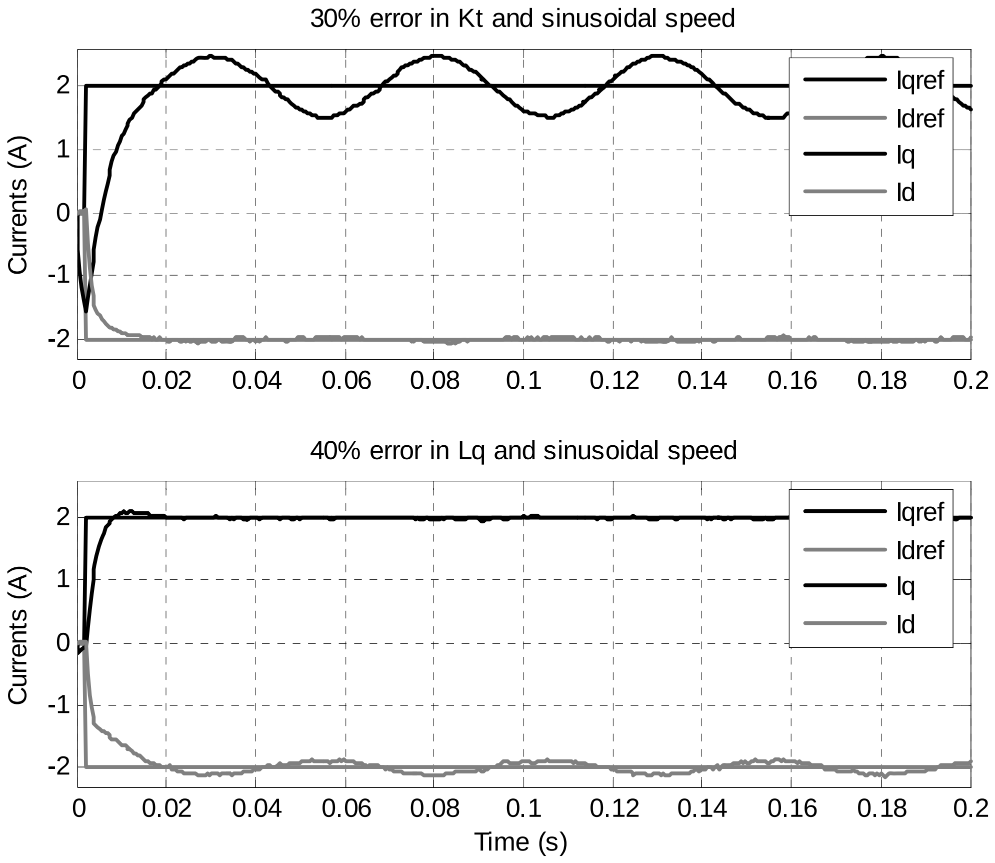

Fig. 3.8: Current controller response with parameter error to speed perturbation.

The first plot in Figure 3.8 shows that a variation in the torque constant translates into a poor compensation of the speed variation in the *q*-axis current. In contrast, a variation of the *q*-axis inductance has no effect on the *q*-axis current, but shows the sinusoidal perturbation on the *d*-axis current. This can be explained by looking at the design equations of the feedforward current controller (2.16).

$$\begin{cases} V_q' = V_q - \omega_e \cdot L_d \cdot i_d - \omega_r \cdot K_T = r_s \cdot i_q + p \cdot L_q \cdot i_q \\ V_d' = V_d + \omega_e \cdot L_q \cdot i_q = r_s \cdot i_d + p \cdot L_d \cdot i_d \end{cases}$$

Due to the cross-coupling effect between *q*- and *d*-axes of the machine, the consequences of a poor estimation of L_q will lead to errors in the *d*-axis current control and vice versa. On the other hand, errors in the torque constant will lead to problems on the *q*-axis. In addition to this, for the machine under study in this research, we have

$K_T >> L_{q\max} \cdot i_{\max}$ or $L_{d\max} \cdot i_{\max}$.

This makes the current controller much more sensitive to errors in the torque constant than in inductance, which can be observed from Figure 3.5.

The effects of stator resistance change only the dynamics of the current controller and affect only mildly the design of the PI controller. Resistance feedback would be useful if the current controller was designed to account for such changes. In the present case, dynamic performance is satisfactory for the purpose of this research. The possible role and importance of stator resistance estimation will be further analyzed later.

3.3 Research Objectives

The objectives for the research presented in this paper are multiple, but its general goal is to find a solution to the problem of on-line parameter estimation of PM machines for the parameters outlined in section 3.1. The algorithm to be found should have the capabilities of tracking parameter variations due to temperature and saturation, and parameter estimation at steady state. Further details on the objectives of the algorithm are presented in the following subsections.

3.3.1 Tracking of Parameter Variations due to Temperature

The method to be found should be able to track parameter variations due to temperature, which are slow compared to the dynamics involved in the machine controller. These temperature effects can affect all four parameters, but they are likely to affect the stator resistance and torque constant most.

Interest in tracking the stator resistance changes has not been emphasized so far, because it can be seen as secondary from the point of view of controller design. The resistance only comes into play in the design of the compensator for the current controller, which is important for very high performance controllers, but requires additional computations as the machine is running. On the other hand, the material properties of electrical conductors are very well known, and it could be possible to use an on-line resistance estimate to get an idea of the machine operating temperature. Monitoring resistance changes could also be interesting for the purpose of machine or inverter diagnostics. This could be done as a system self check before startup. Finally, the most important role of knowledge of stator resistance for this project is that it will allow a much easier operation for the identification of other machine parameters, as it will be shown later. A reliable resistance estimate will likely be crucial for the numerical stability of the entire algorithm. Existing algorithms [18] [19] acknowledge the importance of having good resistance and torque constant estimates for the purpose of identifying the *q*- and *d*-axes inductances.

The machine torque constant is another parameter that changes mostly because of temperature variation. Having a good estimate for this parameter is very important for the operation of the current controller and is appreciable when it comes to the torque controller, as it has been mentioned in sections 3.2.1.1 and 3.2.2. Depending on the magnet material used in the rotor, the torque constant may present a high sensitivity to temperature. As for the stator resistance, the torque constant will also be of importance for the overall stability of the parameter estimation algorithm.

Identification of parameter variation due to temperature changes should be realized with the steady-state machine model in order to maintain numerical simplicity and to avoid the inclusion of current derivative terms that are especially difficult to estimate in motor drive applications. This should not be an issue because of the relative slowness of temperature variation when compared to the motor drive dynamics.

3.3.2 Tracking of Inductance Variation due to Saturation

The problem related to saturation is of a different nature than the one related to temperature changes. As opposed to the latter, saturation effects are directly related to the current levels flowing in the machine, and their dynamics are as fast as the currents'. As a consequence, an on-line estimation algorithm for inductances must operate at a much faster rate than one designed for temperature related phenomena. The algorithm may have to use the machine model including current dynamics, or operate at a rate that makes it possible to neglect them. Another solution could be to use the results given by the algorithm in steady state operation and update a look-up table used by the controller. An advantage in the use of an on-line estimation algorithm is that it can track the effects of cross-saturation transparently from the user point of view. The method should also be able to operate so as not to disturb the one related to temperature changes if they are distinct. These issues will be discussed further in chapters IV and VI.

3.3.3 Steady-State Detection Capability

In order to best use and coordinate the estimation of all electrical machine parameters, an algorithm will have to be developed that can operate when the machine is in steady state. For example, a method based on the steady-state model of the machine is bound to show errors during a change of operating point because the model it uses does not represent the controlled system during that time. This “steady-state detection” algorithm should then be able to enable or disable the operation of the parameter estimation algorithms, and even discard some estimation results. This method will be based on the current or voltage commands that the different controllers issue.

3.4 Conclusion

This chapter presented the phenomena that affect the main machine parameters, and the subsequent effect of parameter variation on controller performance. This analysis served as a basis for the determination of precise objectives for this research project.

CHAPTER IV

ON-LINE PARAMETER ESTIMATION ALGORITHM

The objective of this research is to provide the machine controller with parameter estimates that are calculated as the machine is operated. The Recursive Least Squares (RLS) algorithm was chosen as a basis for this purpose. It is a relatively fast algorithm that provides reliable parameter estimations, even in the presence of white noise. Since the structure of the machine model was known beforehand, this algorithm is suitable for this research.

4.1 Least Squares algorithms

The RLS algorithm is based on the Least Squares (LS) algorithm. When applied to system parameter estimation, the latter simply gives an estimate of unknown model parameters based on a given number of input and output system measurements. These measurements can be done at any point in time prior to the execution of the LS algorithm. On the other hand, the RLS algorithm is one that can be directly adapted to on-line parameter estimation because its structure allows easier expressions for new parameter estimates each time it is supplied with new input and output data.

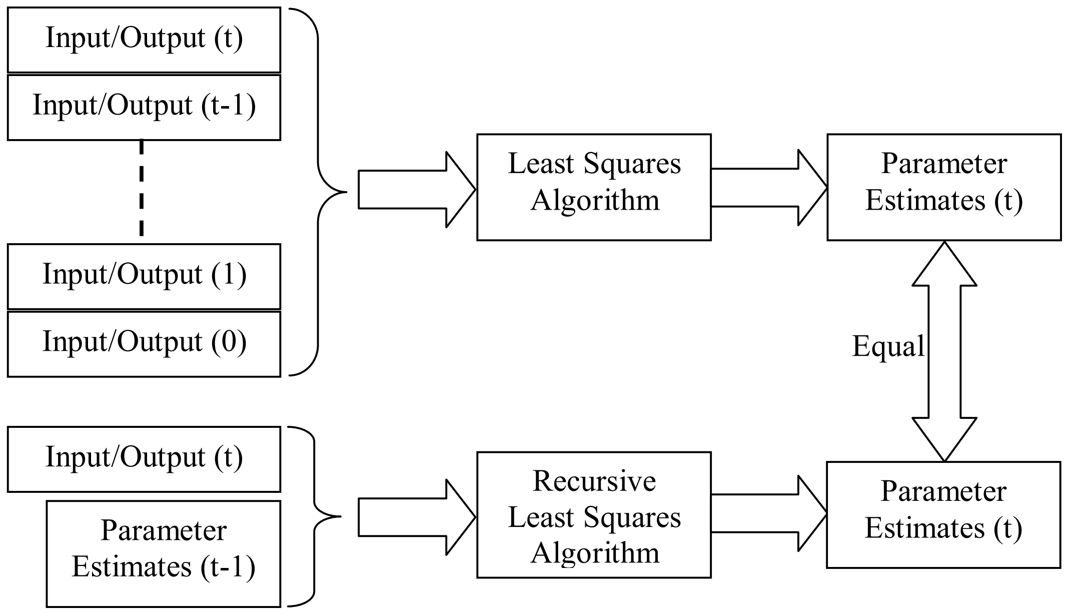

Fig. 4.1: Comparison between RLS and LS algorithm structures.

Figure 4.1 shows the fundamental structural difference between the RLS and LS algorithms. A simple technique to obtain an on-line estimation algorithm from the LS algorithm would be to execute it each time new input/output data is available. Although this could be possible in theory, the numerical computation power required for that would quickly become an impediment for an on-line implementation. One major constraint that one encounters when confronted with real-time numerical control techniques is that the various programs associated with an application have to be executed fast enough in order to be able to maintain the desired sampling rate.

In applications, the sampling rate is set to a value that allows the control system to take into account the main dynamics of the system. The sampling rate is usually deduced from Shannon's law, which states that the sampling rate has to be at least twice that of the highest frequency to be considered. Usually, the sampling rate choice of a control system is a compromise between software complexity and system stability. Control algorithms are typically executed at the sampling rate in order to account for the complete system dynamics. For that to be possible, the various algorithms associated with the application have to be executed in a time that is shorter than the sampling period. If this condition is not realized, the program can easily become unstable. The inherent structure of the LS algorithm makes it unsuitable for on-line applications. The RLS algorithm was developed to overcome this limitation and still give the same estimates for a given set of inputs.

4.2 Introduction to the LS algorithm

For a system described by

$$y_{(k)} = \varphi_{(k)}^{T} \cdot \theta \tag{4.1}$$

where y is the system output, φ represents a measurement vector and θ the system parameter vector that is to be estimated. If we have N equations like equation (4.1), we can write them in a combined form as

$$Y = \Phi \cdot \theta$$

where $\Phi_{(k)} = \left[\varphi_{(1)}, \varphi_{(2)}, \ldots, \varphi_{(k)}\right]^{T}$ and $Y_{(k)} = \left[y_{(1)}, y_{(2)}, \ldots, y_{(k)}\right]^{T}$. The Least Squares algorithm calculates an estimate of the system parameters such that

$$\begin{cases} \theta_{est} = \arg_{\theta}\left(\min\left(V_{(\theta)}\right)\right) \\ V_{(\theta)} = \dfrac{1}{2} \cdot \displaystyle\sum_{k=1}^{N} \left(y_{(k)} - \varphi_{(k)}^{T} \cdot \theta\right)^{2} \end{cases}$$

The solution to this problem is as follows [22]:

$$\theta_{est} = \left(\Phi^{T} \cdot \Phi\right)^{-1} \cdot \Phi^{T} \cdot Y \tag{4.2}$$

One can see that this solution requires the inversion of a matrix that can be large, depending on the number *N* of input/output equations considered. The RLS algorithm is one that would only use data available at a particular point in time and its previous estimate to provide the user with the same result as the LS algorithm.

4.3 RLS algorithm

4.3.1 Definition of the RLS algorithm

The approach taken to get the RLS algorithm from the regular LS algorithm is interesting because it is one that could be applied to other offline algorithms. To obtain a formulation of the RLS algorithm, we start from the result that the LS algorithm would give at the kth sample (Eq (4.2)) as:

$$\theta_{est(k)} = \left(\Phi_{(k)}^T \cdot \Phi_{(k)}\right)^{-1} \cdot \Phi_{(k)}^T \cdot Y_{(k)} \tag{4.3}$$

Introducing $F_{(k)} = \left[\Phi_{(k)}^T \cdot \Phi_{(k)}\right]$, leads to $F_{(k)} \cdot \theta_{est(k)} = \Phi_{(k)}^T \cdot Y_{(k)}$.

By decomposing the matrices at the kth step, one can get

$$F_{(k)} = \left[\Phi_{(k-1)}^T \; \varphi_{(k)}\right] \cdot \begin{bmatrix} \Phi_{(k-1)} \\ \varphi_{(k)}^T \end{bmatrix} = \Phi_{(k-1)}^T \cdot \Phi_{(k-1)} + \varphi_{(k)} \cdot \varphi_{(k)}^T = F_{(k-1)} + \varphi_{(k)} \cdot \varphi_{(k)}^T$$

The *F* matrix can be obtained recursively using the above equation. Going back to the LS algorithm, it is possible to get:

$$\theta_{est(k)} = F_{(k)}^{-1} \cdot \Phi_{(k)}^T \cdot Y_{(k)} = F_{(k)}^{-1} \cdot \left[\Phi_{(k-1)}^T \; \varphi_{(k)}\right] \cdot \begin{bmatrix} Y_{(k-1)} \\ y_{(k)} \end{bmatrix}$$

And then

$$\theta_{est(k)} = F_{(k)}^{-1} \cdot \left[\Phi_{(k-1)}^T \cdot Y_{(k-1)} + \varphi_{(k)} \cdot y_{(k)}\right] = F_{(k)}^{-1} \cdot \left[F_{(k-1)} \cdot \theta_{est(k-1)} + \varphi_{(k)} \cdot y_{(k)}\right]$$

This expression can be further simplified as follows:

$$\theta_{est(k)} = F_{(k)}^{-1} \cdot \left[\left(F_{(k)} - \varphi_{(k)} \cdot \varphi_{(k)}^{T}\right) \cdot \theta_{est(k-1)} + \varphi_{(k)} \cdot y_{(k)}\right]$$
$$\theta_{est(k)} = F_{(k)}^{-1} \cdot \left[F_{(k)} \cdot \theta_{est(k-1)} - \varphi_{(k)} \cdot \varphi_{(k)}^{T} \cdot \theta_{est(k-1)} + \varphi_{(k)} \cdot y_{(k)}\right]$$

And from this last equation, the first version of the RLS algorithm can be deduced:

$$\begin{cases} \theta_{est(k)} = \theta_{est(k-1)} + F_{(k)}^{-1} \cdot \varphi_{(k)} \cdot \left(y_{(k)} - \varphi_{(k)}^{T} \cdot \theta_{est(k-1)}\right) \\ F_{(k)} = F_{(k-1)} + \varphi_{(k)} \cdot \varphi_{(k)}^{T} \end{cases}$$

This algorithm can be practically implemented as it is, but it presents the problem of requiring the inversion of a possibly large matrix at each step. Matrix inversions are extremely time consuming operations and are to be avoided as much as possible in real-time algorithms. As a consequence, this version of the RLS algorithm had to be further modified in order to make it more practical.

A simplified recursive expression for the inverted matrix had to be found. For this purpose, the following matrix is introduced:

$$P_{(k)} = F_{(k)}^{-1}$$

And from

$$F_{(k)} = F_{(k-1)} + \varphi_{(k)} \cdot \varphi_{(k)}^{T}$$

one can obtain

$$P_{(k)}^{-1} = P_{(k-1)}^{-1} + \varphi_{(k)} \cdot \varphi_{(k)}^{T}$$
$$P_{(k)} = \left[P_{(k-1)}^{-1} + \varphi_{(k)} \cdot \varphi_{(k)}^{T}\right]^{-1}$$

At this point, a matrix inversion lemma can be used:

$$\left[A + B \cdot C \cdot D\right]^{-1} = A^{-1} - A^{-1} \cdot B \cdot \left[D \cdot A^{-1} \cdot B + C^{-1}\right]^{-1} D \cdot A^{-1}$$

with $A = P_{(k-1)}^{-1}$, $B = \varphi_{(k)}$, $C = I_n$, $D = \varphi_{(k)}^{T}$ where I_n is the identity matrix.

Therefore,

$$P_{(k)} = P_{(k-1)} - P_{(k-1)} \cdot \varphi_{(k)} \cdot \left[\varphi_{(k)}^{T} \cdot P_{(k-1)} \cdot \varphi_{(k)} + I_{n}\right]^{-1} \cdot \varphi_{(k)}^{T} \cdot P_{(k-1)} .$$

Introducing,

$K_{(k)} = P_{(k)} \cdot \varphi_{(k)}$ and $\varepsilon_{(k)} = y_{(k)} - \varphi_{(k)}^{T} \cdot \theta_{est(k-1)}$,

Rewriting the previous algorithm and obtain the final implementation of the RLS algorithm [22] [23] can be obtained as:

$$\begin{cases} \theta_{est(k)} = \theta_{est(k-1)} + K_{(k)} \cdot \varepsilon_{(k)} \\ K_{(k)} = P_{(k-1)} \cdot \varphi_{(k)} \cdot \left[I_{n} + \varphi_{(k)}^{T} \cdot P_{(k-1)} \cdot \varphi_{(k)}\right]^{-1} \\ P_{(k)} = \left[I_{n} - K_{(k)} \cdot \varphi_{(k)}^{T}\right] \cdot P_{(k-1)} \end{cases} \tag{4.4}$$

One can notice that this algorithm may also require a matrix inversion, but this time the matrix to be inverted is a function of the number of system outputs. In the case of a Single-Input-Single-Output (SISO) system, this matrix is scalar. In the case of interest, the matrix to be inverted will be one with a dimension of 2, which is quite simple to invert.

4.3.2 Modified RLS algorithm

The RLS algorithm as it is presented here is still not suitable for the kind of parameter estimation required by this project. The algorithm gives equal weight to all measurements, just like in the original LS algorithm, and the adaptation gain matrix *K* tends towards the zero matrix as *k* increases. This comes from the assumption that the parameters are constant for all measurements. The aim of this project is to have an algorithm that can estimate motor parameters and track their variations. The RLS algorithm has to be modified in order to give higher importance to the most recent measurements, since they correspond to the most recent motor model parameters.

A solution to this problem can be obtained by changing the initial cost function and introducing an exponential "forgetting factor":

$$\begin{cases} \theta_{est} = \arg_{\theta}\left(\min\left(V_{(\theta)}\right)\right) \\ V_{(\theta)} = \frac{1}{2} \cdot \sum_{k=1}^{N} \left(y_{(k)} - \varphi_{(k)}^{T} \cdot \theta\right)^{2} \lambda^{N-k} \end{cases}$$

The introduction of λ, which is a real number that has a value between 0 and 1, allows the algorithm to weigh the measurements and give more importance to the most recent ones. If the forgetting factor is equal to "1", we will get the regular RLS algorithm, and if it is set at zero, we will get a simplified algorithm called "gradient algorithm". Usually, values of the forgetting factor range from 0.95 to 0.999. If the same approach that gave the RLS algorithm (Eq. (4.4)) is used with this modified cost function, the following algorithm is obtained [23]:

$$\begin{aligned} \theta_{est(k)} &= \theta_{est(k-1)} + K_{(k)} \cdot \varepsilon_{(k)} \\ \varepsilon_{(k)} &= y_{(k)} - \varphi_{(k)}^{T} \cdot \theta_{est(k-1)} \\ K_{(k)} &= P_{(k-1)} \cdot \varphi_{(k)} \cdot \left[\lambda \cdot I_{n} + \varphi_{(k)}^{T} \cdot P_{(k-1)} \cdot \varphi_{(k)}\right]^{-1} \\ P_{(k)} &= \left[I_{n} - K_{(k)} \cdot \varphi_{(k)}^{T}\right] \cdot P_{(k-1)} / \lambda \end{aligned} \qquad (4.5)$$

The above algorithm is the one that will serve as a basis for parameter estimation for the research project.

4.4 Estimation Algorithms

4.4.1 Overview

The parameter vector to be estimated in this project was given in Eq. (3.1).

$$\theta = \begin{bmatrix} L_q \\ L_d \\ r_s \\ K_T \end{bmatrix} \tag{3.1}$$

The inductances can vary because of saturation at a rate that is comparable to that of the machine currents. On the other hand, the most important factor that affects the motor phase resistance and the torque constant is the motor temperature, which varies at a much slower rate than the other electromechanical variables in the machine. A decoupling between the inductances and the latter two parameters was necessary in order to minimize the program execution time. A simple approach to the problem of estimating all four parameters in the machine would be to have a RLS algorithm which would aim at estimating all four parameters at a fast rate, possibly at the sampling rate of the application. This approach is not possible for a few reasons, one of which is that it would be a heavy burden on the control program, and another one will be mentioned later in the chapter. A more simple approach to the problem would be to split the estimation algorithm as a function of the different execution rates required. A simpler algorithm would focus only on the estimation of the *q*- and *d*-axes inductances and run at the sampling rate, with the assumption that the resistance and torque constant remain constant at this rate. The estimation of the remaining two parameters can be done in a separate program that would be run at a lower frequency.

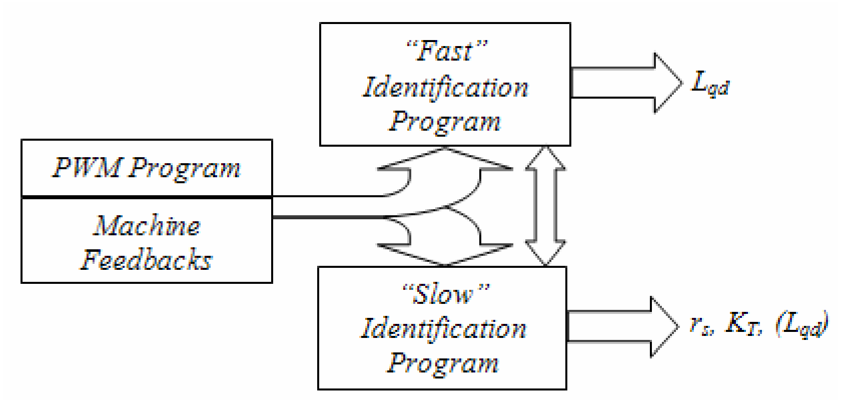

Fig. 4.2: Proposed algorithm structure.

The algorithm chosen for the PM machine parameter estimation has the structure shown in Figure 4.2. It is decomposed into two subroutines, a "fast" one that estimates the inductances, and a slower one that estimates all four parameters. A method where the two pairs of parameters would be completely decoupled was tested but proved to be unstable; an error in any of the four parameters would get amplified by the other estimation program and build up.

4.4.2 Fast Estimation Algorithm

The goal for this portion of the program is to estimate the machine *q*- and *d*-axes inductances, assuming the motor phase resistance and torque constant are known. As mentioned earlier, this routine will run at the fastest rate possible in the program, which is the sampling rate. The sampling frequency was set to 20 kHz for the experimental and simulation setups; this value is commonly used in high performance motor control applications. Because of measurement limitations, the algorithm was developed from the steady state model of the machine (the equations are restated here for clarity):

$$\begin{cases} V_q = r_s \cdot i_q + L_d \cdot \omega_e \cdot i_d + K_T \cdot \omega_r \\ V_d = r_s \cdot i_d - L_q \cdot \omega_e \cdot i_q \end{cases} \tag{2.17}$$

This model does not include the current differential terms because of the limited precision, the signal to noise ratio that was available for the experimental design, and the high sampling rate. These terms are of the following form:

$$A_{qd} = L_{qd} \cdot \frac{di_{qd}}{dt}$$

For a small *dt*, a small error in di_{qd} would lead to a large error in *A*. On the other hand, a larger *dt* would introduce an undesirable lag in the treatment of the measurements. If they had been included in the experimental algorithm, they would have introduced a level of error that would probably outweigh the motivations for their inclusion. Excluding these terms will cause the model to misrepresent the machine behavior whenever there is a change in machine current. These changes however are usually relatively short but depend on the application. The benefits of including these terms will be studied through simulation.

The RLS algorithm is set up as follows. The voltage equations are rewritten in order to isolate the parameters to be identified:

$$\begin{cases} V_q - r_s \cdot i_q - K_T \cdot \omega_r = L_d \cdot \omega_e \cdot i_d \\ V_d - r_s \cdot i_d = -L_q \cdot \omega_e \cdot i_q \end{cases}$$

This can then be written as the RLS matrices:

$$\begin{bmatrix} V_q - r_s \cdot i_q - K_T \cdot \omega_r \\ V_d - r_s \cdot i_d \end{bmatrix} = \begin{bmatrix} 0 & \omega_e \cdot i_d \\ -\omega_e \cdot i_q & 0 \end{bmatrix} \cdot \begin{bmatrix} L_q \\ L_d \end{bmatrix}$$

$$y = \begin{bmatrix} V_q - r_s \cdot i_q - K_T \cdot \omega_r \\ V_d - r_s \cdot i_d \end{bmatrix}, \ \varphi^T = \begin{bmatrix} 0 & \omega_e \cdot i_d \\ -\omega_e \cdot i_q & 0 \end{bmatrix} \text{ and } \theta = \begin{bmatrix} L_q \\ L_d \end{bmatrix}$$

The RLS algorithm can be implemented with the above equations and give estimates for the motor inductances as a function of the inputs and measurements to and from the motor. The only constraint is that the phase resistance and motor torque constant have to be supplied externally. When the machine is at room temperature, these parameters can be set to constant values that are measured offline, but for the final algorithm implementation, these will be supplied from another identification routine. The program structure is given in Figure 4.3.

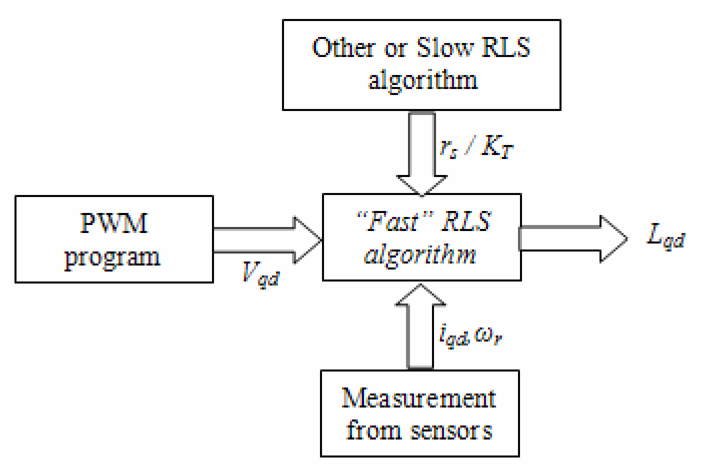

Fig. 4.3: “Fast” estimation program structure.

The primary concern in terms of numerical implementation was the size of the matrix to be inverted in equation (4.5):

$$\lambda \cdot I_n + \varphi_{(k)}^T \cdot P_{(k-1)} \cdot \varphi_{(k)}$$

In this case, both φ and φ^T are 2x2 matrices, and from this we can deduce that P is of dimension 2x2. Finally, the identity matrix must have the same size in order to be added properly. The matrix that has to be inverted is of size 2x2, and its inversion is a simple operation:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \left(\frac{1}{a \cdot d - b \cdot c} \right) \cdot \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

This identification program is suitable to be run continuously and will converge even if the machine stays in one set operating point in terms of currents, voltages and speed. Usually, for the RLS algorithm to converge properly, it needs to be supplied with input and output data that is "rich" enough in information. In the present case, the vector that has to be estimated is of size 2, and one operating point gives two equations, one for the *q*-axis, and another for the *d*-axis. There is therefore enough information in one operating point for the RLS algorithm to converge properly. This would not be true if the vector to be estimated was any larger in size, as it is the case with the second part of the estimation algorithm.

4.4.3 "Slow" Estimation Algorithm

The second part of the complete estimation method is one that was designed to be run at a much slower frequency than the sampling rate (500 to 2000 Hz in simulation and experiments). The purpose of this program is to estimate all four parameters using the RLS algorithm. However, the inductance estimate given by this algorithm is not used in other parts of the program and was introduced in order to avoid isolating the two sets of machine parameters and improve algorithm stability. This algorithm is also based on the steady state *d*-*q* model of the machine, but as opposed to the "fast" identification program, it will only be run during steady states.

This time the machine model equations are written as:

$$\begin{bmatrix} V_q \\ V_d \end{bmatrix} = \begin{bmatrix} 0 & \omega_e \cdot i_d & i_q & \omega_r \\ -\omega_e \cdot i_q & 0 & i_d & 0 \end{bmatrix} \cdot \begin{bmatrix} L_q \\ L_d \\ r_s \\ K_T \end{bmatrix}$$

And the RLS matrices are:

$$y = \begin{bmatrix} V_q \\ V_d \end{bmatrix},\ \varphi^T = \begin{bmatrix} 0 & \omega_e \cdot i_d & i_q & \omega_r \\ -\omega_e \cdot i_q & 0 & i_d & 0 \end{bmatrix} \text{ and } \theta = \begin{bmatrix} L_q \\ L_d \\ r_s \\ K_T \end{bmatrix}$$

In this case, φ is a 4x2 matrix and φ^T is a 2x4 matrix, and from this we can deduce that *P* is of dimension 2x2. The matrix that has to be inverted for this algorithm will be a matrix of size two, just like in the case of the "fast" algorithm. On the other hand, the algorithm will require matrix products and sums that will handle larger matrices than in the case of the "fast" algorithm, and will consequently take a longer time to execute.

The primary constraint associated with the implementation of this RLS algorithm came from the fact that it would require data from more than one operating point in order to be stable. Since the parameter vector is of size four and the system is of size two, this algorithm needs at the very least data from two different operating points in order to operate properly. On the other hand, a design constraint associated with a practical implementation of the algorithm is that it should be unintrusive as possible. In other words, the estimation algorithm does not set the machine operating points; they are defined by the application (speed, torque or position control). The solution that was chosen was a compromise: the estimation algorithm will add a perturbation to the *d*-axis of the machine and get its data from the machine feedback. This perturbation was chosen to be on the *d*-axis because it usually is the least torque producing one, especially in a surface mounted machine or an IPM with low reluctance. In the case of an IPM with very high reluctance, or a synchronous reluctance machine, it may become more interesting to use the *q*-axis.

The *d*-axis perturbation will have to be as small as possible, in order to minimize the torque ripple in the machine, but will have to be large enough to have its effects felt in the controller feedback. It is therefore mostly a function of the hardware in terms of the quality of the machine current feedback. The perturbation was chosen to be a succession of small steps at a rather low frequency, leading to a succession of electrical steady states in the machine. The excitation frequency was chosen to be much lower than the electrical dynamics of the machine, so that the current transients would not dominate a time step. On the other hand, it also had to be chosen high enough in order to be able to feed the RLS algorithm with data that would allow it to track parameter variations. This minimum magnitude is a direct function of the sensing capabilities of the system (resolution). The perturbation magnitude also has to be fast and small enough in order to minimize the machine torque ripple and the perturbation in mechanical dynamics.

Another design constraint associated with the implementation of this algorithm is that the injected current perturbation may also result in an inductance perturbation, due to saturation and/or cross saturation. The solution to this problem was to disable the algorithm during current transients, and to copy the estimated inductances coming from the “fast” algorithm when the “slow” algorithm would be enabled again.

The program structure associated with the implementation of this algorithm is shown on Figure 4.4.

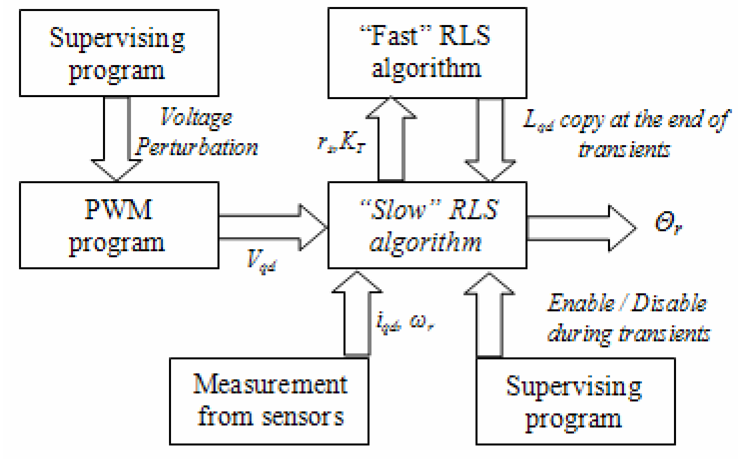

Fig. 4.4: "Slow" estimation program structure.

The algorithm uses the same inputs as the "fast" estimation algorithm, except for the fact that it requires updates in its estimated inductances at the end of current transients. Figure 4.4 shows in details the interaction between the two algorithms during a transient.

When a current transition occurs in the controller, either due to a change in controller operating point, or to the perturbation necessary to excite the "slow" algorithm, the estimation algorithms have to respond accordingly. In the example shown in Figure 4.5, a change in the *d*-axis current was introduced by the controller. The first plot shows how the *d*-axis current reference changes from one level to another, and how the actual *d*-axis current tracks it, this being the result of the operation of the current controller.

The second plot shows the operation of the "fast" estimation algorithm; it is assumed that *d*-axis inductance value is being effectively tracked initially. When the transient occurs, the differential terms start to appear in the machine equations, but are not taken into account by the estimation algorithm. This explains why the estimation cannot track the inductance during the current transient. Once the current has reached steady state again, the estimation algorithm converges again. During that time however, the "slow" estimation algorithm was disabled, and the estimated resistance and torque constant remained unchanged. This approach is valid because current transients are much faster than temperature transients.

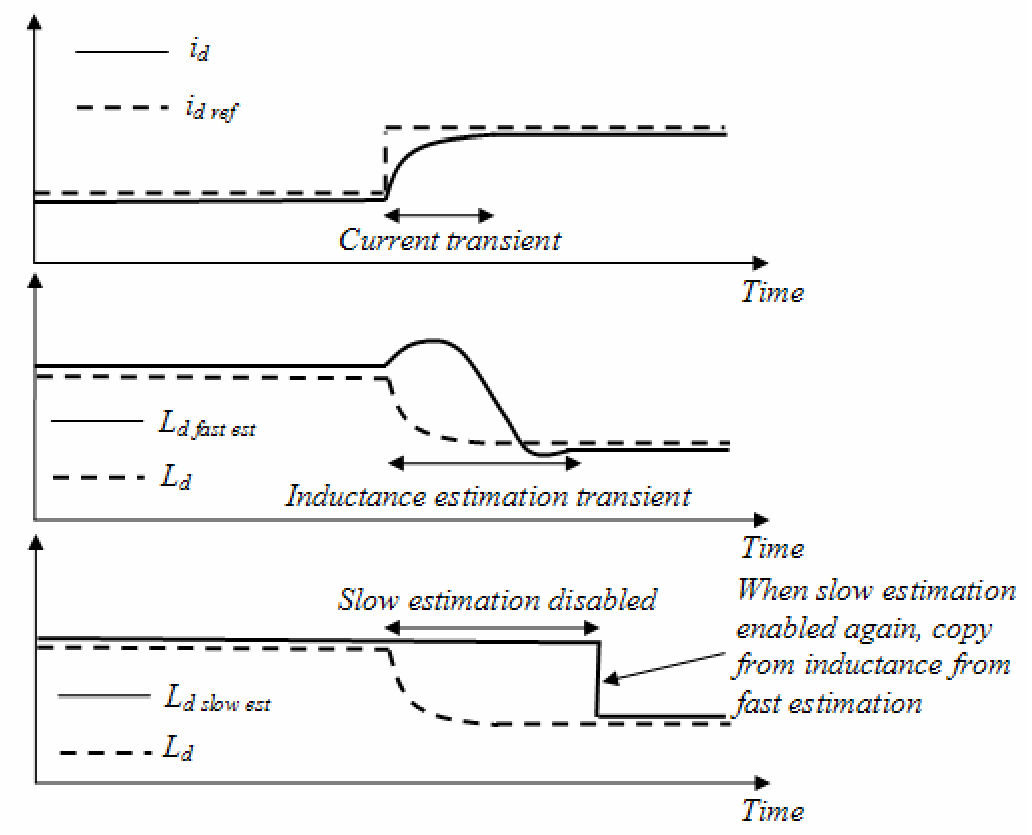

Fig. 4.5: Estimation algorithm behavior during current transient.

However, when the "slow" estimation algorithm resumes its operation, it still has the old *d*-axis inductance value in memory, which does not match the actual machine inductance anymore. The inductance coming from the "fast" algorithm is therefore copied in the "slow" algorithm when it resumes its operation. This allows the current transient to be almost transparent from the point of view of the "slow" algorithm.

4.5 Conclusion

The recursive least squares algorithm and its derivation were presented in this chapter. An on-line parameter estimation algorithm was then introduced, based on this algorithm. The proposed structure takes advantage of the different dynamics of the parameters that had to be estimated and is decomposed into three main blocks. One will be dedicated to machine inductance estimation, another one will try to estimate all four machine parameters, and the last block will supervise their operations. In the following chapters, the effectiveness of this proposed algorithm will be investigated and verified both through simulation and experiments.

CHAPTER V

PARAMETER ESTIMATION SIMULATION MODEL

The first step in developing and validating an algorithm for on-line parameter estimation in PMSM is to develop a machine and controller model that would be used for simulation. This model would serve the purpose of verifying the effectiveness of the proposed algorithm in a simulated environment that is as close as possible to the experimental design. In addition to this, the model could also be used to verify and measure variables that may not be easily accessible in the experimental design.

The platform that was chosen to develop the simulation model was Matlab®, and its extension Simulink®. They were chosen because of the simple visual representation of the system they provide, and also because they provide easy access to any model variable. Designing a model with Simulink® also enables the programmer to visualize the program in ways that are helpful for the final experimental implementation. The simulation model is decomposed into blocks that can be separated or coupled in ways that can reflect the DSP program structure. Newer versions of Simulink® can also produce code from the model that is directly downloadable into a DSP, making software development much faster. The global model structure in Matlab/Simulink environment for this project is given in Figure 5.1.

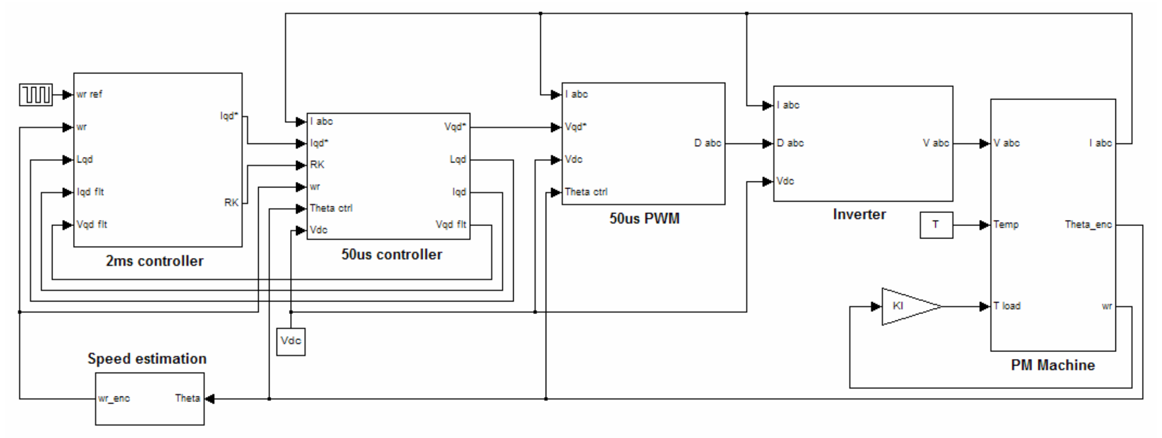

Fig. 5.1: Global controller model structure.

Figure 5.1 shows the decomposition of the model into important subsystems. These will be presented in details next.

5.1 Machine Model

The first step in creating a simulation model for the whole controller is to design a model for the machine behavior. This model is based on the equations given by the *d-q* transformation of a three phase PMSM. This part of the model calculates the phase currents, the machine torque, speed and rotor position for a given input voltages and load torque. It is also necessary for the purpose of this project to be able to have the machine parameters be functions of external or internal factors. The phase resistance and torque constant are consequently chosen to be linear functions of the temperature, and the *d*- and *q*-axes inductances were made functions of the machine currents. The temperature is an external variable, and for most of the experiments, it was set to be constant or a slowly increasing linear function of time.

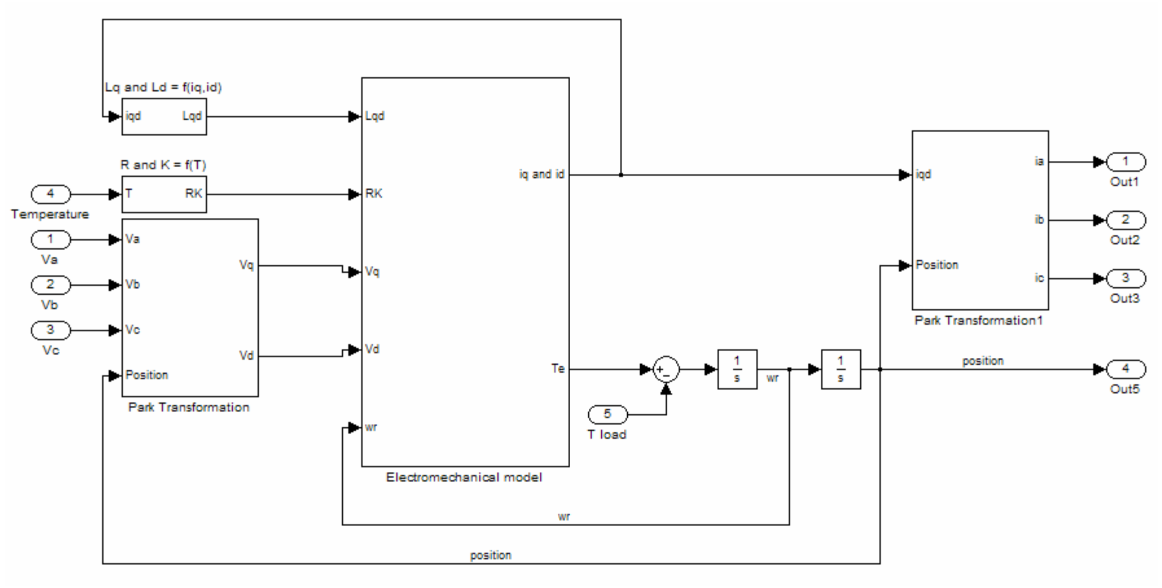

Fig. 5.2: Machine simulation model.

The machine model takes three phase voltages as its inputs, performs the Park transformation on them using the rotor position, and calculates the *d-q* input voltages. The electromechanical model block is the one that performs the integration of the *d-q* currents as a function all the inputs. Equations (2.6) were adapted for that purpose as:

$$
\begin{aligned}
i_q &= \int \frac{V_q - R \cdot i_q - \omega_e \cdot L_d \cdot i_d - K_T \cdot \omega_r}{L_q} \cdot dt \\
i_d &= \int \frac{V_d - R \cdot i_d + \omega_e \cdot L_q \cdot i_q}{L_d} \cdot dt
\end{aligned}
\tag{5.1}
$$

In numerical system modeling, it is usually preferable to avoid differentiations, which can cause instability and accuracy errors, and hence the electrical equations were written in an integral form. The currents are then used in the torque equation to calculate the machine electromechanical torque. Equation (2.5) was used for that purpose and is given here for the sake of clarity.

$$T_e = \frac{3 \cdot P}{2} \cdot \left[\varphi_{mag} \cdot i_q + \left(L_d - L_q\right) \cdot i_d \cdot i_q\right] \tag{2.5}$$

Both the torque and currents are outputs to this block. The currents are transformed back to the *abc* domain using the Park inverse transformation and are sent out, because they are measured in the experimental design and available to the experimental controller. The difference between machine and load torque is integrated to get machine speed (Equation (2.6)), which is then integrated to obtain the rotor position, neglecting rotational losses like bearing friction. This position is also used as an output because of the presence of a position sensor in the experimental design. On the other hand, the machine speed is not set to be an output, because there is no tachometer in the experimental design. The machine speed is estimated from the rotor position feedback information available in the controller. One can also notice the presence of blocks that calculate the values of the machine model parameters as functions of temperature and currents. The structure of the machine model is shown in Figure 5.2.

5.2 Inverter Model

5.2.1 Basic Inverter Operation

The power inverter is the element that converts the small power Pulse Width Modulation (PWM) outputs of the microcontroller into higher power signals that are used by the machine. It is also the source of non-idealities and discrepancies between the desired machine voltages and their actual value. The inverter is a device that is made of six switching cells, each of them being subdivided into two semiconductor switches. One of them is a controllable switch, which will turn on and off as a function of its input voltage. These controllable switches can be from different families, like the Insulated Gate Bipolar Transistors (IGBT), the Metal Oxide Semiconductor Field Effect

Transistors (MOSFET), or the Bipolar Junction Transistors (BJT). They also are usually limited in their operation to one voltage-current quadrant: they are unidirectional in both voltage and current. To overcome this limitation, common inverter designs attach a diode in parallel to the controllable switch. This diode is not controllable by any external signal, and its state will be dictated by the external circuit.

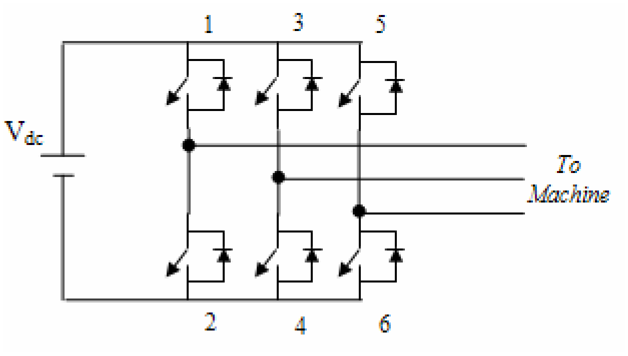

Fig. 5.3: Three phase bridge inverter.

Fig. 5.3 shows the circuit used to design a classical bridge inverter structure. The six switching cells (labeled 1 to 6) are arranged into three legs (1 and 2, 3 and 4, 5 and 6) and each of these legs is connected to a machine phase in the experimental design.

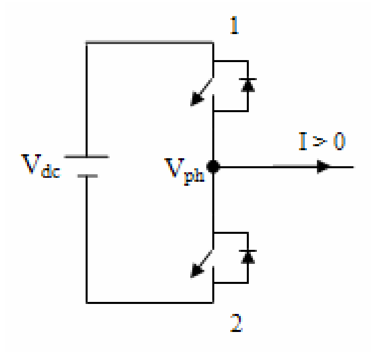

Fig. 5.4: Inverter leg.

Figure 5.4 can be used to describe the operation of an ideal inverter leg. The same analysis can be conducted on the complete three phase bridge inverter. If we assume the current labeled “I” has a return path through other parts of the circuit and that the inverter switches (transistor and diode) are ideal, the voltage V_{ph} at the output will take values as described in Table 5.1. It is necessary here to point out that in such an inverter leg, the top and bottom switching cells are never ordered to be “on” at the same time, since that would result in a source short circuit. In addition to this, in a typical PMSM application, a configuration where both switching cells are ordered “off” does not occur.

Table 5.1: Operation of ideal inverter leg.

Switch commanded “on”	**Sign(I)**	$\mathbf{V_{ph}}$
1	+	V_{dc}
1	-	V_{dc}
2	+	0
2	-	0

One can see from Table 5.1 that if the top cell is commanded to be on, the inverter leg will act as if the cell was a closed switch; the current sign (direction) only determines which switch (transistor or diode) is conducting. Likewise, when the bottom cell is commanded to be on, the inverter acts as if the bottom cell was closed.

5.2.2 Inverter Nonlinearities

A real inverter, like the one used in the experimental design associated with this project, differs to some extent in its operation from its ideal model. In most motor control applications, these differences do not affect the motor control much and they are ignored. However, in the present application, any error between the voltages desired in the controller and the voltages applied to the machine will have a direct impact on the parameter estimation performance. This impact will be the subject of a later chapter.

The main errors introduced by the inverter will be studied in detail in the following sections.

5.2.2.1 Deadtime

The switches that compose the inverter cannot change state instantaneously, and in order to avoid DC bus short circuits, it is necessary to introduce a deadtime in the PWM algorithm during which both switches in transition are ordered to be open. This deadtime is a delay that is introduced whenever there is a transition between commanding top and bottom switch conductions.

The simultaneous turn-on of both the switches in one leg of the inverter due to switch non-idealities is prevented by the insertion of this deadtime period, whenever a switch transition command comes from the PWM algorithm. In the experimental design, the DSP controls the deadtime period by ordering both the switches "off" for a short predetermined duration.

Figure 5.5 shows the transient associated with the turn-on of one of the experimental design IGBTs. The inverter circuit used in this project was a single integrated circuit, including both switches and gate drivers (part #IRAMX16UP60A). It is possible to see that it takes approximately 200 μs before the switch can be considered "on". Diode conduction transients and IGBT turn-off characteristics also present nonlinear responses. The inverter manufacturer recommends the introduction of a 300 μs deadtime for safe inverter operation which is fast for an IGBT based inverter. These types of inverters usually require deadtime delays ranging up to the order of a microsecond.

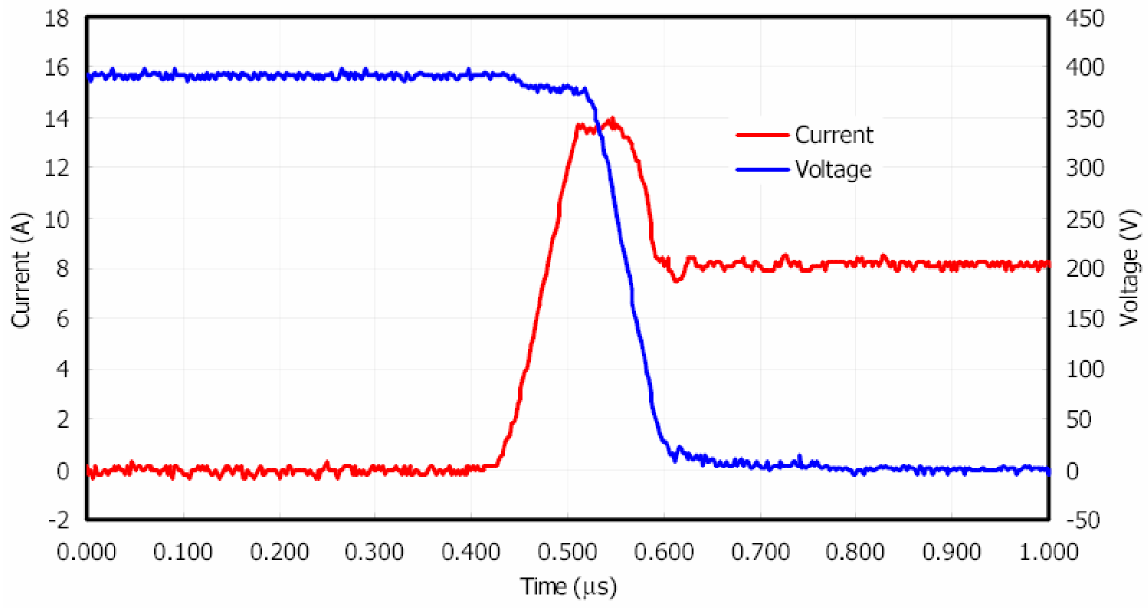

Fig. 5.5: Inverter IGBT turn-on transient (Junction Temperature = 150˚C) [24].

Figure 5.6 shows an example of deadtime insertion by the DSP. The ratio between PWM period and deadtime was made larger in the plots than in the experimental case to be able to visualize it. In the experimental design the deadtime is 160 times smaller than the PWM period T_{PWM}. The delay $t_2 – t_1$ is equal to $t_4 – t_3$ and they are both equal to the desired deadtime. It is also possible to see from Figure 5.6 that the time when the switch "1" is "on" is different from the time in the ideal case. This error is small in most cases but can become significant when the desired switch "on" time is small.

An important thing to notice for the effect of the deadtime delay is that its effect on the output voltage is a function of the phase current. If, in Figure 5.6, the phase current is sufficiently large and positive (flowing towards the machine), the diode at the bottom of the inverter leg will conduct during the deadtime. On the other hand, if the current is sufficiently large and negative, the top diode will conduct during the deadtime. Having a current that is sufficiently large in these cases means that the current is large enough to maintain its polarity throughout the PWM period.

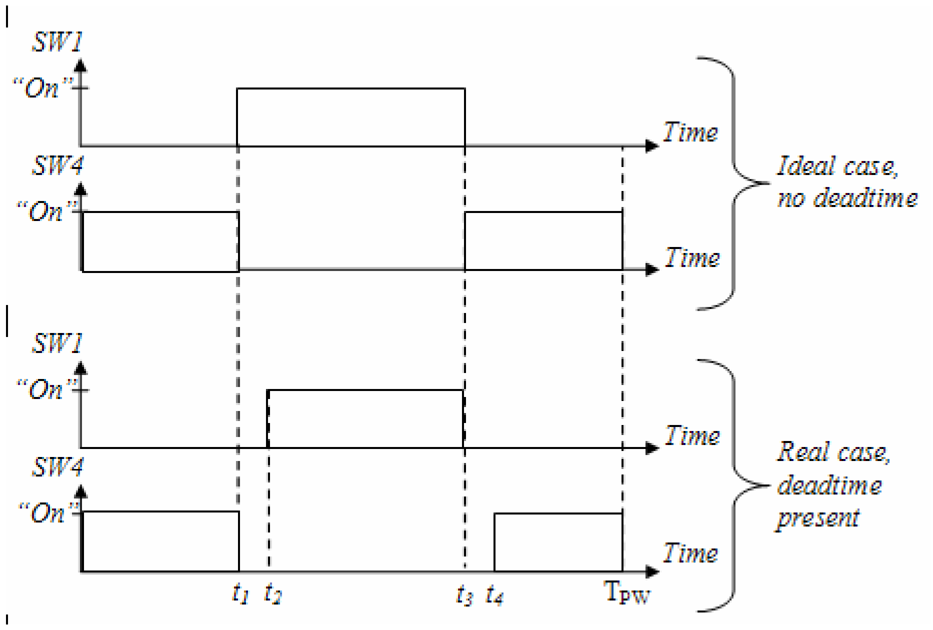

Fig. 5.6: DSP Deadtime insertion.

5.2.2.2 Switch Conduction Transients

Another problem introduced by the real inverter as opposed to the ideal one is also shown in Fig. 5.5. The switch transient behavior differs from an ideal one where it is assumed that the currents and voltages would instantaneously assume their new values. The values taken by the switch current and voltage during transient are also sources of discrepancies between the desired voltage in the controller and the voltage at the output of the inverter. This error is very small and would require extensive knowledge of the inverter switches for any type of compensation to be implemented. It would also require an extremely small simulation time step, which would lead to extremely long simulation times. It is consequently neglected in this project.

5.2.2.3 Switch Steady State Voltage Drops

In the experimental design for this project, the inverter transistors are IGBTs. This type of transistor presents a voltage drop when it conducts current, and so does the diode connected in parallel with it. These voltage drops change slightly the applied voltage to the machine from the ideal situation. The voltage drops introduced by the inverter switches are not constant, but are functions of the current flowing through the switch at a given time. The voltages applied to the motor phase windings after taking the non-idealities into account can be derived according to the logic outlined in Table 5.2. In the table, V_{IGBT} represents the voltage drop in the IGBT and V_D represents the voltage drop in the diode.

Table 5.2: Steady state operation of an actual inverter leg.

Switch commanded “on”	**Sign(I)**	$\mathbf{V_{ph}}$
1	+	$V_{dc} - V_{IGBT(I)}$
1	-	$V_{dc} + V_{D(I)}$
4	+	$0 - V_{D(I)}$
4	-	$0 + V_{IGBT(I)}$

The switch voltage characteristics were measured experimentally and are compiled in Figure 5.7.

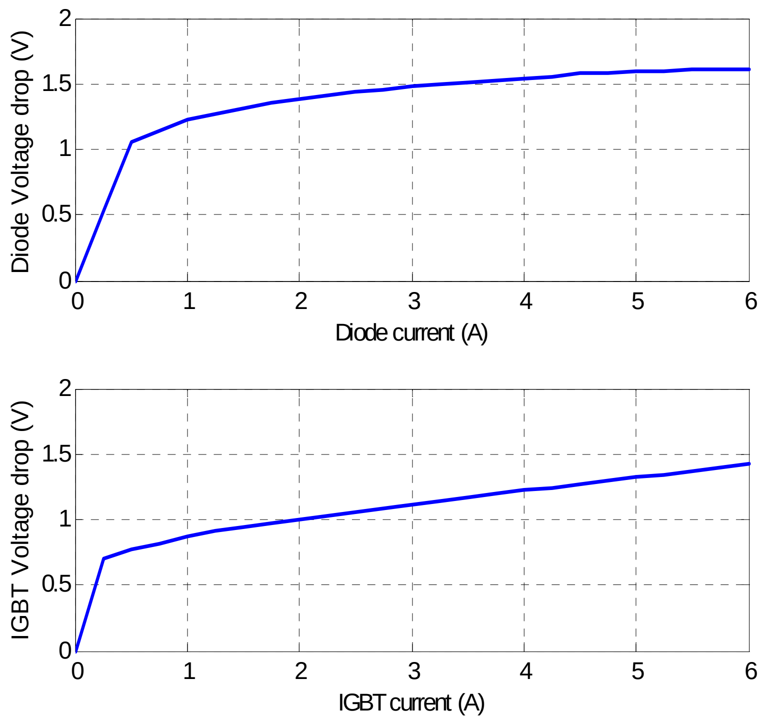

Fig. 5.7: Inverter switch voltage drops.

5.2.3 Simulation Model

The inverter model used for the simulation model for this project includes the deadtime delay generator block, and also the steady state switch characteristics shown in Figure 5.7. However, it does not include the switch transient behavior, because it would require a much longer simulation time for results that would not be affected noticeably. On the same note, deadtime was not always included in the simulations, because it was effectively compensated for in the experimental design, and it would also require a smaller than practical time step in order to be simulated effectively. The deadtime imposed in the experimental design was 300 ns, and the time step that was desired for the simulations was 400 ns, which is a compromise between simulation speed and result precision. From the point of view of the controller, the existence of the deadtime is transparent, since it was compensated for in the PWM block, for the controller the deadtime it was as if it did not exist. The switch voltage drops were also compensated for in the experimental design, and hence, were not included in simulation results shown in Chapter VI. Both the deadtime and switch voltage drops compensation algorithms will be explained in details in Chapter VII.

Initial simulations for this project were done with an ideal inverter because the effect of the inverter nonlinearities was assumed to be negligible. However, experiments showed that the errors introduced were quite large, sometimes even comparable to values of certain terms in the voltage equations that the estimation algorithms had to estimate.

The effect of the inverter non-linearities on the machine applied voltage were simulated using the developed simulation model. Figures 5.8 and 5.9 show the result of a low speed test over one mechanical revolution where the control algorithm fixed the *d-q* input voltages of the inverter. The DC bus voltage was set at 50 V, because the experimental results that were obtained for similar results were done with a voltage around this value.

Figure 5.8 shows the results obtained with an ideal inverter configuration, where the deadtime delay is set to zero, and where the switch voltage drops are also set to zero. The results seen in that figure are the ones that we would like to achieve in the experimental design, because they correspond to a case where the inverter is almost transparent from the point of view of the controller.

It is necessary to bear in mind however, that even an ideal inverter will introduce perturbations in the machine phase voltages, because of the lagging response of the

PWM, and because of the high frequency content of the square wave signals that are applied to the machine. In most cases however these affects can be neglected.

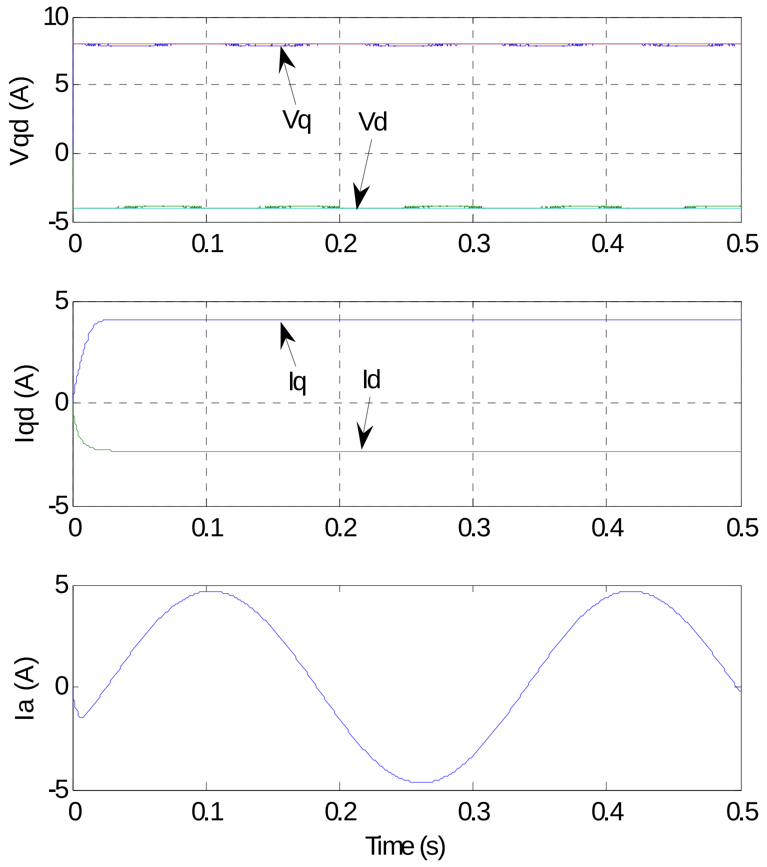

Fig. 5.8: Simulated ideal inverter response to constant *d-q* voltage inputs.

Figure 5.8 shows the *d-q* voltages and currents in the machine, as well as the phase “a” current. The *d-q* voltages were not directly accessible as a continuous signal because of the nature of the inverter output voltages. They were obtained by filtering the output of the Park transformation at the input of the machine model. The plot that shows the *d-q* voltages of the machine also shows the reference voltages at the input of the controller, which were constant at:

$$\begin{cases} V_q = 8V \\ V_d = -4V \end{cases}$$

However, it is difficult to distinguish them in Figure 5.8 because they match closely.

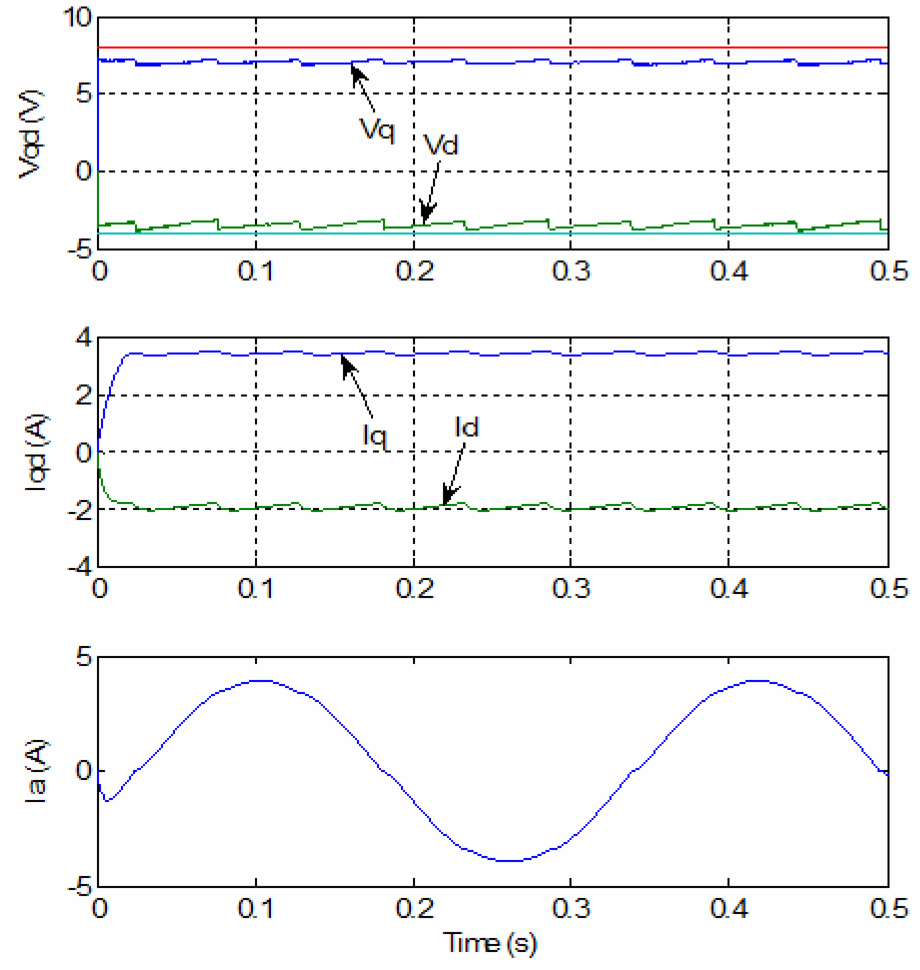

Fig. 5.9: Simulated real inverter response to constant *d-q* voltage inputs.

In Figure 5.9, the simulation model of the inverter included both the deadtime delay, and the steady state switch voltage drops. One can see significant changes on each of the three plots from the ideal case shown in Figure 5.8. In terms of the *d-q* voltages of the machine, it is possible to see both a distortion of the signals from their reference, and a change in their average level. Additional simulations were run in order to isolate the problems, and showed that most of the distortions were due to the deadtime, because the switch voltage drops tend to cause a constant drop in the *d-q* voltages. These results show a significant problem from the controller point of view: while the controller thinks it is sending the reference *d-q* voltages (V_d = -4V, V_q = 8V), the inverter is only applying (V_d = -3.5V, V_q = 7V), and the machine feedback is a response to this latter pair. This error, if not compensated or taken into account by the estimation algorithms, can have an enormous impact on the performance of the controller.

In the case of the *d-q* currents, the values obtained are slightly different from the ones in the ideal case, and the oscillations in the machine voltages have an impact on the current waveforms. Another way to look at the distortion caused in the current waveforms is to look directly at the phase currents; phase "a" current was shown in both figures 5.8 and 5.9. With the realistic inverter results, we can see some small distortions in the current waveform, mostly around zero current, from the waveform obtained in the ideal case, which was sinusoidal.

The distortion around zero comes from the fact that in the case of small currents, the effect of the deadtime on the controller voltage changes because the phase current get small enough to allow the diode to reach an "open" state or even change the conducting diode during the PWM period. On the other hand, when the current is sufficiently large, the effect of deadtime is easy to estimate because the diode remains in a known state of conduction. The effect of the deadtime for large currents is to introduce a voltage error whose polarity is a function of the sign of the current, because of the diode that conducts during the deadtime delay. This is not the case when the current gets smaller and the distortion comes from the change of behavior of the perturbation.

It is also necessary to note that whereas the voltage disturbance due to the switch voltage drops is a function of the current flowing through them, the deadtime effect, except for smaller currents, is a function of the switch duty cycles. Since the deadtime is a fixed delay, the ratio between the "on" time of the switch and the desired "on" time is a function of the latter. The impact of the deadtime delay for larger currents will be felt mostly in the case of smaller duty cycles.

A series of tests was done in order to estimate and model the error in the controller voltage introduced by the inverter. The experimental approach chosen was designed to use the experimental machine in the simplest way. The machine was connected to the inverter and the controller applied different sets of constant *q*-axis input voltages (the *d*-axis voltage remaining at zero). The machine rotor was locked and current measurements were taken in steady state. In these conditions, the machine electrical model reduces to:

$$\begin{cases} V_q = r_s \cdot i_q \\ V_d = r_s \cdot i_d \end{cases}$$

The tests were conducted with the machine at room temperature and were completed within a short time, so that we could assume the machine temperature did not change significantly. As a result, it was possible to assume the stator resistance remained constant at a value that was measured offline. The error introduced by the inverter was then estimated by taking the difference between the controller voltage and the estimated resistive drop.

The results shown in the lower part of Figure 5.10 (Figure 5.10(b)) show that the error introduced by the inverter for the locked position and the type of voltage used in the experiment is pretty much constant. Theoretically, there should be a small increase in the error as the current gets larger with the switch characteristics shown in Figure 5.7, but the noise level was too high for it to appear clearly. These results show, however, that the error introduced by the inverter is quite significant and requires compensation. A similar plot was obtained in the case of the *d*-axis.

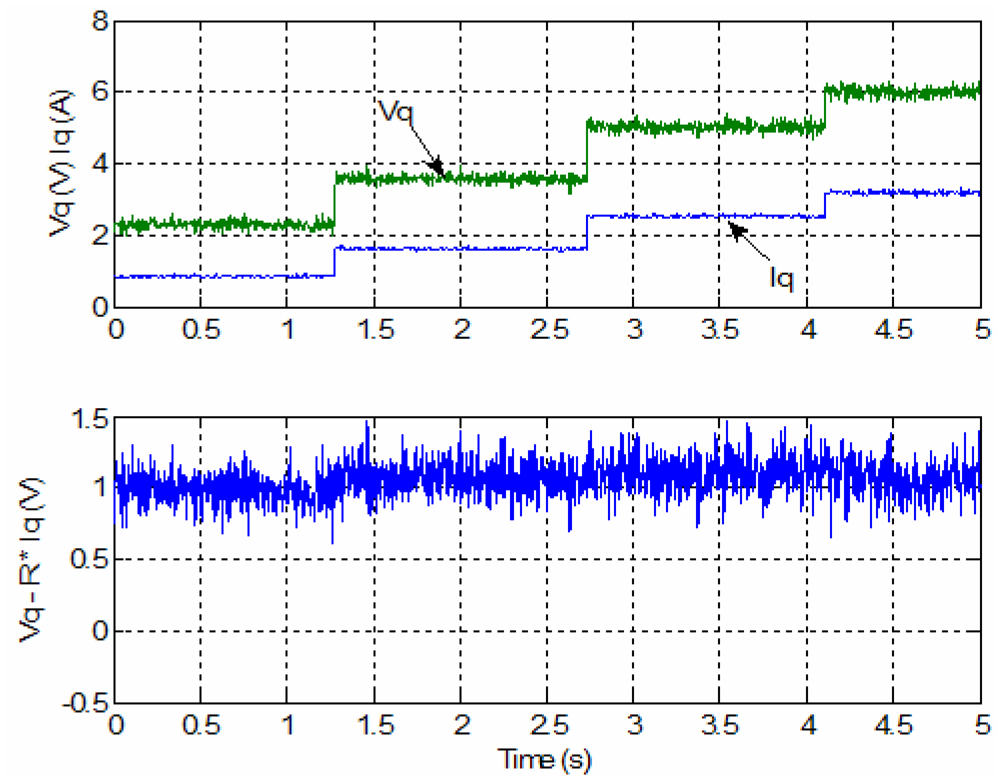

Fig. 5.10: Experimental inverter response with locked rotor.

5.3 PWM Algorithm

The PWM algorithm block performs the transformation from the desired *d*-*q* machine voltages to the inverter switch duty cycles. The duty cycle for a power electronics switch is the ratio between its "on" time and the switching period. This is the first block of the simulation model that is part of the machine controller, which is not run continuously at the simulation time step. In the experimental design, the various segments of the control program are either run periodically or on the occurrence of certain events. The controller simulation model is executed at a frequency matching the experimental design, in order to represent the real situation more accurately. In the project, the PWM frequency was set at 20 kHz, and the simulated PWM was executed at the same rate. For comparison, the simulation frequency (inverse of time step), which is used by the machine and inverter blocks as “real time”, was 2.5 MHz. In addition to being more realistic, having portions of the simulation model execute at lower frequencies allows the complete model to run much faster than it would if everything was executed at each time step.

The inverse Park transformation is the first step in the implementation of the conversion *d-q* reference voltages into duty cycles. The transformation is obtained from equation (2.3):

$$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} = \begin{bmatrix} -\sin(\theta_r) & \cos(\theta_r) \\ -\sin(\theta_r - 2\cdot\pi/3) & \cos(\theta_r - 2\cdot\pi/3) \\ -\sin(\theta_r + 2\cdot\pi/3) & \cos(\theta_r + 2\cdot\pi/3) \end{bmatrix} \cdot \begin{bmatrix} V_q \\ V_d \end{bmatrix} \tag{5.2}$$

The next step is to scale the three phase voltages in order to obtain the correct duty cycles at the inverter input. The solution chosen for this project consists in scaling the three desired phase voltages with respect to the available DC voltage at the source of the inverter, and then creating a virtual zero for a duty cycle of 50%. As a consequence, negative voltages will result in duty cycles that are smaller than 50%, and positive voltages give duty cycles larger than this value. There are two immediate consequences of using this algorithm: First, a strong homopolar voltage component is applied to the machine stator (which does not affect operation or performance), and the second is that the maximum achievable phase voltages are half the DC bus voltage. On the other hand, using this method makes it easier to compensate switch non-idealities.

Another method was also considered, which was based on the use of the smallest of the three phase voltages as a fictitious reference and expresses the other two phase voltages as differences between their initial values and the new reference. This algorithm uses phase to phase voltages instead of individual phase voltages.

Let us see on an example how both these algorithms would operate on a set of input voltages to obtain the duty cycles d_{abc}.

For

$$\begin{cases} V_a = 10V \\ V_b = 5V \\ V_c = -15V \end{cases} \text{ and } V_{DC} = 100V$$

The first algorithm gives

$$\begin{cases} d_a = (0.5 + V_a / V_{DC}) \\ d_b = (0.5 + V_b / V_{DC}) \\ d_c = (0.5 - V_c / V_{DC}) \end{cases} \text{ and finally } \begin{cases} d_a = (0.5 + 10/100) = 60\% \\ d_b = (0.5 + 5/100) = 55\% \\ d_c = (0.5 - 15/100) = 35\% \end{cases}$$

And the second PWM algorithm gives

$$V_{\min} = V_c = -15V$$

$$\begin{cases} d_a = (V_a - V_{\min})/V_{DC} = (10 - (-15))/100 = 25\% \\ d_a = (V_b - V_{\min})/V_{DC} = (5 - (-15))/100 = 20\% \\ d_a = (V_c - V_{\min})/V_{DC} = (-15 - (-15))/100 = 0\% \end{cases}$$

The immediate advantage that one can see in using the second algorithm is that it always keeps one of the three phases at a zero duty cycle, which means the corresponding phase voltage remains unchanged for the PWM period. This helps minimizing the switching losses in the inverter. The switching losses are a consequence of the non-ideal behavior of the inverter switches depicted in Figure 5.5: during a switching transient, the product between voltage and current in the switch becomes nonzero and translates into heat losses. By switching only two phases out of the three, one reduces significantly these losses. On the other hand, maintaining a phase at a zero duty cycle makes it impossible for the voltage compensation algorithm to operate optimally.

The first algorithm has been chosen, since adequation between machine model and machine response is one of the primary concerns of this project.

The inverter compensation algorithms have been implemented inside the PWM model block to achieve a proper match between the controller voltages and the inverter outputs.

5.4 Current Controller

The current controller model block calculates the *d-q* voltages required to obtain the desired levels in *d-q* currents and is similar to the experimental DSP program. This segment of the program is executed at the PWM frequency of 20 kHz which is the same as in the PWM block.

The first step in achieving current control is to have adequate feedback. The currents that are measured in the experimental design are not the *d-q* currents, but usually two of the *abc* phase currents. It is therefore necessary to perform the Park transformation on them in order to obtain the corresponding *d-q* currents. We can use equations (2.2) for this purpose:

$$\begin{bmatrix} i_q \\ i_d \end{bmatrix} = \frac{2}{3} \cdot \begin{bmatrix} -\sin(\theta_r) & -\sin(\theta_r - 2\cdot\pi/3) & -\sin(\theta_r + 2\cdot\pi/3) \\ \cos(\theta_r) & \cos(\theta_r - 2\cdot\pi/3) & \cos(\theta_r + 2\cdot\pi/3) \end{bmatrix} \cdot \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{5.3}$$

The simulation model also incorporates the precision of the Analog-to-Digital Converter (ADC) of the DSP at this point, which is 12 bits in theory. However, measurement noise and the real precision of the ADC reduced the available resolution to about 9-10 bits. This effect was not simulated except through a decrease in simulated ADC resolution.

The next task for the current controller is to extract the command voltages required to establish the desired machine currents. The controller algorithm uses the machine electrical model along with the current feedbacks.

The transfer functions that describe the operation of the current controller with feedforward have already been presented and are repeated here for the sake of convenience:

$$\begin{cases} \dfrac{i_q}{V_q^{'}} = \dfrac{1}{r_s + p \cdot L_q} \\ \dfrac{i_d}{V_d^{'}} = \dfrac{1}{r_s + p \cdot L_d} \end{cases}$$

where

$$\begin{cases} V_q^{'} = V_q - \omega_e \cdot L_d \cdot i_d - \omega_r \cdot K_T = r_s \cdot i_q + p \cdot L_q \cdot i_q \\ V_d^{'} = V_d + \omega_e \cdot L_q \cdot i_q = r_s \cdot i_d + p \cdot L_d \cdot i_d \end{cases}$$

The error between the *d-q* reference currents and the feedback is sent to a Proportional and Integral (PI) controller, whose output is shifted by the feedforward terms so that the system is linear from the controller's perspective.

5.5 Speed Estimation

The machine speed is an important variable in the program, but is not readily available to the controller. The position signal given by the optical encoder needs to be integrated to obtain the speed information. This sensor provides a relative rotor position with a precision of 4096 counts per mechanical revolution, or 2048 counts per electrical revolution, since the machine has four rotor poles. To obtain the motor speed, the program subtracts the current position with the one it had during its last execution, and applies a low-pass filter to the result. The filter pole is adjusted to get a compromise between ripple and response speed.

5.6 Fast Identification Algorithm

Another program that is executed at the sampling rate is the fast parameter estimation algorithm. The algorithm uses the *d-q* current feedback calculated by the Park transformation and the output voltages sent to the inverter to estimate the *d-q* machine inductances. Two other parameters, K_T and r_s are required in this fast algorithm. The operation of this algorithm and the equations on which it is based have already been presented in Chapter IV. The fast identification algorithm is included in the current control block of the global simulation model for convenience.

5.7 Low Frequency Controller

The last block that is a part of both the simulation model and the experimental program runs non time-critical tasks and executes at a slower rate than the sampling frequency. The cycle frequency for this block was chosen to be 2 kHz.

The speed controller has been included in this block, because the mechanical dynamics are much slower than electrical dynamics. The speed controller uses a PI compensator on the error between the estimated speed feedback and the desired reference speed, giving the reference magnitude of the stator current. This magnitude is then converted into the *d-q* currents by using either the maximum Torque per Ampere algorithm, or the zero *d*-axis current algorithm, both of which were presented in Chapter II. It is to be noted that the speed controller is to be used only when the application demands it. Other cases exist where the parameter to be controlled is the machine torque or rotor position.

Another program that was executed at the lower frequency is the slow identification algorithm, which is mostly used to track the variations in the stator resistance and the machine torque constant. The equations and operation of this slow identification algorithm were discussed in Chapter IV.

5.8 Conclusion

This chapter has introduced the reader with the simulation model developed for this project. The various blocks of the model, and the constraints present and necessary to be modeled were discussed. Chapter VI will focus on how this simulation model was used to develop and verify the parameter estimation algorithm in PMSM that is the primary objective of this research.

CHAPTER VI

PARAMETER ESTIMATION SIMULATION RESULTS

The first step in validating the proposed algorithm was to verify its effectiveness using the simulation model presented in the previous chapter. For this purpose, the algorithm was tested on two different machine models. One of these models matches the motor that was used experimentally for this project. A problem that was encountered with this machine is that it does not show significant saturation characteristics, and as a consequence it could not be used to show some of the results that motivated this research. The second motor model chosen displayed significant inductance saturation at higher torque levels; it was obtained from the motor data used in [19]. This latter motor was selected for simulation to show the possible effects of saturation on the maximum Torque per Ampere algorithm in Chapter III.

6.1 Machine Models

In this section the machine model parameters that were used for simulation will be presented. The parameters of the two machines are shown in Table 6.1. The machine "A" has been used experimentally to demonstrate the functionality of the developed algorithm. Parameter variations due to temperature were introduced as variations from the nominal values given in Table 6.1.

Parameter variation due to saturation was introduced by changing the *d-q* inductance values in the program as a function of the machine currents.

Table 6.1: Simulation model parameters.

Parameter name	Machine "A"	Machine "B"
Stator resistance, r_s (Ω)	1.55	1.45
Torque constant, K_T (V.s)	0.207	0.172
L_q (mH) no saturation	9.6	18
L_d (mH) no saturation	5.1	6
Rotor inertia, J (N.m.s^2)	46.1e-6	99.6e-6
Number of magnet pole pairs, P	2	2
Rated current, I_{max} (A)	5	25

6.1.1 Machine "A" Inductances

In the case of the experimental machine, the inductance waveforms were measured offline by taking steady state measurements for a variety of operating points and extracting the corresponding inductance values from the model equations. These tests were done with the machine at room temperature so that the resistance and torque constant could be assumed to be at their nominal values: These parameter values were also measured with other tests. The phase resistance was measured directly with a multimeter, and the torque constant was obtained from an open-circuit test described in section 7.2.3 of Chapter VII. The mechanical load in the experimental design was a controlled brake, that was not able to produce motoring torque on the motor shaft. Because of this it was not possible to do the tests with the machine *q*-axis current set to zero since they would not produce any motoring torque from the machine, and the motor speed would be zero. A non-zero speed is required for the inductance terms to appear in the steady state model equations. The *q*-axis current is required to get motoring torque (see equation (2.5)).

For a given electrical operating point, the machine was run at a constant speed in steady state and the controller recorded a large number of data points (compensated command voltages and measured currents). An average of these points was done in order to improve precision, and the steady state model of the machine was used to extract the machine inductances from equation (2.17), which is stated again below for convenience:

$$\begin{cases} V_q = r_s \cdot i_q + K_T \cdot \omega_r + L_d \cdot \omega_e \cdot i_d \\ V_d = r_s \cdot i_d - L_q \cdot \omega_e \cdot i_q \end{cases} \tag{2.17}$$

Which gives

$$\begin{cases} L_q = \dfrac{r_s \cdot i_d - V_d}{\omega_e \cdot i_q} \\ L_d = \dfrac{V_q - r_s \cdot i_q - K_T \cdot \omega_r}{\omega_e \cdot i_d} \end{cases}.$$

Results were only obtained for a negative *d*-axis and a positive *q*-axis current because positive *d*-axis currents are never used in practice for motoring applications (they generate negative torque) and the quadrant that corresponds to the negative *q*-axis current is symmetrical to the positive side. The values of the inductance for the applied *d*-*q* axes currents are given in Tables 6.2 and 6.3.

Table 6.2: Machine “A” *q*-axis inductance in mH.

***i*$_q$ \ -*i*$_d$ (A)**	0	1	1.5	2	2.5	3
0	**9.6**	**9.9**	**10.13**	**10.48**	**10.89**	**11.6**
0.75	**9.6**	**9.9**	**10.13**	**10.48**	**10.89**	**11.6**
1.25	9.6	**9.9**	**10.13**	**10.48**	**10.89**	11.46
1.8	9.5	9.9	10.13	10.48	10.89	11.34
2.35	9.4	9.88	10.1	10.43	10.78	**11.34**
2.9	9.15	9.78	10	**10.43**	**10.78**	**11.34**

Table 6.3: Machine “A” *d*-axis inductance in mH.

***i*$_q$ \ -*i*$_d$ (A)**	1	1.5	2	2.5	3
0.75	**5.87**	6.68	6.89	6.95	7.11
1.25	5.87	5.89	6.4	6.69	6.91
1.8	5.56	5.57	6.12	6.44	6.57
2.35	5.35	5.55	5.61	5.96	**6.57**
2.9	5.27	5.46	**5.61**	**5.96**	**6.57**

In tables 6.2 and 6.3, the values in bold correspond to inductance values that were extrapolated (clamped) from neighboring results. The reason for extrapolation is that for small values of current for both the *q*- and *d*- axes, the inductance term is too small to overcome the feedback noise. Other measurements that correspond to the largest values of *d*- and *q*- axes currents were not obtained because they were too close to the machine ratings. The corresponding surface plots are shown in figures 6.1 and 6.2.

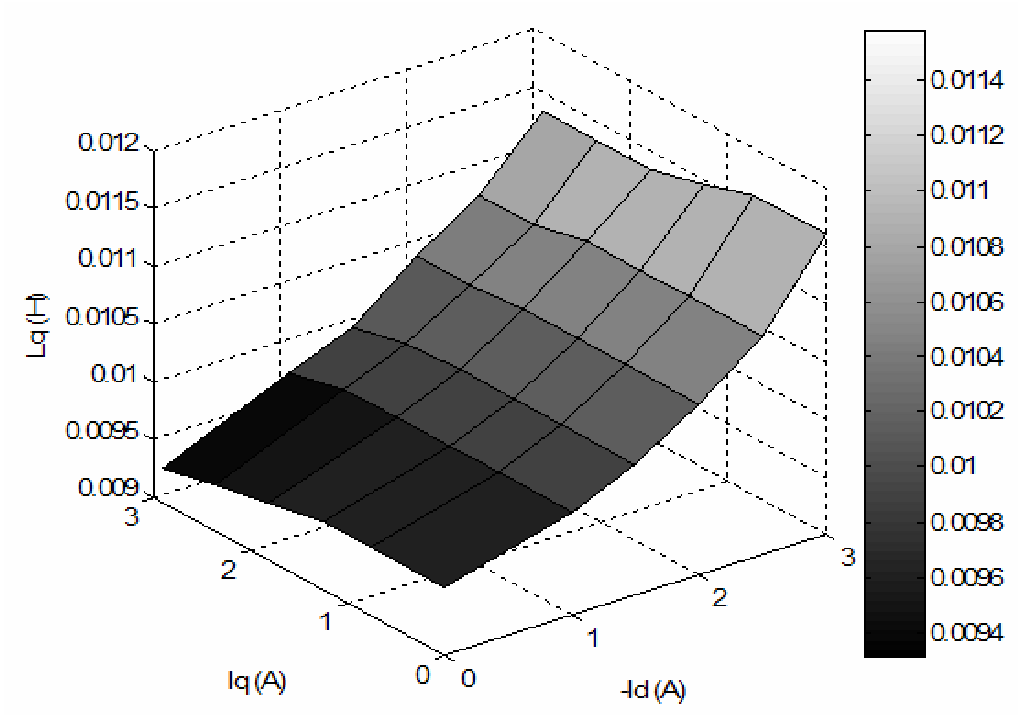

Fig. 6.1: Machine “A” *q*-axis inductance.

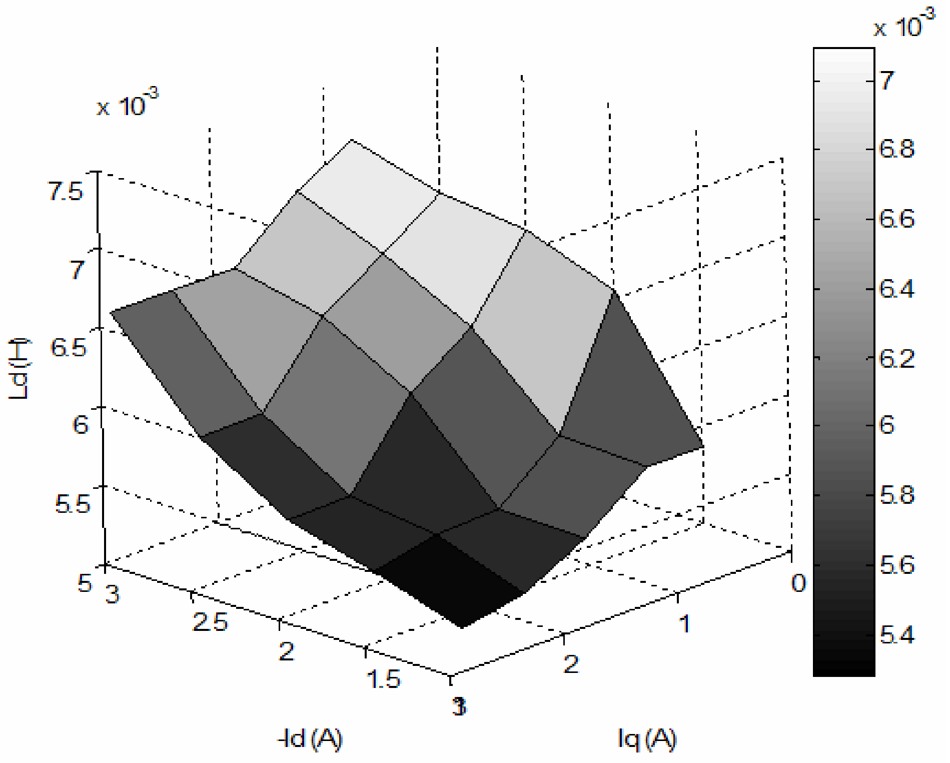

Fig. 6.2: Machine “A” *d*-axis inductance.

The measurements done to get the *d*- and *q*-axes inductances show the presence of cross-coupling between the two axes. Each of the two inductances is a function of both the *q*-axis and the *d*-axis currents.

For a constant *d*-axis current, the *q*-axis inductance is decreasing as a function of the *q*-axis current because of magnetic saturation in the corresponding iron path. In the case of a constant *q*-axis current, the *q*-axis inductance is increasing as a function of the *d*-axis current magnitude. The *d*-axis flux works against the magnet flux as the *d*-axis current becomes negative, thereby reducing the global amount of flux in the machine and saturation, and hence the inductance increases. This dependency of the *q*-axis inductance on *d*-axis current is referred to as cross-coupling phenomenon.

The *d*-axis inductance plots can also be explained similarly accounting for the cross-coupling effects. Increasing the *d*-axis current magnitude reduces the *d*-axis flux for a constant *q*-axis current, and consequently the corresponding inductance gets closer to its linear region value and increases in value. Reducing the magnitude of the *q*-axis current for a constant *d*-axis current reduces the *q*-axis flux and the cross coupling flux, which increases the *d*-axis inductance.

6.1.2 Machine “B” Inductances

In the case of machine “B”, the data available provided only the *q*-axis inductance as a function of the *q*-axis current. The *d*-axis inductance waveform was provided only for positive *d*-axis currents, which have not been used in this project. The author of the paper [19] stated that the *d*-axis inductance did not vary much and we made the assumption that the *d*-axis inductance was constant at the value given for a zero *d*-axis current.

The inductance characteristics of machine “B” are shown in Chapter III, in Figure 3.5. This machine shows a more significant saliency ratio than machine “A”. As a consequence, the saliency torque producing term will be more important to the controller and the machine will be more suitable for high speed operation. However, this motor also shows a more significant saturation characteristic than the other one and a consequence of this is that motor drive performance may get degraded if it is not taken into account. Figure 3.2 also shows a region, where the *q*-axis inductance could get smaller than the *d*-axis inductance, making the reluctance torque zero or even negative.

6.2 Novel Parameter Estimation Algorithm

6.2.1 Algorithm Structure

The novel parameter estimation algorithm proposed for this research has a modular structure. The algorithm can be divided into three main components:

- The first one is the “fast” identification algorithm. It is a part of the program that runs continuously and provides estimates of the machine inductances from the other two parameters and the machine inputs and feedback. This program is non-intrusive, and could also be used independently of the rest of the program if it were provided with estimates of the stator resistance and the torque constant.
- The second part is the one referred to as the “slow” identification algorithm. As opposed to the previous program, this one is more demanding in terms of inputs. The primary factor for its effective execution is that it needs to be supplied with

data that corresponds to steady state operating points. An error in its inputs could have important repercussions considering its slow rate of execution. Another constraint associated with this program is the necessity to provide it with data that is “rich” enough with information about the system. The data for the slow algorithm was enriched with a small perturbation signal was introduced on the *d*-axis. This program can therefore be seen as intrusive in the controller operation. However, considering the very slow rate of change of the stator resistance and torque constant, it is conceivable to disable the algorithm for long periods of time and only activate it when new estimates for these two parameters are required.

- The third and last component of the proposed parameter estimation algorithm is one that controls the interactions between the first two programs and also between the motor drive and them. This block detects when the motor drive is operating in steady state and enables or disables the “slow” identification program. It also controls the times at which the “fast” program will update the parameters in the “slow” one. This block also generates the small *d*-axis perturbation used by the “slow” algorithm. Figure 6.3 shows a clearer picture of the interaction between the three blocks.

Figure 6.3 shows a typical machine startup operation with machine “B”. The first plot shows the machine *d*-*q* currents as a function of time for a period of one second. The *q*-axis current is set at a constant value, which is the case in most current or speed control applications in steady state. The *d*-axis current, on the other hand, shows small step variations around the set point. These are the consequence of the small perturbation injected in the *d*-axis of the controller. The second plot shows a closer look on the *d*-axis current, and also shows the signal that enables or disables the "slow" estimation algorithm. When this signal is at "1", the "slow" algorithm is enabled and will run continuously, though at a slow rate.

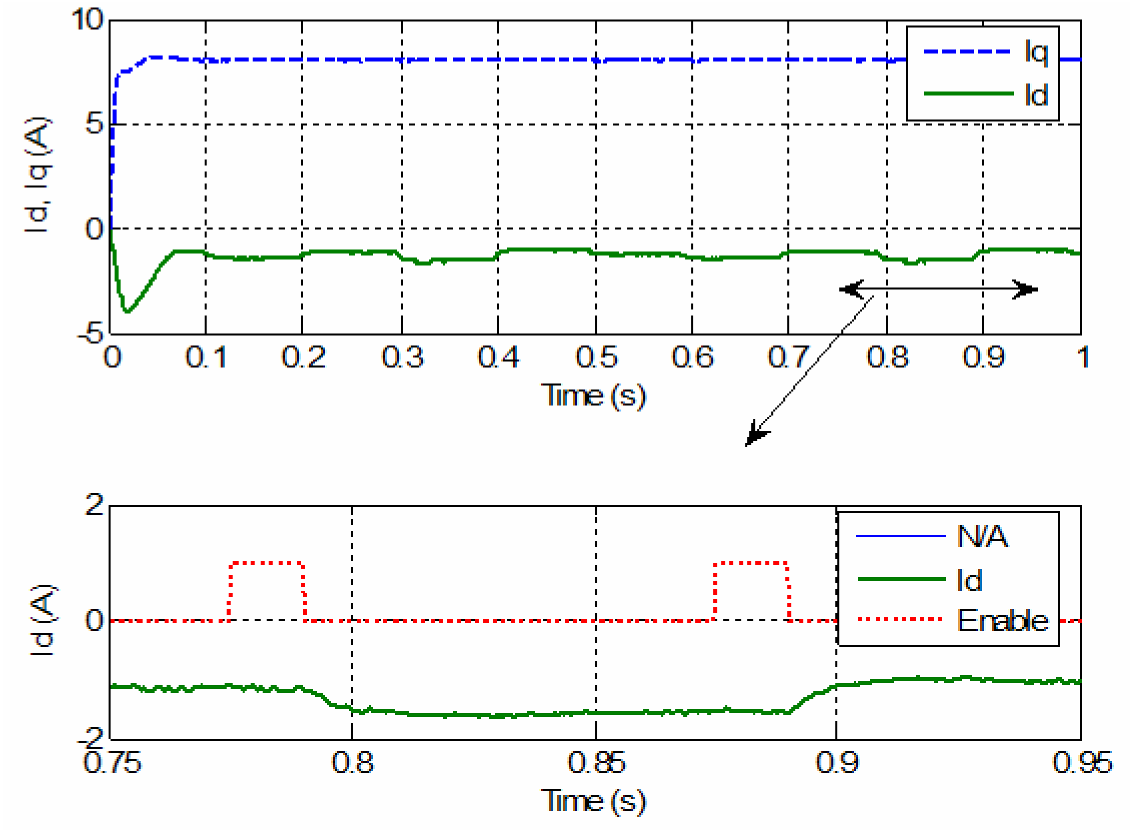

Fig. 6.3: Example of interaction between elements of algorithm.

It can be observed that this "Enable" signal does not really correspond to times where the machine is in steady state. For example, between the time "0.82" and the time at which the enable signal turns to "1", the machine currents can be considered constant and the machine could be considered to be in steady state. The reason for this is that enabling the "slow" algorithm for a too long period of time on the information of a single operating point will decrease the "richness" of its input and may lead to instability of the algorithm. In this example, the “slow” algorithm is executed about 10 times during an “enable” period (for a 500 Hz execution loop).

6.2.2 Supervising Program

The program that decides whether or not to enable the “slow” identification program operates by using both the current references in the controller and the current feedback. In all cases, if the current feedback shows a change in the *d*-*q* currents that is larger than a threshold, the algorithm will be disabled. The advantage in using the current references is that they usually lead the actual machine currents and allow the controller to disable the “slow” algorithm before it is too late. Another advantage is that the perturbation that is used in the *d*-axis is known and its transitions can be monitored precisely. Another feature that is included in this portion of the program is the introduction of a delay when the disabling of the parameter estimation algorithm is triggered by the reference current. This delay allows the user to control the window during which the identification program is active. The appropriate length of this window and the frequency of the *d*-axis perturbation will vary from motor drive to motor drive, mainly as a function of the RLS forgetting factor.

Another feature of the supervising program concerns the monitoring of times when the parameter estimates coming from the fast algorithm are copied into the slow one. This aspect was introduced to solve the following problem: when there is a large change in the operating point of the machine, the inductances may suddenly vary depending on the amount of saturation. The “fast” algorithm will follow accurately this possible change, and the “slow” algorithm is disabled during that time. However, when the latter program resumes its operation, a large error would have been introduced in its estimation and the program will have to reconverge, possibly causing transient errors in the other parameters. Such transients are usually fast compared to the mechanical and temperature dynamics in the machine. It is consequently a valid assumption to assume the stator resistance and the torque constant do not change during a current transient. The "fast" algorithm should operate properly given the set of machine parameters at the beginning of the transient and for its duration.

On the other hand, from the "slow" algorithm's point of view, its parameter estimates after resuming operation should be a closer match than if its execution had not been disabled. Combining the two algorithms in the way described above allows the parameter estimation algorithm to track fast inductance changes without affecting the estimation of the other two machine parameters.

An example of the operation during an inductance step change with machine "B" can be seen in Figure 6.4. In this figure, the top and middle plots show the *q*-axis inductance and its controller estimate. The top plot shows a larger time scale than the middle, which focuses on a step change in the inductance, which was itself caused by a step change in the *q*-axis current from 12 A to 8 A and back. In the middle plot, the machine inductance matches the controller "fast" estimate, labeled "Lqf", and the "slow" algorithm estimate is labeled "Lqs". This nomenclature will appear throughout this chapter and is also used for the *d*-axis inductances.

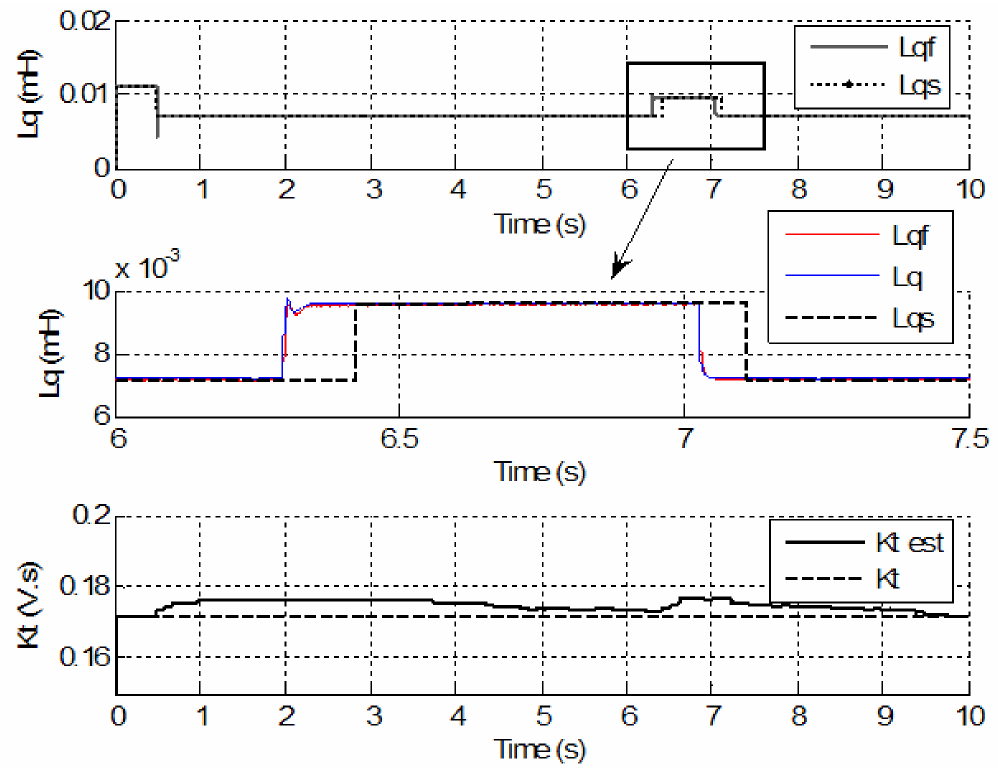

Fig. 6.4: Simulation example of inductance step change.

The bottom plot of Fig. 6.4 shows the machine torque constant (“Kt”) and its estimate, labeled “Kt est”. This plot demonstrates that the estimation of the torque constant is not affected by the transient. The operation of the supervising algorithm can be seen from the middle plot. At the beginning of the transient, the “slow” parameter estimation is disabled and the corresponding parameter estimates are “frozen”. The “fast” estimation algorithm follows the variation in inductance effectively, and at the end of the transition (6.4s), the “slow” algorithm is enabled again and uses the new “fast” inductance estimates instead of the values it was using before the transient.

For comparison, Figure 6.5 shows simulation for the same conditions as in Figure 6.4, but with the “fast” inductance copy to the slow algorithm disabled. Before the transient, the results in both figures match closely. When the inductance step occurs, the “slow” identification algorithm is disabled and operation is the same as in the previous case, but as soon as the algorithm resumes its estimation (at about 6.4s), the error caused by the mismatch between the “slow” algorithm inductance estimate and the correct estimate provided by the fast algorithm causes a large error in the torque constant estimate. It can also be observed that the error in the resistance and torque constant estimates affects the “fast” algorithm by looking at the “fast” estimate after the problem starts. The algorithms are able to converge after some time, but such an operation and parameter error has adverse effects on the torque and current controllers.

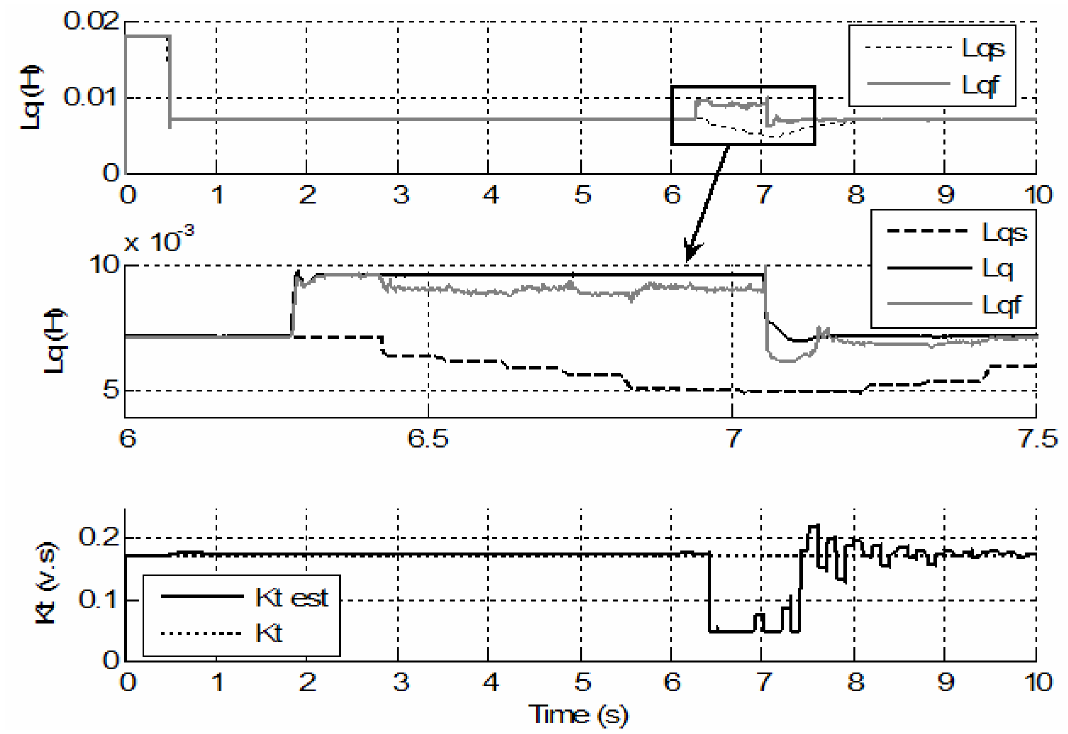

Fig. 6.5: Inductance step change without “fast” inductance copy.

6.3 Effects of Neglecting Differential Terms

It has been mentioned in Chapter IV that the “fast” RLS algorithm implemented in the experimental design associated with this project does not include the differential terms from the electrical model equations.

These terms are of the following form:

$$A = L_q \cdot \frac{di_q}{dt} \text{ and } B = L_d \cdot \frac{di_d}{dt}$$

These terms are present on both the *d*- and *q*-axes. This omission is equivalent to making the assumption that the algorithm is executed at steady state. However, whether it is caused by the *d*-axis perturbation or a fast change in operating point, the "fast" identification algorithm will at times be executed in a case where the differential terms are not zero.

The effect of such an omission is presented in Figure 6.6, where results from the "fast" algorithm are obtained in the case where the differential terms are neglected and in the case where they are not. These terms could easily be accounted for with a different and improved experimental setup, but in the present implementation the noise level increases the errors in estimation when these terms are included.

The conclusion that can be drawn from Figure 6.6 is that the "fast" algorithm is affected by the exclusion of the differential terms. The small pikes observed in the estimated *q*-axis inductance show this. On the other hand, as soon as the transient ends, the identification algorithm converges again to the same value as in the case where the differential terms were included. Special care should be taken during transient operation and when results are copied from the "fast" algorithm into the "slow" one in the case of an implementation without the differential terms. The effect of differential terms could also be low-pass filtered to extract the desired signal.

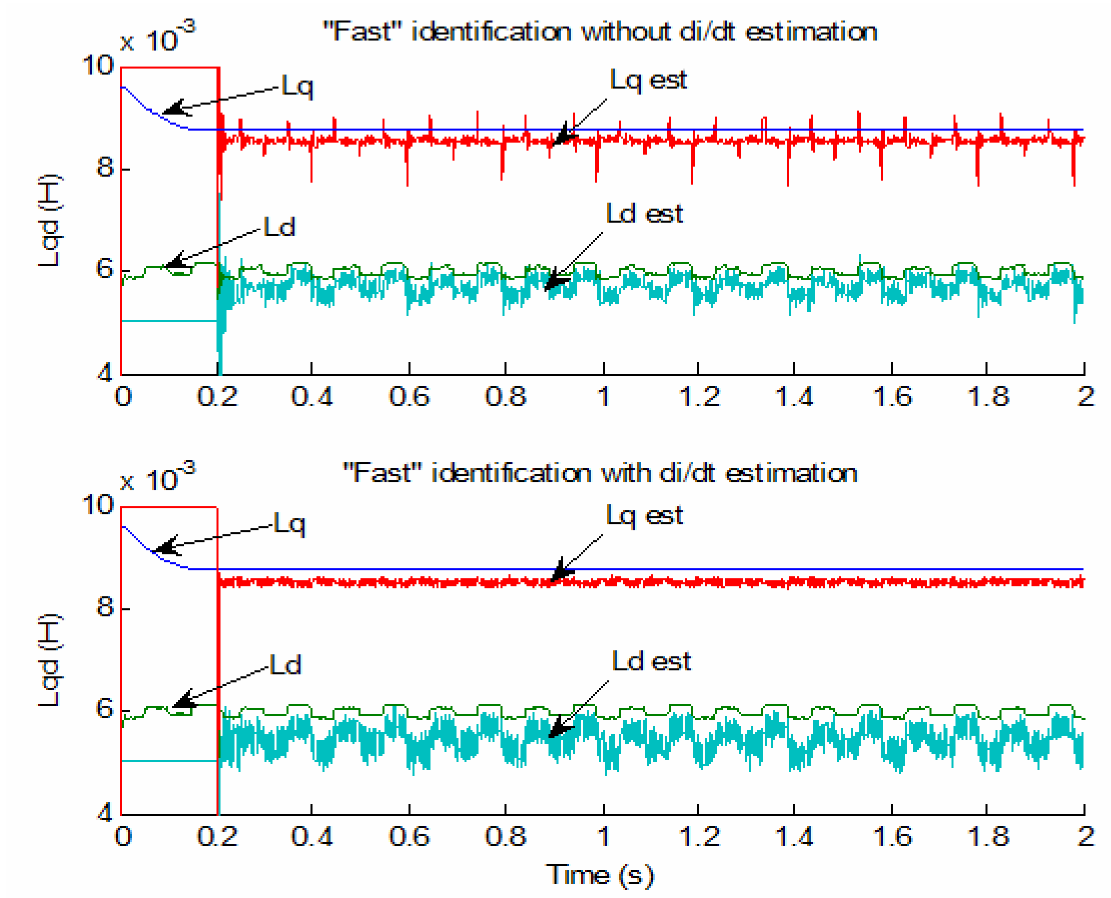

Fig. 6.6: Comparison between inclusion and exclusion of differential term.

6.4 Sensitivity Analysis of "Fast" Algorithm Estimates

The performance of the fast identification algorithm is discussed in this section. In contrast to the "slow" algorithm, if the fast algorithm is provided with accurate estimates of the stator resistance and the torque constant, it will be able to provide accurate estimates of the machine inductances. The analysis in this section is presented with the assumption that there are no errors in the machine currents. This is a valid assumption since in the experimental design the currents are directly measured and except for their average signal to noise ratio, these measurements are accurate.

Let us consider the machine electrical equations in steady state:

$$\begin{cases} V_q = r_s \cdot i_q + K_T \cdot \omega_r + L_d \cdot \omega_e \cdot i_d \\ V_d = r_s \cdot i_d - L_q \cdot \omega_e \cdot i_q \end{cases}$$

The following expressions can be extracted from the above equation

$$\begin{cases} L_q = \dfrac{r_s \cdot i_d - V_d}{\omega_e \cdot i_q} \\ L_d = \dfrac{V_q - r_s \cdot i_q - K_T \cdot \omega_r}{\omega_e \cdot i_d} \end{cases}$$

In the context of the parameter estimation algorithm, conditions are sought for a set of four parameters that will verify these equations. If a fixed operating point is considered such that the inductances are constant, the estimated inductances will satisfy the following relationship

$$\begin{cases} L_{q\,est} = \dfrac{r_{s\,est} \cdot i_d - V_{d\,ctrl}}{\omega_e \cdot i_q} \\ L_{d\,est} = \dfrac{V_{q\,ctrl} - r_{s\,est} \cdot i_q - K_{T\,est} \cdot \omega_r}{\omega_e \cdot i_d} \end{cases} \tag{6.1}$$

In these equations, the subscript "est" denotes an estimate of a parameter, and the subscript "ctrl" refers to variables as they are thought to be by the controller. As shown in Chapter IV, there can be a mismatch between the desired machine voltages in the controller and the voltages at the output of the inverter because of its non-idealities. In the experimental program, this error is compensated using the method that will be described in Chapter VII, but its effects will be investigated here.

For the operating point under consideration one can obtain

$$dL_{q\,est} = \frac{\partial L_{q\,est}}{\partial r_{s\,est}} \cdot dr_{s\,est} + \frac{\partial L_{q\,est}}{\partial K_{T\,est}} \cdot dK_{T\,est} + \frac{\partial L_{q\,est}}{\partial L_{d\,est}} \cdot dL_{d\,est} + \frac{\partial L_{q\,est}}{\partial V_{q\,ctrl}} \cdot dV_{q\,ctrl} + \frac{\partial L_{q\,est}}{\partial V_{d\,ctrl}} \cdot dV_{d\,ctrl}$$

and

$$dL_{d\,est} = \frac{\partial L_{d\,est}}{\partial r_{s\,est}} \cdot dr_{s\,est} + \frac{\partial L_{d\,est}}{\partial K_{T\,est}} \cdot dK_{T\,est} + \frac{\partial L_{d\,est}}{\partial L_{q\,est}} \cdot dL_{q\,est} + \frac{\partial L_{d\,est}}{\partial V_{q\,ctrl}} \cdot dV_{q\,ctrl} + \frac{\partial L_{d\,est}}{\partial V_{d\,ctrl}} \cdot dV_{d\,ctrl}$$

Eq. (6.1) gives

$$dL_{q\,est} = \frac{i_d}{\omega_e \cdot i_q} \cdot dr_{s\,est} - \frac{1}{\omega_e \cdot i_q} \cdot dV_{d\,ctrl}$$

and

$$dL_{d\,est} = \frac{-i_q}{\omega_e \cdot i_d} \cdot dr_{s\,est} - \frac{1}{i_d} \cdot dK_{T\,est} + \frac{1}{\omega_e \cdot i_d} \cdot dV_{q\,ctrl}$$

In the experimental case, the inverter voltage error is compensated for and there is an accurate match between the controller and machine voltages. In the end, the previous sensitivity equations reduce to:

$$dL_{q\,est} = \frac{i_d}{\omega_e \cdot i_q} \cdot dr_{s\,est}$$

$$dL_{d\,est} = \frac{-i_q}{\omega_e \cdot i_d} \cdot dr_{s\,est} - \frac{1}{i_d} \cdot dK_{T\,est}$$

Both the inductance estimates will be affected by an error in the stator resistance. The impact of error on the *q*-axis inductance will be smaller for operating points that present a large *q*-axis current, high speed and low *d*-axis current. Usually, in IPMs the high speed operation translates into flux weakening operation, which reduces the *q*-axis current and increases the *d*-axis one. In the case of a surface mount machine, the *d*-axis current is set to zero for operation below base speed, which minimizes the resistance error impact. On the other hand, the *d*-axis inductance estimation requires low *q*-axis and high *d*-axis currents and speed to minimize the error. Operation in the flux weakening region will consequently be best suited for this estimate.

An important problem associated with the *d*-axis inductance estimation is the impact that an error in the torque constant can have on its accuracy. The corresponding sensitivity term is only a function of the *d*-axis current and can be quite large.

The conclusion of this analysis is that the estimation of the *d*-axis inductance will be more sensitive to errors in the other parameter estimates than the estimation of the *q*-axis inductance. Consequently, this estimation will probably be of poorer quality than the latter. However, it is common in permanent magnet machines to have a much lower *d*-axis inductance variation due to saturation than on the *q*-axis and the *d*-axis in such cases could be considered constant for this problem. Such a solution can also be considered for cases where the *d*-axis current is too small to extract the *d*-axis inductance, in low torque and speed for an IPM or below base speed operation for a surface mount machine. In this case, the *d*-axis inductance is considered constant at a value that can be measured offline, and the "fast" estimation algorithm can be rewritten to only estimate the *q*-axis inductance.

Figure 6.7 shows a simple representation of the torque-speed characteristic of a PM machine. The space within the envelope of the characteristic can be divided into four regions depending on the performance of the parameter estimation algorithm.

The primary concern in estimating the machine inductance is that the corresponding term in the electrical equation, $L_{qd} \cdot \omega_e \cdot i_{qd}$ has to be measurable. For this reason, if the currents or the machine speed are too small, the inductance cannot be extracted from the electrical model. In the case of low torque operation, the controller can use the inductance values given by the manufacturer or values measured offline. On the other hand, operation at low speed, whether it is at low or high torque, is not suitable for this algorithm.

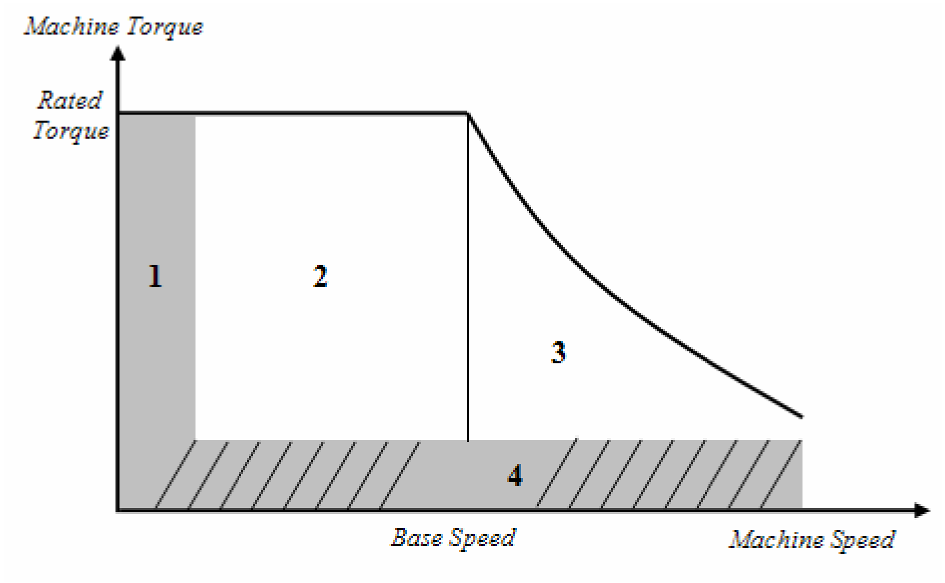

Fig. 6.7: Torque-speed regions for parameter estimation.

A different method that relies on the differential terms in the electrical equation could be implemented, but would be very difficult to implement experimentally in a non-intrusive manner because of the noise associated with estimation of the differential terms.

Table 6.4: Performance of parameter estimation in Torque-speed regions.

Region	***Surface mount PMSM***	***IPM***
1	Speed is too low for operation	Speed is too low for operation
2	L_d unimportant Good performance for $L_{q\ est}$	Average performance for $L_{d\ est}$ Good performance for $L_{q\ est}$
3	Average performance for $L_{d\ est}$ Average performance for $L_{q\ est}$	Better performance for $L_{d\ est}$ Average performance for $L_{q\ est}$
4	*d-q* currents are too small	*d-q* currents are too small

The difference between surface mount and IPM machines in region “3” of Figure 6.7 is due to the fact that usually IPM machines require a larger *d*-axis current than surface mount PM machines.

6.5 Algorithm Initialization

The RLS algorithm needs to be properly initialized when the machine starts to achieve maximum stability of the system. At machine startup, the assumption is that the machine is at room temperature, so that the stator resistance and torque constant are at their nominal values. This does not necessarily require offline measurements, because these values are usually given by the manufacturer. Offline measurements would be more accurate since these parameters can change slightly from one machine to another. On the other hand, the information of the *d-q* inductances is seldom available and it was assumed that only a rough estimate was available.

The first step in starting up the parameter estimation algorithm is to make sure that the “fast” identification program has converged before the other algorithm is started. Initially, only the “fast” algorithm will be run, after startup, and will use the room temperature estimates of the resistance and torque constant. After the “fast” algorithm has converged and steady state has been reached, the “slow” algorithm begins its operation using the estimates from the “fast” algorithm as its initial values for inductance, and the complete algorithm gets underway.

An example of initial convergence of the algorithm can be seen in Figure 6.8. The top plot shows the *d-q* inductances and their estimates. The fast estimation algorithm is enabled at 0.2s and converges towards the correct value for the *q*-axis inductance very quickly.

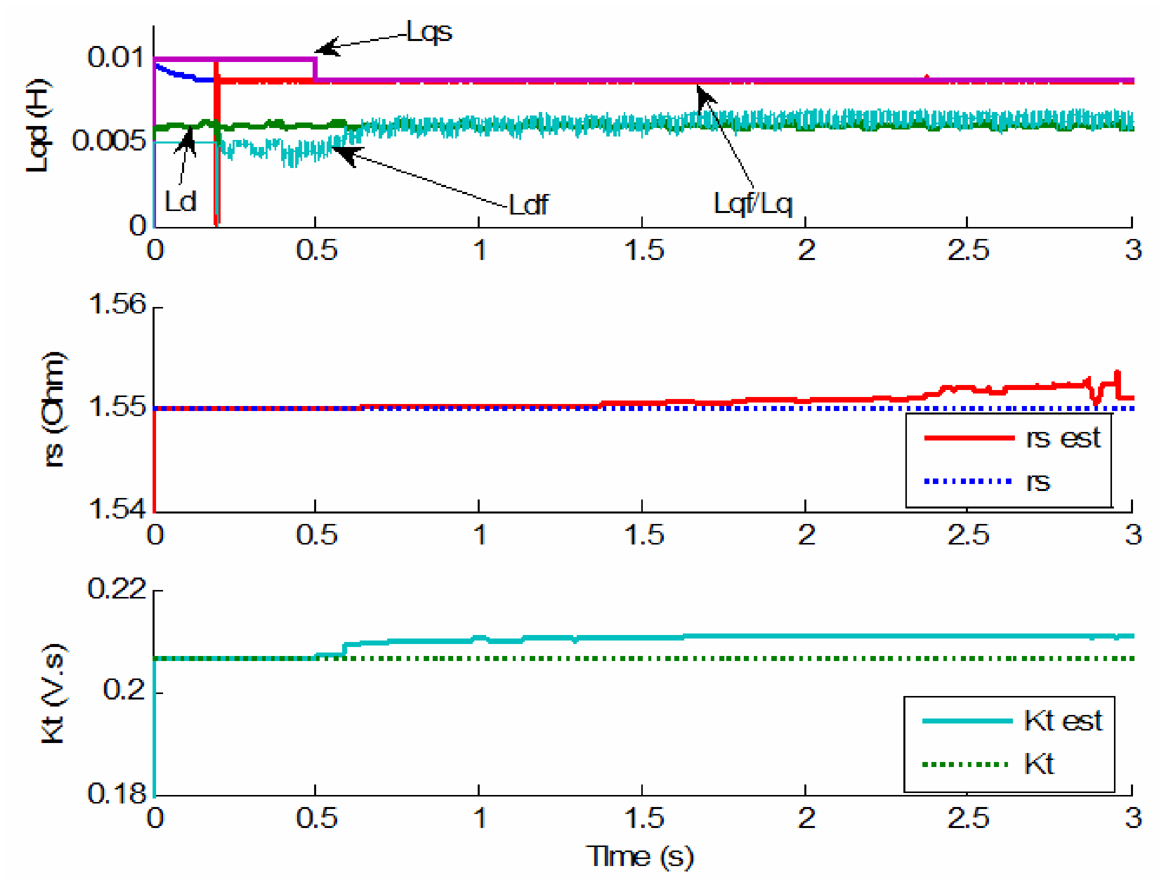

Fig. 6.8: Initial parameter estimation convergence.

The slow algorithm starts its operation at 0.5s, and uses the results from the “fast” estimation algorithm. After that point, the three waveforms are indiscernible. In the case of the *d*-axis inductance, the same conclusions apply (the result from the “slow” estimation program was omitted for clarity) except for the fact that the estimate appears to be more noisy and has a small error until the slow algorithm starts. This is consistent with our previous analysis, which stated that the *d*-axis inductance estimate was more likely to be poor than the *q*-axis inductance estimate. In this simulation, the inductances were initialized with values matching the machine inductances when there is no current (nominal values). There was therefore an error present when the “fast” algorithm was first enabled, which was reduced in a few cycles. One can also notice the small step waveform that the *d*-axis inductance has; this is due to the introduction of the *d*-axis perturbation necessary for the “slow” algorithm to converge.

The middle and bottom plots of Figure 6.8 show the stator resistance and the torque constant along with their estimates. Just like in the first plot, the execution of the parameter identification program that estimates them does not start until 0.5s. A small steady state error is present after the initial transient. The small oscillations before the final convergence at 3 s do not affect the algorithm stability. These are due to the limited simulated precision in controller measurements.

Another test was conducted with the same conditions as that of the first test except that the initial conditions on the stator resistance and the torque constant were off by 20%. These results are shown on Figure 6.9. The objective of these results was to show that the algorithm could converge even if the initial guesses for the machine parameters were wrong. This could happen if the machine was started without being at room temperature, for example after a long high torque operation. The parameters take a longer time than in Figure 6.8 to converge, but they do so quite effectively. It is possible to note that the error in the stator resistance reflects directly in the error in *q*-axis inductance (similar waveform). On the other hand, both the error in resistance and torque constant affect the *d*-axis inductance, with the initial 20% error causing more than 300% error in the inductance estimate. This numerical property of the *d*-axis inductance estimate will definitely be a problem in some cases. Note that parameter estimates are clamped for increased stability and values that would threaten system stability should not be allowed.

The results shown in Figure 6.9 also indicate that the algorithm may be able to run intermittently. In other words, to minimize the intrusive aspect of the “slow” identification program knowing that the stator resistance and the torque constant change slowly as a function of time, it may be possible to run this algorithm for limited periods in time. This allows the algorithm to reconverge to possibly new parameter values in a manner such as the one seen in Figure 6.9 if parameters needed to be updated significantly. On the other hand, the execution of the “fast” algorithm is non-intrusive and should be executed at all times.

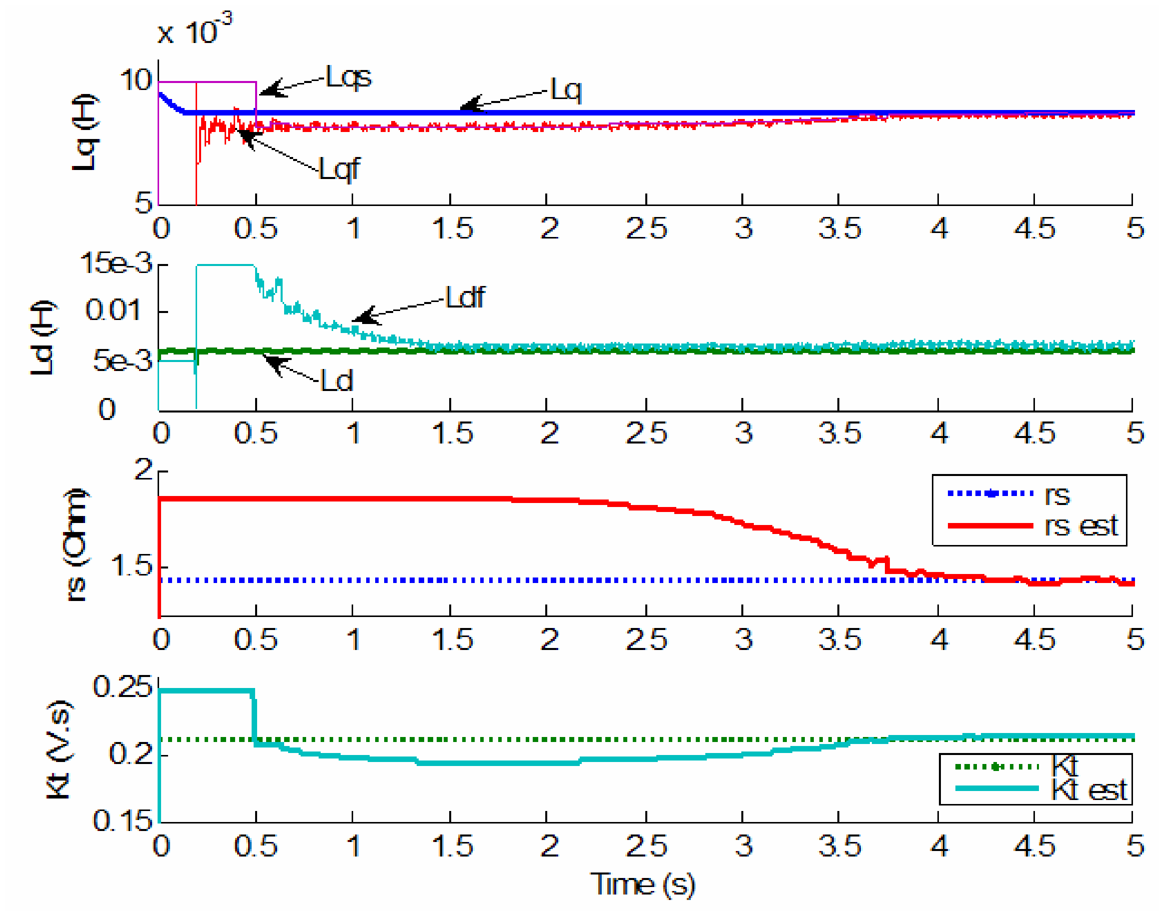

Fig. 6.9: Initial parameter convergence with 20% initial error.

6.6 Effect of Cross-Saturation

Cross-saturation is a phenomenon that makes both the *d*- and *q*-axes inductances functions of both *d*- and *q*-axes currents. In other words, the magnetic flux associated with magnetic paths aligned with the rotor permanent magnets affects the magnetic flux that is in quadrature with it, and vice versa. Cross-saturation surfaces were measured experimentally for machine “a” and the results were shown in figures 6.1 and 6.2.

The cross-saturation has one adverse effect with regards to the task of parameter estimation: the *d*-axis perturbation, which in the case where no cross saturation is present affects only the *d*-axis inductance, may now affect both *d*- and *q*-axes inductances. In most cases, however, the *d*-axis inductance variation is small, but the introduction of cross-saturation may affect the *q*-axis inductance significantly.

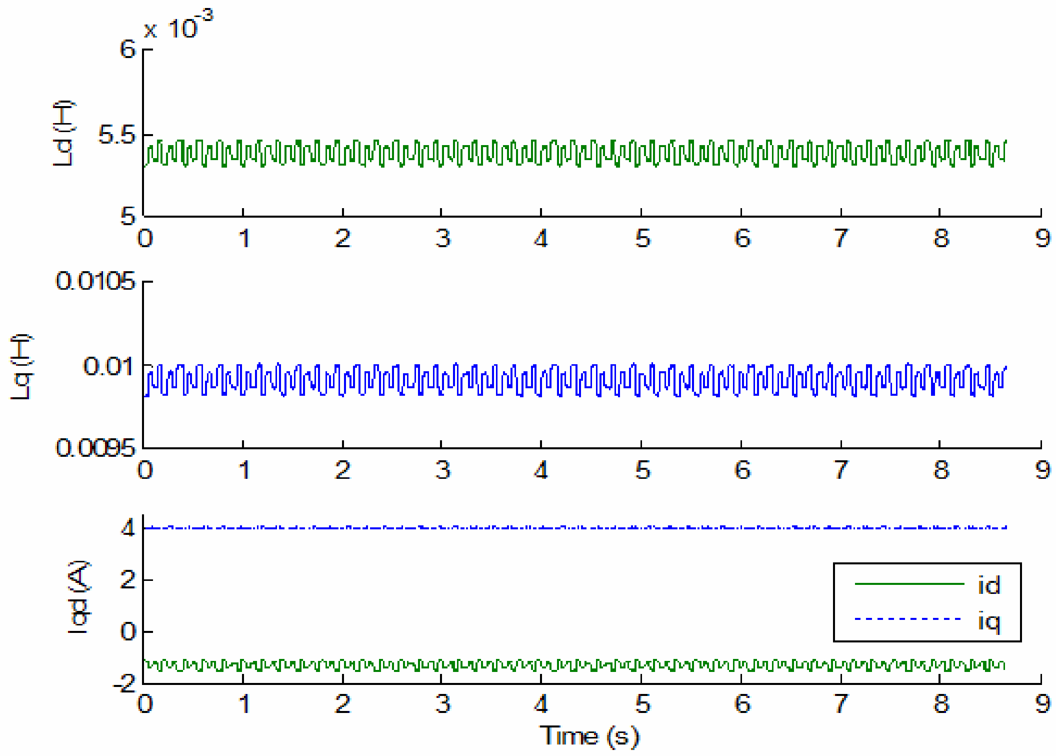

Fig. 6.10: Cross-saturation effect on machine inductances.

Figure 6.10 shows the effect of cross-saturation on both the *q*- and *d*-axes inductances in the context of execution of the parameter estimation algorithm. A variation of about 25% in the *d*-axis current is the cause of a variation of 2% in *q*-axis inductance, and about 4% in *d*-axis inductance. Although these changes are not significant, they cause the system to have different inductance parameters at each step of the perturbation, as opposed to the normal case, where only the *d*-axis inductance is affected.

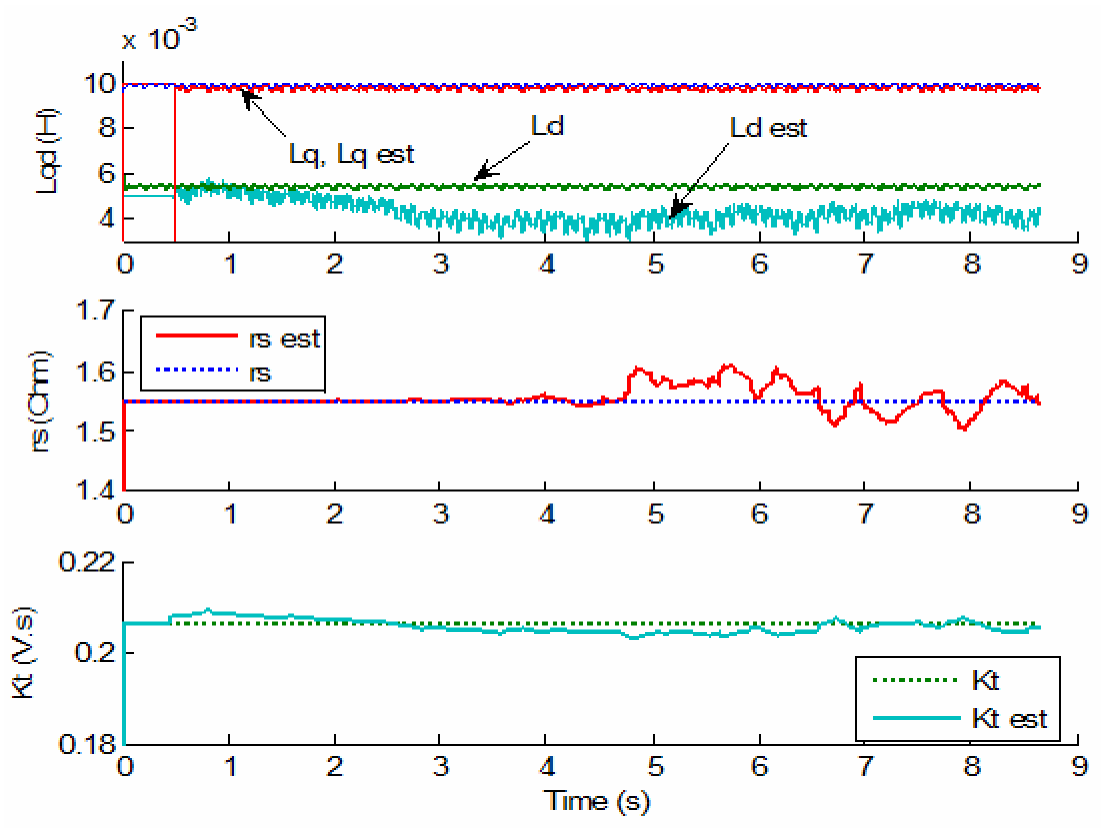

Fig. 6.11: Parameter estimation results with cross-saturation.

Parameter estimation results are shown in Figure 6.11 in the case where saturation is present in machine “A”. The *q*-axis inductance, stator resistance and torque constant have fairly accurate estimates, but the *d*-axis inductance estimate is of poor quality. This is because the *d*-axis inductance is also a function of the *q*-axis current, which is quite large in this case. Having a large *q*-axis current increases the amount of flux present in the stator iron and consequently increases saturation. A consequence of this is that the *d*-axis inductance becomes smaller than in the case where the *q*-axis current is zero (no cross saturation case). Having a smaller *d*-axis inductance term will make the corresponding inductive drop smaller in the electrical model equations, and more difficult to estimate. Another reason for the poor estimate is that the small inductance changes measured by the "fast" inductance algorithm are not copied into the "slow" one at every cycle, since it would nullify the inductance estimation capability of the latter program, and make the overall algorithm less stable. The solution adopted in the experimental case is to set the *d*-axis inductance to be constant, since small errors in its estimation have negligible effect on the other parameter estimates.

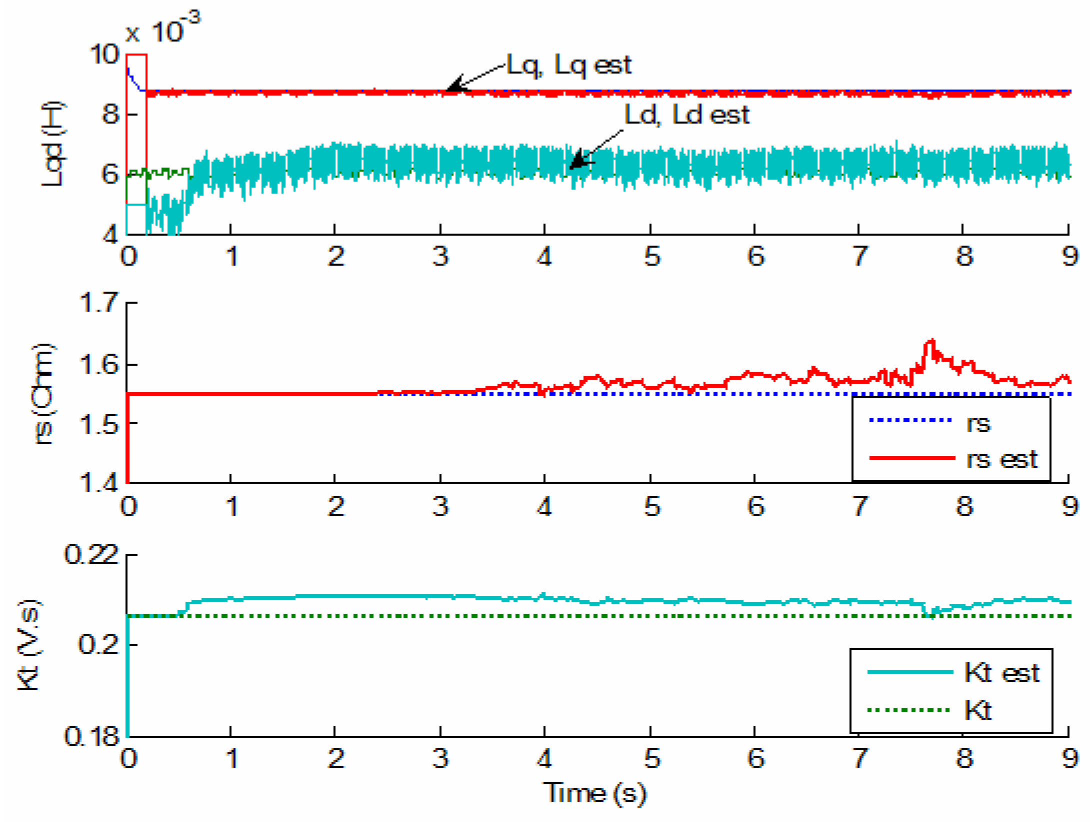

Fig. 6.12: Parameter estimation results with no cross-saturation.

For comparison purposes, simulation results obtained without cross-saturation in machine “A” are shown in Figure 6.12. In this case, the *d*-axis inductance estimate is quite accurate, as well as the other three parameter estimates.

6.7 Tracking of Temperature Effects

In this section, the simulation results that correspond to cases where the machine temperature increases linearly as a function of time are shown. The temperature change affects the stator resistance and the machine torque constant directly. It is important for the parameter estimation algorithm to be able to track these changes.

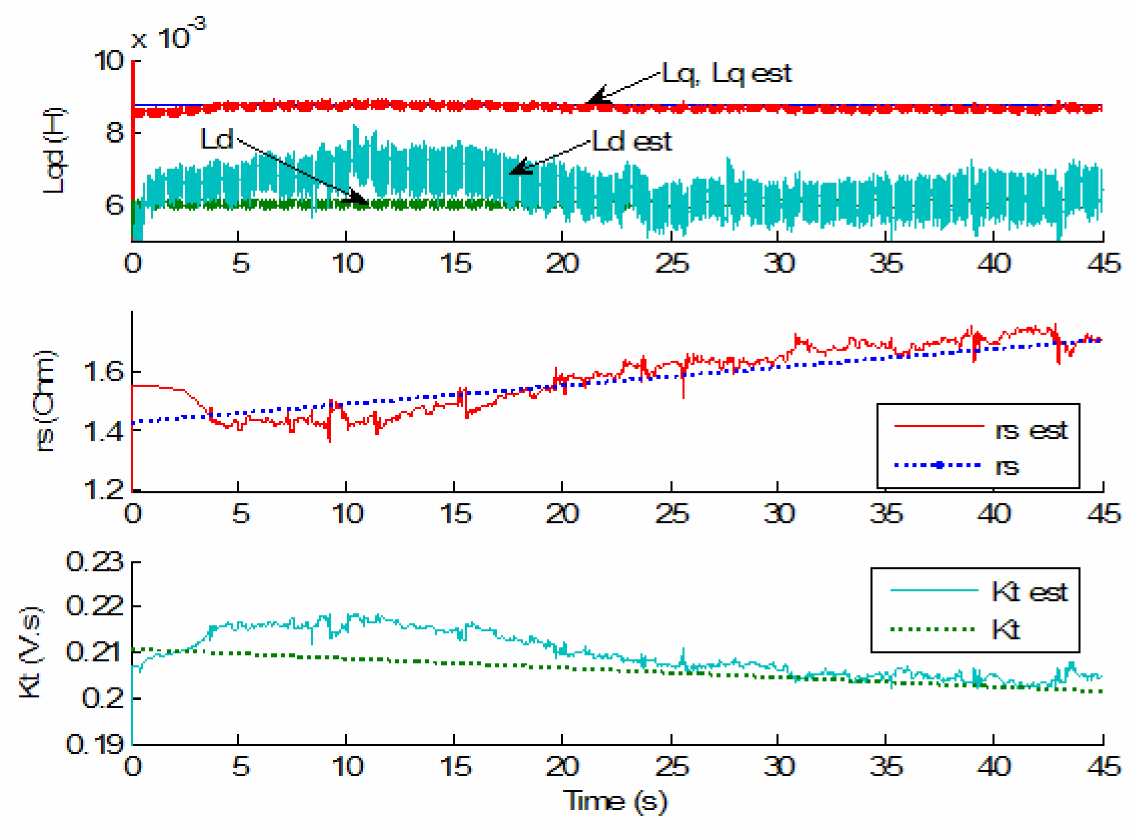

Fig. 6.13: Parameter estimation with +1˚C/s ramp temperature on machine “A”.

In Figure 6.13, parameter estimation results are shown in the case of a ramp temperature increase in machine “A”. The rate of change was set to be quite fast compared to common temperature dynamics in electrical machines. The three plots show that the different parameter estimates converge within acceptable range of the real machine parameters. The fact that the algorithm converges with such a temperature rate of change is a good indication that the algorithm will be able to track parameter variations due to temperature changes in any experimental case.

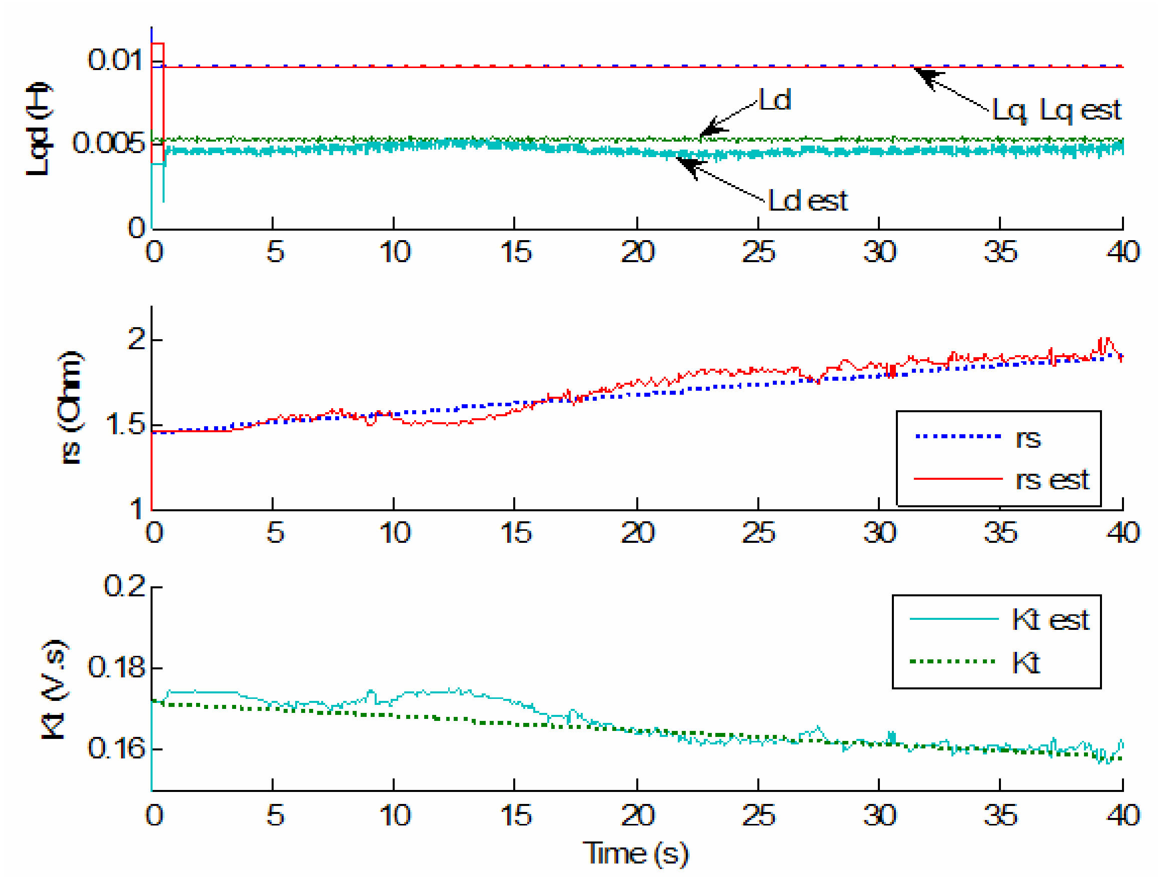

Fig. 6.14: Parameter estimation with +2˚C/s ramp temperature on machine "B".

Similar results and conclusion were obtained with machine "B" and are shown in Figure 6.14. The temperature rate of change in this simulation was set to be twice as large (2˚C/s) as in Figure 6.12, and the results were still satisfactory.

6.8 Effect of Back-Emf Harmonics

A common non-ideality associated with permanent magnet machines and the *d-q* model is that the back electromotive force is not a perfect sine wave. The presence of higher order harmonics is sometimes felt, especially in the case of IPMs. The purpose of this section is to investigate the effect that such harmonic perturbations can have on the parameter estimation algorithm. The analysis can easily be done with the three phase machine model expressed in the *d-q* domain. To begin this analysis, it is easier to go back to the three phase domain. Going back to the Park transform equations, presented in Chapter II, we have

$$\begin{bmatrix} f_a \\ f_b \\ f_c \end{bmatrix} = \begin{bmatrix} -\sin(\theta_r) & \cos(\theta_r) & 1 \\ -\sin(\theta_r - 2\cdot\pi/3) & \cos(\theta_r - 2\cdot\pi/3) & 1 \\ -\sin(\theta_r + 2\cdot\pi/3) & \cos(\theta_r + 2\cdot\pi/3) & 1 \end{bmatrix} \cdot \begin{bmatrix} f_q \\ f_d \\ f_0 \end{bmatrix}$$

Considering the back emf term $K_T \cdot \omega_r$ term in the *q*-axis electrical equation, the inverse Park transform of this term will only have a component on the *q*-axis. This component is constant at constant speeds, neglecting signal harmonics. After we apply the inverse Park transform to this *q*-axis term, we can see that it corresponds to a three-phase signal as follows:

$$\begin{bmatrix} E_a \\ E_b \\ E_c \end{bmatrix} = K_T \cdot \omega_r \cdot \begin{bmatrix} -\sin(\theta_r) \\ -\sin(\theta_r - 2\cdot\pi/3) \\ -\sin(\theta_r + 2\cdot\pi/3) \end{bmatrix}$$

In this above equation, E_x (x =*a*, *b* or *c*) symbolizes the phase back-emf. The introduction of an additional term which corresponds to the kth harmonic will consequently lead to the following form:

$$\begin{bmatrix} E_a \\ E_b \\ E_c \end{bmatrix} = K_T \cdot \omega_r \cdot \begin{bmatrix} -\sin(\theta_r) \\ -\sin(\theta_r - 2\cdot\pi/3) \\ -\sin(\theta_r + 2\cdot\pi/3) \end{bmatrix} + K_{Tk} \cdot \begin{bmatrix} -\sin(k\cdot\theta_r) \\ -\sin(k\cdot(\theta_r - 2\cdot\pi/3)) \\ -\sin(k\cdot(\theta_r + 2\cdot\pi/3)) \end{bmatrix}$$

where K_{Tk} is the magnitude coefficient of the kth harmonic. It is now necessary to go back to the *d-q* domain, using the Park transform equations, to see the effects of the introduction of harmonics on the *d-q* model of the machine. The *d-q* transformation equations are repeated here for convenience.

$$\begin{bmatrix} f_q \\ f_d \\ f_0 \end{bmatrix} = \frac{2}{3} \cdot \begin{bmatrix} -\sin(\theta_r) & -\sin(\theta_r - 2\cdot\pi/3) & -\sin(\theta_r + 2\cdot\pi/3) \\ \cos(\theta_r) & \cos(\theta_r - 2\cdot\pi/3) & \cos(\theta_r + 2\cdot\pi/3) \\ 1/2 & 1/2 & 1/2 \end{bmatrix} \cdot \begin{bmatrix} f_a \\ f_b \\ f_c \end{bmatrix}$$

Using the distributive property of the Park transform, we can separate the Park transform of the fundamental back-emf ($K_T \cdot \omega_r$ in the *q*-axis) from the Park transform of the kth harmonic. An interesting property arises in the case of harmonics of ranks that are multiples of 3 (k = 3.p, for some integer p):

$$E_q = \frac{2 \cdot K_{T3p}}{3} \cdot \begin{bmatrix} \sin(\theta)\cdot\sin(3\cdot p\cdot\theta) + \sin\left(\theta - \frac{2\pi}{3}\right)\cdot\sin\left(3\cdot p\cdot\left(\theta - \frac{2\pi}{3}\right)\right) \\ +\sin\left(\theta + \frac{2\pi}{3}\right)\cdot\sin\left(3\cdot p\cdot\left(\theta - \frac{2\pi}{3}\right)\right) \end{bmatrix}$$

$$E_q = \frac{2 \cdot K_{T3p}}{3} \cdot \sin(\theta) \cdot \left[\sin(\theta) + \sin\left(\theta - \frac{2\pi}{3}\right) + \sin\left(\theta + \frac{2\pi}{3}\right)\right] = 0$$

And in the case of the *d*-axis:

$$E_d = \frac{2 \cdot K_{T3p}}{3} \cdot \begin{bmatrix} \cos(\theta)\cdot\cos(3\cdot p\cdot\theta) + \cos\left(\theta - \frac{2\pi}{3}\right)\cdot\cos\left(3\cdot p\cdot\left(\theta - \frac{2\pi}{3}\right)\right) \\ +\cos\left(\theta + \frac{2\pi}{3}\right)\cdot\cos\left(3\cdot p\cdot\left(\theta - \frac{2\pi}{3}\right)\right) \end{bmatrix}$$

$$E_d = \frac{2 \cdot K_{T3p}}{3} \cdot \cos(\theta) \cdot \left[\cos(\theta) + \cos\left(\theta - \frac{2\pi}{3}\right) + \cos\left(\theta + \frac{2\pi}{3}\right) \right] = 0$$

A consequence of these results is that harmonics of orders that are multiples of 3 will not affect the result of the Park transformation. As a result, the *d-q* model will not be affected by such harmonics, and the results obtained in this chapter would not be affected by their introduction.

However, harmonics that are not of such orders will introduce a perturbation voltage on both the *d-* and *q*-axes of the machine electrical model. As an example, we chose to study the effect of the 7th harmonic, which is often present in the back-emf of PM machines. The effect of introducing this harmonic on the *d-q* model of the machine can be seen in Figure 6.15.

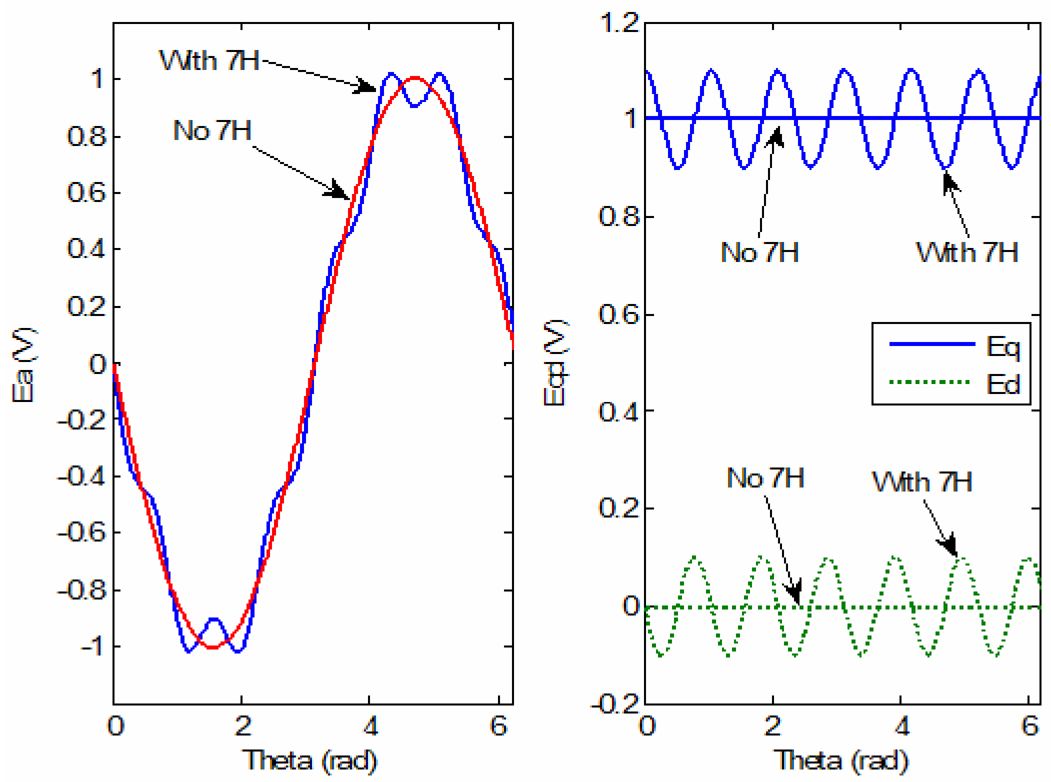

Fig. 6.15: Effect of 7th harmonic on *d-q* model ((a) Phase “a” / (b) *d-q* equivalent).

In Figure 6.15, the effect of a 10% seventh harmonic is shown when applied to a fundamental back-emf of unity magnitude. The left hand side of the Figure (Fig. 6.15 (a)) shows the phase back-emf waveform, with and without the additional harmonic, and the right hand side (Fig. 6.15 (b)) shows the corresponding *d-q* voltages. One can see that the perturbation associated with the seventh harmonic directly affects both the *d*- and *q*-axes back emfs. The effect of this additional term will consequently be a perturbation on both axes of the *d-q* model of the machine. This perturbation is not accounted for by the controller and the parameter estimation algorithm. In order to simulate the effects of such a perturbation on the performance of parameter estimation algorithm, the perturbation was introduced on both the *d*- and *q*-axes in the form of a sine function (cosine for the *q*-axis) of the position having a magnitude proportional to the motor speed. A test similar to the one that gave Figure 6.12 was conducted with the introduction of this perturbation and the results are shown in Figure 6.16.

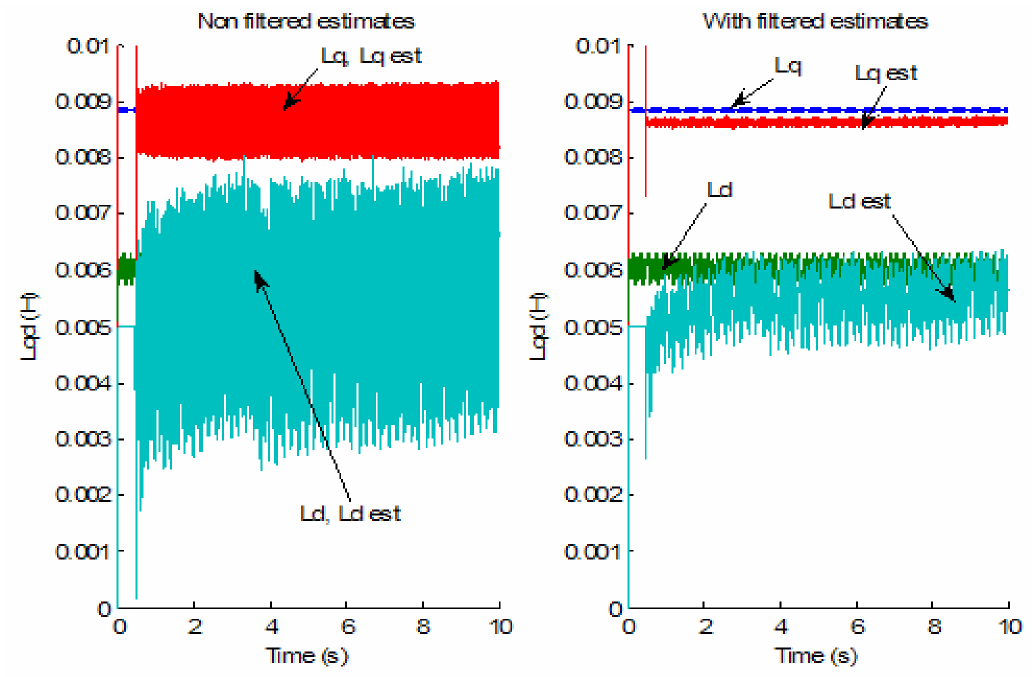

Fig. 6.16: Effect of 7th harmonic on parameter estimation ((a) non filtered / (b) filtered).

Figure 6.16 has two sets of results: Figure 6.16 (a) shows the results that would be obtained directly from the parameter estimation algorithm, and Figure 6.16 (b) comes from estimation results that have been filtered. The motivation for filtering is apparent from the noise present in the results of Figure 6.16 (a), which is a direct consequence of the perturbation. It can be observed that the perturbation affects the *d*-axis inductance more that the *q*-axis one, although the perturbation is of same magnitude on both axes and the term corresponding to the *d*-axis inductance drop is smaller than the *q*-axis one. One can see that once the perturbation has been filtered, the results are acceptable. The lag introduced by the filtering is acceptable and has no effect on the "slow" parameter estimation program as long as it is not enabled during an inductance transient, which is usually not the case.

6.9 Conclusion

This chapter was dedicated to the presentation of simulation results that were a first step in validating the proposed algorithm. These results were either related to difficulties or concerns at the algorithm development stage, or were related to actual results obtained with the proposed parameter estimation algorithm in various situations. Two machine models were used to validate the algorithm performance, and the results were satisfactory. The following chapters will focus on the experimental verification of the effectiveness of the parameter estimation algorithm.

CHAPTER VII

PMSM DRIVE EXPERIMENTAL DESIGN

This chapter introduces the different elements of the experimental design created and used to verify the effectiveness of the proposed algorithm. Each of these elements is introduced in details, with special emphasis on features that could affect the project.

The motor model parameters were presented in Chapter VI and the inverter was introduced in Chapter V. This chapter follows with the details of the feedback circuit design and the digital processor.

7.1 Inverter

7.1.1 Inverter Choice

The inverter chosen for this project has the classical three-phase bridge structure presented in the previous chapters. The package chosen included all six switches and the required driver in one integrated circuit. In addition to saving space, this structure also allows a better match between switch characteristics and propagation delays. However, the problem with this structure is that the whole integrated circuit has to be replaced even if only one switch is damaged. The inverter ratings were chosen to largely exceed the machine ratings in terms of voltage and current, in order to minimize damage.

The switch driver in the package included a temperature and a current monitor that could be used for shutdown in the event of a problem. These features were disabled for this project. Another interesting feature is the insertion of a forced deadtime and the prevention of inverter short circuit even when a faulty controller would command it.

7.1.2 Inverter Voltage Compensation

The non-linear characteristics of the experimental inverter were presented in Chapter V. Their effects were investigated and it was shown that they would need to be minimized in the context of this project. The compensation algorithm used for this project treated two problems. The first one being the deadtime compensation and the second is the switch voltage drop. The goal of this algorithm is to get a better match between the controller voltage and the inverter output.

7.1.2.1 Deadtime Compensation

The method chosen for the purpose of compensating the deadtime delay in the controller is a simple one. It was derived from Figure 5.6 in Chapter V and observations made about it. The introduction of a deadtime delay changes the voltage applied to the machine by changing the effective input duty cycles of the inverter.

Figure 7.1 shows the DSP outputs that correspond to a case of deadtime introduction, and the corresponding phase voltage at the machine terminal. Two assumptions were made in this compensation technique; the first one is that the phase current does not change sign during the switch deadtime, and the second one is that the switch voltage drops were negligible. The second assumption is valid because the switch voltage drops are compensated for by another algorithm. The assumption that the phase current does not change sign may in some cases not be valid, but even if the current changes sign, the diode reverse recovery time (the time it takes for a diode to switch "off") will maintain the voltage error constant longer. There will be cases, however, where this assumption is not valid, but they will introduce a very small error, for a very short time.

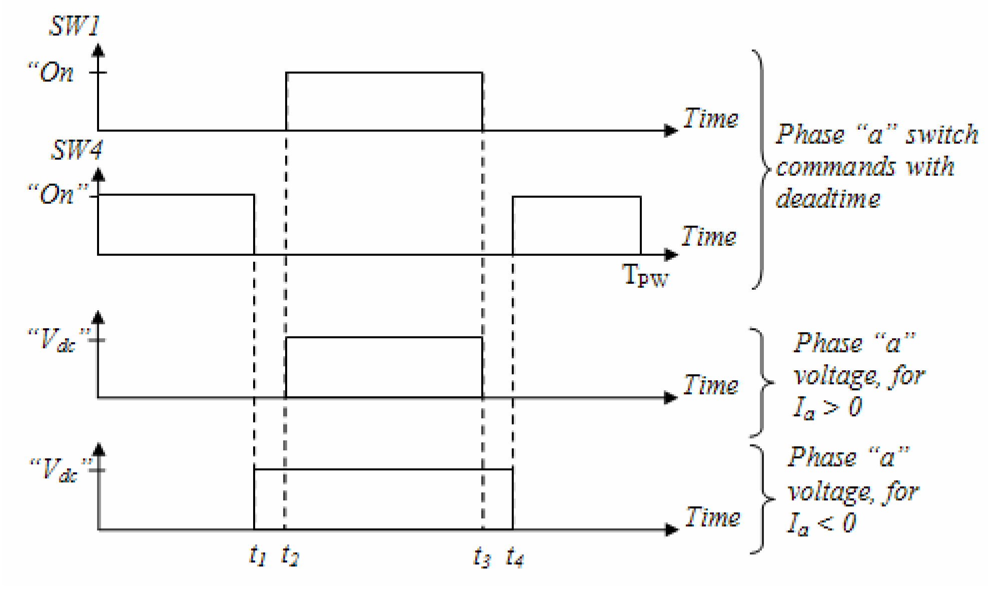
SW1
"On
Time
SW4
"On"
Time
TPW
Phase "a" switch commands with deadtime
"Vdc"
Time
Phase "a" voltage, for Ia > 0
"Vdc"
Time
Phase "a" voltage, for Ia < 0
t1 t2 t3 t4

Fig. 7.1: Deadtime error as a function of phase current.

The error introduced by the deadtime delay T_{dead} is a function of the sign of the phase current. The explanation for this is that when both switches are commanded to open, the inductive nature of the motor circuit will prevent the current from switching to zero and will force one of the leg diodes to turn "on". The diode that turns "on" depends on the sign of the current. If it is positive (going toward the machine), the bottom diode will conduct and the phase voltage will be zero, and in the other case the top diode will conduct and the phase voltage will be high, as shown in Fig. 7.1. If we refer back to the ideal case, shown in Figure 5.6 of Chapter V, we know that the controller command voltage would have the inverter switch at times t_1 and t_3.

It follows that

$t_2 - t_1 = t_4 - t_3 = T_{dead}$.

The voltage error can be expressed as:

$$V_{err\,dead} = V_{ctrl} - V_{inverter} = -V_{dc} \cdot \frac{T_{dead}}{T_{PWM}} \cdot sign(i_{ph})$$

where T_{PWM} is the PWM period, V_{ctrl} is the controller command voltage, $V_{inverter}$ is the actual inverter output voltage and i_{ph} is the phase current. The voltage error term is introduced in the controller at the PWM stage to compensate the error given by the above equation. This way, the compensation is transparent from the controller's point of view.

7.1.2.2 Switch Drop Compensation

As mentioned in Chapter V, the inverter switches do not have ideal characteristics, and when they conduct, both diodes and IGBTs introduce a voltage drop. The method presented in this chapter is specific to the hardware that was used in this project. Other power electronics devices, such as MOSFETs, would not present the same characteristics, since when they are "on" they act like a small resistance. In the case of IGBTs and diodes, a voltage drop is present during conduction that is a non-linear function of the current going through the switch. The voltage drop was recorded as a function of this current and the results were shown in Chapter V for both types of switch.

For the analysis, it will be assumed that deadtime is not present (or compensated), and that the phase current does not change sign during the PWM period. The effect of the switch voltage drops is a function of the phase current sign, similar to the case of deadtime compensation as shown in Table 5.2.

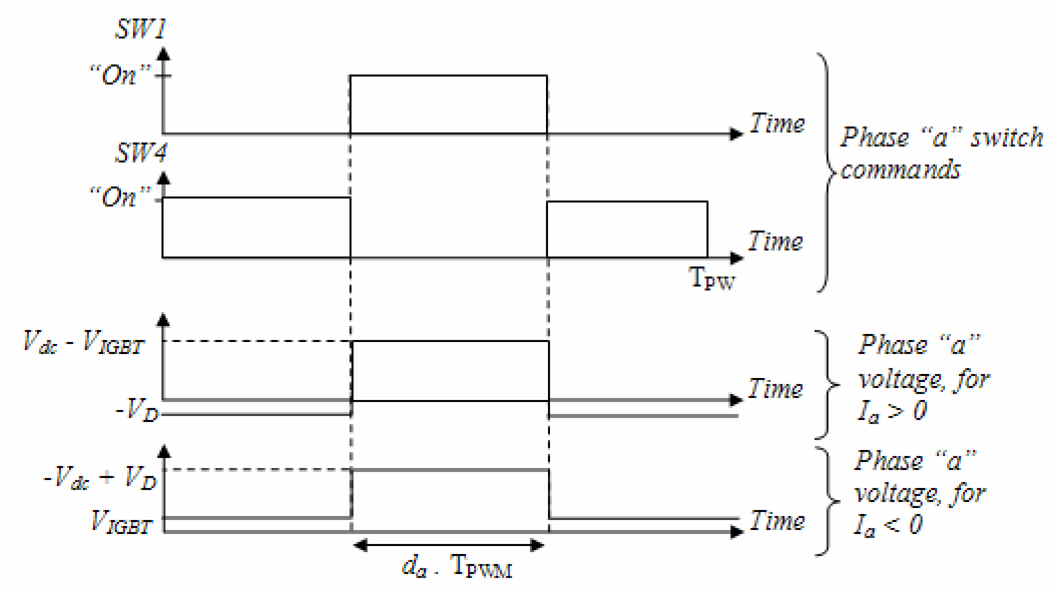
SW1
"On"
Time
SW4
"On"
Time
TPW
Phase "a" switch commands
Vdc - VIGBT
-VD
Time
Phase "a" voltage, for Ia > 0
-Vdc + VD
VIGBT
Time
Phase "a" voltage, for Ia < 0
da . TPWM

Fig. 7.2: Switch voltage drop error as a function of phase current.

Figure 7.2 shows the two possible cases that can be obtained from the inverter as a function of the sign of the phase current in the case of phase "a". Both the diode and IGBT voltage drops are also function of the magnitude of the phase current, as shown in Chapter V.

In an ideal case, both the diode and IGBT voltage drops would be zero, and the phase duty cycle could be calculated as:

$$d_a = \frac{V_{a\,ctrl}}{V_{dc}}$$

where $V_{a\,ctrl}$ is the phase "a" voltage desired by the controller. If the switch voltage drops are taken into consideration, this equation is not valid anymore, and compensation is necessary to match the controller and inverter voltages. To obtain an expression for the compensation voltages, or the duty cycle, it is necessary to consider two cases separately. In the case of a positive phase current, we have:

$$V_{a\,ctrl} = \left(V_{dc} - V_{IGBT}\right) \cdot d_a - \left(1 - d_a\right) \cdot V_D$$

where the controller is assumed to have the same voltage command as the inverter output voltage. The phase duty cycle can then be extracted as:

$$d_a = \frac{V_{a\,ctrl} + V_D}{V_{dc} + V_D - V_{IGBT}}$$

In this equation, the diode and IGBT voltage drops are positive and quite close to each other considering the results shown in Figure 5.7. These two variables can also be considered to be much smaller than V_{dc}, which is at least larger than 50 V in this project. On the other hand, the phase "a" controller voltage is a variable and may not be considered to be much larger than the diode forward voltage drop. As a consequence, the compensated duty cycle can be rewritten as

$$d_a = \frac{V_{a\,ctrl} + V_D}{V_{dc}} \tag{7.1}$$

In the case of a negative phase current, a similar analysis can be conducted. The controller phase voltage can be expressed as

$$V_{a\,ctrl} = \left(V_{dc} + V_D\right) \cdot d_a - \left(1 - d_a\right) \cdot V_{IGBT}$$

And the compensated duty cycle can be extracted as

$$d_a = \frac{V_{a\,ctrl} - V_{IGBT}}{V_{dc} + V_D - V_{IGBT}}$$

In this case the denominator is the same as in the previous case, and the same conclusion applies. The final compensated duty cycle can be obtained as

$$d_a = \frac{V_{a\,ctrl} - V_{IGBT}}{V_{dc}} \tag{7.2}$$

One can notice that in the case of a positive phase current, the error induced by the switch voltage drops is dominated by the diode drop, and in the case of a negative current the IGBT drop dominates.

The compensation algorithm uses look-up tables to store the results shown in Figure 5.7. Interpolation functions could also be used and this choice depends on a compromise between calculation complexity and program memory requirement. In the PWM program, the measured currents are used to obtain the switch voltage drops as a function of each phase current. The corresponding three phase duty cycles are calculated from equations (7.1) and (7.2) and the deadtime compensation algorithm is applied to the result.

7.1.2.3 Compensation Verification

To verify the effectiveness of the combined compensation algorithms, an experiment similar to the one conducted to obtain Figure 5.10 was done. The *q*-axis voltage was varied in steps for a locked rotor condition, and the inverter voltage error was measured by taking the difference between the resistive voltage drop, and the controller voltage. These results are shown in Figure 7.3. If the results on this plot are compared with the ones shown on Figure 5.10, one can see a substantial improvement. The voltage error in the case that had no compensation was around 1 V for the considered conditions, and is now around zero. As a consequence, the controller voltage matches the inverter one much more accurately.

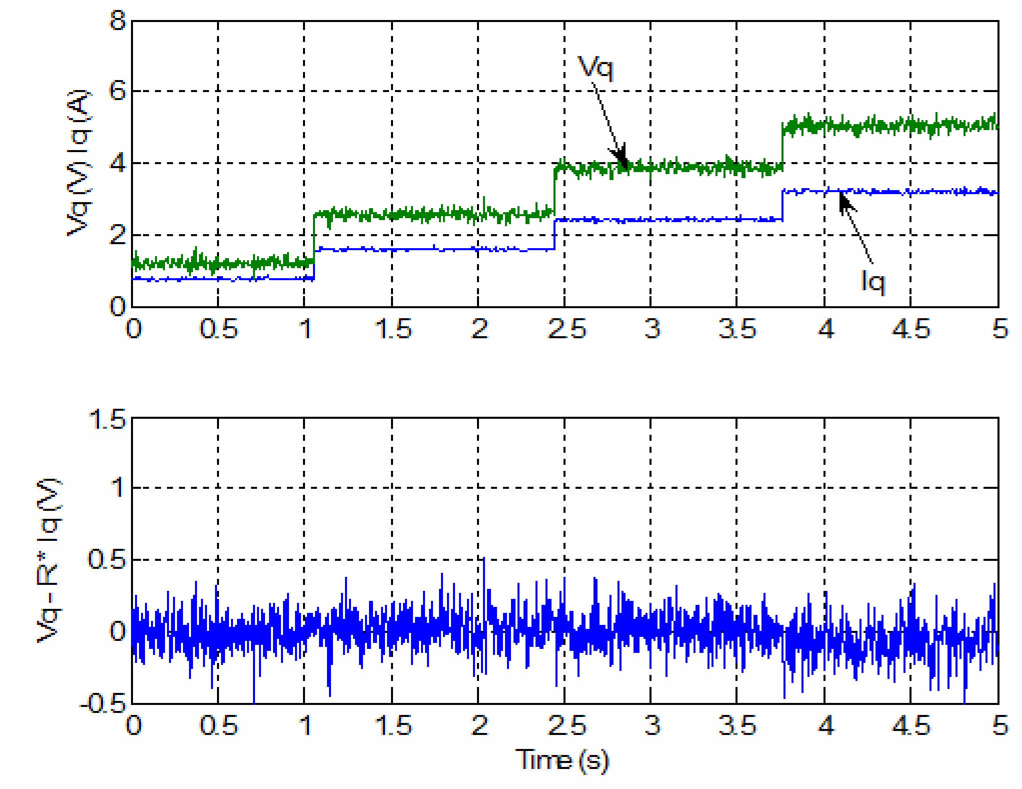

Fig. 7.3: Experimental inverter response with locked rotor and voltage compensation.

Another way of showing the effect of the voltage and deadtime compensations is to look at the controller voltages for a given operating point for different cases of compensation.

Table 7.1: Effects of controller voltage compensation.

Types of compensation	V_q (V)	V_d (V)	$L_{q\ est}$ (mh)
None	24.6	4.5	11.8
Deadtime	24	4.26	11.1
Deadtime and switch voltage	22.8	3.82	9.7

Table 7.1 shows results that were obtained in the case of a current-control experiment at constant speed. The results shown correspond to an operating point that has the *q*-axis current at 1.5 A, and the *d*-axis current at -0.5 A at a speed of 1000 RPM. The

controller voltages are shown, as well as the estimated *q*-axis inductance, in order to show the effect of the voltage error on parameter estimation. Since the operating point is the same in the three cases, we can deduce that the inverter output voltage remains constant too. If we refer to the inductance table shown in Chapter VI (Table 6.2), we see that the *q*-axis inductance for the considered operating point is between 9.6 and 9.9 mH. The case that includes both voltage compensations matches this value closely, but this is not the case with the other experiments. The omission of the voltage compensation algorithms has a significant effect on the inductance estimation for this operating point.

7.2 Digital Signal Processor

The DSP that was chosen for this project is the TMS320F2812 from Texas Instruments. Its design is optimized for motor control operations. Consequently, it offers many interesting features for this project which will be highlighted in the following sections.

7.2.1 PWM Generation

The DSP features a number of digital PWM outputs, with associated timers that can generate PWM signals with minimal CPU supervision. In other words, after the dedicated registers are initialized, the only operation needed from the CPU is to update the duty cycle.

The way PWM is generated is by having a timer counting either upwards, or upwards and downwards (symmetric or asymmetric PWM) up to a value chosen by the user, at a rate chosen by the user. Consequently, both PWM precision and frequency can be accurately selected. The timer value is then compared at each step with a variable contained in another register, called the compare register. Depending on the desired output logic, the PWM output will have a state that will depend on whether or not the timer value is larger than the compare value. It is also possible to associate two PWM outputs with one timer and compare value, in order to control the two switches that form an inverter leg.

Various interrupts can be generated to synchronize the programs with PWM. An interrupt function is a program that is executed when a preprogrammed event occurs, such as PWM period or underflow. This can be useful for example to synchronize the various program measurements (analog to digital conversions) with the PWM.

An additional feature that the DSP offers in terms of PWM generation is the automatic introduction of a user-specified deadtime delay. This feature is necessary for a PMSM drive and its effects, and the compensation techniques for them have been discussed in previous sections.

7.2.2 Analog to Digital Converters

The 2812 DSP has 16 possible analog to digital (AD) inputs that are multiplexed with two sample and hold units and one analog to digital converter (ADC). Two inputs can be sampled simultaneously, and then converted sequentially by the ADC. This is convenient for applications where synchronization is important. For example, the machine phase currents require at least the measurements of two phase currents simultaneously for optimal precision. If the conversions are not taken at the same time, it may have an effect on the measurement quality. This is however a minor concern and there are structures where a single current sensor is used in combination with a variable delay to reconstruct the three phase currents.

Another very interesting aspect of the ADC in the 2812 DSP is the speed at which they operate. They are able to provide a precision of up to 12 bits for a conversion time of 60 to 200ns. This allows very high sampling frequencies and the possibility of oversampling, in order to improve measurement quality and precision. Twelve bits of precision give a maximum accuracy of 2.5 mA for a maximum current of 10 A.

The ADC can also be automatically synchronized with the PWM circuits so that conversions can be started when specific events occur, without any CPU supervision. This can be helpful in some current sampling cases, like the one that was mentioned previously (single shunt current sensor for three phase machine).

However, in reality, for the present project and its associated hardware, the effective precision of the ADC was closer to 10 bits. The ADC input voltage range is 3 V, which was made to correspond to 20 A and this gives about

$$\frac{20}{2^{10}} \approx 20\,\text{mA}$$

for maximum accuracy. Another problem introduced by the ADC was the introduction of an offset and ramp error. These errors were specific to each DSP chip and had to be compensated. The response of the ADC can be described by the following equation

$$ADC_{out} = \frac{4096}{3} \cdot \left(G \cdot V_{in} + Off\right) \tag{7.3}$$

where ADC_{out} is the digital output of the ADC (between 0 and 2^{12} - 1), G is the ADC gain, V_{in} is the ADC input voltage, and Off is the ADC offset. The “4096/3” gain comes from the 12 bit precision of the ADC and its maximum input voltage, which is 3 V.

Ideally, *G* would be equal to "1" and *Off* would be zero, but this was not the case in practice. For example, the last DSP used for this project had a 32% full range gain error and a 40 mV offset. These were measured with known fixed voltages, which were compared with the conversion result. Each time the DSP had to be changed, the gain and offset parameters had to be measured and were introduced in the program, so that their effect could be compensated. If these errors were not compensated, the ADC would introduce significant errors when measuring sinusoidal signals, because its error would be largely different from the maximum of the signal to its minimum, introducing significant distortion.

Another constraint that was associated with the ADC was the relatively low maximum input voltage. In order to get maximum precision from the DSP, it was necessary to scale all input signals so that they would strictly remain within the 0-3 V region. If an ADC input signal were to be outside of this region, even for a very short time, the ADC circuits would most likely be damaged. The level of noise introduced by the inverter was important; and ADC damage was definitely an inconvenience, since the whole processor had to be changed.

7.2.3 Encoder Interface

The DSP has dedicated circuits that can be interfaced with a digital encoder, which is the type of position sensor used for this project. The encoder outputs, which are two square-waves in quadrature, are decoded so that they either increment or decrement the counter value in a timer. The DSP detects the rotor movement direction from the leading of the two signals and the timer's value is updated accordingly. In addition to this, the encoder had an index output, which is a digital output that gives a small pulse once per revolution. This allows the program to get absolute rotor position. This required an additional offline test where the motor was spun with open terminals. The machine phase to phase back-emf was measured with an oscilloscope and was displayed with the encoder index output. The result is shown in Figure 7.4.

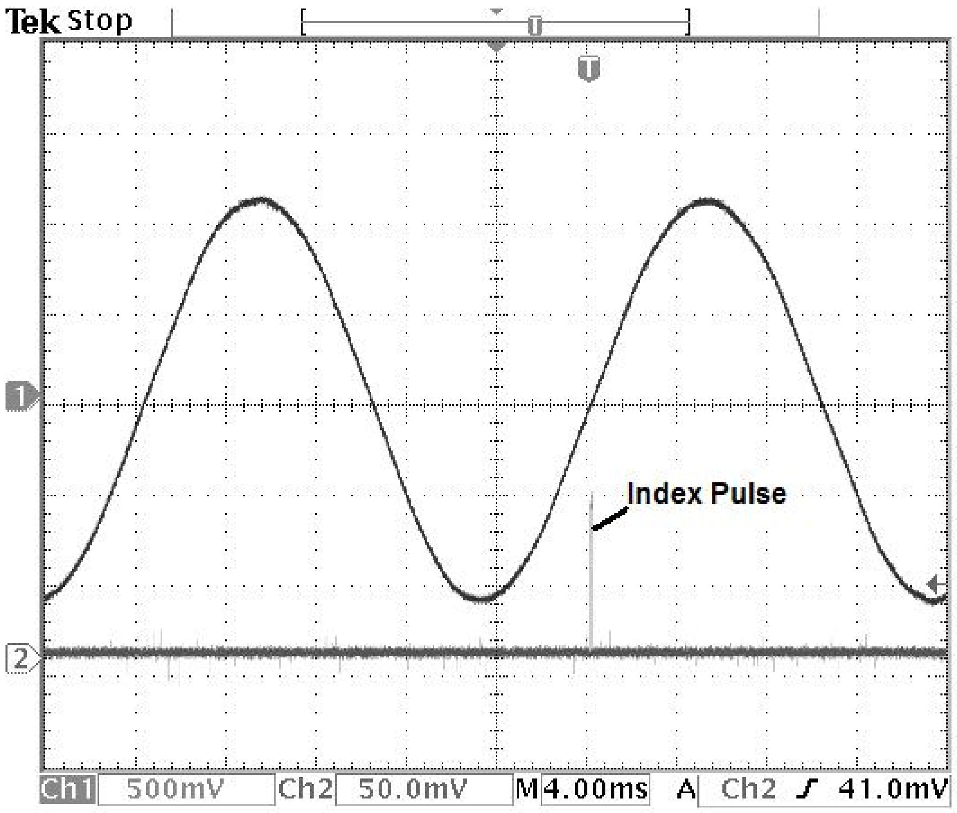

Fig. 7.4: Machine bench test result.

It is possible to see from Figure 7.4 that the index pulse of the encoder was synchronized with the zero crossing point of the phase to phase (a-b) back-emf of the machine. The zero position chosen for the Park transform is the zero crossing point of the phase “a” back-emf, which could not be directly measured (neutral point not accessible), but is 30 electrical degrees away from the phase to phase zero crossing. As a result, when the DSP receives a pulse from the associated encoder output, it executes an interrupt function that resets the controller position at 30˚. This allows the controller to avoid accumulation errors.

The test that was conducted in order to get Figure 7.4 was also used to get an estimate of the torque constant. The peak value of the phase to phase back emf is directly related to the torque constant value by equation (7.4).

$$K_T = \frac{V_{peak}}{\sqrt{3} \cdot \omega_r} \tag{7.4}$$

where V_{peak} is the peak value of the phase to phase back-emf, and the motor speed is the one that was measured for the test (which can also be obtained from the back-emf frequency).

7.2.4 DSP Central Processing Unit

The 2812 DSP is a 32-bit fixed point processor that is able to process up to 150 millions of instructions per second (MIPS). It was chosen for this project because it is a processor that is becoming increasingly popular in the industry and it meets the performance requirements. However, working with a fixed point DSP makes it more difficult to calculate mathematical expressions that involve non-integer variables. A floating point DSP on the other hand would have solved this issue, but these are more expensive processors, and are consequently less popular.

7.2.4.1 Introduction to Fixed-Point Mathematics

In the software program that was written for this project, decimal numbers had to be represented in a special way because only integers can be directly represented in the

CPU. Depending on its range and the precision we would want with it, each variable was assigned a certain number of bits (binary digits) after their decimal point. A simple example given below explains how this was done.

Let us consider the calculation where the product of *A* and *B* for $A = 1.25$ and $B = 0.04$ is desired. If a precision of 4 bits after the decimal point for *A*, and 8 bits for *B* is chosen, the product can be written as

$$A = \frac{1.25 \cdot 16}{16} = \frac{20}{16} \text{ and } B = \frac{0.04 \cdot 2^8}{2^8} = \frac{0.04 \cdot 256}{256} = \frac{10.24}{256} \approx \frac{10}{256} = 0.0391$$

The number *A* can directly be represented by the integer “20” if we keep in mind that it has four bits after the decimal point. On the other hand, *B* can be rounded to “10” and we would have eight bits after the decimal point. A simple notation for that was introduced: $A_{q4} = 20 \text{ and } B_{q8} = 10$, where the subscript marks the precision associated with each number. It is possible to see that in the case of *B*, an error was introduced by this method. This error could be reduced by introducing more bits after the decimal point. For example, $B - B_{q8} \approx 10^{-3}$ and $B - B_{q10} \approx 4 \cdot 10^{-5}$. The product of the two variables can be obtained as follows:

$$A_{q4} \cdot B_{q8} = \frac{20}{16} \cdot \frac{10}{256} = \frac{200}{2^{12}} = 200_{q12}$$

The same example could be followed for the product of a “*q x*” variable and a “*q y*” one, which would always give a “*q (x + y)*” result. From the processor’s point of view in this example, the only operation necessary is the product of “10” and “20”, but the user has to take into account the precision of the result. For comparison purposes, $1.25 \cdot 0.04 = 0.05$, and $200_{q12} = 0.0488$. If the user wanted the result to be of any different precision, all that is needed is to multiply or divide by the appropriate power of two.

Divisions are a very time consuming operation for fixed point processors and have to be avoided at all costs. Fortunately, multiplying or dividing by a power of two is the same as doing a register shift left or right by the corresponding number of bits, which comes from the fact that the CPU operates in base two. An analogy can be made with dividing by “10” in base “10”, which amounts to shifting the decimal point to the left.

It was possible to see from this example that the error introduced by the rounding of *B* had an important consequence on the result of the product. It is a difficult problem to find the optimal precision to associate with each variable. One could think that having more than enough precision for each variable would be a solution, but this can also introduce problems.

To illustrate the difficulty, let us consider we are working with an eight bit CPU, which can only work with integers that are between 0 and 255 (or -128 to +127 if signed variables are used). If one attempted to increase the accuracy by increasing the precision on *B*, the result would be

$$A_{q4} \cdot B_{q10} = 20_{q4} \cdot 41_{q10} = 820_{q14}$$

The above result cannot be easily represented in the CPU because it is outside of the range of integers that can be represented, as opposed to the result obtained when *B* was a “q8”. Choosing the precision for each variable is consequently a compromise between the actual precision desired and the capacity of the CPU.

7.2.4.2 Known Limitations

Fortunately, the 2812 DSP has a 32 bit CPU which can effectively multiply 32 bit variables, because it has a 64 bit register that can be used to extract a result if it exceeds 32 bits. However, one very important limitation introduced in the program was from the interfaces with external hardware, the ADC and the position sensor.

The analog to digital converter has an effective precision that ranges from 9 to 12 bits, and the position sensor has a precision of 11 bits per electrical cycle. If we take the case of the ADC and current sampling, the worst case scenario gives us a precision of about 0.02 A. For such a case, it is useless to represent the associated integer variables with a precision greater than "q9". In the case of the rotor position, the maximum precision is around 0.003 rd and anything larger than a "q9" would also be useless. These precision limitations may have an effect on the precision of calculations that are present in the motor control programs.

7.2.4.3 Program Execution Times

A constraint associated with real time implementation of a program is that its different elements have to execute fast enough to sustain the sampling rate and not interfere with each other. In the final program implementation, there were two sorts of programs that were implemented:

- Programs that would be synchronized with the PWM or another event, such as data acquisition and treatment, current control, PWM and parameter estimation algorithms. The encoder index could also trigger a program to reset the rotor position. These programs are executed from an interrupt routine which is triggered when specific events happen. In the case of PWM synchronization, timer period match and underflow were used, and for the encoder index, an input capture interrupt was used.

- Programs that would not require execution at specific times. The main example for this is the serial communication interface that was implemented to transmit data to a computer (to display results and debug). The communication program would run with a low priority and its execution would be stopped by any interrupt event and resume when the DSP would have executed the interrupt program. DSP diagnostics and other non time critical programs could be executed like that.

The way various program sections were executed as a function of time is explained in Figure 7.5.

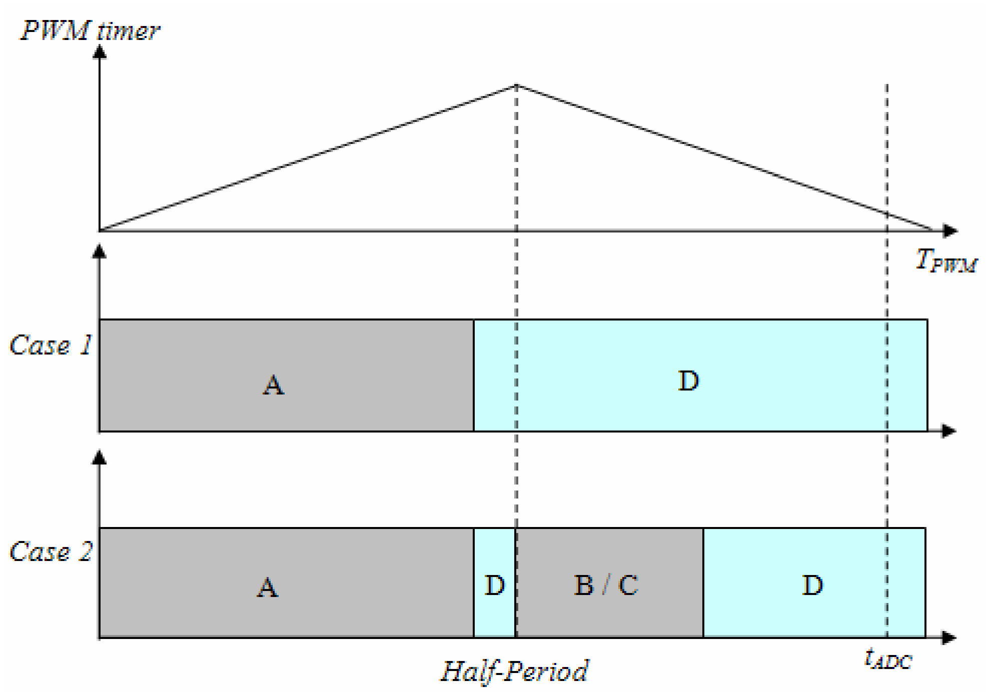

Fig. 7.5: Program executions as a function of time.

In Figure 7.5, different letters correspond to different types of programs:

- A: This type of program includes high priority time critical applications. They are executed on a timer underflow (zero value) interrupt, at the beginning of the PWM period. These programs include: ADC result treatment, Park transformation of currents, position and speed update, current control, PWM algorithm, voltage compensation and "fast" parameter estimation algorithm. The total maximum execution time for this block is around 19 μs.
- B/C: The "slow" parameter estimation algorithm was split into two blocks because of its high execution time, so that time was available for "D" programs to execute in the PWM period. In addition to the parameter estimation algorithm, the speed control program could also be included in these blocks. These programs were chosen to be synchronized with the timer period interrupt and consequently have a high priority. Execution times are around 7.9 μs for "B" and 11.4 μs for "C".
- D: Most of the programs included in this block have a low priority and stop their execution whenever an interrupt occurs. There are two exceptions for that. The first one was with the interrupt associated with the encoder index, which resets the rotor position and has a very small execution time. The second one is the interrupt associated with the reloading of the serial communication transmit register, whenever its buffer is empty. The other programs in this category are not time critical

and include data conversion and serial transmission to the computer. They could also include diagnostics and a supervising program.

Figure 7.5 contains a lot of information about how the different portions of the program were arranged together in order to provide smooth execution. The top diagram shows the value that the PWM counter takes as a function of time. A symmetric PWM carrier was chosen for this project, and the PWM timer counts up and down from zero to a chosen period value that sets the PWM frequency.

The other two diagrams show how programs are executed in two cases. Case "1" corresponds to program execution that occurs most of the time, almost every 50 µs. Case "2" happens when a program needs to execute at a specific frequency lower than the PWM frequency, like the "slow" identification algorithm. For example, if a frequency of 2 kHz was chosen for the "slow" identification program, case "2" would take place two times (one for B and another for C) every ten PWM cycles.

The time noted "t_{ADC}" in Figure 7.5 corresponds to the time at which ADC conversions are triggered. This time was chosen so that the data used in the "A" programs would use measurements that would be as recent as possible.

The software program was written with special care so that they would execute as fast as possible. Time consuming operations, such as divisions, trigonometric functions and repetitions were avoided as much as possible. For example, the inversion of a 2x2 matrix was calculated by multiplying each element of the transposed comatrix by the inverse of the original matrix's determinant (calculated once), instead of dividing each transposed comatrix element by the determinant (four divisions). It is very important that the various program elements do not “overlap” because that could lead to program instability and the loss of synchronization with PWM.

7.3 Feedback Circuits

The last experimental design part that needs to be mentioned in this chapter is the analog feedback circuits. As stated earlier in the chapter, the analog inputs to the DSP have to be strictly within the 0 – 3 V range. The sensor outputs of the analog values that were measured were not in that range and additional circuits had to be designed to adapt them.

7.3.1 Current Sensing

In the case of current sensing, Hall effect sensors were used and their output is a bidirectional current that is proportional to the current they are sensing. This current had to be converted into a voltage that would be within the ADC input range. Figure 7.6 shows the analog circuit that was designed for this purpose.

The output of the Hall effect sensor had to be converted first into a voltage. This was done with a simple resistor “R_1” whose value was designed to introduce a gain such that the largest possible phase current would not induce a voltage larger (in magnitude) than 1.5 V. This voltage was consequently proportional to the machine phase current and could be negative, because of the sinusoidal nature of the machine currents. It was then necessary to shift this voltage by 1.5 V and create a virtual zero for the DSP analog to digital converter.

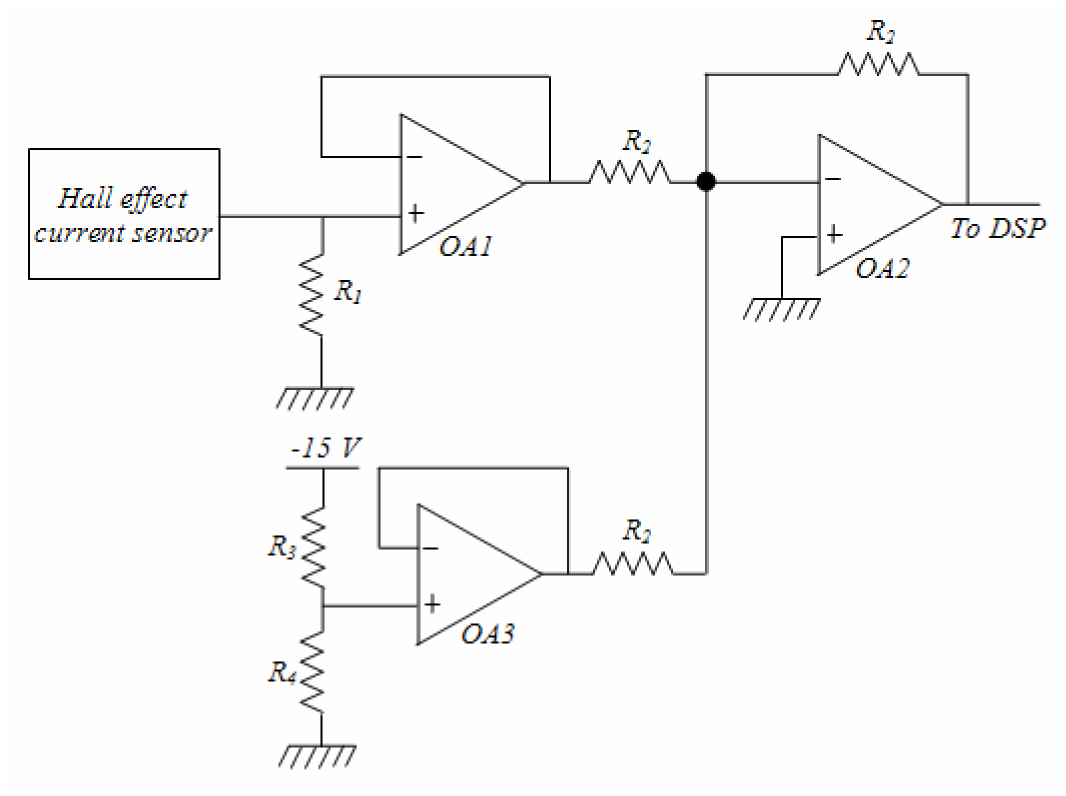

Fig. 7.6: Analog current feedback circuit.

The "*OA1*" and "*OA3*" operational amplifiers have a "follower" function, which means their output is equal to their input, but they load their input in a minimal way. The voltage divider at the input of "*OA3*" is set to provide a 1.5 V reference. The resistors R_3 and R_4 were chosen so that

$$-15 \cdot \frac{R_4}{R_3 + R_4} = -1.5$$

The last operational amplifier, *"OA2"*, is an inverting adder whose gain is simply "-1" in this case. It is important to note that for a sinusoidal input current, the output will be a sinusoidal voltage oscillating around 1.5 V, with a 180 degrees phase shift from the original current.

Two of the circuits shown in Figure 7.6 were used for this project. Upon initialization, before the machine is started, the DSP program measures the input voltages coming from these circuits and saves them. These values, which are around 1.5 V, correspond to a zero current and will be subtracted from the later measurements in order to obtain current samples oscillating around zero. In addition to this, the DSP will also have to invert the obtained waveform to compensate the inverting gain of "*OA2*".

7.3.2 Voltage Sensing

In the case of the DC bus voltage feedback, the problem was simpler, because it is a strictly positive voltage. The only necessary operation was to scale down the high voltage such that ADC ratings are never exceeded. This was done with a combination of a voltage divider and a voltage follower, as shown in Figure 7.7.

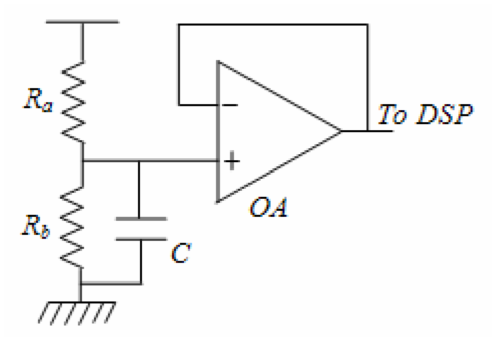

Fig. 7.7: Analog voltage feedback circuit.

The resistors in this circuit (Figure 7.7) were chosen according to the following inequality

$$V_{DC\max} \cdot \frac{R_b}{R_a + R_b} < 3$$

7.3.3 Feedback Filtering

The presence of the capacitor "*C*" in Figure 7.7 is the only difference of this circuit when compared with the current feedback circuits. It was mentioned throughout this dissertation that noise was an important issue for this project. The capacitor, combined with the two resistors, provides a low-pass filter that helps improve the feedback quality.

On the other hand, such an approach could not be taken with the current feedback circuits, because the machine currents are sinusoidal waveforms, and would be affected by a phase shift (lag) if a low pass filter was applied to them, even in the steady state. A software filter was chosen in the case of the currents. However, it did not act on the three phase currents but rather on the *d-q* currents obtained from them with the Park transform. In steady state, the effect of this filter is to reduce the noise level in the current feedback. It only has a small adverse effect during transients.

7.4 Conclusion

This chapter has introduced the reader with the experimental design and the various issues that had to be resolved for this project. The introduction of the inverter made it necessary for some of its non-linearities to be compensated. The DSP and its features were presented, as well as the feedback hardware. The next chapter will focus on experimental results that were obtained with this experimental setup.

CHAPTER VIII

PARAMETER ESTIMATION EXPERIMENTAL RESULTS

This chapter presents the results that were obtained with the experimental design in order to verify the effectiveness of the proposed parameter estimation algorithm. These results include data from the “fast” algorithm either alone or combined with the “slow” one. Current controller performance improvements are also shown.

8.1 Time Scale Uncertainty

It was mentioned in Chapter VII that the serial communication program that was used to retrieve data from the DSP was executed in an asynchronous way. This means that depending on the tasks with a higher priority that the DSP has to execute, the transfer rate will not be constant. The data transmission software was programmed in two parts:

- The first one accepted a number of integer inputs, corresponding to the value of variables to be transmitted (in a preset order), and converted them into ASCII (American Standard Code for Information Interchange) code. For example, the number “73”, which is coded by “0100 1001” in the DSP, will be coded as two characters, “7” and “3”, which are represented respectively by “0011 0111” and “0011 0011”. This conversion made it easier on the PC side to retrieve data and store it with a minimum processing, but put an additional burden on the DSP.

Different variables could be separated by a “space” character and data corresponding to different times could be separated by a “carriage return”. The program used on the computer side should use the same norms.

- The second part configured the DSP serial communication hardware so that it would send the characters that had to be transmitted. After the transmission of the first characters, it uses an interrupt to reload the transmission buffer register until the data corresponding to a full message has been sent. Once this has been done, it goes back to the first program which converts the next message data. The serial data transmission speed was set to 115200 baud, or bits per second, but with an additional start bit, a stop bit, and a parity bit.

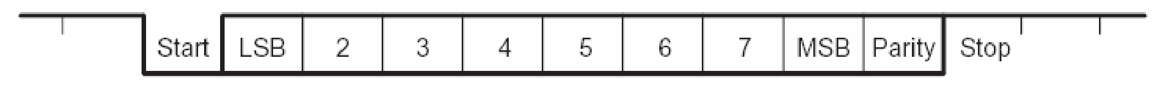

Fig. 8.1: Transmission of one data byte.

Figure 8.1 shows the transmission of one byte of data through the serial communication hardware. When the serial line is at rest, it is at a “high” state. The “start” bit notifies the receiver that transmission is starting, and the eight bits that follow are the actual data to be transmitted, one at a time, where LSB is “Least Significant Bit” and MSB is “Most Significant Bit”. The parity bit is an interesting feature that prevents some transmission errors. Both transmitter and receiver count the number of “high” states in the data byte, and if for example, an even parity was chosen, the parity bit will be at zero if that number is even. The transmitter sends this bit, and the receiver compares it to what it counted; if there is a mismatch, the data byte is discarded. The stop bit notifies the receiver of the end of transmission of the current data byte. One can see that it takes 11 bits to send an 8 bit piece of data. With a transmission rate of 115200 baud, the maximum byte transmission rate is around 10 kHz. The number “73” takes two characters and will consequently take at least 0.2 ms to send.

A consequence of this structure is that depending on how busy the CPU is and how long the variables to be sent are, the effective transmission rate will be affected. A larger number will take longer to convert than a smaller one, and will also require more characters (bytes) to be sent. Consequently, it was difficult to get an exact estimate of the times to which each data sample corresponded on the PC side. A solution to this problem would be to include a variable in the transmitted data that would indicate the exact time at which the sample was taken, but this would further decrease the transmission speed. The solution chosen was to have only a rough estimate of the transmission rate as a function of the number of integer variables sent. Time was not a critical piece of information for this project and an estimate of it was enough for most results. These variables were scaled to a reduced precision so that only the most important bits would be sent. For most of the results shown in this chapter, the transmission speed measured was around 500 Hz. The variables sent through serial communication are in integer format, and were then scaled to decimal numbers depending on their precision.

The low transmission speed and time uncertainty was an important hardware limitation for this project. However, if the time scale is not exactly accurate, results shown in this chapter were synchronized together. In other words, for each of the plots that follow, all the data was taken with the same time scale.

8.2 Current Controller Performance

This section presents the results obtained with the current controller in order to verify the need for accurate parameter feedback for optimal performance. These results can be related with the simulation results shown in section 3.1.2 of Chapter III. The machine was run in current control mode (i_q = 1.5 V and i_d = -0.5 V) with a load that limited its speed at 1000 RPM. After steady state was reached, a perturbation was introduced in the load that affected the machine speed. This perturbation was introduced in the form of friction and was not exactly repeatable. However, the conclusions that will be drawn are not affected by this. The effects of this perturbation were then studied in different cases of current controller parameter feedforward.

The equations related to these terms are shown here for the sake of clarity:

$$\begin{cases} V_q^{'} = V_q - \omega_e \cdot L_d \cdot i_d - \omega_r \cdot K_T = r_s \cdot i_q + p \cdot L_q \cdot i_q \\ V_d^{'} = V_d + \omega_e \cdot L_q \cdot i_q = r_s \cdot i_d + p \cdot L_d \cdot i_d \end{cases} \tag{2.16}$$

The following are referred to as feedforward terms

$$\begin{cases} V_{q\,ff} = L_d \cdot \omega_e \cdot i_d + K_T \cdot \omega_r \\ V_{d\,ff} = -L_q \cdot \omega_e \cdot i_q \end{cases}$$

Figure 8.2 shows the results obtained in the case of a current control that does not include any feedforward term ($V_{qd}^{'} = V_{qd}$). The error obtained from the difference between reference and feedback current went through a PI controller and its output was used as a voltage command for both *q*- and *d*- axes. The speed perturbation was of fairly small magnitude in this experiment, but one can see that the *q*-axis current control was directly affected by it.

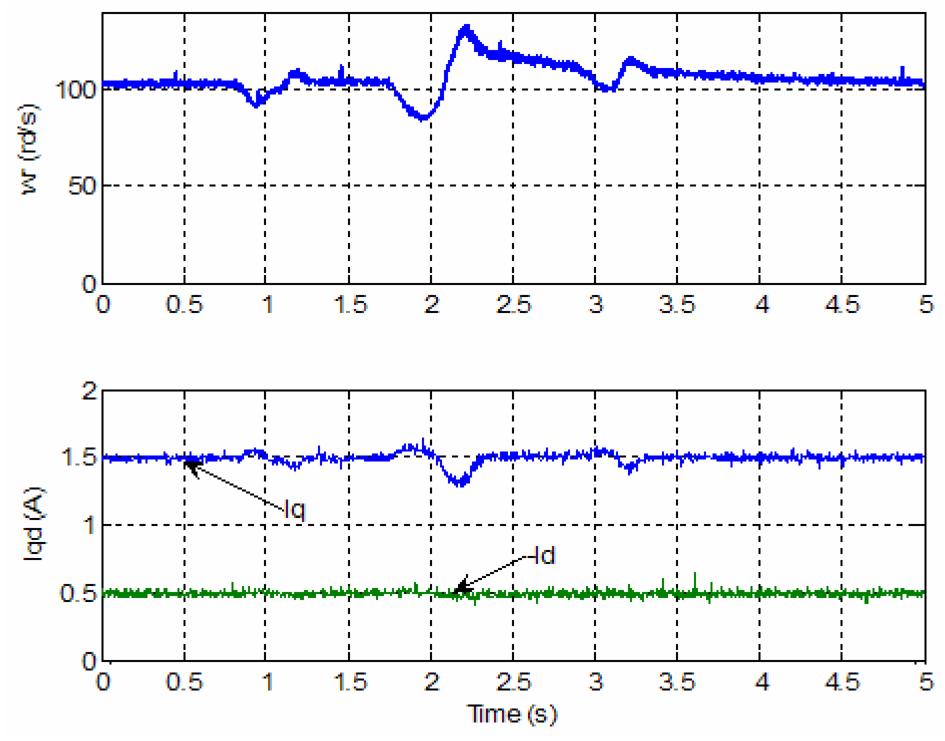

Fig. 8.2: Current controller response without feedforward term.

In Figure 8.2, the *d*-axis current shows small variations due to the perturbation, and they are negligible. The difference in *d*- and *q*- axes currents for the chosen operation arises from the dominating back-emf term in the electrical model of the machine compared to the other terms. This back-emf term only affects the *q*-axis and is directly proportional to the rotor speed. For example, if the speed decreases suddenly, the back-emf will decrease, and the current controller PI will have to compensate that voltage to maintain the *q*-axis current constant. However, its response is not instantaneous and this is why we can see perturbations in the *q*-axis control. For example, at 2.2 s, the machine shows a fast acceleration and the current decreases from its set point. In this Figure, at 2.2 s in Figure 8.2, a variation of 27% in speed induces an error of 11% in current control.

In the case of the *d*-axis current, the only term that could affect current control is the inductive drop $L_q \cdot \omega_e \cdot i_q$, but it is small and its variations are quickly compensated for by the PI controller.

To prove the effectiveness of a current controller with accurate feedforward compensation, another test was conducted in similar conditions but with feedforward compensation and its results are shown in Figure 8.3.

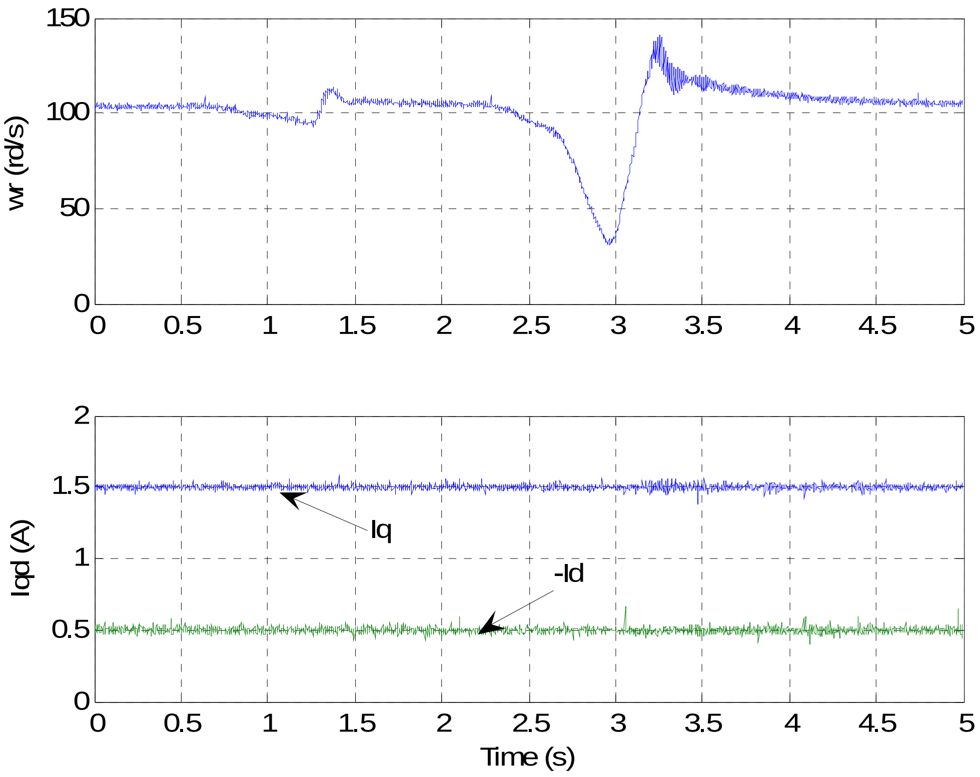

Fig. 8.3: Current controller response with feedforward.

In the plots shown in Figure 8.3, a small speed perturbation was followed by a very large one (70%). The *d-q* currents were not affected at all by this perturbation and remained constant and equal to the reference currents. The reason is the speed perturbation does not change anything from the PI compensator's point of view. The difference in voltage required by a change in speed is directly taken into account by the feedforward terms and is transparent from the compensator's viewpoint. In this last case the perturbation was quite large, and one can imagine that a controller without feedforward terms would have been strongly affected.

One last test was then conducted to check the effects of poor parameter estimation on controller performance. In this test, a 30% error on the torque constant was introduced in the feedforward compensation. The corresponding results are shown in Figure 8.4.

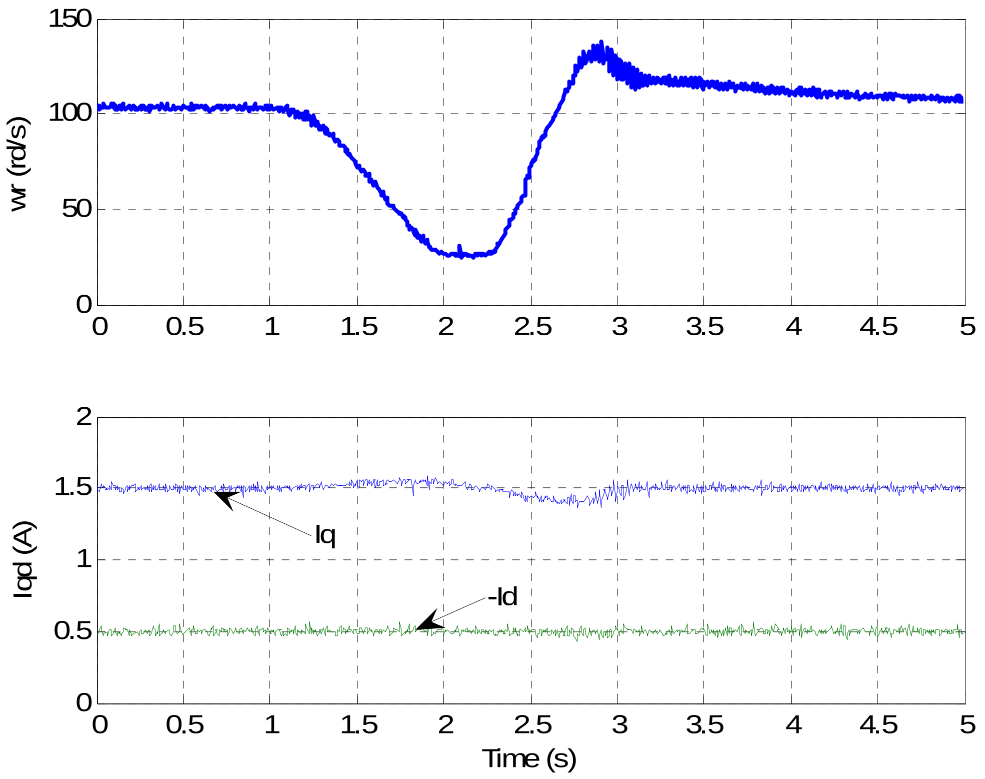

Fig. 8.4: Current controller response with feedforward (30% K_T error).

Figure 8.4 shows that the torque constant error makes the controller sensitive to changes in speed. However, when these results are compared with those of Figure 8.2, one can consider that the problem has been compensated by at least by 70% (70% from torque constant plus the inductive drops). It is also possible to see that the error seems to be more important when the acceleration or deceleration is most important. For example, the error introduced in Figure 8.4 from 1.2s to 2s is smaller than the one introduced from 2.3s to 2.8s. The reason for this is that in the case of a slow variation, it will be easier for the PI controller to compensate the perturbation.

The test results shown in figures 8.2, 8.3 and 8.4 show that for optimum current controller robustness, it is important to have accurate parameter estimates with a feedforward compensator for optimum current controller robustness. For applications where speed and torque transients are not important, such as pumps and fans, this might not be necessary. On the other hand, high performance servo controllers can take advantage of this feature.

8.3 Inductance Estimation

8.3.1 Steady State Operation

Several results that were used to build the inductance tables shown in Chapter VI (tables 6.2 and 6.3) were compared with results obtained using the “fast” identification algorithm with the experimental hardware. A similar table was built to verify the effectiveness of this part of the algorithm in the hardware.

Table 8.1 shows results obtained from the “fast” estimation algorithm for the same tests that were used to build tables 6.2 and 6.3. The currents are shown in Amperes, and the internal cells give the estimated inductances in mH, and the percentage of error between these results and the ones obtained in Table 6.2. This table shows a good match between estimated and calculated inductances, especially for larger currents, where the signal-to-noise ratio gets larger. The cells that were not filled correspond to results that were extrapolated in Table 6.2.

Table 8.1: q-axis inductance estimated by "fast" algorithm.

$i_q \setminus i_d$	1	1.5	2	2.5	3
1.25	9.26 – 3.5%	N/A	10.4 – N/A	10.95 – N/A	11.83 – 3.2%
1.8	9.5 – 4%	9.71 – 4.1%	10.3 – 1.7%	10.85 – 0.4%	11.42 – 0.7%
2.35	9.33 – 5.6%	9.83 – 2.7%	10.1 – 3.2%	10.55 – 2.1%	N/A
2.9	9.21 – 5.8%	9.5 – 5%	N/A	N/A	N/A

8.3.2 Initial Convergence

An experimental test was conducted to show how the "fast" estimation algorithm converges towards a satisfactory estimate for machine inductance. The machine was started in current control mode with its speed set at 1000 RPM, and the parameter estimation program started after the initial transient. The q-axis current was set at a 2.5 A reference and the d-axis current was set at -0.5 A. Figure 8.5 shows the associated results.

In Figure 8.5, the top plot shows estimation results obtained from this initial convergence test for a 0.028 s window, and the bottom plot is a zoom on the first part of the top plot. In this test, the serial communication program had to be changed in order to be able to see the algorithm operate at the PWM frequency. The samples that had to be transmitted were stored in a memory table in the DSP at the PWM frequency, starting from the time when the "fast" algorithm was activated. Once the table was filled, the communication program was started and transmitted the data at low speed. This way, it was possible to extract a short window of high frequency data with an exact time scale.

The DSP memory was too small to be able to use this method for an extended amount of time, and would be a limitation for other experimental results shown in this chapter.

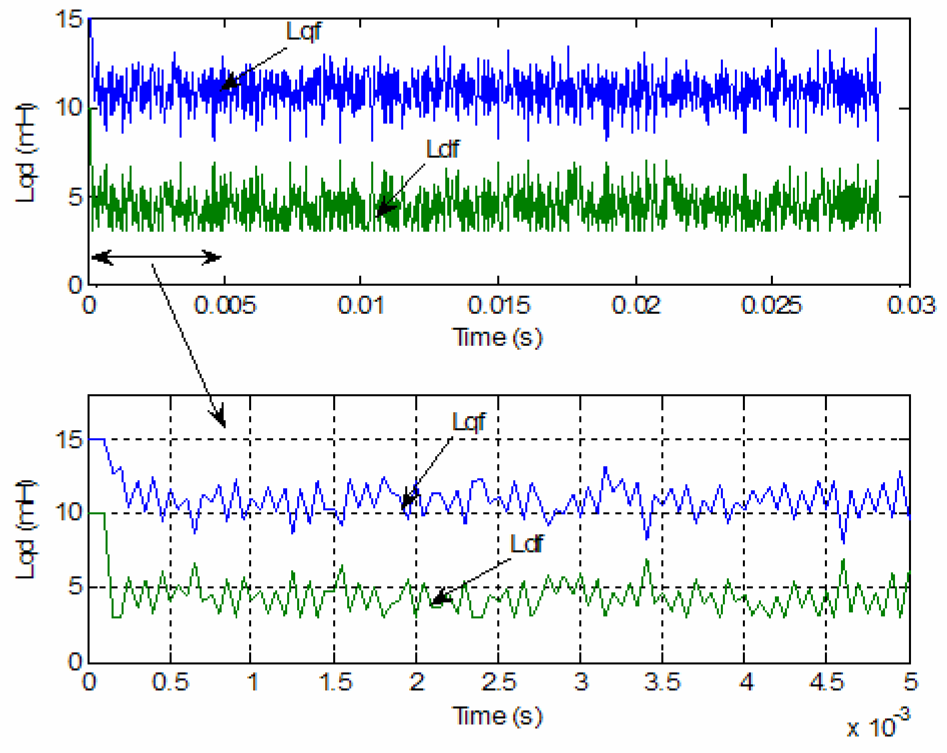

Fig. 8.5: Initial convergence of “fast” algorithm.

It is possible to see from Figure 8.5 that algorithm convergence was very fast, taking less than ten PWM cycles to converge towards its final value from initial values of 15 mH for the *q*-axis inductance and 10 mH for the *d*-axis inductance. The algorithm was able to converge even in the case of a large initial error, because of the inherent robustness of the fast algorithm. It is also possible to see the high level of noise present in the estimation, which is mostly due to the noise level in the current feedback. A low pass filter was later used when the “fast” estimation algorithm was interfaced with other parts of the controller program.

8.3.3 Tracking Ability

The "fast" estimation algorithm was tested for transient operation. In the following experiment, the machine was run at constant speed under current control (speed limiting load), and a step in *q*-axis current was introduced. The *d*-axis current was set at zero for this test. The step went from one-third to about two-thirds of the machine rated current. The current waveform and the corresponding *q*-axis inductance estimate are plotted in Figure 8.6.

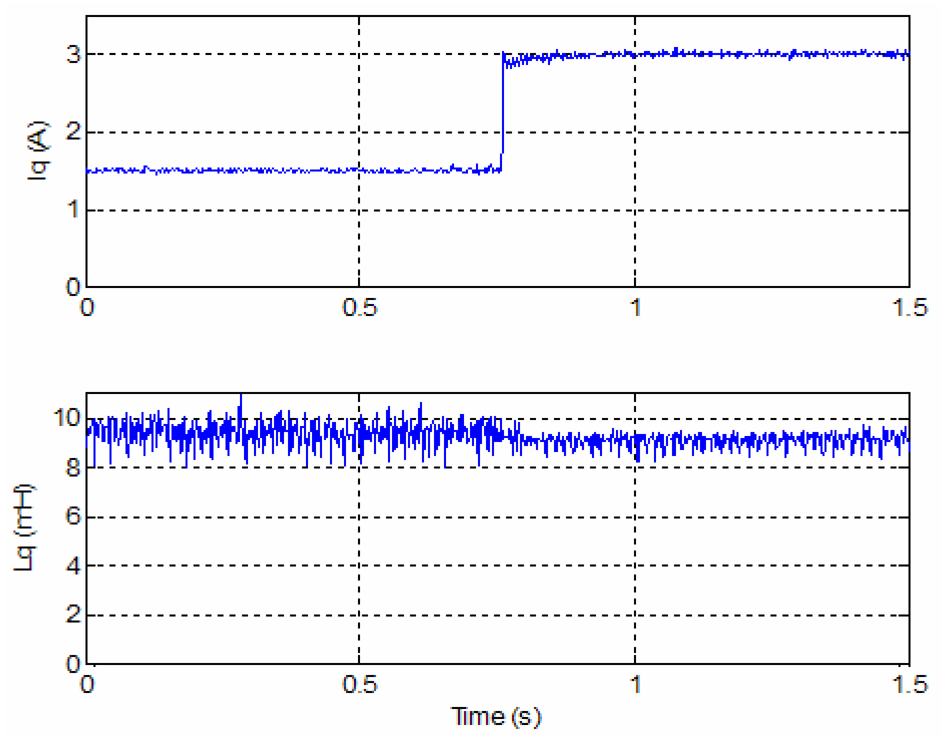

Fig. 8.6: "Fast" algorithm step response.

The top plot in Figure 8.6 shows the filtered *q*-axis current in the machine for this test; it is a step waveform from 1.5A to 3A. The bottom plot shows the corresponding *q*-axis inductance estimated by the "fast" identification algorithm. It is possible to see a slight change in the inductance estimate, as well as a reduction in its noise level. This reduction comes from the fact that the current magnitude was doubled, thereby reducing the corresponding signal-to-noise ratio. The inductance waveform had an average around 9.35mH before the step, and 9.07mH after the step. These values roughly match the ones found in Table 6.2.

8.4 Complete Algorithm Tests

8.4.1 Current Waveforms

The results shown in the remaining sections of this chapter correspond to current control operations, with a constant or varying *q*-axis current, and a three-level perturbation *d*-axis current with a frequency around 10Hz. A typical corresponding current waveform is shown in Figure 8.7.

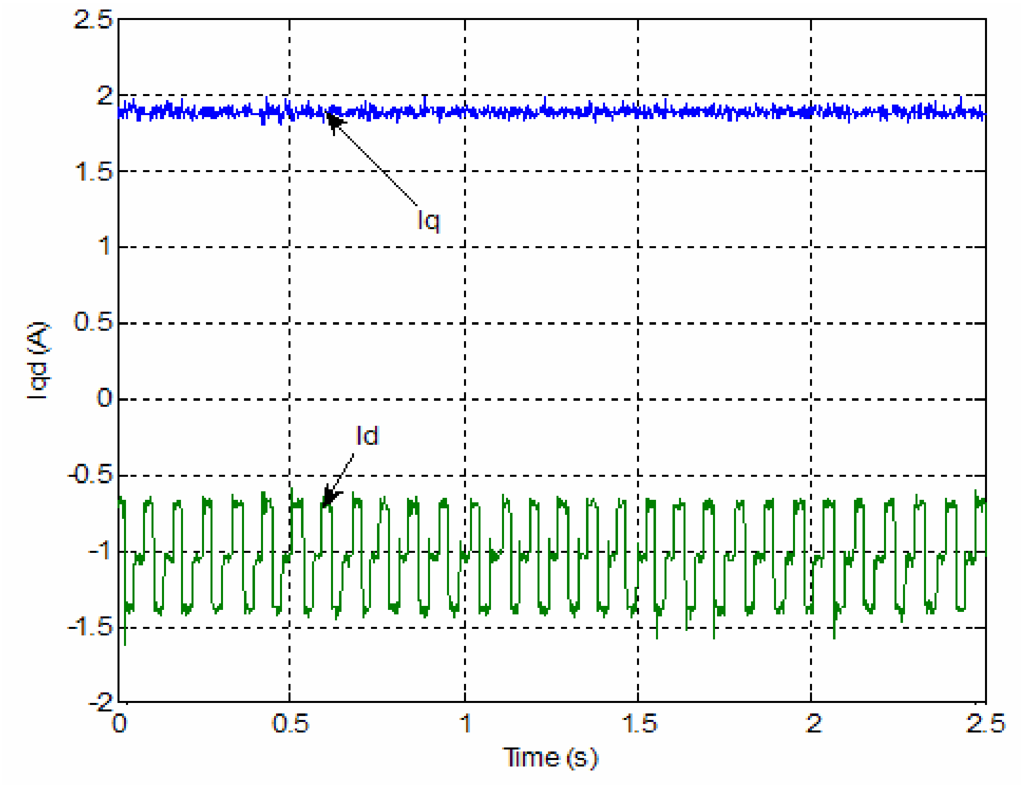

Fig. 8.7: Typical current waveforms.

8.4.2 *d*-axis Inductance Estimation

It has been shown in Chapter VII that the *d*-axis inductance was the most sensitive parameter that was to be estimated by the algorithm. In addition to this, it is also the smallest one, and corresponds to the smallest term ($\omega_e \cdot L_d \cdot i_d$) in the electrical model equations. For the machine used on the experiments, the saliency ratio (L_q / L_d) was fairly small: as a consequence, this machine would require a fairly small *d*-axis current for optimum operation. This made it all the more difficult for the parameter estimation algorithm to extract this parameter. Several attempts were made to estimate the *d*-axis inductance for low *d*-axis current values but they did not yield satisfactory results.

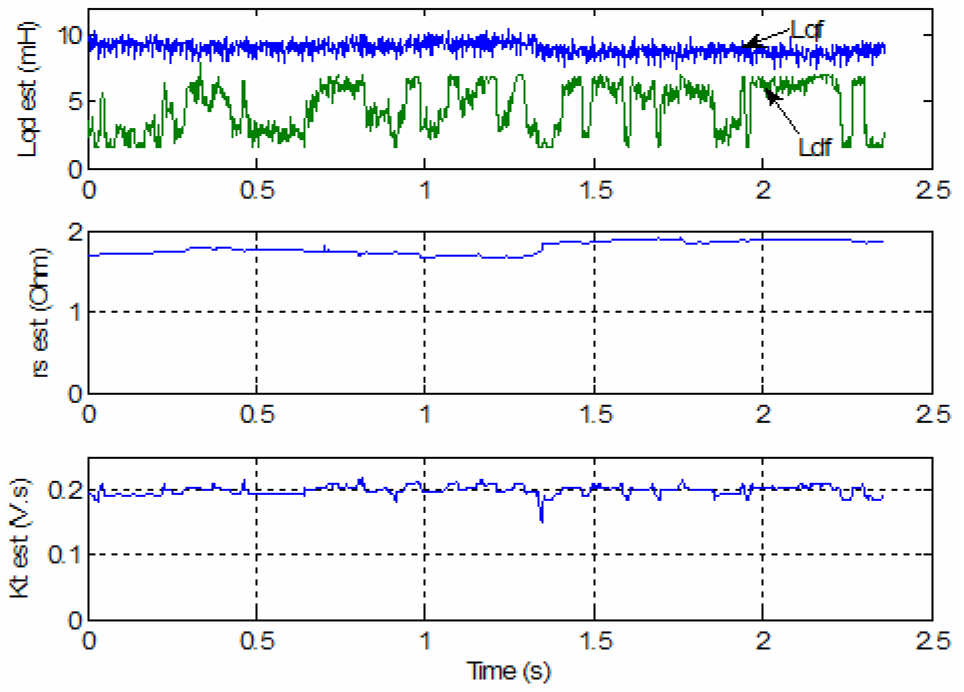

Fig. 8.8: *d*-axis inductance estimation problem.

Figure 8.8 shows the results obtained for closed loop parameter estimation of all four machine parameters. The two identification algorithms were interfaced in the way that has been presented earlier in Chapter VI. All four parameter estimates are shown in the plots, and it can be observed that three out of the four parameters were relatively stable and remained in the vicinity of the actual motor parameters (resistance around 1.55Ω at 25˚C, torque constant around 0.2V.s at 25˚C and *q*-axis inductance around 10mH). On the other hand, the *d*-axis inductance estimate is very unstable and bounces off the boundaries of its estimation region. A positive conclusion that can be drawn from this plot is that the poor estimation of the *d*-axis inductance does not seem to affect the other parameters in a significant way. The reason for this is the rather small value of the inductive term in the *q*-axis electrical equation. However, it does introduce a problem that could possibly destabilize the algorithm and would disrupt controller operation. A consequence of such results is that for the experimental machine, the estimation of the *d*-axis inductance had more of an adverse effect than a beneficial one. In addition to that, the fact that we know the machine would operate at a low *d*-axis current in most cases allows us to think that we could consider this parameter to be constant at a value obtained from Table 6.3 (or given by the motor manufacturer).

If a different motor had been considered, such as machine "B", the *d*-axis inductance estimation would have been less of a problem. Figure 3.3 in Chapter III showed that this machine operates with a larger *d*-axis current (due to the higher saliency ratio) and usually such machines are designed for high speed operation. Therefore the corresponding *d*-axis inductance term would be easier to estimate than for machine "A" which is the machine used in the experiments.

In the results that follow this section, the *d*-axis inductance estimate was set to be constant in the "fast" estimation algorithm.

8.4.3 Initial Convergence

The complete algorithm was tested under current control mode. Results corresponding to its initialization will be shown in this section. The machine was started and the “fast” parameter estimation algorithm was activated; after a delay, the “slow” parameter estimation algorithm was enabled and the complete algorithm was run in the way described earlier in Chapter VI. The algorithm was started with a large error on the initial estimation of the stator resistance (about 30%).

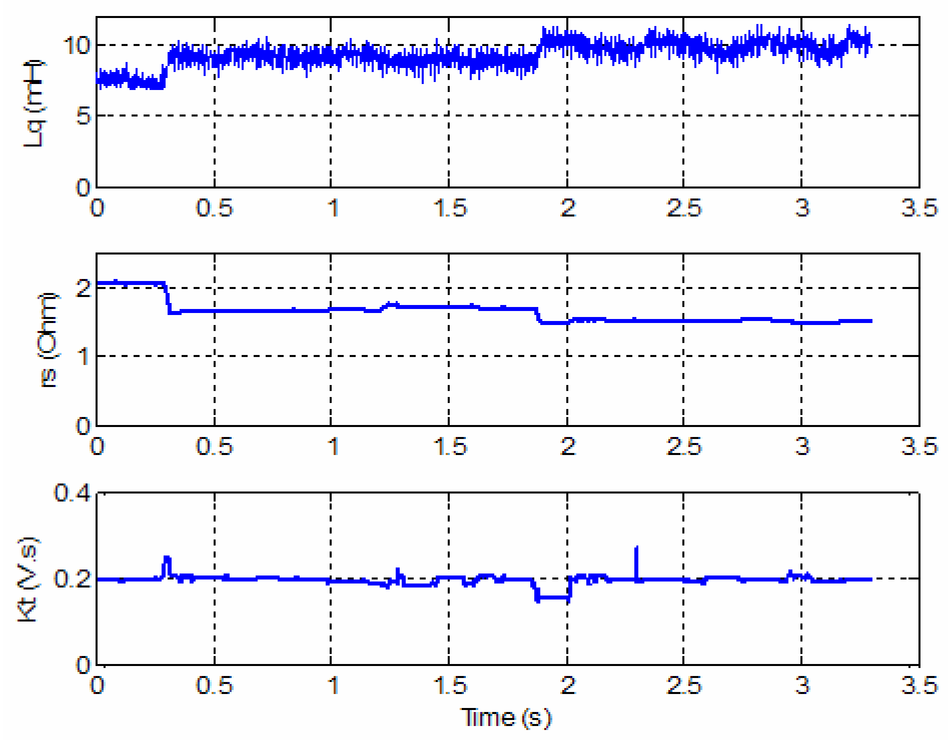

Fig. 8.9: Initial convergence of complete algorithm.

The parameter estimation results corresponding to this complete algorithm experiment are shown in Figure 8.9. At 0s, the “fast” algorithm was started and ran with the resistance error until the “slow” algorithm started at about 0.3s. The parameter estimation then converged towards its final estimates. It can be observed that the large error in stator resistance has a large effect on the initial *q*-axis inductance estimate.

However, as the algorithm converges and the stator resistance estimates improves, there is a direct improvement on the inductance estimation. Except for the glitch around 2s which was caused by the jump in resistance estimation, the torque constant estimation was fairly robust and stayed around the nominal value. The reason for this robustness is that the back-emf term $K_T \cdot \omega_r$ usually dominates the other terms in the *q*-axis electrical model, and is consequently less sensitive to noise. It also does not depend on the machine currents, which are the noisiest signals in the controller feedback.

The small glitch seen in the torque constant estimate around 2.3s is most likely an error introduced by the serial communication. If such an error occurred in the estimation of this parameter, it would have a large direct effect on the *q*-axis inductance (and possibly also on the stator resistance). The serial communication line was subject to a significant amount of noise (from both the inverter and mechanical load drive) and presented errors in some cases. Some of these errors were easily noticeable, such as impossible characters (letters and symbols), and others such as the one mentioned above where one or more of the digits in the transmitted data have changed.

8.4.4 Steady State Operation

Experimental results corresponding to a longer steady state operation are shown in Figure 8.10. For this test, the machine was run at constant speed and *d*-*q* currents. The *q*-axis inductance and the machine torque constant seemed to remain constant over the duration of this test. On the other hand, the machine resistance showed a slight increase which is explained by a rising machine temperature due to continued operation. The step-like appearance of the torque constant waveform comes from the fact that it was transmitted as a “q8” variable, which limits the precision to 4.10^{-3} but reduces the size of the transmitted variable. This discretization only affected displayed results, since the corresponding variable in the DSP program remained as a “q12” ($0.25.10^{-3}$ precision) variable.

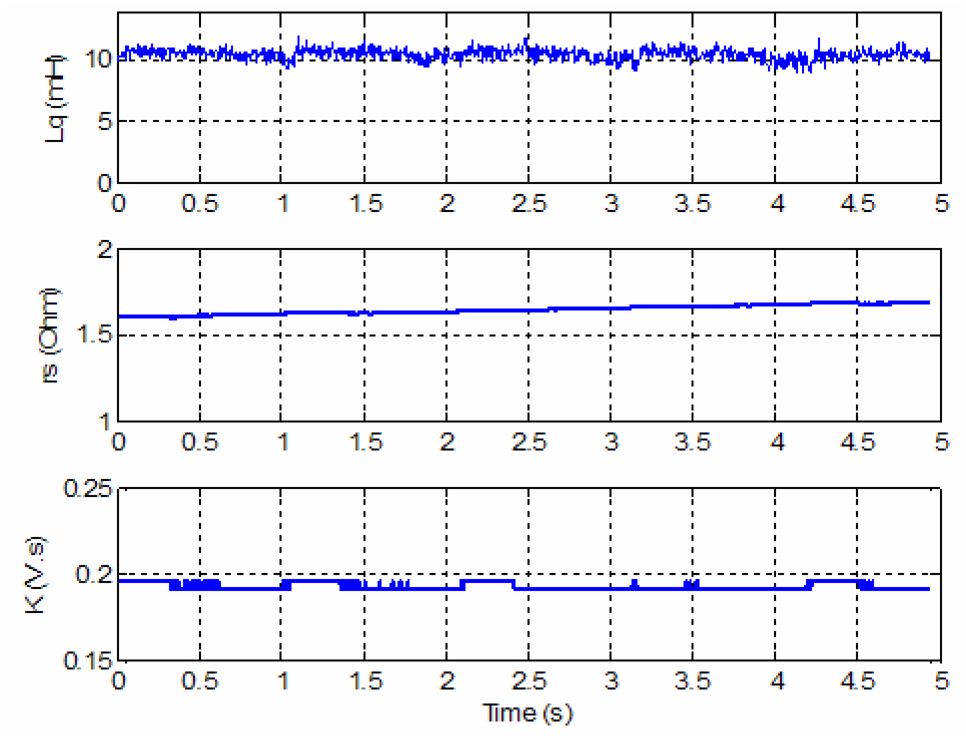

Fig. 8.10: Algorithm steady state operation.

8.4.5 Operation During Transient

Another experiment was conducted to verify the stability of the proposed algorithm under a machine transient operation. The corresponding results are displayed in Figure 8.11. The machine and algorithm were started under current control at 1000 RPM, and at the point in time that corresponds to 0s in the plots the *q*-axis current reference was increased from 1 A to 2 A with an external potentiometer connected to an ADC pin.

It can be observed from these plots that algorithm operation was not disrupted by the transient; the stator resistance and machine torque constant estimates were not affected by the change in the operating point. On the other hand, the *q*-axis inductance estimate diminished slightly as the current increased, because of saturation. There is however a small error in the inductance estimate from the values obtained from Table 6.2.

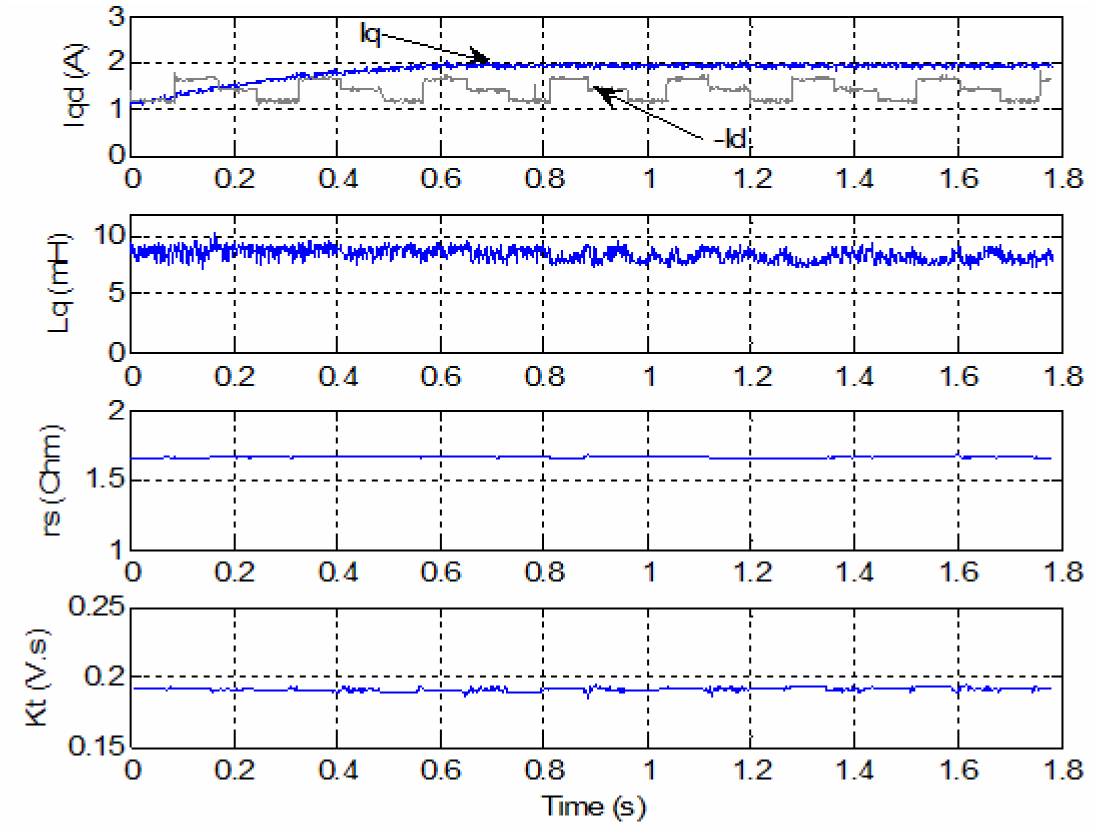

Fig. 8.11: Operation during transient.

One can notice that for this test the *d*-axis current perturbation was introduced at a lower frequency than in Figure 8.7. Different frequencies were tried for the project, and as a general rule higher frequencies are to be preferred because the mechanical load will filter them out and it will provide “richer” information to the “slow“ algorithm, but the current controller capability is the limitation for the high end of the *d*-axis current perturbation frequency. A perturbation frequency of 12 Hz was used in the experiment corresponding to Figure 8.11.

8.5 Conclusions

This chapter presented results obtained with the experimental machine of this project (machine “A”), which is an IPM with a low saliency ratio. Results have verified the performance improvement with accurate parameter estimates in the case of the current controller. The parameter estimation algorithm was also tested; its performance and stability were verified under different conditions. The “fast” algorithm was tested separately, and its initial convergence, steady state operation and transient behavior were verified. A similar treatment was done with the complete algorithm.

CHAPTER IX

CONCLUSIONS AND FUTURE WORK

9.1 Introduction

This dissertation focused on on-line parameter estimation in three-phase permanent magnet machines. The research motivation came from an analysis of the phenomena that affect machine parameters and the effects of parameter variation on motor drive performance (Chapter III). An on-line parameter estimation algorithm was developed in order to solve this problem and its effectiveness was verified.

This parameter estimation algorithm is based on a combination of two recursive least squares programs. One of them is dedicated to the estimation of the *d*-*q* machine inductances and is to be run at a fast rate. The other one is structured to estimate all four machine parameters at a slower rate. This latter program also requires external stimulation in the form of a *d*-axis current perturbation to operate properly.

A simulation model that represented the complete motor drive (motor, inverter and digital controller) was created in order to study the performance of the proposed algorithm. This model was also used to design the parameter estimation structure which helped solve numerous problems with the algorithm. The simulation model was used on a machine that was available for experiments so that simulation results could be verified experimentally.

Another machine was studied through simulation that had a larger saliency ratio, so that possible performance improvements for such machines could be studied. The simulation model allowed us to create test scenarios that could not have been implemented experimentally. It also enabled the user to isolate problems and test them separately. Such an approach is very important to the motor drive designer.

The parameter estimation algorithm was also tested experimentally with a low saliency IPM machine. A complete motor drive was designed using a DSP and inverter along with the necessary feedback circuits. The results obtained verified the stability and effectiveness of the novel algorithm.

9.2 Research Contributions

The unique contributions of this dissertation can be summarized as follows:

- Analysis of PMSM parameter variation
    - The sources of parameter variation and their characteristics were investigated
    - The effect of parameter variation was studied through simulation for both torque and current controllers
- Development of a new on-line parameter estimation algorithm
    - An algorithm was developed and first validated with simulation
    - The algorithm estimates all four machine electrical parameters and is able to track sudden changes in saturation as well as slow changes in temperature

  - The complete algorithm included inverter non-linearity compensation in both simulations and experiments

9.3 Limitations in Experimental Setup

The experimental design in the current setup presented several problems that affected the parameter estimation performance. The issues encountered are mentioned in the following.

- The analog to digital converter did not perform according to its specifications. It was extremely sensitive to noise and its effective precision was much lower than expected. This was an issue for the parameter estimation algorithm because it limited the numerical precision available for the measured variables. The ADC also presented significant gain and offset errors that had to be compensated. The structure of the analog circuits designed for this project may also have affected ADC performance because of grounding problems and the use of long wires.
- The hardware circuit designed for this project was not suitable for a high performance motor drive. The main reason for this was that the electrical ground for the power circuits (motor and inverter) was not isolated from the digital and analog grounds of the DSP. As a result, there were high levels of noise in the ground which affected the analog feedback. Consequently, the program required larger currents than it should have needed in order to run properly, so that the signal to noise ratio was satisfactory.
- The use of a fixed point DSP made program design more complex and may have introduced precision limitations in the parameter estimation algorithm.

- The serial communication developed to send data to a computer for debugging and displaying purposes was slow and sensitive to noise. This made data analysis difficult in some cases.

9.3 Suggested Future Work

There are a few extensions of the research accomplished in this project that may be of interest for future work. The first one would be to try the algorithm on a new experimental design, with adequate isolation between the control and power stages (such as opto-couplers) for better noise immunity. A different DSP, or a different ADC design than the one presented in the current project could also show improvements on the results.

It would also be interesting to try the algorithm on different machine types, such as a surface-mount machine or a machine with a high saliency ratio such as machine “B”. The algorithm, by its design, should operate independently of the machine type, but the benefits that could be obtained with it are machine dependent. This is also why machine “B” was studied through simulation. The benefits of using the algorithm on surface mount PMSM could also be of interest and improve knowledge and possibly performance of corresponding motor drives.

Finally, it was mentioned in Chapter II that core losses were omitted in the machine model and the parameter estimation algorithm. Although these are usually small and consequently neglected, they can become important in some cases and could potentially have an effect on machine parameter estimation. A further study of the effect of these losses and a possible new parameter estimation algorithm could be derived. This algorithm could be based on the structure that has been presented, with the same “fast” identification program dedicated to machine inductances and the “slow” one including all machine parameters, plus a new one representing core losses.

REFERENCES

[1] S. Evershed, “Permanent Magnets in Theory and Practice”, *IEE Journal*, Vol 58, No. 295, 1920.

[2] S. Evershed, “Permanent Magnets in Theory and Practice”, *IEE Journal*, Vol 63, No. 344, 1925.

[3] W. Kober, “Permanent Magnet Alternators”, Report E-1004, Signal Corps Engineering Laboratories, Bradley Beach, N. J., Sept 16, 1946.

[4] R. M. Saunders, R. H. Weakley, “Design of Permanent Magnet Alternators”, *AIEE transactions*, Vol 70, part II, 1951, pages 1578-1581.

[5] M. Burth, G. C. Verghese, M. Velez-Reyes, “Subset Selection for Improved Parameter Estimation in On-Line Identification of a Synchronous Generator”, *IEEE Transactions on Power Systems*, Vol. 14, No. 1, Feb 1999, pp. 218-225.

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

[7] P. Pillay, R. Krishnan, “Modeling, Simulation and Analysis of PM Motor Drives, Part I: the Permanent Magnet Synchronous Motor Drive”, *IEEE transactions on Industry Applications*, Vol. 25, NO. 2, March/April 1989.

[8] S. Morimoto, Y. Takeda, T. Hirasa, K. Taniguchi, “Expansion of Operating Limits for Permanent Magnet Motor by Current Vector Considering Inverter Capacity”, *IEEE Trans. on Industry Applications*, IA-26, 5, pp. 866-871 (1990).

[9] S. Morimoto, “IPM Vector Control and Flux Weakening”, appearing in “Design, Analysis, and Control of Interior PM Synchronous Machines”, *Tutorial course notes*, Chapter 8, IEEE IAS conference, Seattle 2004.

[10] K.-W. Lee, D.-H. Jung, I.-J. Ha, “An Online Identification Method for Both Stator Resistance and Back-EMF Coefficients of PMSMs without Rotational Transducers”, *IEEE Transactions on Industrial Electronics*, Vol. 51, No. 2, April 2004.

[11] C.-C. Wang, Y.H. Cho, "Effects of Leakage Flux on Magnetic Fields of Interior Permanent Magnet Synchronous Motors", *IEEE Transactions on Magnetics*, Vol. 37, No. 4, July 2001, pp. 3021-3024.

[12] B. Stumberger, G. Stumberger, D. Dolinar, A. Hamler, M. Trlep, "Evaluation of Saturation and Cross-Magnatization Effects in Interior Permanent-Magnet Synchronous Motor", *IEEE Transactions on Industry Applications*, Vol. 39, No. 5, Sept/Oct 2003, pp. 1264-1271.

[13] A. Kilthau, J.M. Pacas, "Parameter-Measurement and Control of the Synchronous Reluctance Machine Including Cross Saturation", Industry Applications Conference, 2001. Thirty-Sixth IAS Annual Meeting. *Conference Record of the 2001 IEEE IAS annual meeting*, Vol. 4, 30 Sept.-4 Oct. 2001 Pages:2302 – 2309.

[14] U. Schaible, B. Szabados, "Dynamic Motor Parameter Identification for High Speed Flux Weakening of Brushless Permanent Magnet Synchronous Machines", *IEEE Transactions on Energy Conversion*, Vol. 14, No. 3, Sept 99, pp. 486-492.

[15] K.M. Rahman, S.Hiti, "Identification of Machine Parameters of a Synchronous Motor", *Conference Record of the 38th IAS Annual Meeting*, pp. 409 - 415 vol.1.

[16] S. Morimoto, K. Hatanaka, Y. Tong, Y. Takeda, T. Hirasa, "Servo Drive and Control Characteristics of Salient Pole Permanent Magnet Synchronous Motor", *IEEE Transactions on Industry Applications*, Vol. 29, Issue 2, pp. 338-343.

[17] S. Morimoto, T. Ueno, M. Sanada, A. Yamagiwa, Y. Takeda, T. Hirasa, "Effects and Compensation of Magnetic Saturation in Permanent Magnet Synchronous Motor Drives", *Conference Records of the 1993 IEEE IAS Annual Meeting*, Vol. 1, pp. 59-64.

[18] K. Hyunbae, R. D. Lorentz, "Improved Current Regulators for IPM Machine Drives Using On-Line Parameter Estimation", *Conference Records of the 2001 IEEE IAS Annual Meeting*, Vol. 1, pp. 86-91.

[19] K. Hyunbae, J. Hartwig, R. D. Lorentz, "Using On-Line Parameter Estimation to Improve Efficiency of IPM Drives", *2002 Power Electronics Specialists Conference (PESC)*, Vol. 2, pp. 815-820.

[20] D. Jiles, *Introduction to Magnetism and Magnetic Materials*, CRC Press, 2nd Edition, June 1998.

[21] T. Gopalarathnam, R. McCann, "Saturation and Armature Reaction Effects in Surface-Mount PMAC Motors", Electric Machines and Drives Conference, *IEMDC 2001*. IEEE International 2001 Page(s):618 – 621.

[22] L. Ljung, *System Identification – Theory for the user*, Prentice-Hall 1987.

[23] L. Ljung, T. Söderström, *Theory and Practice of Recursive Identification*, MIT Press, 1987.

[24] International Rectifier, *IRAMX16UP60A iMOTION Series*, No. P*D*-94684 Rev. B.

APPENDIX

Sample “C” code for identification algorithms

```
#include "DSP281x_Device.h"
#include "IQmathLib.h"

const S32 two30 = 1073741824;
const S32 two24 = 16777216;
const S32 two26 = 67108864;
const S32 two16 = 65536;


/*********************************************************************/
/*                        "Fast" identification algorithm
                                                                    */
/*********************************************************************/

extern S32 w_rds_q4,wflt_rds_q4,weflt_rds_q4;          // Mechanical speed
extern S32 Vqref_q16,Vdref_q16;                        // Voltage commands
extern S32 Iq_q8,Id_q8;                                // Current feedback
extern S32 Iq_flt_q8,Id_flt_q8;                        // Filtered Current feedback
extern S32 Iq_fltfast_q8,Id_fltfast_q8;
extern S32 Lqini_q16,Ldini_q16;                        // Initial parameters
extern S32 P_fast_ini_q20;
extern S32 lamda_fast_q16;                   // Forgetting factor
extern S32 Pole_pairs;

extern S32 Ke_cst_q12;                       // External required parameters
extern S32 R_cst_q12;
extern S32 RK_q12[2];
extern S32 Kmax_fast_q20;                    // Saturation boundaries
extern S32 Pmax_fast_q20;
extern S32 Pmin_fast_q20;
extern S32 Lqmax_q16,Lqmin_q16;
extern S32 Ldmax_q16,Ldmin_q16;
extern S32 we_rds_q4;
extern S32 Idmin_q8,Iqmin_q8;
extern U32 ISR2mscount;
extern S32 R_flt_q12,K_flt_q12;

S32 Ke_fast_q12,R_fast_q12;                    // External parameters
S32 RIq_q16,RId_q16,Kw_q16;
S32 Y_fast_q16[2],Y_fast_est_q16[2];           //Algorithm Matrices
S32 P_Phi_q16[2][2],PhiT_P_Phi_q16[2][2];
S32 ToBe_Inv_q16[2][2];
S32 det_fast_q2,inv_det_fast_q28;
S32 inverted_fast_q16[2][2];
S32 K_fast_q20[2][2];
S32 KPhiT_fast_q16[2][2];
S32 temp1_fast_q16[2][2],temp2_fast_q20[2][2];
S32 Err_fast_q16[2],dpara_fast_q16[2];

S32 enable_fast=0;                             //Used to enable/ disable algorithm
S32 Lq_flt_q16=0,Ld_flt_q16=0;

S32 Lqd_fast_q16[2];                           // Identif output
S32 P_fast_q20[2][2];
S32 lamda_inv_fast_q16;                        // 1/lamda
S32 PhiT_fast_q8[2][2],Phi_fast_q8[2][2];
S32 neg_Pmax_fast_q20,neg_Pmin_fast_q20,neg_Kmax_fast_q20; //Negative saturation

void identif_Lqd_init(void)
// Initialize identif variables for first run
{
Lqd_fast_q16[0]=Lqini_q16;
Lqd_fast_q16[1]=Ldini_q16;
P_fast_q20[0][0]=P_fast_ini_q20;
P_fast_q20[1][1]=P_fast_ini_q20;
P_fast_q20[0][1]=0;
P_fast_q20[1][0]=0;

lamda_inv_fast_q16 = _IQ16div(65536,lamda_fast_q16);
neg_Pmax_fast_q20=-Pmax_fast_q20;
neg_Pmin_fast_q20=-Pmin_fast_q20;
neg_Kmax_fast_q20=-Kmax_fast_q20;
// These terms will remain at zero
PhiT_fast_q8[0][0] = 0;
PhiT_fast_q8[1][1] = 0;
Phi_fast_q8[0][0] = 0;
Phi_fast_q8[1][1] = 0;
}
```

```
void identif_fast(void)
// Given R and K, uses RLS algorithm to get Lq and Ld
// Assumes steady state because running at 20kHz, not disabled when steady=0
{
if(ISR2mscount>1500)         //uses parameters from slow identification after 3s
        {
        Ke_fast_q12 = RK_q12[1];//Ke_cst_q12;//RK_q12[1];
        R_fast_q12 = RK_q12[0];//R_cst_q12;//RK_q12[0];
        }
else
        {
        Ke_fast_q12 = Ke_cst_q12;
        R_fast_q12 = R_cst_q12;
        }

if(enable_fast==1)
        {
        // Calculate system output matrix
        Y_fast_q16[0] = Vqref_old_q16-((R_fast_q12*Iq_fltfast_q8)>>4)-
        (Ke_fast_q12*wflt_rds_q4);
        Y_fast_q16[1] = Vdref_old_q16-((R_fast_q12 * Id_fltfast_q8)>>4);

        // Update feedback matrix
        PhiT_fast_q8[0][1] = ((-weflt_rds_q4) * Id_fltfast_q8)>>4;
        PhiT_fast_q8[1][0] = ((weflt_rds_q4) * Iq_fltfast_q8)>>4;
        Phi_fast_q8[0][1] = PhiT_fast_q8[1][0];
        Phi_fast_q8[1][0] = PhiT_fast_q8[0][1];

        // Calculate PxPhi
        P_Phi_q16[0][0] = _IQ12mpy(P_fast_q20[0][1],Phi_fast_q8[1][0]);
        P_Phi_q16[0][1] = _IQ12mpy(P_fast_q20[0][0],Phi_fast_q8[0][1]);
        P_Phi_q16[1][0] = _IQ12mpy(P_fast_q20[1][1],Phi_fast_q8[1][0]);
        P_Phi_q16[1][1] = _IQ12mpy(P_fast_q20[1][0],Phi_fast_q8[0][1]);

        // Calculate lamda * I + PhiT * P * Phi
        PhiT_P_Phi_q16[0][0] = _IQ8mpy(PhiT_fast_q8[0][1],P_Phi_q16[1][0]);
        ToBe_Inv_q16[0][0]   =  lamda_fast_q16 + PhiT_P_Phi_q16[0][0];
        PhiT_P_Phi_q16[0][1] = _IQ8mpy(PhiT_fast_q8[0][1],P_Phi_q16[1][1]);
        ToBe_Inv_q16[0][1]   =  PhiT_P_Phi_q16[0][1];
        PhiT_P_Phi_q16[1][0] = _IQ8mpy(PhiT_fast_q8[1][0],P_Phi_q16[0][0]);
        ToBe_Inv_q16[1][0]   =  PhiT_P_Phi_q16[1][0];
        PhiT_P_Phi_q16[1][1] = _IQ8mpy((PhiT_fast_q8[1][0]),P_Phi_q16[0][1]);
        ToBe_Inv_q16[1][1]   =  lamda_fast_q16 + PhiT_P_Phi_q16[1][1];
```

```
// 2x2 Matrix inversion of ToBe_Inv_q16       :
det_fast_q2 = (_IQ30mpy(ToBe_Inv_q16[0][0],ToBe_Inv_q16[1][1])
                  -_IQ30mpy(ToBe_Inv_q16[1][0],ToBe_Inv_q16[0][1]));
if(det_fast_q2==0)inv_det_fast_q28 = 0;
else inv_det_fast_q28 = two30 / det_fast_q2;

inverted_fast_q16[0][0] = _IQ28mpy(inv_det_fast_q28,ToBe_Inv_q16[1][1]);
inverted_fast_q16[1][1] = _IQ28mpy(inv_det_fast_q28,ToBe_Inv_q16[0][0]);
inverted_fast_q16[1][0] = _IQ28mpy(-inv_det_fast_q28,ToBe_Inv_q16[1][0]);
inverted_fast_q16[0][1] = _IQ28mpy(-inv_det_fast_q28,ToBe_Inv_q16[0][1]);

// Calculate gain matrix for parameter update
K_fast_q20[0][0] = _IQ12mpy(P_Phi_q16[0][0],inverted_fast_q16[0][0])
                   +_IQ12mpy(P_Phi_q16[0][1],inverted_fast_q16[1][0]);
if(K_fast_q20[0][0]>Kmax_fast_q20) K_fast_q20[0][0]=Kmax_fast_q20;
if(K_fast_q20[0][0]<neg_Kmax_fast_q20)
                        K_fast_q20[0][0]=neg_Kmax_fast_q20;
K_fast_q20[0][1] = _IQ12mpy(P_Phi_q16[0][0],inverted_fast_q16[0][1])
                   +_IQ12mpy(P_Phi_q16[0][1],inverted_fast_q16[1][1]);
if(K_fast_q20[0][1]>Kmax_fast_q20) K_fast_q20[0][1]=Kmax_fast_q20;
if(K_fast_q20[0][1]<neg_Kmax_fast_q20)
                        K_fast_q20[0][1]=neg_Kmax_fast_q20;
K_fast_q20[1][0] = _IQ12mpy(P_Phi_q16[1][0],inverted_fast_q16[0][0])
                   +_IQ12mpy(P_Phi_q16[1][1],inverted_fast_q16[1][0]);
if(K_fast_q20[1][0]>Kmax_fast_q20) K_fast_q20[1][0]=Kmax_fast_q20;
if(K_fast_q20[1][0]<neg_Kmax_fast_q20)
                        K_fast_q20[1][0]=neg_Kmax_fast_q20;
K_fast_q20[1][1] = _IQ12mpy(P_Phi_q16[1][0],inverted_fast_q16[0][1])
                   +_IQ12mpy(P_Phi_q16[1][1],inverted_fast_q16[1][1]);
if(K_fast_q20[1][1]>Kmax_fast_q20) K_fast_q20[1][1]=Kmax_fast_q20;
if(K_fast_q20[1][1]<neg_Kmax_fast_q20)
                        K_fast_q20[1][1]=neg_Kmax_fast_q20;

// Calculate K x PhiT
KPhiT_fast_q16[0][0] = _IQ12mpy(K_fast_q20[0][1],PhiT_fast_q8[1][0]);
KPhiT_fast_q16[0][1] = _IQ12mpy(K_fast_q20[0][0],PhiT_fast_q8[0][1]);
KPhiT_fast_q16[1][0] = _IQ12mpy(K_fast_q20[1][1],PhiT_fast_q8[1][0]);
KPhiT_fast_q16[1][1] = _IQ12mpy(K_fast_q20[1][0],PhiT_fast_q8[0][1]);

// Calculate I - K x PhiT
temp1_fast_q16[0][0] = 65536 - KPhiT_fast_q16[0][0];
temp1_fast_q16[0][1] = 0 - KPhiT_fast_q16[0][1];
temp1_fast_q16[1][0] = 0 - KPhiT_fast_q16[1][0];
temp1_fast_q16[1][1] = 65536 - KPhiT_fast_q16[1][1];
```

```
// Prepare covariance matrix update
temp2_fast_q20[0][0] = _IQ16mpy(temp1_fast_q16[0][0],P_fast_q20[0][0])
                      +_IQ16mpy(temp1_fast_q16[0][1],P_fast_q20[1][0]);
temp2_fast_q20[0][0] = _IQ16mpy(lamda_inv_fast_q16,temp2_fast_q20[0][0]);

temp2_fast_q20[0][1] = _IQ16mpy(temp1_fast_q16[0][0],P_fast_q20[0][1])
                      +_IQ16mpy(temp1_fast_q16[0][1],P_fast_q20[1][1]);
temp2_fast_q20[0][1] = _IQ16mpy(lamda_inv_fast_q16,temp2_fast_q20[0][1]);

temp2_fast_q20[1][0] = _IQ16mpy(temp1_fast_q16[1][0],P_fast_q20[0][0])
                      +_IQ16mpy(temp1_fast_q16[1][1],P_fast_q20[1][0]);
temp2_fast_q20[1][0] = _IQ16mpy(lamda_inv_fast_q16,temp2_fast_q20[1][0]);

temp2_fast_q20[1][1] = _IQ16mpy(temp1_fast_q16[1][0],P_fast_q20[0][1])
                      +_IQ16mpy(temp1_fast_q16[1][1],P_fast_q20[1][1]);
temp2_fast_q20[1][1] = _IQ16mpy(lamda_inv_fast_q16,temp2_fast_q20[1][1]);

// Update covariance matrix
P_fast_q20[0][0]=temp2_fast_q20[0][0];
if(P_fast_q20[0][0]>0)
        {
        if(P_fast_q20[0][0]>Pmax_fast_q20)P_fast_q20[0][0]=Pmax_fast_q20;
        if(P_fast_q20[0][0]<Pmin_fast_q20)P_fast_q20[0][0]=Pmin_fast_q20;
        }
else
        {
if(P_fast_q20[0][0]<neg_Pmax_fast_q20)P_fast_q20[0][0]=neg_Pmax_fast_q20;
if(P_fast_q20[0][0]>neg_Pmin_fast_q20)P_fast_q20[0][0]=neg_Pmin_fast_q20;
        }

P_fast_q20[0][1]=temp2_fast_q20[0][1];
if(P_fast_q20[0][1]>0)
        {
        if(P_fast_q20[0][1]>Pmax_fast_q20)P_fast_q20[0][1]=Pmax_fast_q20;
        }
else
        {
if(P_fast_q20[0][1]<neg_Pmax_fast_q20)P_fast_q20[0][1]=neg_Pmax_fast_q20;
        }

P_fast_q20[1][0]=temp2_fast_q20[1][0];
if(P_fast_q20[1][0]>0)
        {
        if(P_fast_q20[1][0]>Pmax_fast_q20)P_fast_q20[1][0]=Pmax_fast_q20;
        //if(P_fast_q20[1][0]<Pmin_fast_q20)P_fast_q20[1][0]=Pmin_fast_q20;
```

```
		}
	else
		{
	if(P_fast_q20[1][0]<neg_Pmax_fast_q20)P_fast_q20[1][0]=neg_Pmax_fast_q20;
		}

	P_fast_q20[1][1]=temp2_fast_q20[1][1];
	if(P_fast_q20[1][1]>0)
		{
		if(P_fast_q20[1][1]>Pmax_fast_q20)P_fast_q20[1][1]=Pmax_fast_q20;
		if(P_fast_q20[1][1]<Pmin_fast_q20)P_fast_q20[1][1]=Pmin_fast_q20;
		}
	else
		{
	if(P_fast_q20[1][1]<neg_Pmax_fast_q20)P_fast_q20[1][1]=neg_Pmax_fast_q20;
	if(P_fast_q20[1][1]>neg_Pmin_fast_q20)P_fast_q20[1][1]=neg_Pmin_fast_q20;
		}

	// Calculate estimated output:
	Y_fast_est_q16[0] = (PhiT_fast_q8[0][1]*Lqd_fast_q16[1])>>8;
	Y_fast_est_q16[1] = (PhiT_fast_q8[1][0]*Lqd_fast_q16[0])>>8;

	// Error between system and estimated outputs
	Err_fast_q16[0] = Y_fast_q16[0] - Y_fast_est_q16[0];
	Err_fast_q16[1] = Y_fast_q16[1] - Y_fast_est_q16[1];

	// Error x gain -> change in parameters
	dpara_fast_q16[0] = _IQ20mpy(K_fast_q20[0][0],Err_fast_q16[0])
	                  + _IQ20mpy(K_fast_q20[0][1],Err_fast_q16[1]);
	dpara_fast_q16[1] = _IQ20mpy(K_fast_q20[1][0],Err_fast_q16[0])
	                  + _IQ20mpy(K_fast_q20[1][1],Err_fast_q16[1]);

	// Parameters' update and saturation
	Lqd_fast_q16[0] += dpara_fast_q16[0];
	Lqd_fast_q16[1] += dpara_fast_q16[1];
	if(Lqd_fast_q16[0]>Lqmax_q16)Lqd_fast_q16[0]=Lqmax_q16;
	if(Lqd_fast_q16[0]<Lqmin_q16)Lqd_fast_q16[0]=Lqmin_q16;
	if(Lqd_fast_q16[1]>Ldmax_q16)Lqd_fast_q16[1]=Ldmax_q16;
	if(Lqd_fast_q16[1]<Ldmin_q16)Lqd_fast_q16[1]=Ldmin_q16;

	// Low pass, pole at 0.98 q12
	Lq_flt_q16 = ((82*Lqd_fast_q16[0])>>12) + ((4014 * Lq_flt_q16)>>12);
	Ld_flt_q16 = ((82*Lqd_fast_q16[1])>>12) + ((4014 * Ld_flt_q16)>>12);
	}
}
```

```
/*****************************************************************/
/*                   "Slow" identification algorithm
                                                                */
/*****************************************************************/

extern S32 Vqref_flt_q16,Vdref_flt_q16;        // Voltage commands
extern S32 Iq_flt_q8,Id_flt_q8;                // Current feedback
extern S32 Lqini_q16,Ldini_q16;
extern S32 Rsini_q16,Keini_q16;
extern S32 P_slow_ini_L_q20,P_slow_ini_RK_q20;
extern S32 lamda_slow_q16;
extern S32 weflt_rds_q4;
extern S32 wflt_rds_q4;

extern S32 Lqmin_q16,Lqmax_q16;
extern S32 Ldmin_q16,Ldmax_q16;
extern S32 Rmin_q16,Rmax_q16;
extern S32 Kmin_q16,Kmax_q16;

extern S32 Kmax_slow_q20;
extern S32 Pmax_slow_q20;
extern S32 Pmin_slow_q20;
extern S32 Pmin_slowL_q20;
extern S32 Lqd_fast_q16[2];
extern S32 Lq_flt_q16,Ld_flt_q16;
extern S32 Ld_zero_q16;

S32 Y_slow_q16[2],Y_slow_est_q16[2],Err_slow_q16[2];
S32 P_Phi_slow_q16[4][2];
S32 PhiT_P_Phi_slow_q16[2][2];
S32 ToBe_Inv_slow_q16[2][2],inverted_slow_q16[2][2];
S32 det_slow_q2,inv_det_slow_q28;
S32 K_slow_q20[4][2];
S32 KPhiT_slow_q16[4][4];
S32 temp1_slow_q16[4][4];
S32 temp2_slow_q20[4][4];
S32 enable_slow=1;

S32 R_flt_q12=0,K_flt_q12=0;
S32 LRK_q16[4];
S32 RK_q12[2];
S32 P_slow_q20[4][4];
S32 lamda_inv_slow_q16;                  // 1/lamda
S32 PhiT_slow_q8[2][4];
S32 Phi_slow_q8[4][2];
```

```
S32 dpara_slow_q16[4];
S32 neg_Kmax_slow_q20;
S32 neg_Pmax_slow_q20;
S32 neg_Pmin_slow_q20;
S32 neg_Pmin_slowL_q20;


void identif_LRK_init(void)
// Initialize identif variables for first run
{
U16 i,j;
LRK_q16[0]=Lqini_q16;
LRK_q16[1]=Ldini_q16;
LRK_q16[2]=Rsini_q16;
LRK_q16[3]=Keini_q16;

RK_q12[0] = LRK_q16[2]>>4;
RK_q12[1] = LRK_q16[3]>>4;

for(i=0;i<4;i++)
        for(j=0;j<4;j++)
                {
                P_slow_q20[i][j]=0;
                }
P_slow_q20[0][0]=P_slow_ini_L_q20;
P_slow_q20[1][1]=P_slow_ini_L_q20;
P_slow_q20[2][2]=P_slow_ini_RK_q20;
P_slow_q20[3][3]=P_slow_ini_RK_q20;

lamda_inv_slow_q16 = _IQ16div(65536,lamda_slow_q16);
for(i=0;i<2;i++)
        for(j=0;j<4;j++)
                {
                PhiT_slow_q8[i][j]=0;
                Phi_slow_q8[j][i]=0;
                }
neg_Kmax_slow_q20 = -Kmax_slow_q20;
neg_Pmax_slow_q20 = -Pmax_slow_q20;
neg_Pmin_slow_q20 = -Pmin_slow_q20;
neg_Pmin_slowL_q20= -Pmin_slowL_q20;
}


void identif_slowA(void)
// Assuming steady state, tries to estimate Lq, Ld, Rs and Ke, needs "rich" inputs
// Disabled when system in transient
// Code avoids loops on purpose, for faster execution
//Some Matrix product functions omitted because of their length
{
if(enable_slow==1)
        {
        // System output matrix
        Y_slow_q16[0] = Vqref_flt_q16;
        Y_slow_q16[1] = Vdref_flt_q16;

        // Update feedback matrix
        PhiT_slow_q8[0][1] = ((-weflt_rds_q4) * Id_flt_q8)>>4;
        PhiT_slow_q8[0][2] = Iq_flt_q8;
        PhiT_slow_q8[0][3] = wflt_rds_q4<<4;
        PhiT_slow_q8[1][0] = ((weflt_rds_q4) * Iq_flt_q8)>>4;
        PhiT_slow_q8[1][2] = Id_flt_q8;

        Phi_slow_q8[0][1] = PhiT_slow_q8[1][0];
        Phi_slow_q8[1][0] = PhiT_slow_q8[0][1];
        Phi_slow_q8[2][0] = PhiT_slow_q8[0][2];
        Phi_slow_q8[2][1] = PhiT_slow_q8[1][2];
        Phi_slow_q8[3][0] = PhiT_slow_q8[0][3];

        Calc_P_Phi();           //Matrix product function

        // Calculate lamda x I + PhiT x P x Phi
        PhiT_P_Phi_slow_q16[0][0] =
        _IQ8mpy(PhiT_slow_q8[0][1],P_Phi_slow_q16[1][0])
        + _IQ8mpy(PhiT_slow_q8[0][2],P_Phi_slow_q16[2][0])
        + _IQ8mpy(PhiT_slow_q8[0][3],P_Phi_slow_q16[3][0]);
        ToBe_Inv_slow_q16[0][0] = lamda_slow_q16 + PhiT_P_Phi_slow_q16[0][0];

        PhiT_P_Phi_slow_q16[0][1] =
        _IQ8mpy(PhiT_slow_q8[0][1],P_Phi_slow_q16[1][1])
        + _IQ8mpy(PhiT_slow_q8[0][2],P_Phi_slow_q16[2][1])
        + _IQ8mpy(PhiT_slow_q8[0][3],P_Phi_slow_q16[3][1]);
        ToBe_Inv_slow_q16[0][1] = PhiT_P_Phi_slow_q16[0][1];

        PhiT_P_Phi_slow_q16[1][0] =
        _IQ8mpy(PhiT_slow_q8[1][0],P_Phi_slow_q16[0][0])
        + _IQ8mpy(PhiT_slow_q8[1][2],P_Phi_slow_q16[2][0]);
        ToBe_Inv_slow_q16[1][0] = PhiT_P_Phi_slow_q16[1][0];

        PhiT_P_Phi_slow_q16[1][1] =
        _IQ8mpy(PhiT_slow_q8[1][0],P_Phi_slow_q16[0][1])
```

```
          + _IQ8mpy(PhiT_slow_q8[1][2],P_Phi_slow_q16[2][1]);
          ToBe_Inv_slow_q16[1][1] = lamda_slow_q16 + PhiT_P_Phi_slow_q16[1][1];

          // 2x2 Matrix inversion of ToBe_Inv_q16       :
          det_slow_q2 =
          (_IQ30mpy(ToBe_Inv_slow_q16[0][0],ToBe_Inv_slow_q16[1][1])
          -  _IQ30mpy(ToBe_Inv_slow_q16[1][0],ToBe_Inv_slow_q16[0][1]));
          if(det_slow_q2==0)inv_det_slow_q28 = 0;
          else inv_det_slow_q28 = two30 / det_slow_q2;

          inverted_slow_q16[0][0] =
          _IQ28mpy(inv_det_slow_q28,ToBe_Inv_slow_q16[1][1]);
          inverted_slow_q16[1][1] =
          _IQ28mpy(inv_det_slow_q28,ToBe_Inv_slow_q16[0][0]);
          inverted_slow_q16[1][0] = _IQ28mpy(-
          inv_det_slow_q28,ToBe_Inv_slow_q16[1][0]);
          inverted_slow_q16[0][1] = _IQ28mpy(-
          inv_det_slow_q28,ToBe_Inv_slow_q16[0][1]);

          // Calculate gain matrix for parameter update
          Calc_Kslow();

          // Calculate I - K x PhiT
          Calc_IminKPhiT();
          // ... Continued in identif_slowB()...
          }
}


void identif_slowB(void)
{
if(enable_slow==1)
          {
          // Calculate New covariance matrix
          Calc_Ptemp();             //Similar to identify_fast() code only longer
          Upd_P();

          // Calculate estimated output:
          Y_slow_est_q16[0] = ((PhiT_slow_q8[0][1]*LRK_q16[1])
                                        +  (PhiT_slow_q8[0][2]*LRK_q16[2])
                                        +  (PhiT_slow_q8[0][3]*LRK_q16[3]))>>8;
          Y_slow_est_q16[1] = ((PhiT_slow_q8[1][0]*LRK_q16[0])
                                        +  (PhiT_slow_q8[1][2]*LRK_q16[2]))>>8;

          // Error between system and estimated outputs
```

```
        Err_slow_q16[0] = Y_slow_q16[0] - Y_slow_est_q16[0];
        Err_slow_q16[1] = Y_slow_q16[1] - Y_slow_est_q16[1];

        // Error x gain -> change in parameters
        dpara_slow_q16[0] = _IQ20mpy(K_slow_q20[0][0],Err_slow_q16[0])
                            + _IQ20mpy(K_slow_q20[0][1],Err_slow_q16[1]);
        dpara_slow_q16[1] = _IQ20mpy(K_slow_q20[1][0],Err_slow_q16[0])
                            + _IQ20mpy(K_slow_q20[1][1],Err_slow_q16[1]);
        dpara_slow_q16[2] = _IQ20mpy(K_slow_q20[2][0],Err_slow_q16[0])
                            + _IQ20mpy(K_slow_q20[2][1],Err_slow_q16[1]);
        dpara_slow_q16[3] = _IQ20mpy(K_slow_q20[3][0],Err_slow_q16[0])
                            + _IQ20mpy(K_slow_q20[3][1],Err_slow_q16[1]);

        // Estimated parameters' update
        LRK_q16[0] += dpara_slow_q16[0];
        LRK_q16[1] += dpara_slow_q16[1];
        LRK_q16[2] += dpara_slow_q16[2];
        LRK_q16[3] += dpara_slow_q16[3];

        //Saturation
        if(LRK_q16[0]>Lqmax_q16)LRK_q16[0]=Lqmax_q16;
        if(LRK_q16[0]<Lqmin_q16)LRK_q16[0]=Lqmin_q16;
        if(LRK_q16[1]>Ldmax_q16)LRK_q16[1]=Ldmax_q16;
        if(LRK_q16[1]<Ldmin_q16)LRK_q16[1]=Ldmin_q16;
        if(LRK_q16[2]>Rmax_q16)LRK_q16[2]=Rmax_q16;
        if(LRK_q16[2]<Rmin_q16)LRK_q16[2]=Rmin_q16;
        if(LRK_q16[3]>Kmax_q16)LRK_q16[3]=Kmax_q16;
        if(LRK_q16[3]<Kmin_q16)LRK_q16[3]=Kmin_q16;

        RK_q12[0] = LRK_q16[2]>>4;
        RK_q12[1] = LRK_q16[3]>>4;
        R_flt_q12 = ((410*RK_q12[0])>>12) + ((3686 * R_flt_q12)>>12);
        K_flt_q12 = ((410*RK_q12[1])>>12) + ((3686 * K_flt_q12)>>12);
        }
        //When algorithm is disabled, Lqd are copied from identify_fast()
        else{
        LRK_q16[0]=Lq_flt_q16;
        LRK_q16[1]=Ld_flt_q16;
        }

}
```