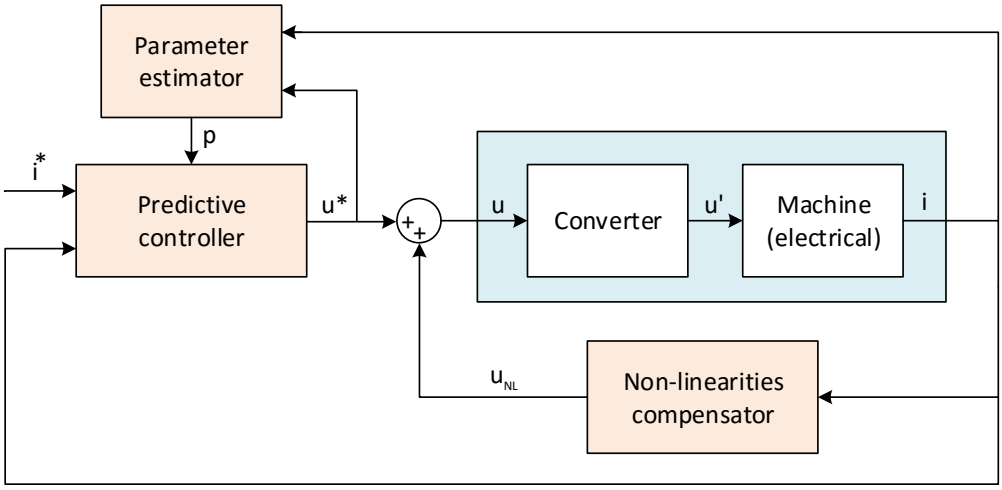
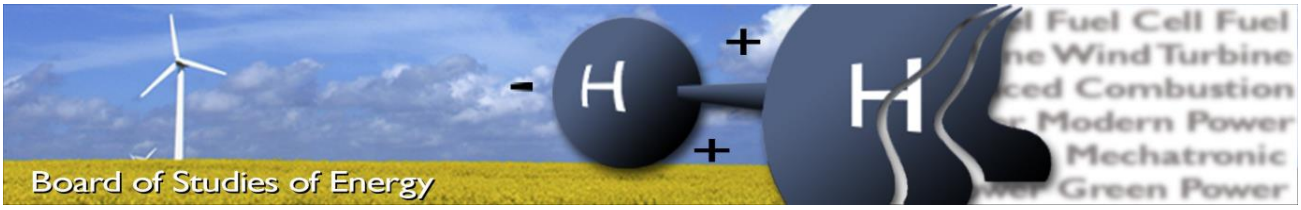
Predictive control for PMSM



Master's Thesis
PED4-944
Carlos Gómez Suárez

January 6, 2016

AALBORG UNIVERSITET



Title:	**Predictive control for PMSM**
Semester:	**3th-4th semester M.Sc.**
Semester theme:	**Master thesis – Power Electronics and Drives**
Project period:	**01-02-2015 – 29-01-2016**
ECTS:	**50**
Supervisor:	**Kaiyuan Lu**
Project group:	**PED4-944**

SYNOPSIS:
This thesis focuses in the design, analysis and implementation of a predictive current controller for the control of a Permanent Magnet Synchronous Machine (PMSM). This thesis studies the predictive controller including the main problems presented with its implementation. The effect of errors in the system parameters is studied and different non-linearities are analyzed and compensated. Online parameter estimations are also developed as a way for the compensator to adapt to changes in them. As a result of the thesis the predictive controller is seen as a simple controller with excellent performance when the different disturbances that affect the system are accounted for.

Carlos Gómez Suárez

Copies:	[0]
Pages, total:	[135]
Appendix:	[4]
Supplements:	[1 CD]

By signing this document, each member of the group confirms that all group members have participated in the project work, and thereby all members are collectively liable for the contents of the report. Furthermore, all group members confirm that the report does not include plagiarism.

PREFACE

This master thesis is written by Carlos Gomez, a 10^{th} master student in the Department of Energy Technology in Aalborg University. The semester theme is Power Electronics and Drives.

Reading instructions The denotation of employed equations, figures, tables and codes trough the thesis is based in the notation (X.Y) which means the Y^{th} item belonging to the X^{th} chapter. The units used are placed on the right of the numbers and are usually SI units.

At the end of the thesis the appendices can be found which supplement the information provided. Bibliography is placed after them. The references follow $[k]$ where k represents the k reference in the bibliography list. As a further attachment a CD with simulations is provided as well as a PDF version of the thesis.

ABSTRACT

This thesis focuses in the design, analysis and implementation of a predictive current controller for the control of a Permanent Magnet Synchronous Machine (PMSM).

The theory regarding the PMSM and converter system is described mathematically in order to have models to better understand the different problems. Trough the different parts of the project several approximations are needed to yield solutions to the problems presented and trough simulation and experiments the hypothesis taken are validated.

This thesis studies the predictive controller including the main problems presented with its implementation. The effect of changes in system parameters is studied and different non-linearities are analyzed and compensated. Finally online parameter estimations are also developed as a way for the compensator to adapt to changes in them.

As a result of the thesis the predictive controller is seen as a simple controller with excellent performance when the different compensations needed are accounted for.

CONTENTS

1

INTRODUCTION

Development of semiconductor devices and powerful cost efficient Digital Signal Processing (DSP)s give opportunities for applications in different areas including AC-machine drives.

Permanent Magnet Synchronous Machine (PMSM) drives present several advantages over other drives due to its high efficiency and high power density capabilities. The control of this drives can be performed trough vector control which was developed as a way to get better torque responses trough the decoupling of the system in two different controls, one for the torque and one for the flux [1]. This is illustrated in Figure 1.1. The focus of the project consist in the inner loop or current controller.

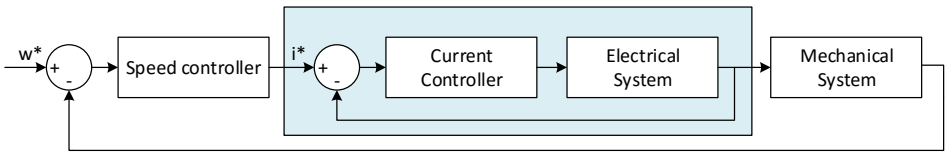


Figure 1.1.: System controllers

The flux, controlled by the field current can be performed trough the conventional use of PI controllers which offer a simple solution to the problem. However the transient response can be improved with different techniques which have been studied over the years from which predictive controller is a promising method.

1.1 PREDICTIVE CONTROL METHODS

Predictive controllers use models of the system to create predictions of future states and variables to control. With this information the actuation in the system is obtained according to the method used. Several classifications can be made as suggested in [2]. The main strategies are: hysteresis based, trajectory based, model predictive control (MPC) and deadbeat.

Hysteresis based which needs no modulator bounds control variables within a tolerance band and calculates the instants at which the switch states must change. This requires a variable frequency.

Trajectory based calculates different optimum control strategies based in different states and applies them directly. There is no need for cascade control and speed can be controlled directly from the speed error without a current controller. The switching states of the converter are classified into categories such as "increasing torque", "reducing torque slowly", etc. and the instants to change are also calculated so variable frequency is needed.

Model predictive control can use modulators (and fixed switching frequency) depending of the implementation or a finite control set where the different switching states are tried within an horizon. A cost function with the errors between variables is created and optimized by modifying the different control variables trough an optimization algorithm. In the case of finite set for example the 8 different switching states can be tested to see which one produces the best (least error in the optimization function) response in the next period.

Finally deadbeat controller uses a modulator (and fixed frequency) for the current controller. Based on system equations the voltage to apply to reach the current reference is estimated.

Based in the main principles of the controllers and previous work conducted in [3] - [4] deadbeat controller is chosen. In [4] both MPC and deadbeat are implemented. MPC is observed to present a much bigger ripple in the current due to the lack of modulator, simple switching strategy and sampling frequency of 25 kHz imposed as a limit by the setup. Deadbeat is only worse in steady-state errors due to simulated system parameter changes. In [3] deadbeat controller presents also good results.

1.2 PROBLEM STATEMENT

A PMSM can be controlled by means of two different set of controllers as shown in Figure 1.1. The outside loop regulates the speed by adjusting the current that passes trough the motor as this modifies the electrical torque which in the end modifies the speed.

On the other hand, inside a current loop (blue rectangle) regulates the voltage applied to the motor in order to achieve the desired current. This project is focused in this second controller. The speed controller used may be a PI while a predictive controller that uses the machine equations to obtain a better response is studied in this project in contrast to the classical PI for current control

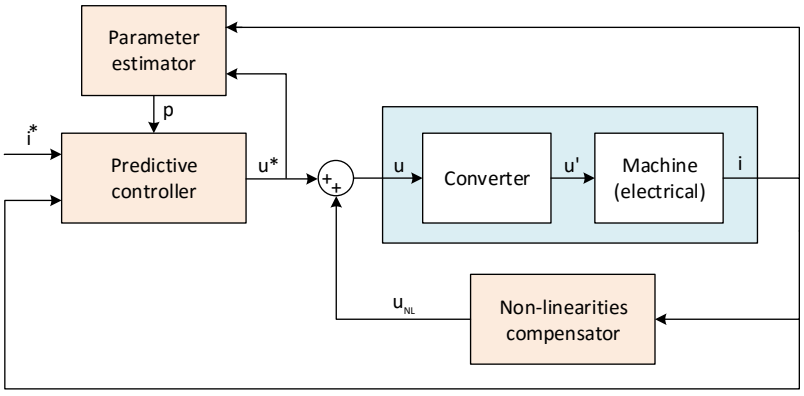


Figure 1.2.: Predictive current controller, non-linearity compensation and online parameter estimator represented in blocks

Further study of the electrical system shows that there are 3 main problems to solve as shown in Figure 1.2 (pink rectangles) in which the project will have its focus:

- Predictive controller: Using models that predict different states a control can be made with great performance relying on the predictions of the variables.
- Non-linearity compensation: Since the predictive controller uses models of the system everything needs to be accounted for in order to compensate for it. When a voltage command is sent to the converter it is converted from u to u' as illustrated in Figure 1.2 and can be compensated by adding an extra voltage u_{NL} to account for it. The non-linearities studied in this project are the deadtime in the gate signals and the voltage drop due to snubbers, Insulated-Gate Bipolar Transistors (IGBT) and diodes.
- Online parameter estimator: In normal operation machine parameters can vary slowly due to temperature. However inductors which are fundamental for the stability of the controller can change abruptly due to saturation. Therefore an online estimator to work with the predictive controller which uses the current response and voltage command to adapt the parameters is also developed.

1.3 OBJECTIVE

The goals of this project include:

- Implementation of deadbeat controller
- Compensation of the different delays present in the system
- Study of the effect of parameter variations in the algorithm. How do parameter changes compromise stability? How does the controller behaves with parameters changed at different states (loads, speeds)? How does the switching frequency affects the stability and steady-state errors when there are parameter errors?
- Study and compensation of the different non-linearities present in the converter
- As a way to mitigate the errors due to system parameter changes implementation of different parameter estimation algorithms

1.4 THESIS OUTLINE

This thesis is structured as follows. After the introduction the description of the system is done in Chapter 2 where the experimental setup is shown followed by a more in depth description of the converter and machine.

In Chapter 3 the field-oriented control with PIs is revised and analyzed as it is commonly used as a way to control this type of machines. The gains are calculated based on classical control theory and a compromise between speed and stability is done so

the response can be later compared with the predictive controller which is presented in Chapter 4 which explains the deduction of the controller equations. The effect of parameter variations in the controller (stability, steady-state errors, transient response) is studied in Chapter 5.

The non-linearities in the converter are described in Chapter 6 which begins with the induced signal delay in the pulses. It continues considering the snubbers and ends up studying the IGBTs and diodes voltage drops. Compensation for all this components are calculated and a compensator is proposed.

In Chapter 7 the determination of online parameters is analyzed with two different algorithms, the most common RLS and also a gradient method. At the end experimental tests show that the estimators are able to detect the saturation of the inductors.

Lastly, some appendices are included to compliment the reading. In Appendix A the offline tests performed to obtain the machine and converter parameters are described. In Appendix B some derivations are presented. The most relevant models used in the project are shown in Appendix C. Finally in Appendix D the main codes used are presented both for its Simulink and PLECS implementation.

1.5 LIMITATIONS AND ASSUMPTIONS

PMSM Model

The dq model is used and some assumptions are therefore made. Stator windings are modeled as a DC offset and a purely sinusoidal varying component. Rotor permanent magnets are modeled as sinusoidal varying components. Modification of components as a function of temperature, speed or load are not modeled even though their effect in the control is studied and online estimations are developed to account for them. Core losses are neglected. The mechanical model is simplified as a one mass system containing the PMSM and Induction Machine (IM).

Load control

Different load conditions are tested with an induction IM connected mechanically with the PMSM of which torque is controlled trough a PI. Any coupling due to this controller which may produce a small oscillating disturbance in the load is neglected.

Inverter

The modifications done on the inverter ideal model are the inclusion of dead-time ($2.5\mu s$), diode and transistor voltage drop (in ohmic region) and snubbers modeled as capacitors.

Pulse Width Modulation (PWM)

The PWM is modeled either as a Zero-Order Hold (ZOH) or an average in the equations and thus its pulsated nature is neglected. If the switching frequency is small this can

cause dissimilarities between model and reality but both simulation and experimental results show at the frequencies used it can be neglected.

Simulink and DSpace

PWM behavior in DSpace is inferred from experimental results concluding that there is no period delay in the voltage command. Under this behavior the computational time could distort the voltage command but as long as it is small enough it can be neglected due to the symmetric nature of the PWM. Trough the experimental results since the performance seems adequate and results tend to follow the simulations this computational time is assumed to be small and possible to be neglected.

From the block descriptions it is understood that adding a period delay block to the voltage command would not help as it would delay the voltage command one period (after it is calculated) and would have no effect in removing this computational time distortion as it would still be presented.

Models are developed in Simulink and blocks are expected to behave as they should but since the implementation is hidden one must assume they behave as the are advertised. Since the final code is never checked as it is machine made it is also assumed to reflect the models originally created.

Information shown in Control Desk is assumed to be an accurate representation of the variables in real-time.

Measurement devices are assumed to be calibrated and their values provided trusted. Several identical setups have been tested without any discernible change between them.

2

SYSTEM DESCRIPTION

This chapter begins with a general description of the system used and follows explaining more in detail the converter and PMSM.

2.1 SYSTEM SETUP

To test the algorithms proposed a PMSM is controlled by means of an inverter that regulates the 3-phase voltages applied. To simulate different conditions the PMSM is mechanically connected to an induction machine that is used to simulate different torques. The schematics are shown in Figure 2.1 where the green lines represent power and the blue ones signals.

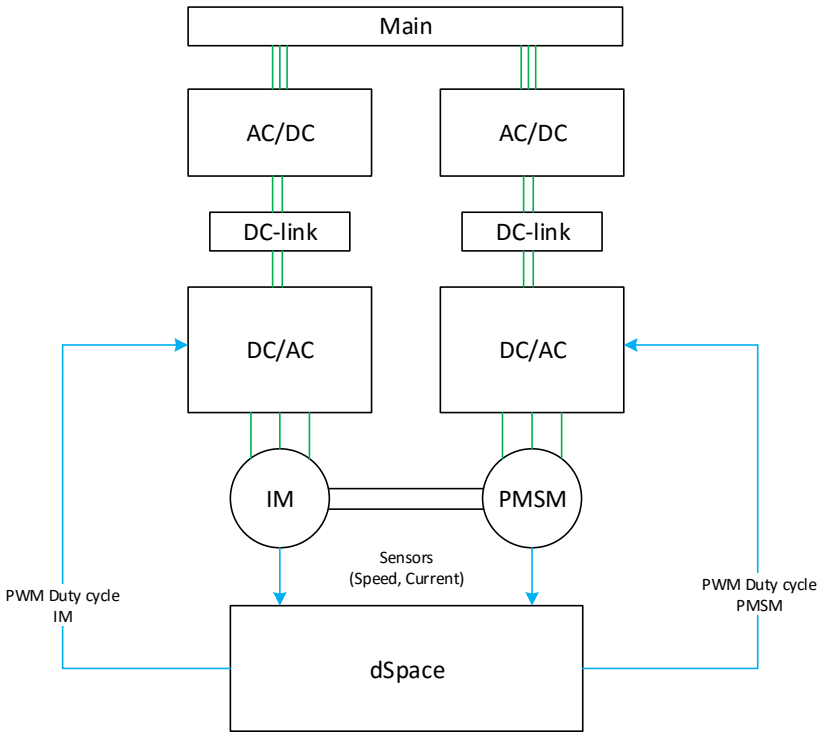


Figure 2.1.: Experimental system setup

The power from the main is transformed to DC by means of a diode bridge for the IM and PMSM and thus there is no control on this part. The DC is then transformed into AC by means of an inverter that is regulated trough DSpace. Different signals from the sensors are sent to the controller that uses this information and in return calculates

the duty cycles signals that are sent to the PWM hardware in order to modify the AC voltage in the machine terminals to achieve the desired references.

The control of the IM is independent of the PMSM and the only objective is to regulate the torque applied to the later so different conditions can be tested.

The control of both machines is done trough DSpace. The code is written in Simulink where common blocks are accessible including Matlab functions and finally the model is compiled to C that is pushed to the DSP. There is an interface with the computer trough Control Desk where variables can be seen and modified in real-time. Sensors can be read in Simulink by using the Analog to Digital Conversion (ADC) blocks provided and the only signals sent to the device are the PWM duty cycles trough the PWM block.

The whole program is executed at a fixed frequency following Figure 2.2. In each new tick of the clock represented by the green ball the program written in Simulink performs the calculations needed to modulate the PWM which take some small computation time represented by the grey square. It will begin reading the information from the sensors and at the end calculate the PWM duty cycles that will be sent to the PWM hardware with this small computation delay. This is based on tests performed where if the voltage command is changed the current is seen modified instantly in the next period.

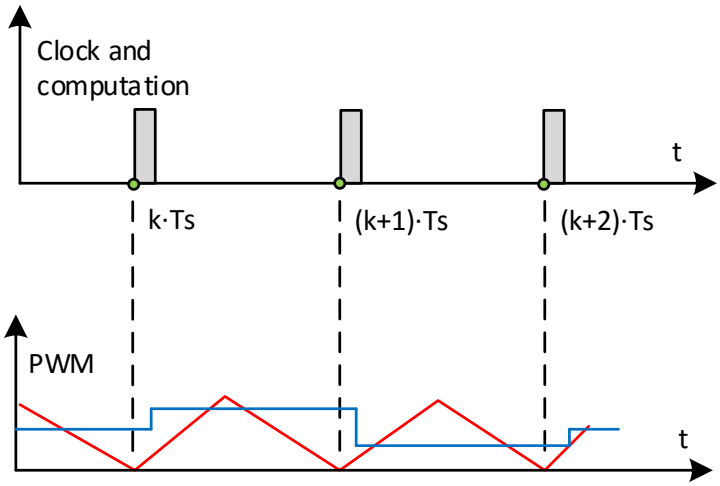


Figure 2.2.: Execution sequence of the program

In the bottom graph it can be seen the PWM carriers in red syncrhonized with the tick and receiving the new duty cycle in blue delayed by the computation time and an even smaller time due to the hardware. As long as the computation time can be kept small in comparison with the frequency this delay will not disturb the command in a noticeable manner. Another option is to force the new PWM to enter in the next period so the delay can be compensated since it would be constant as shown in Figure 2.3 where the grey box represents again the computation time but in this case the new duty cycles are sent in $k+1$ instead of k plus the computation delay. This is the most typical way to do the

control but since DSpace is used it has not been possible to change the PWM behavior. For this reason two set of controllers are developed, a 1-period controller for the system implemented in the lab from Figure 2.2 and a 2-period version for the most common implementation shown in Figure 2.3 that is used is most of the simulations.

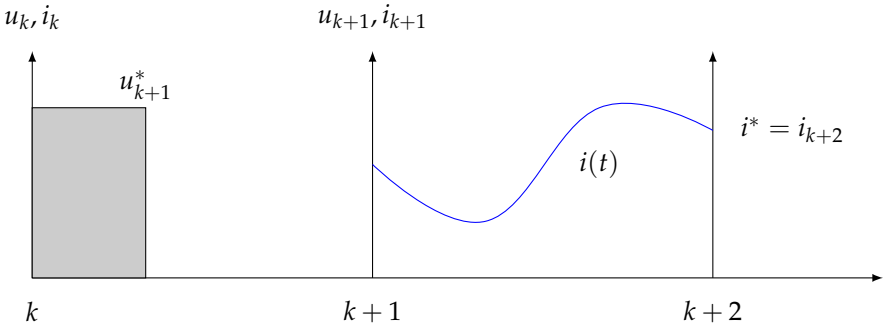


Figure 2.3.: PWM modulation with a period delay in the command

The advantages of the first method is that the current controller can theoretically achieve the reference using a predictive controller in only one switching period while the second requires two. On the other hand, the second is more precise since the delay can be accounted for and fully compensated while in the first it can only be neglected and if the computation time is considerably big it could make distortions in the voltage command. However based in the experimental results the controller works adequately and the more common implementation with the DSP computation delay is also developed in simulations.

2.2 VOLTAGE SOURCE INVERTER

A 2 level-converter is used which is able to modulate the AC voltage in the output at the desired level and frequency that will drive the machine. A schematic of the converter is presented in Figure 2.4.

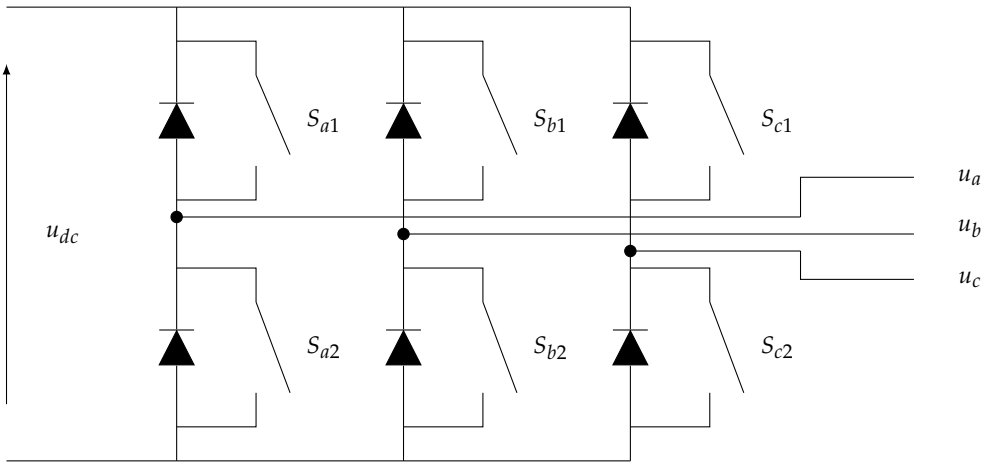


Figure 2.4.: 2-level converter

The input is the voltage u_{dc} which is considered constant and there are three legs for each AC output. For each leg, either the top or the bottom switch can be closed but not both as that would short-circuit the source. Because the switches are not ideal they need a time to commute from one state to another and there is a dead-time forced in the signals to ensure there is time for the current to commute from one path to the other.

If the reference of the voltages is set in the negative side of the dc-link, when the top switch of a leg is closed the voltage to neutral seen is u_{dc} and when the bottom is closed there is no voltage. If when the top switch of a leg is closed the state is given by 1 and 0 otherwise the line-line voltages as a function of the switch states can be given simply by Equation 2.1.

$$\begin{bmatrix} u_{ab} \\ u_{bc} \\ u_{ca} \end{bmatrix} = u_{dc} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_a \\ S_b \\ S_c \end{bmatrix} \tag{2.1}$$

The machine connected will have a different neutral-point than the converter and so the actual line-neutral voltages seen by the PMSM will be different. They can be obtained by assuming that the system is balanced. For such a system Equation 2.2 - 2.5 are true. One of the first three equations is redundant and is not used. Therefore, the linear system with 3 unknowns and equations can be solved and the relation between line-neutral and line-line voltages is given by Equation 2.6 if the third equation is removed. Therefore to obtain the line-neutral voltages seen by the machine, Equation 2.1 can be multiplied on the left by Equation 2.6 resulting in Equation 2.7.

$$u_{ab} = u_{an} - u_{bn} \tag{2.2}$$

$$u_{bc} = u_{bn} - u_{cn} \tag{2.3}$$

$$u_{ca} = u_{cn} - u_{bn} \tag{2.4}$$

$$u_{an} + u_{bn} + u_{cn} = 0 \tag{2.5}$$

$$\begin{bmatrix} u_{an} \\ u_{bn} \\ u_{cn} \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 2 & 1 & 0 \\ 1 & -1 & 0 \\ -1 & -2 & 0 \end{bmatrix} \begin{bmatrix} u_{ab} \\ u_{bc} \\ u_{ca} \end{bmatrix} \tag{2.6}$$

$$\begin{bmatrix} u_{an} \\ u_{bn} \\ u_{cn} \end{bmatrix} = \frac{u_{dc}}{3} \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} S_a \\ S_b \\ S_c \end{bmatrix} \tag{2.7}$$

Finally, the voltages can be transformed into $\alpha\beta$ by multiplying by the transformation matrix $K_{abc}^{\alpha\beta}$ which is omitted here for brevity. It can be calculated using the projection method explained in Section B.1. When iterating trough all the possible switching states 8 different vectors in this reference frame can be obtained and they are shown in Figure 2.5.

2.2.1 *Space Vector Modulation (SVM)*

SVM is a modulation technique that can be used to achieve the reference vector by means of alternating between the different possible vectors the converter can generate for a given time which is function of the reference. Normally the three closest vectors are chosen and the reference vector can be generated by modulating those vectors in a switching period such as the average is the reference.

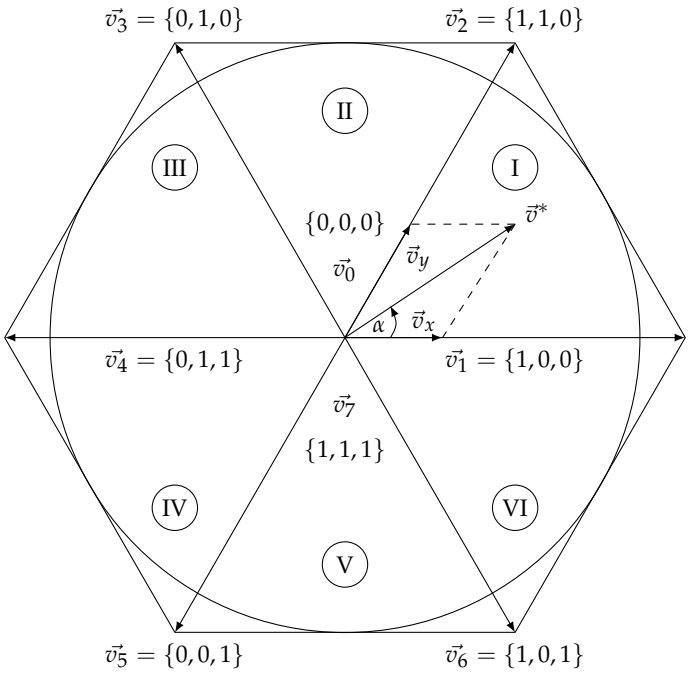


Figure 2.5.: All the different vectors in $\alpha\beta$ for a 2-level converter

Each region limited by 2 vectors in a 2-level SVM is called a sector. Therefore there are 6 sectors as shown in Figure 2.5 with the roman numerals. The different vectors are placed in the drawing with the switching state that creates them. Vectors 1-6 are called active as they contain a value different from zero on contrast with 0 and 7. The reference vector will always be in a sector and the closest vector to the right can be denoted as v_x while the one on the left as v_y.

To determine the sector the angle α can be obtained by Equation 2.8 where v_α and v_β are the α and β components of the reference vector $\vec{v}^*$ and then the sector is given by Equation 2.9.

$$\alpha = \tan^{-1}\frac{v_\beta}{v_\alpha}, \alpha \in [0, 2\pi) \tag{2.8}$$

$$Sector = floor\,(3\alpha/\pi) + 1 \tag{2.9}$$

Then using the closest vectors, the times can be obtained by solving d_x, d_y and d_0 where $d_i = T_i/T_s$ in Equation 2.10 - 2.11.

$$\vec{v}^* = \vec{v}_x d_x + \vec{v}_y d_y + \vec{v}_0 d_0 \tag{2.10}$$

$$d_x + d_y + d_0 = 0 \tag{2.11}$$

The solution is then given by Equation 2.12 - 2.14 where $\beta = \alpha - (Sector - 1) \times \pi/3$.

$$d_x = \frac{2}{\sqrt{3}} \frac{v^*}{v_x} \sin(\frac{\pi}{3} - \beta) \tag{2.12}$$

$$d_y = \frac{2}{\sqrt{3}} \frac{v^*}{v_y} \sin(\beta) \tag{2.13}$$

$$d_0 = 1 - d_x - d_y \tag{2.14}$$

After the duty cycles of the vectors are calculated they can be applied either trough software with timers by modulating the vectors for a given amount of time or by transforming the duty cycles in the vectors to duty cycles in legs and using standard PWM hardware as is done in the experiment.

The transformation from the duty cycles in each vector in $xy0$ to abc can be made by summing the result of multiplying each duty cycle by the vector it represents (following a notation where 1 is up and 0 is down). The result will have three components where each index represents the leg, that is the first is a, second is b and third us c.

The duty cycle per leg d_{abc} can then be fed to a comparator with a carrier at f_{sw} of which output will generate the switching state per leg (turning on the top igbt or the lower). Later it will be explained that this generated signals to the IGBT will be delayed a constant time (dead-time) to prevent short-circuits due to the finite time it takes for the current to commute from one path to another.

2.3 PERMANENT MAGNET SYNCHRONOUS MACHINE

A representation of a PMSM with one pole-pair is depicted in Figure 2.6. When current passes trough a stator winding it will generate a flux perpendicular to its plane. Therefore if positive current is applied to phase-a it will produce a flux in the upwards/downwards direction which is labeled with a-axis in the picture. The magnet in the rotor will then try to follow the flux and will change its position so its in phase with the flux. The extra phases are added as with only one phase there are positions in which the rotor would stay stuck. For example if the rotor flux is perpendicular to the stator it will not rotate. It can be observed that using thee phases the flux can be made to point in any direction. If a balanced current is applied trough the stator windings it will point in a rotating direction and the rotor will follow.

More pole-pairs can be added by maintaining the symmetry. Adding an extra pole-pair in the rotor would require to set it perpendicular to the old one. Three pole-pairs would be displaced 120 degrees and so on. For each new pole-pair another set of windings is also added following the same logic. It can be observed that with this configuration the

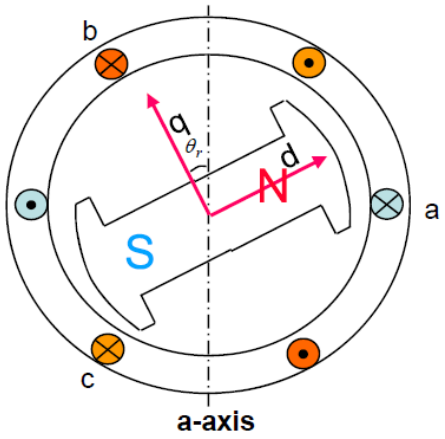


Figure 2.6.: One pole-pair PMSM representation as taken from [5]

effect is equivalent to having one pole-pair but with the current rotating as many times as number of pole-pairs faster than the mechanical speed. Therefore a machine with any number of poles can be simplified and studied as a one pole-pair machine taking into account that the electrical angle will always be $\theta_{el} = N_{pp}\theta_{mec}$ and then the electrical speed $\omega_{el} = N_{pp}\omega_{mec}$.

2.3.1 *Electrical machine model*

With reference to the neutral of the machine the voltage equations in *abc* are given by Equation 2.15. The equations will be described in *abc* and later transformed to *dq*0 since they become simpler. Most derivations are based in the lectures from [5].

$$\begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix} = R \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + \frac{d}{dt} \begin{bmatrix} \lambda_a \\ \lambda_b \\ \lambda_c \end{bmatrix} \tag{2.15}$$

The flux has the form shown in Equation 2.16 where $L_{leakage,abc}$ and $L_{main,abc}$ are 3x3 matrices. The components of those matrices are position dependent and can be derived by using the vector projection method also described in [5] and are omitted here.

$$\lambda = \begin{bmatrix} \lambda_a \\ \lambda_b \\ \lambda_c \end{bmatrix} = L_{leakage,abc} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + L_{main,abc} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + \lambda_{pm,abc} \tag{2.16}$$

$$\lambda_{pm,abc} = \lambda_{mpm} \begin{bmatrix} \cos(\theta_e) \\ \cos(\theta_e + 2\pi/3) \\ \cos(\theta_e - 2\pi/3) \end{bmatrix} \tag{2.17}$$

The inductor values which are present in $L_{leakage,abc}$ and $L_{main,abc}$ and which sum can be denoted simply as L_{abc} are function of the air-gap and therefore in a fixed reference frame such as *abc* the terms of Equation 2.16 will not be constant if the machine is salient. The term $\lambda_{pm,abc}$ is also position dependent and is given by Equation 2.17. Having this into account changing the reference frame to one that is fixed in the rotor and therefore

always points at the same direction will simplify the position-dependent formulas and make them constant. Some derivations about reference frame theory are added in Section B.1.

In a $qd0$ reference frame the d-axis points at the direction of the north of the rotor and the q-axis is rotated 90 degrees positively as is shown in Figure 2.6. The transformation matrix from abc to $qd0$ can be denoted as K_{abc}^{qd0}. To transform from abc then to this coordinate system both sides of Equation 2.15 can be multiplied by this matrix on the left-side. This will produce Equation 2.18.

$$K_{abc}^{qd0}\begin{bmatrix}u_a\\u_b\\u_c\end{bmatrix} = u_{qd0} = K_{abc}^{qd0}R\begin{bmatrix}i_a\\i_b\\i_c\end{bmatrix} + K_{abc}^{qd0}\frac{d}{dt}\begin{bmatrix}\lambda_a\\\lambda_b\\\lambda_c\end{bmatrix} \tag{2.18}$$

Since $R = R \times I$ the transformation matrix can enter in i_{abc} producing Ri_{qd0}. This cannot be done in the second term which needs a special treatment. Denoting the time derivative of $x(t)$ as $x'(t)$ it is always true that $(a(t)b(t))' = a'(t)b(t) + a(t)b'(t)$ and therefore $a(t)b'(t) = (a(t)b(t))' - a'(t)b(t)$. In this case $a(t) = K_{abc}^{qd0}$ and $b(t) = \lambda_{abc}$. The first term produces directly $\frac{d}{dt}(\lambda_{qd0})$. In the second λ_{abc} can be multiplied by $K_{qd0}^{abc}K_{abc}^{qd0}$ since is I. This will produce $\frac{d}{dt}(K_{abc}^{qd0})K_{qd0}^{abc}$ which can be operated and is simplified to $-\omega_e T$ where T is given by Equation 2.22.

The term λ_{qd0} is calculated as depicted in Equation 2.21. The term $K_{abc}^{qd0}i_{abc}$ is simply i_{qd0} and $K_{abc}^{qd0}L_{abc}K_{qd0}^{abc}$ can be denoted as L_{qd0} which is shown in Equation 2.20. As for $K_{abc}^{qd0}\lambda_{pm,abc}$ it can be operated to $\lambda_{pm,qd0}$ which is shown in Equation 2.21.

$$\lambda_{qd0} = K_{abc}^{qd0}\lambda_{abc} = K_{abc}^{qd0}L_{abc}K_{qd0}^{abc}K_{abc}^{qd0}i_{abc} + K_{abc}^{qd0}\lambda_{pm,abc} = L_{qd0}i_{qd0} + \lambda_{pm,qd0} \tag{2.19}$$

$$L_{qd0} = K_{abc}^{qd0}L_{abc}K_{qd0}^{abc} = \begin{bmatrix}L_q & 0 & 0\\0 & L_d & 0\\0 & 0 & L_0\end{bmatrix} \tag{2.20}$$

$$\lambda_{pm,qd0} = K_{abc}^{qd0}\lambda_{pm,abc} = \begin{bmatrix}0\\\lambda_{mpm}\\0\end{bmatrix} \tag{2.21}$$

$$T = \begin{bmatrix}0 & 1 & 0\\-1 & 0 & 0\\0 & 0 & 0\end{bmatrix} \tag{2.22}$$

For reference the final electrical equations in $qd0$ of the PMSM are depicted in Equation 2.23.

$$u_{qd0} = Ri_{qd0} + \frac{d}{dt}\lambda_{qd0} + \omega_e T\lambda_{qd0} \tag{2.23}$$

If the flux and inductors are treated as a constant then the differential term in Equation 2.23 can be further simplified and the final equations are given by Equation 2.24.

$$u_{qd0} = Ri_{qd0} + L_{qd0}\frac{d}{dt}i_{qd0} + \omega_e T\lambda_{qd0} \tag{2.24}$$

2.3.2 *Mechanical machine model*

The instantaneous power in the electrical machine can be calculated as in any other electrical system by multiplying the voltage by the current. In $qd0$ however this the power formulas must be derived. The abc variables in the power equation given by $P_{abc} = i_{abc}^T u_{abc}$ can be substituted using $x_{abc} = K_{qd0}^{abc} x_{qd0}$ in both current and voltage. Simplifying terms the power equation is found to be Equation 2.25.

$$P_{qd0} = \frac{3}{2}(i_q u_q + i_d u_d + 2i_0 u_0) \tag{2.25}$$

The voltage equation in Equation 2.24 can then be multiplied on the left side by $3/2 \begin{bmatrix} i_q & i_d & 2i_0 \end{bmatrix}$ to obtain the electrical power. The instantaneous power in the machine is then given by Equation 2.26 - 2.29.

$$P_{qd0} = P_q + P_d + P_0 \tag{2.26}$$

$$P_q = \frac{3}{2}(Ri_q^2 + L_q i_q \frac{d}{dt} i_q + \omega_e(\lambda_{mpm} + L_d i_d) i_q) \tag{2.27}$$

$$P_d = \frac{3}{2}(Ri_d^2 + L_d i_d \frac{d}{dt} i_d - \omega_e L_q i_q i_d) \tag{2.28}$$

$$P_0 = 3(Ri_0^2 + L_0 i_0 \frac{d}{dt} i_0) \tag{2.29}$$

There will be a heat generated in the machine inside P_{qd0} that will travel outside and will generate no torque. This term can be identified in Equation 2.26 - 2.29 as the resistive term, Ri_j^2. It can also be seen that the term $L_j i_j \frac{d}{dt} i_j$ is the power stored in the inductor and therefore cannot produce any real power in a period basis. Based on this it is evident that the 0 component produces no torque. The terms that produce useful mechanical power have been added together in Equation 2.30.

$$P_{qd0,useful} = \frac{3}{2}\omega_e i_q(\lambda_{mpm} + i_d(L_d - L_q)) \tag{2.30}$$

The power in Equation 2.30 will then be equal to the mechanical power given by $T_{el}\omega_{el}/N_{pp}$. Equating both terms result in the torque equation depicted in Equation 2.31.

$$T_{el} = \frac{3}{2}N_{pp} i_q(\lambda_{mpm} + i_d(L_d - L_q)) \tag{2.31}$$

If $L_q \neq L_d$ there are then multiple ways to generate a given torque. In the experimental setup the machine follows that $L_q \approx L_d$ and thus the torque may be simplified as Equation 2.32 where it can be seen that only the q current affects it in this case. Therefore it seems sensible to set $i_d^* = 0$ as it will only increase copper losses without producing any real power. There are also some core losses that depend of i_d and an optimization may be done to find the optimum i_d^* but it is considered outside of the scope of this project.

$$T_{el} = \frac{3}{2}N_{pp}\lambda_{mpm} i_q \tag{2.32}$$

Finally once the torque is calculated it can be used to obtain the speed by using Equation 2.33 where it is stated that the total torque is equal to the inertia times the angular speed plus a term that is speed dependent and J_0. Those parameters can be found in the datasheet.

$$T_{el} - T_{load} = J_m \frac{d}{dt}\omega_{mec} + B\omega_{mec} + J_0 \tag{2.33}$$

2.4 SYSTEM PARAMETERS

The diagram of the experiment setup if shown in Figure 2.7 which follows the diagram previously illustrated in Figure 2.1. On the left the two converters controlled trough DSpace are connected to the IM and PMSM mechanically coupled on the right. DSpace is also connected to the computer and variables can be seen and modified on real-time trough the Control Desk application.

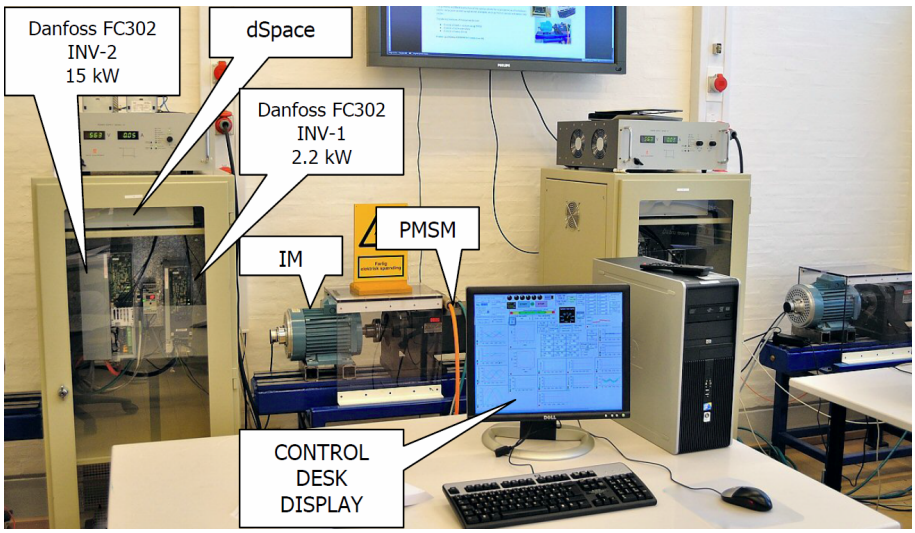


Figure 2.7.: Experiment setup

The relevant parameters of the different systems used in the experimental setup are shown below. Fields marked with * were obtained experimentally and the tests are described in Appendix A. The converter and machine power limits were taken from previous work from [3].

Parameter	Symbol	Value	Unit
Rated speed (mechanical)	n_n	4500	rev/min
Pole pairs	N_{pp}	4	$-$
Rated current	I_n	24.5	A
Rated power	P_n	9.42	kW
Rated torque	T_n	20	Nm
Machine + wire resistance	R	0.25*	Ω
Inductor q axis	L_q	2.15*	mH
Inductor d axis	L_d	2.03*	mH
Magnetic flux	λ_{mpm}	0.12*	Wb

Table 2.1.: Machine electrical parameters

Parameter	Symbol	Value	Unit
Drive shaft inertia	J_m	0.113*	Nms^2
Viscous damping	B	0.456*	$mNms$
Coulomb friction	J_0	0.194*	mH

Table 2.2.: Machine mechanical parameters

Parameter	Symbol	Value	Unit
Rated input current	I_{in}	29	A
Rated output current	I_{out}	32	A
Deadtime	t_d	2.5	μs
IGBT/diode on-voltage	v_{on}	1.2*	V
IGBT/diode resistive part	r_{on}	0.03*	Ω
Snubber capacitance	C_s	1.26	nF
Switching frequency	f_{sw}	3000	Hz
Sampling frequency	f_s	3000	Hz

Table 2.3.: Converter parameters

3 FIELD ORIENTED CONTROL

In this chapter the control of the PMSM with means of PI compensators is revised as they are typically used to control this type of machines. The controllers are explained and tuned analytically and lastly simulation and experimental results are presented.

3.1 CURRENT CONTROL

The current control of the system can be made by means of a PI. The blocks of the system and the controller are presented in Figure 3.1 - 3.2. The current error is fed into the PI which produces the voltage command that is fed to the SVM. The converter will have a delay that can be modeled as a first order system with time constant $\tau = 1.5T_s s$. The derivation is explained in several papers and [6] can be cited. There is a one period delay due to the DSP computation delay. Then the system may be modeled in s but with a zero order hold and sampler used which can be simplified to a delay of $0.5T_s s$. Hence the total delay is given by $1.5T_s$. The delay given by $e^{-1.5T_s}$ may then be approximated as a first order transfer function of $\frac{1}{1.5T_s s+1}$ so it is easier to handle. To simplify the design of the controller, the coupling terms are added with contrary sign after the controller so when they are added back inside the machine they can be considered canceled and the current control is a Single Input Single Output (SISO).

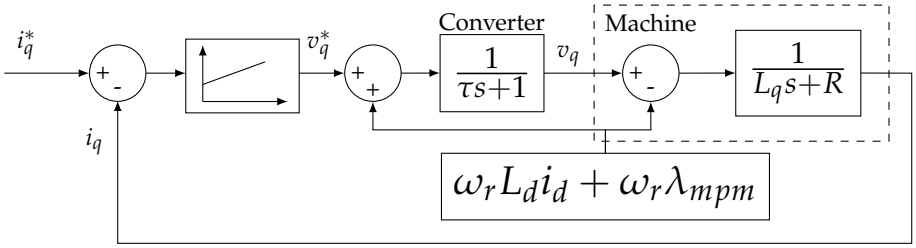


Figure 3.1.: Inner-loop for i_q

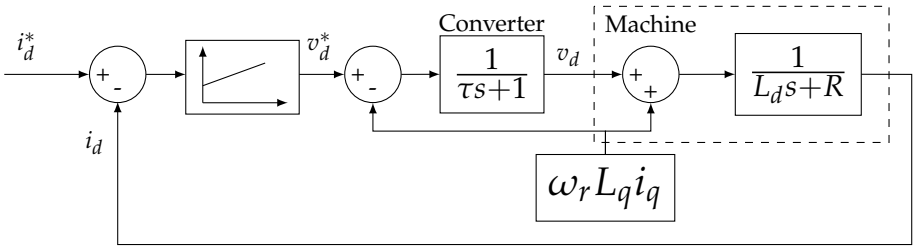


Figure 3.2.: Inner-loop for i_d

With the coupling terms neglected, classical control theory can be used to tune the parameters of the compensator. A way to tune the inner-loop could be by eliminating the

slow pole due to the RL system and then adjusting the gain for the desired performance. The PI is described by Equation 3.1 and thus to eliminate the slow pole of the machine rewritten as Equation 3.2 then Equation 3.3 is imposed.

$$D(s) = K_p + \frac{K_i}{s} = K_i \frac{\frac{Kp}{Ki}s + 1}{s} \tag{3.1}$$

$$G(s) = \frac{1}{L_q s + R} = \frac{1/R}{\frac{Lq}{R}s + 1} \tag{3.2}$$

$$\frac{Kp}{Ki} = \frac{L_q}{R} \tag{3.3}$$

With this compensator the whole system is now second order. Another equation can be imposed referred to the damping coefficient ξ. The term ξ affects mostly the overshoot and the term ω_n mostly the oscillations frequency. Both also affect the time constant. This is illustrated with step responses as depicted in Figure 3.3. The OL of the system is given by Equation 3.4 and the CL can be calculated as Equation 3.5.

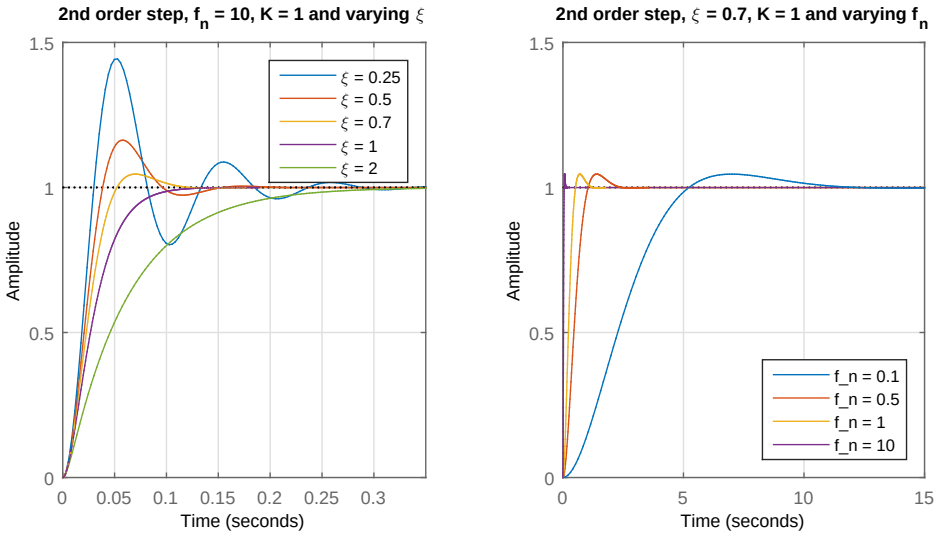


Figure 3.3.: Inner-loop for i_q

$$G_{ol}(s) = \frac{K_i}{R} \frac{1}{s\,(\tau s + 1)} \tag{3.4}$$

$$G_{cl}(s) = \frac{K_i/(R\tau)}{s^2 + s/\tau + K_i/(R\tau)} \tag{3.5}$$

By comparing Equation 3.5 with a second order system K_i can be obtained so the system has a determined damping ξ. The value obtained is given by Equation 3.6. Therefore the only parameter to choose is ξ and can be made $\xi = 0.7$ since it provides a 5% overshoot.

$$K_i = \frac{R}{4\xi^2\tau} \tag{3.6}$$

3.2 SPEED CONTROL

To achieve the desired speed the electrical torque can be adjusted by means of i_q as is shown in Equation 2.32 - 2.33. The current command is given by a PI where the speed error is fed as an input. The blocks are shown in Figure 3.4.

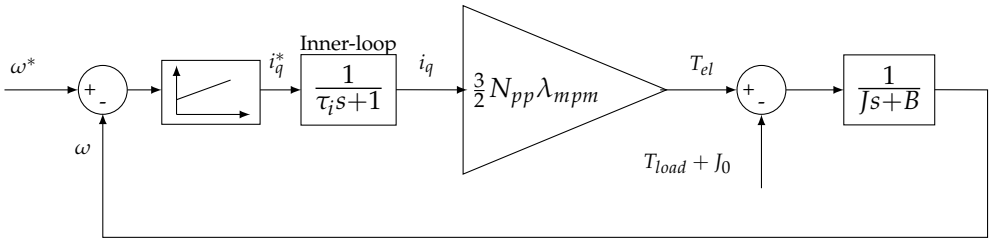


Figure 3.4.: Outer-loop for ω

Classical control theory can be used to tune the parameters of the compensator. To simplify the design, the inner-loop is simplified as a first order system. This approximation is valid since the inner-loop (both in the case of the deadbeat or the PI controller) is designed to have a small overshoot and thus the behavior is similar to one of a first-order system. The time constant of the inner-loop for the PI can be obtained from Equation 3.5 since the close-loop transfer function is a second order system and is given by Equation 3.7.

$$\tau_i = \frac{1}{\xi\omega_n} = 2\tau \tag{3.7}$$

The open-loop transfer function of the simplified speed-loop is shown in Equation 3.8 and it is adjusted for the implementation in the experiment where $K_{rad}^{rpm} = 60/(2\pi)$ since the experiment model was later designed with the mechanical speed in rpm as input of the controller. On the other hand the output of the controller is set as torque and therefore the constant $\frac{3}{2}N_{pp}\lambda_{mpm}$ does not appear in the OL transfer function. It will appear as a gain later in the implementation to transform to current.

$$G_{speed}(s) = K_{rad}^{rpm}\left(K_p + \frac{K_i}{s}\right)\frac{1}{\tau_i s+1}\frac{1}{Js+B} \tag{3.8}$$

The pole/zero placement of Equation 3.8 is depicted in Figure 3.5 for $K_p = 0.1$ and $K_i = 1$. The system has initially a slow pole due to the mechanical nature at B/J and a fast one due to the current loop at $1/\tau_i$. The PI introduces a pole in the origin and a zero that can be placed anywhere with the correct gains.

Using Sisotool in Matlab it can be seen that there are several solutions that in simulations provide similar results. The speed controller can be set at least 10 times slower than the current loop so they do not couple and interfere with each other. Going to this limit however gives high gains that cannot be used experimentally.

One proposed solution is $K_p = 0.0725$ and $K_i = 2.5$. The pole/zero placement for those gains is depicted in Figure 3.6.

The step response with the proposed controller is shown in Figure 3.7.

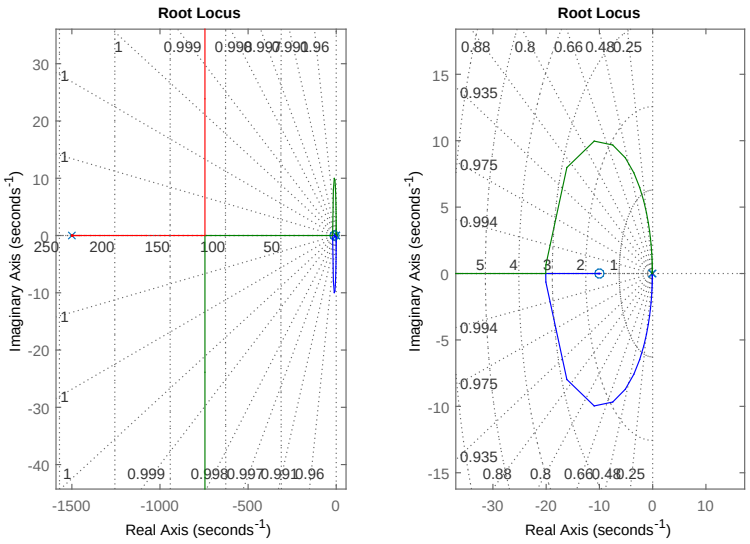


Figure 3.5.: Pole/zero placement of speed loop

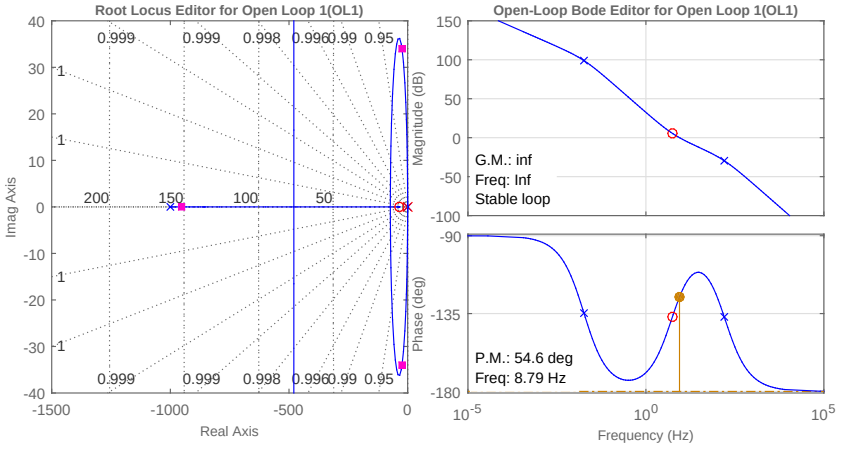


Figure 3.6.: Pole/zero placement of speed loop

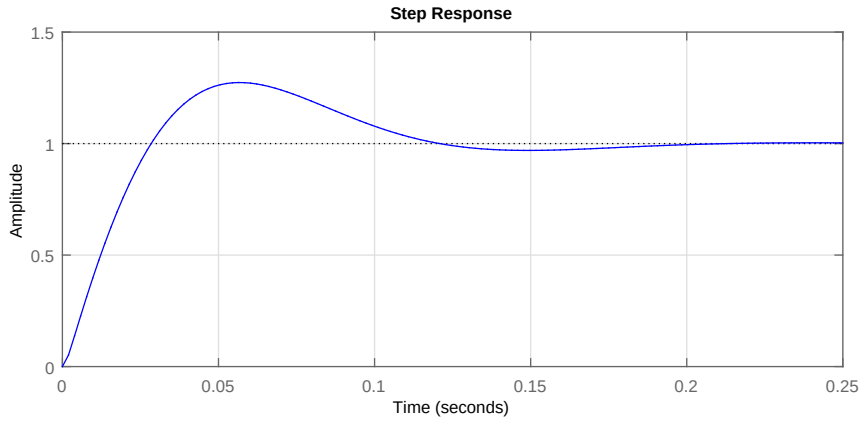


Figure 3.7.: Speed step with tuned parameters

If the gains are multiplied by 8 a faster controller with less overshoot is proposed and the step responses are depicted in Figure 3.8. While it is 10 times faster than the proposed one it still meets the requirements of being 10 times slower than the current loop.

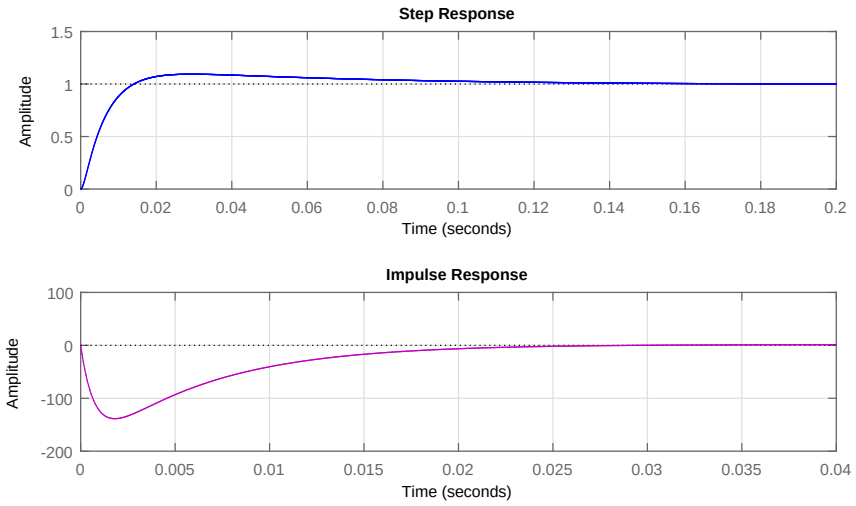


Figure 3.8.: Speed step and noise impulse response for faster solution

This solution however does not work experimentally. Some other controllers have been tested but it has been impossible to experimentally implement a controller much faster than the one proposed and it is hypothesized is due to the lack of noise rejection. Also the parameters used in the mechanical system were obtained in tests and are subject to errors. Moreover the IM torque controller may couple with the PMSM and influences the response. If the overshoot is checked in a step response, when the IM is connected it grows noticeably even when the torque is set to zero. While the controller was tested with the IM disconnected so the torque controller does not couple, in the real testing it will affect the performance.

3.3 ANTI WIND-UP

The PIs will give the command to the plant without any constraint. If the error is big enough the controller may require too much command that would damage the system. For this purpose based on the limits imposed by the system a saturation block can be placed at the end of the voltage command. However the integrator in the PI will continue integrating even though it will have no effect and would take a lot of time to reset after the error is finally drop. For this reason an anti-windup technique can be implemented in any of the PI used in the system. The schematics are shown in Figure 3.9.

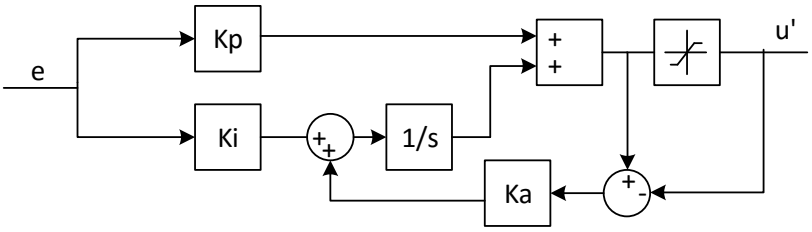


Figure 3.9.: Anti wind-up scheme

When the command is too big and it is saturated the difference is fed to the integrator to stop it from keeping integrating. When the command given by the controller is below the limits then the anti wind-up does nothing.

3.4 SIMULATION RESULTS

3.4.1 *Current loop*

The current response for the PI in continuous and discrete time (with forward-euler discretization) for $f_s = 3000Hz$ and $\xi = 0.7$ using Equation 3.3 - 3.6 is shown in Figure 3.10. It is also shown the simulated response using Equation 3.5.

The coupling terms have been added as a feed-forward term to the voltage command. The response in continuous and discrete time (with tustin) is almost the same and therefore there is no need to design the controller in $\mathcal{Z}$. The response in the q axis is also very similar to the modeled one and the origins of the errors may be due to the PWM approximations taken. As for the d axis the changes when i_q changes may be to the coupling terms not being perfectly canceled out in the transient.

3.4.2 *Speed loop*

The comparison of the proposed controller in the analytical model and simulations is depicted in Figure 3.11. It can be seen a great concordance between both models. In the response it is also shown the difference between including or neglecting the first order approximation of the current loop which can be seen to not be important and could be neglected.

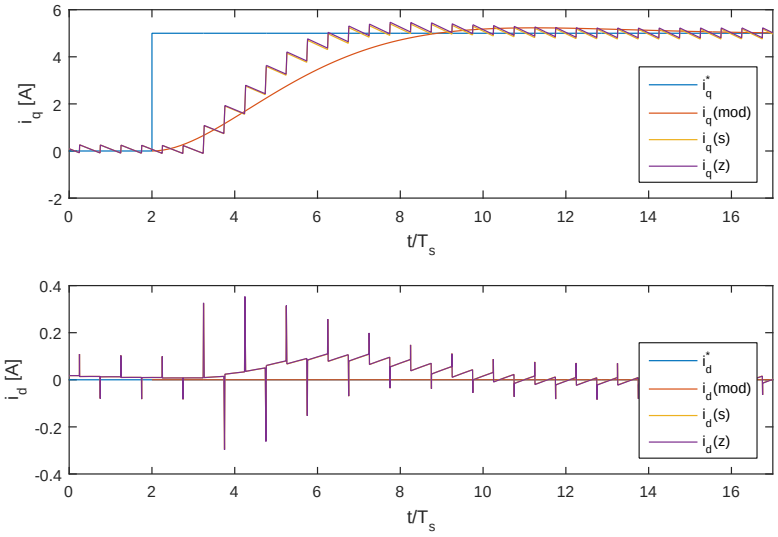


Figure 3.10.: PI current response for $\xi = 0.7$

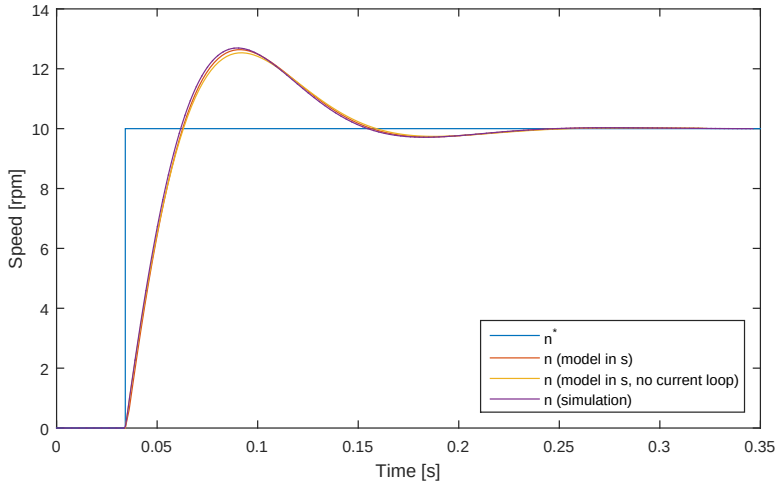


Figure 3.11.: Speed loop response comparison simulation vs modeling

3.5 EXPERIMENTAL RESULTS

3.5.1 *Current controller*

Different controllers have been tested and the results are shown later in Figure 4.4 with the predictive controller also.

3.5.2 *Speed controller*

The speed controller has been tested in the experiment with the gains proposed. The results are depicted in Figure 3.12. The settling time follows the theory but the overshoot is increased almost to double and the shape of the speed is also modified. Since the focus of the project is the predictive controller a ramp limitation as discussed in [1] has been added to the speed error with a relative low limit of $2000 rpm/s$ to smooth the response and is shown in the same graph.

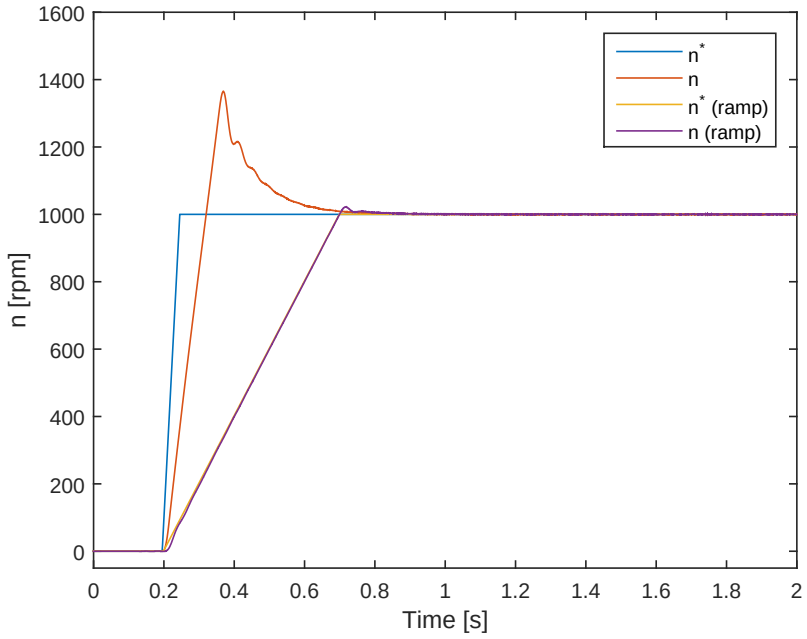


Figure 3.12.: Speed controller step response in the experiment

4 PREDICTIVE CURRENT CONTROL

In this chapter the predictive controller is presented. The controller which uses machine equations and different estimations of variables is derived. At the end simulation and experimental results are shown.

4.1 DESCRIPTION OF THE ALGORITHM

Predictive control in a PMSM is theoretically able to produce the desired reference in a minimum amount of time without compromising the stability or inducing steady-state errors. However this kind of performance can only be achieved if the model of the system is good enough and the system parameters are accurately determined. Because of this, parameter sensibility analysis are performed in Chapter 5. Non-linearities are compensated in Chapter 6 and an online parameter estimator method is developed in Chapter 7.

The voltage needed to achieve the current reference is calculated in the deadbeat controller by using the system equations and calculating it based on the previous current and speed. Depending of how the PWM is handled the reference may be achieved either in one or two period but never in less than one. If the PWM signal enters in the next period the reference can be achieved in 2 since the current will achieve the true value after the voltage was applied. On the other hand if the PWM enters shortly after calculation the current reference can be achieved then in only one period as happens in the experiment. Saturation of voltage commands or the average approximations taken may make the real controller be slower in certain conditions where a big change is demanded but otherwise the response can be kept within a minimum time without compromising the transient stability.

The idea of this type of controller is to use the machine equations to calculate the voltage needed to achieve the given reference. Since approximations of the means are used and the PWM effect neglected on reality a value very close to the reference is usually achieved in the first iteration. In comparison the classical PI is not able to achieve the reference in such a short time and reducing the time will only produce overshoot [7].

Because the voltage command takes some computation time it is forced to be delayed strictly one period. In the experiment DSpace passes the command immediately to the PWM and then two different controllers are developed. A first approach will consider the common DSP delay and compensate it while the second one designed for this experimental testing without the delay. Both controllers are based in the same idea of using averages in both sides of the equations.

4.1.1 2-period controller

Assuming we are in period k we want to calculate the voltage needed to be applied in periods $k+1$ that will produce the current in $k+2$ as represented in Figure 2.3. If that is the case, the equations of the PMSM can be rewritten using averages for those 2 future periods. The average of a continuous function is defined as $\frac{1}{T}\int_0^T x(t)dt$ and therefore when doing it in both sides of the motor equations the substitutions in Equation 4.1 - Equation 4.3 can be made which are also used in [7].

$$\langle u \rangle = \frac{1}{2T_{sw}} \int_0^{2T_{sw}} u(t)dt = \frac{u_k + u_{k+1}}{2} \tag{4.1}$$

$$\langle i \rangle = \frac{1}{2T_{sw}} \int_0^{2T_{sw}} i(t)dt \approx \frac{i_k + i_{k+2}}{2} \tag{4.2}$$

$$\left\langle \frac{di}{dt} \right\rangle = \frac{1}{2T_{sw}} \int_0^{2T_{sw}} \frac{di}{dt}dt = \frac{1}{2T_{sw}} \int_0^{2T_{sw}} di = \frac{i_{k+2} - i_k}{2T_s} \tag{4.3}$$

Furthermore, the simplification that ω is constant is done and the value of $\omega(k)$ is taken. To get the desired current, then $i_{k+2} = i_k^*$. The final result for both the d and q axes is shown in Equation 4.9 - 4.10. It is important to note that Equation 4.1 and 4.3 are exact calculations while Equation 4.2 is an approximation. It could be improved by predicting the evolution of the current over the 2 periods. A third point can be added in the middle between both points at i_{k+1} that can be calculated from the voltages given in the previous periods. Then Equation 4.2 may be substituted by Equation 4.4 to have a better response.

$$\langle i \rangle \approx \frac{i_k + i_{k+1} + i_{k+2}}{3} \tag{4.4}$$

The predicted current can be obtained as follow. If there was no saturation in the voltage $i_{k+1} = z^{-1}i^*$. Otherwise it can be estimated. The voltage that is going to be used in the PWM is the previous one given by the controller, named u_{qd}. Knowing the sampled current in the period i_{qd} then the predicted current i_{qd}^p may be estimated. The machine equations can be written with averages as Equation 4.5. Then the current in the next period which wants to be predicted can be solved as Equation 4.8 where A and B are given in Equation 4.6 - 4.7.

$$u_{qd} = Ai_{qd}^p + Bi_{qd} + \begin{bmatrix} \lambda_{mpm}\omega_e \\ 0 \end{bmatrix} \tag{4.5}$$

$$A = \begin{bmatrix} \frac{R}{2} + \frac{L_q}{T_s} & \omega_e \frac{L_d}{2} \\ -\omega_e \frac{L_q}{2} & \frac{R}{2} + \frac{L_d}{T_s} \end{bmatrix} \tag{4.6}$$

$$B = \begin{bmatrix} \frac{R}{2} - \frac{L_q}{T_s} & \omega_e \frac{L_d}{2} \\ -\omega_e \frac{L_q}{2} & \frac{R}{2} - \frac{L_d}{T_s} \end{bmatrix} \tag{4.7}$$

$$i_{qd}^p = A^{-1}\left(u_{qd} - Bi_{qd} - \begin{bmatrix} \lambda_{mpm}\omega_e \\ 0 \end{bmatrix}\right) \tag{4.8}$$

To save computation time the matrix A can be modified and only contain the differential terms and in B remove the 2 in the denominator of the terms divided by it. This is equivalent to approximating the change using the previous current in the terms without the differentiator and provides the advantage that the inversion of A is $[T_s/L_q, 0; 0, T_s/L_d]$ and thus little computation time is needed since a simple equation can be derived without $2x2$ matrices inversions.

$$u_q(k+1) = 2\left(R\frac{i_q^* + i_q(k)}{2} + L_q\frac{i_q^* - i_q(k)}{2T_s} + \omega(k)\left(L_d\frac{i_d^* + i_d(k)}{2} + \lambda_{mpm}\right)\right) - u_q(k) \tag{4.9}$$

$$u_d(k+1) = 2\left(R\frac{i_d^* + i_d(k)}{2} + L_d\frac{i_d^* - i_d(k)}{2T_s} - \omega(k)L_q\frac{i_q^* + i_q(k)}{2}\right) - u_d(k) \tag{4.10}$$

4.1.2 *1-period controller*

Based on experiments the way the PWM is handled in DSpace seems a bit different and the voltage command enters as soon as it is calculated as it was explained before in Figure 2.2 and therefore the controller can be modified to work in such a system.

The PMSM equations are modified as it was done in the 2-period controller by calculating averages an in this case the following substitutions can be done as shown in Equation 4.11 - Equation 4.13.

$$\langle u \rangle = \frac{1}{T_{sw}}\int_0^{T_{sw}} u(t)dt = u_k \tag{4.11}$$

$$\langle i \rangle = \frac{1}{T_{sw}}\int_0^{T_{sw}} i(t)dt \approx \frac{i_k + i_{k+1}}{2} \tag{4.12}$$

$$\left\langle \frac{di}{dt} \right\rangle = \frac{1}{T_{sw}}\int_0^{T_{sw}} \frac{di}{dt}dt = \frac{1}{T_{sw}}\int_0^{T_{sw}} di = \frac{i_{k+1} - i_k}{T_s} \tag{4.13}$$

Since the current is desired to be the reference in $k+1$ then $i_{k+1} = i^*$ and the final controller equations are given by Equation 4.14 - Equation 4.15.

$$u_q(k+1) = R\frac{i_q^* + i_q(k)}{2} + L_q\frac{i_q^* - i_q(k)}{T_s} + \omega(k)\left(L_d\frac{i_d^* + i_d(k)}{2} + \lambda_{mpm}\right) \tag{4.14}$$

$$u_d(k+1) = R\frac{i_d^* + i_d(k)}{2} + L_d\frac{i_d^* - i_d(k)}{T_s} - \omega(k)L_q\frac{i_q^* + i_q(k)}{2} \tag{4.15}$$

4.2 ANGLE COMPENSATION

At high speeds it is important to estimate the angle to use in the dq transformation to $\alpha\beta$ used in the modulator. Since the command is always delayed the angle measured will not be the same as the angle when the voltage commanded enters the system. This is represented in Figure 4.1.

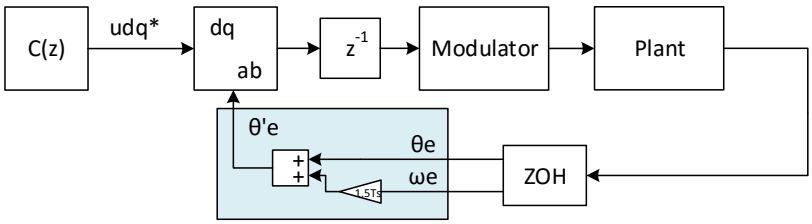


Figure 4.1.: Angle compensation diagram

In the 2-period version the average angle will be close to $\theta'_e = \theta_e + \omega_e 1.5T_s$ since the command will enter after T_s and will be until $2T_s$. If the speed is constant then the average angle will be modified by $\omega_e 1.5T_s$. In the 1-period version it would be $\theta'_e = \theta_e + \omega_e 0.5T_s$.

4.3 SIMULATION RESULTS

A simulation response to two steps in q and d of $10A$ setting the other reference to $10A$ is shown in Figure 4.2 for a speed of $2\pi 50$ which is $1/6$ of nominal. The non-linearities in the converter are simulated and compensated with the discrete implementation later presented.

The results in Figure 4.2 show the predictive controller is able to track the reference in almost the 2 periods needed due to the delay in the voltage command. The small error is a combination of the the non-linearity compensation as the current changes completely from one period to another and there may be a small error in the transient and the average approximation used to deduct the equations. In the same graph the PI tuned for $\xi = 0.7$ is also shown. It can be seen the response is slower in the PI which takes around 4 more periods. It also presents a small overshoot of 5.4% as expected from the analytical tuning. It can be seen the coupling in the currents is also bigger in the PI.

In Figure 4.3 the same test is repeated for a speed of $2\pi 250$ (83% rated). It can be seen the predictive controller behaves similarly although the coupling errors have been increased. In the case of the PI the response is worse as the overshoot is increased and the step would need even more time to reach the steady-state. The coupling errors have also been increased. The reason of such changes of performance with speed in the PI may be due to the coupling terms. Even though they have been compensated the implementation is clearly worse than in the predictive controller where the current is predicted.

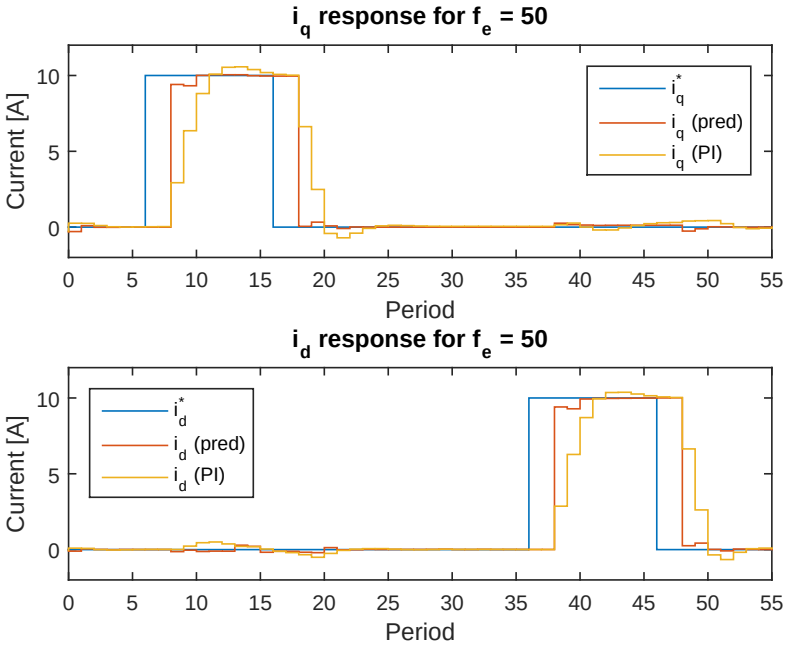


Figure 4.2.: Step response at 50 Hz

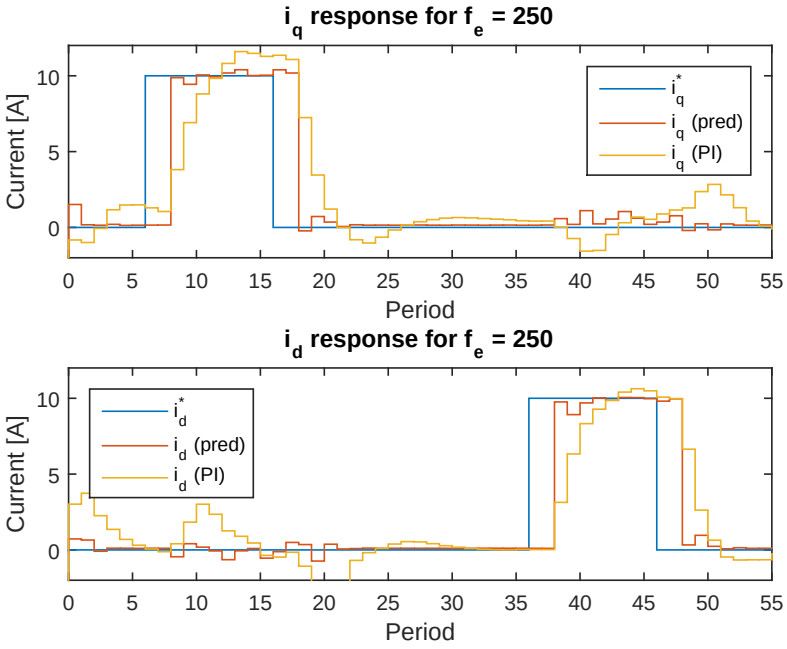


Figure 4.3.: Step response at 250 Hz

4.4 EXPERIMENTAL RESULTS

Results of the comparison between the predictive controller with 1-period delay and different PIs set for different gains are shown in Figure 4.4. With the machine running a step is given to i_d^* since it is easier than to i_q as it does not affect the torque. The responses in Figure 4.4 show great similarity with Figure 4.2 (apart form the part of the period difference due to the 1-period implementation).

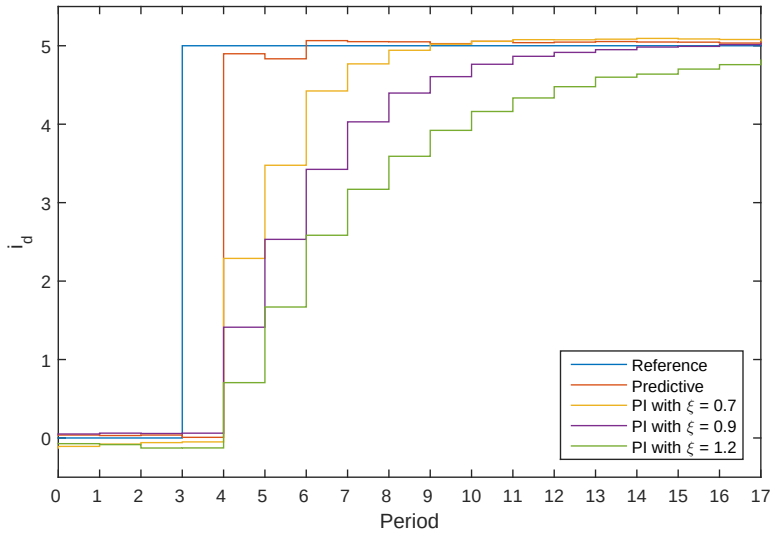


Figure 4.4.: Comparaison between response of predictive controller and PIs

The graph shows that as studied the predictive controller is able to achieve the reference in one period only. This plot also shows that the non-linearites have been correctly compensated as the SS-error is close to zero. It can also be seen that the predictive controller reaches the reference in only one period and thus shows that effectively DSpace does not introduce the delay after the command and that the results are adequate (voltage distortion due to computational delay can be neglected).

On the other hand the PI for $\xi = 0.7$ also follow a similar response to the one in simulations for Figure 3.10 which was tuned the same way.

5 PARAMETER SENSITIVITY ANALYSIS

In this chapter the effect of parameter variation in the predictive controller is studied. Since this compensator does not have adapt mechanisms the performance can be weakened by errors in the machine parameters.

The system is modeled using Multiple Inputs Multiple Outputs (MIMO) theory as the dq axes are not decoupled. Then the controller is also realized in matrix form. The whole system can be represented with linear matrices and the stability assessed with a pole map. Simulations and experiments are performed to validate the claims.

Later steady-state errors calculation is shown. Because several factors play a role on them after the analytical derivation an approximation will be derived to have a better understanding of the effect of each element in the global error. Simulations are performed to validate the approximations.

In the last part the transient is analyzed. Since there are several poles and zeros between system and controller and they are quite close in the pole/zero map it is not easy to develop intuitions about the effect of each parameter change in the transient so some approximations are also taken to understand the main factors that play a bigger role in the transient. At the end simulation for different conditions are shown to validate the simplifications.

5.1 SYSTEM MODEL

In this section the analytical model of the system used in stability will be derived. First the state-space model of the plant is derived. Later the discretization method is shown and lastly the predictive controller is also derived.

5.1.1 *State-space plant*

In state-space a system can be represented by Equation 5.1 - 5.2. The block diagram of such a system is depicted in Figure 5.1 where each state is a vector and each block a matrix.

The x vector is a state-variable, while u is a vector command. By modifying u the values of x are changed over time so in the case of the PMSM x can be seen as the current and u as the voltage. To only obtain the desired state-variables in the final result the relevant x and u variables can be selected by choosing C and D.

$$\frac{d}{dt}x = Ax(t) + Bu(t) \tag{5.1}$$

$$y(t) = Cx(t) + Du(t) \tag{5.2}$$

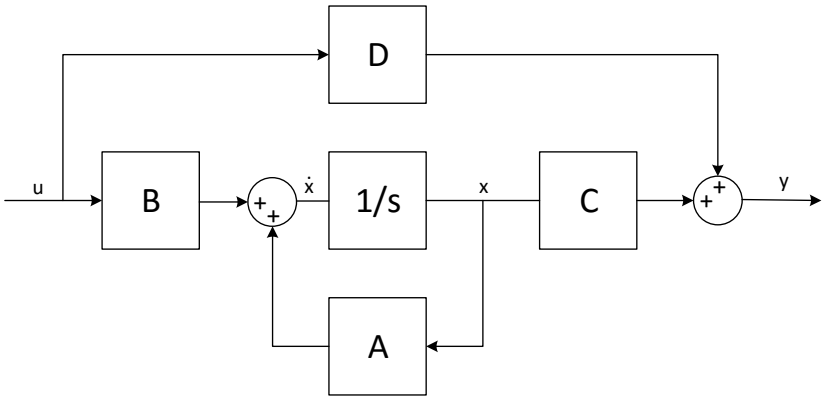


Figure 5.1.: State-space representation

The machine equations can be written in matrix form as Equation 5.3 where the flux linkage has been removed. The speed is considered to change much slower than the current and therefore will be treated as a constant from the mathematical point of view as otherwise the system would be non-linear and the analysis would be much more complicated. Under this assumption the flux voltage term can then be considered as a perturbation to the system that can be later added and considered.

$$u'_{qd} = \begin{bmatrix} R & \omega_e L_d \\ -\omega_e L_q & R \end{bmatrix} i_{qd} + s \begin{bmatrix} Lq & 0 \\ 0 & Ld \end{bmatrix} i_{qd} \tag{5.3}$$

By comparing Equation 5.3 with Equation 5.1 - 5.2 A and B can be determined for the case of $x = i_{qd}$ and $u = u'_{qd}$. The result is found by solving si_{qd} which respect to i_{qd} and u'_{qd}. The matrices that multiply those variables are directly A and B and are written in Equation 5.4 - 5.5.

$$A = \begin{bmatrix} \frac{R}{Lq} & \frac{L_d}{L_q}\omega_e \\ -\frac{L_q}{L_d}\omega_e & \frac{R}{Ld} \end{bmatrix} \tag{5.4}$$

$$B = \begin{bmatrix} \frac{1}{Lq} & 0 \\ 0 & \frac{1}{Ld} \end{bmatrix} \tag{5.5}$$

As we are interested in knowing the current we can set $y = i_{qd}$ by simply setting $C = I$ (2x2 unity matrix) and $D = 0$ (2x2 zero matrix). The block diagram of a state-space system is depicted in Figure 5.1 and using the cascade and feedback rule for MIMO systems the transfer function $G_{pmsm}(s)$ of i_{qd}/u'_{qd} can be calculated as Equation 5.6.

$$G_{pmsm}(s) = C\left(sI - A\right)^{-1} B + D \tag{5.6}$$

5.1.2 *Machine discretization*

The real machine is a continuous system but is driven by PWM which has a discrete nature. Taking into account the PWM is complicated and usually averages or other

simplifications are done. Simulations can be done in circuit simulators such as PLECS where those components are considered and will be seen that the PWM effect can be neglected as long as the switching frequency is fast enough. If it is not the performance of the controller would otherwise be also affected and to get a solution some simplifications must be done.

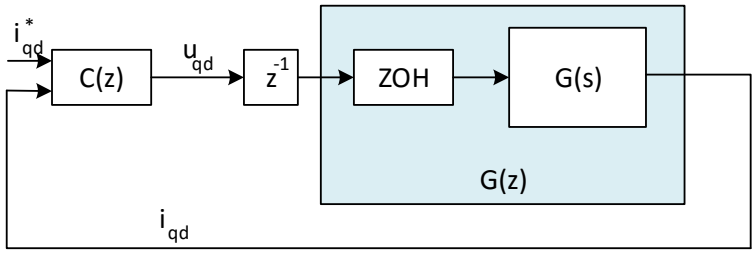


Figure 5.2.: System modeling with ZOH

The proposed technique to have an analytical model is shown in Figure 5.2. The discrete controller (predictive controller in this case) will give the reference to the PWM after a delay to simulate the DSP computation time. The reference then is passed trough a ZOH and therefore the plant may be discretized in Matlab by using the $c2d(G, Ts,' zoh')$ where Ts is the sampling time. This ZOH simulates the PWM behavior as the voltage reference is obtained in average during the period. Same techniques are used to model other PWM driven systems. For example in grid connected inverters the same approximation is taken in [8] which can be cited as a reference.

5.1.3 *Predictive controller*

The predictive controller can be modeled as the diagram shown in Figure 5.3 for the 2-period version. It can be derived by realizing it is the same as the 1-period controller formula with $2T_s$ instead of T_s and by modifying the voltage as $u' = 2u - z^{-1}u'$ which is the 2-period modification to compensate the delay if the predicted current in the average current is neglected as can be seen in Equation 4.9 - 4.10.

Finally from Figure 5.3 the close-loop transfer function of the controller and plant can be derived. From before the double sum to u_{qd} the diagram can be simplified as $D(z) = 2\frac{1}{1+z^{-1}}z^{-1}$. Then plant $G(z)$ and $D(Z)$ can be combined into $G'(z) = D(z)G(z)$. Finally the close-loop function is given by Equation 5.7.

$$G_{cl}(z) = G'(z)\left(I - K_2 G'(z)\right)^{-1} K_1 \tag{5.7}$$

The matrices K_1 and K_2 are shown in Equation 5.8 - 5.9 and are the implementation of Equation 4.9 - 4.10 in matrix form neglecting the predicted current.

$$K_1 = \begin{bmatrix} \frac{R'}{2} + \frac{L'_q}{2T_s} & \omega_e \frac{L'_d}{2} \\ -\omega_e \frac{L'_q}{2} & \frac{R'}{2} + \frac{L'_d}{2T_s} \end{bmatrix} \tag{5.8}$$

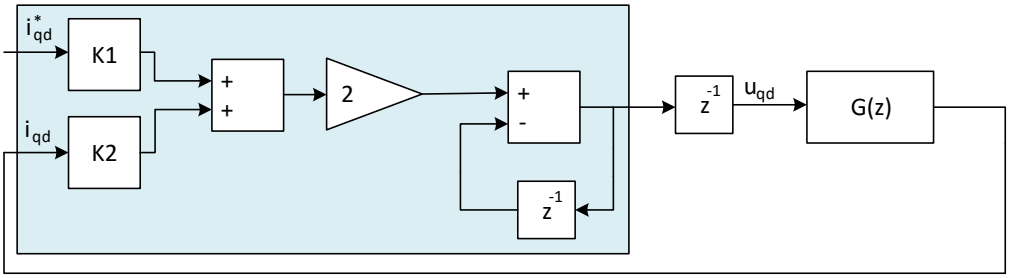


Figure 5.3.: Predictive controller 2-period model

$$K_2 = \begin{bmatrix} \frac{R'}{2} - \frac{L'_q}{2T_s} & \omega_e \frac{L'_d}{2} \\ -\omega_e \frac{L'_q}{2} & \frac{R'}{2} - \frac{L'_d}{2T_s} \end{bmatrix} \tag{5.9}$$

As for the 1-period version the model is depicted in Figure 5.4 and the matrix gains are shown in Equation 5.10 - 5.11. It can be seen are the same as Equation 5.8 - 5.9 but with T_s instead of $2T_s$. The close-loop transfer function is equivalent to the 2-period version but with $G'(z) = G(z)$ in Equation 5.7.

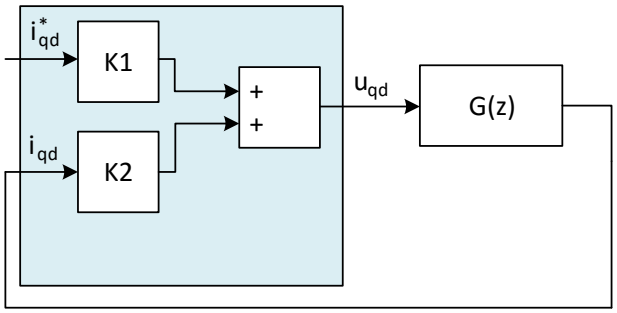


Figure 5.4.: Predictive controller 1-period model

The equations of the predictive controller will be rewritten using averages as was shown in for the 1-period version. The new matrix gains are then given by Equation 5.10 - 5.11.

$$K_1 = \begin{bmatrix} \frac{R'}{2} + \frac{L'_q}{T_s} & \omega_e \frac{L'_d}{2} \\ -\omega_e \frac{L'_q}{2} & \frac{R'}{2} + \frac{L'_d}{T_s} \end{bmatrix} \tag{5.10}$$

$$K_2 = \begin{bmatrix} \frac{R'}{2} - \frac{L'_q}{T_s} & \omega_e \frac{L'_d}{2} \\ -\omega_e \frac{L'_q}{2} & \frac{R'}{2} - \frac{L'_d}{T_s} \end{bmatrix} \tag{5.11}$$

5.2 STABILITY

The stability of the controller can be assessed by modifying each system parameter independently (denoted without an apostrophe) and seeing the limits after where the system becomes unstable as the controller is not updating the parameters (denoted with an apostrophe). A simple way to represent the stability would be to perform a DC sweep in each system parameter and plot the poles position (of each element in the matrix $G_{cl}(z)$) in a pole map. If all the poles are inside the unit circle the system is stable. To see the data better another graph can be added where the maximum absolute pole value is plot vs the parameter change proportion. As long as this value is kept below 1 the system is stable.

The system stability is dependent of the speed and sampling time. Another set of graphs could be done where those variables are also modified to see the effect in the system stability. However the result of testing at different speeds and switching frequencies showed almost no changes in the limits and is omitted for visibility of the pole map.

5.2.1 *Resistor stability*

The two graphs proposed are shown for the resistor in Figure 5.5 for both controllers where the controller parameters are set to the system's except for the resistor of which the proportion of the system value over the controller (R/R') is changed from -1 to 5.

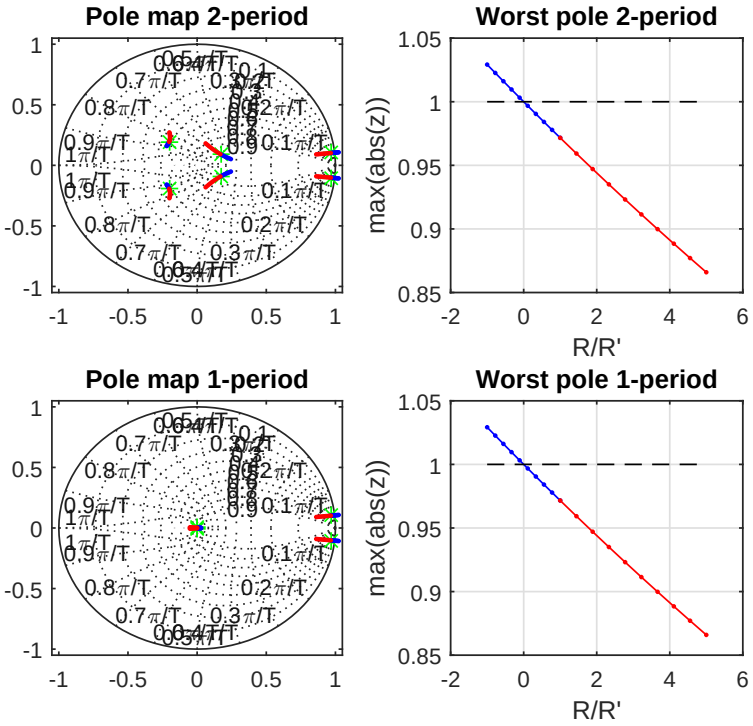


Figure 5.5.: Resistor error effect in stability in ZOH model

The blue values represent reductions in R and the red ones increments. It can be seen on the left the movement of all the system poles over the change in the ratio R/R'. The start is the green dot ($R = R'$). For simple results the worst pole location is shown in the right graph where the limit of stability is found at 0.

Even though the real resistor cannot be negative it is interesting to see there is a lower limit at 0 for which the system becomes unstable. This limit can be verified in the analytical model without PWM. There is no upper limit, therefore the results of the analysis with a ZOH predict that the resistor alone cannot make the system unstable unless negative (which is not physically possible) and offer perspectives regarding the fact that this element may not play a big role in stability. It is also interesting to note that the real resistor will increase in value due to temperature and the graph of the right of Figure 5.5 shows that as R is increased the system becomes more stable so it is expected that the resistor does not play a role in stability.

5.2.2 L_q *stability*

The graph for the analysis done in L_q where the system value is changed over the controller constant value L_q' is shown in Figure 5.6. As it happened with the resistor there are also zeros placed at almost the same position of the poles and zeros are omitted in the graph for readability as poles set the stability.

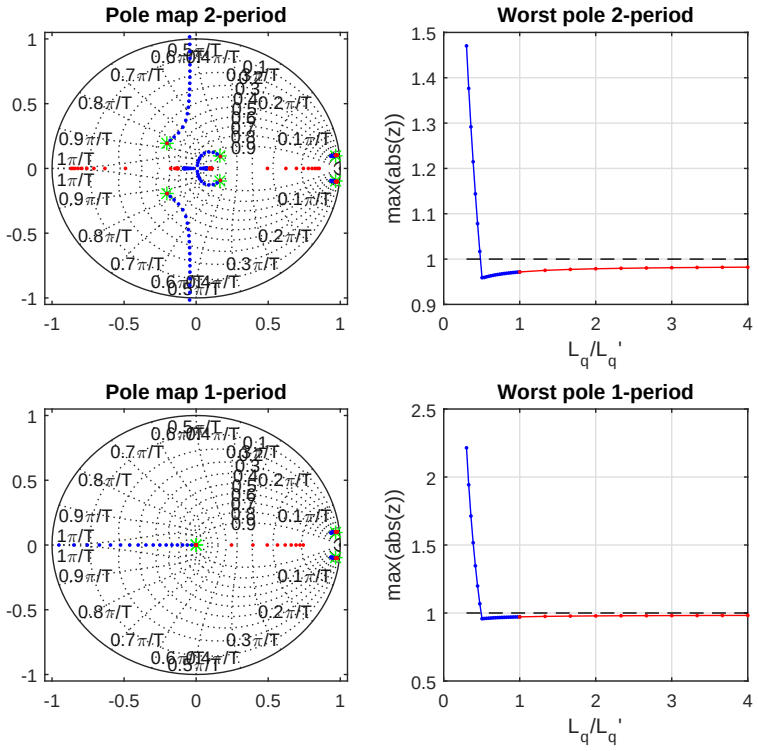


Figure 5.6.: L_q error effect in stability in ZOH model

It can be seen on the graph on the right that the limit of stability is almost the one for when the real inductor drops to half the one used in the controller. This limit makes sense as later in Equation 5.19 is approximated the overshoot as a function of L/L' and it is found that when $L = 0.5L'$ the overshoot is 100% so if the real inductor L drops more than that the overshoot is bigger than 100% so it would tend to amplify any error.

5.2.3 *L_d stability*

The graph for the analysis done in L_d where the system value is changed over the controller constant value L'_d is shown in Figure 5.7.

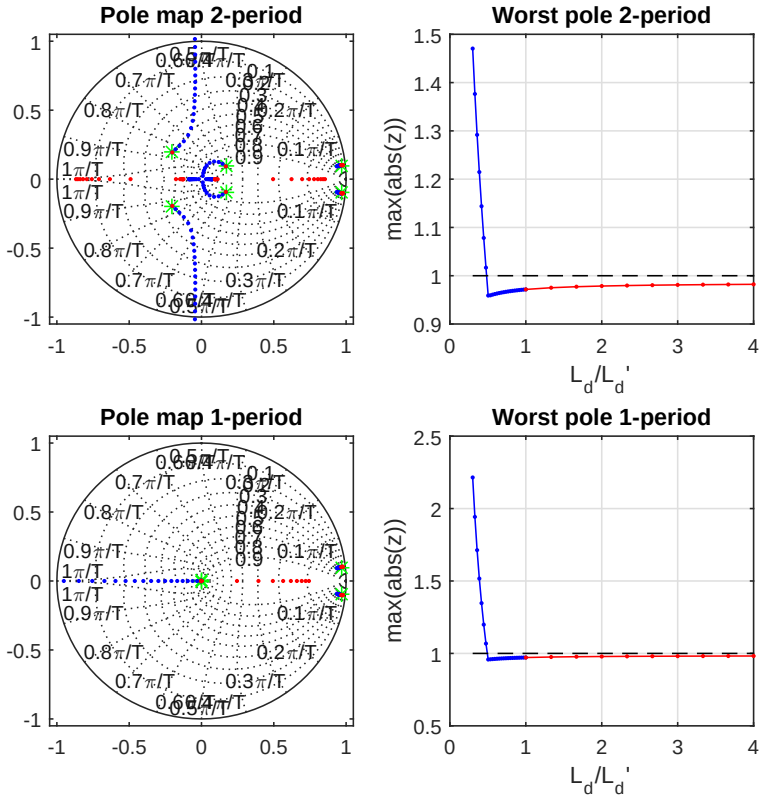


Figure 5.7.: L_d error effect in stability in ZOH model

As expected the results seem to be the same as for L_q. The limit is the same for the 1-period version.

5.2.4 *Flux stability*

The flux has been left out of the equations for simplicity as it can be seen as DC term in the voltage command from a mathematical point of view and therefore it is treated as a perturbation. The flux could be considered by adding the term $\left(\lambda'_{mpm} - \lambda_{mpm}\right)\omega_e$ to the voltage q axes. This is represented in Figure 5.8.

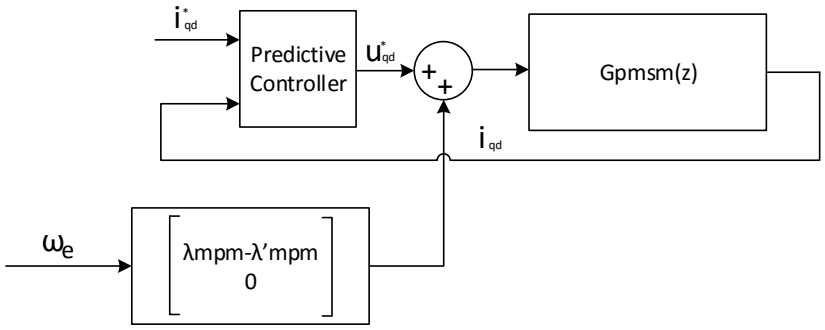


Figure 5.8.: Flux estimation error perturbance

Since the system is linear the response of the system can be seen as the combination of the controller voltage plus the flux difference due to errors in the parameter estimation of the flux. If the controller+system is stable the flux then cannot make the response unstable as it is a constant DC term added to the q voltage.

5.2.5 *Conclusion*

From the results presented it can be seen only inductors play a role in stability and make the system unstable when dropped to half. This value changes slightly with speed and T_s but it is very close to 0.5 always. The resistor on the other hand only makes the system unstable if negative which could be considered a theoretical limit.

5.2.6 *Simulation results*

The limits obtained analytically with the ZOH approximation can be validated in simulation where the PWM at $f_s = 3000Hz$ is no longer neglected. To sum up the results the following was found out:

- A negative resistor makes the system unstable.
- A drop in any inductor by half makes the system unstable.
- The flux does not affect stability if the speed is considered to change slowly compared with the current.

Resistor limits

It has not been possible to simulate a case in which the resistor makes the system unstable with the predictive controller.

Inductor stability

With the 2-period controller the system becomes unstable in simulations when any inductor drops to around 0.47 times its initial value. This value is very close to the one estimated of 0.5.

Flux stability

As predicted it has been impossible to make the system unstable by modifying the flux. Different values from up to ± 10 times have been tried. What has been found is that as long as the speed is not changing since its effect is a DC perturbation it does not affect the transient. If the speed changes considerably it can also affect the transient since it is no longer a constant DC perturbation but since changes are slow it does not play a big role in the transient either.

Conclusion

Based on the simulation results it can be concluded that the only elements that seem to play a role in the stability are the inductors. As long as the speed is not changing the flux error is only a DC offset. Both resistor and flux cannot make the system unstable but they can affect the transient response (the flux only when the speed is changing considerably) and the resistor at low load/speed conditions when it has an important weight in the voltage equation.

5.2.7 *Experimental results*

The limits of stability calculated analytically and tested under simulations can be validated experimentally. It has been found in simulations that the limits estimated analytically coincide with the ones in simulations except for the resistor which seems to not be able to induce instabilities contrary with the results obtained from the root-locus plot. Since the main difference between the analytical model and the simulations is the PWM it is expected for this to be the reason of discrepancies and therefore in the experimental results this is expected to happen:

- Resistor will not play a role in stability.
- A drop in any inductor by half makes the system unstable.
- The flux does not affect stability if the speed is considered to change slowly compared with the current.

To validate all those claims the following tests are proposed. The machine is set running and the parameters used in the controlled are changed to ensure the same proportions found in the limits analytically. Then a step in i_d will be commanded and the response recorded. For a stable system the controller will be able to settle to a value and if the system becomes unstable it will start to oscillate with bigger oscillations each period which will trigger the protections and shut down the system.

5.2.8 *Resistor limits*

Different values have been tested from $-2R$ to $2R$ and the system is always stable. The only change is a steady-state error and at low speeds the transient is also affected.

5.2.9 *Inductor limits*

As the value of L_d used in the controller is increased it is seen that around $4.4mH$ the system starts presenting big oscillations as depicted in Figure 5.9.

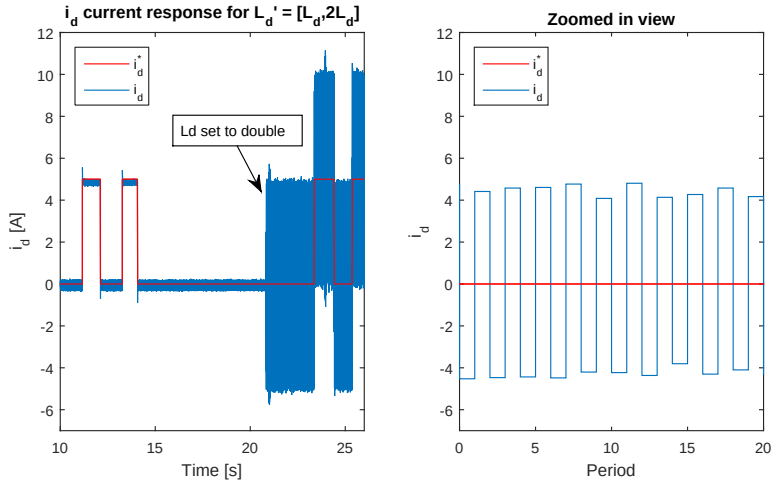


Figure 5.9.: Response on i_d as controller inductor is doubled

Moreover a high pitch noise is heard which makes sense as based in the current response every period the current changes almost $10A$ so a noise of $2f_{sw}$ is expected to be heard.

5.2.10 *Flux limits*

Different values have been tested from $-2\lambda_{mpm}$ to $2\lambda_{mpm}$ and the system stability has not been compromised.

5.3 STEADY-STATE ERRORS

In this section the effect of errors in the parameters will be considered to come up with an idea of how they affect the steady-state errors. The controller and machine equations will be rewritten for the steady-state case by removing the differential terms when appropiate.

5.3.1 *System equations*

In steady-state the current can be considered constant within periods and therefore, on average the differential terms do not appear in the voltage on the system. The current will change due to the PWM nature but the average will still be close to the value at the beggining and end of the period (which will be the same).

Under this considerations the steady-state machine equations can be rewritten in matrix form as Equation 5.12.

$$\begin{bmatrix} u_q \\ u_d \end{bmatrix} = M_{SS} \begin{bmatrix} i_q \\ i_d \end{bmatrix} + \lambda = \begin{bmatrix} R & \omega_e L_d \\ -\omega_e L_q & R \end{bmatrix} \begin{bmatrix} i_q \\ i_d \end{bmatrix} + \omega_e \begin{bmatrix} \lambda_{mpm} \\ 0 \end{bmatrix} \tag{5.12}$$

5.3.2 *Controller equations*

In steady-state the differential terms of the controller will be zero only if there is no steady-state error. However, if there is some steady-state error this difference will still be fed to the differential terms and therefore they cannot be removed. There will be two variables in the controller side, the reference currents and the actual value.

Recalling from before K_1 was defined in Equation 5.10 and K_2 in Equation 5.11 for the 1-period controller and the controller equations are then given by Equation 5.13 where the last term will be denoted simply as λ'.

$$\begin{bmatrix} u_q \\ u_d \end{bmatrix}' = K_1 \begin{bmatrix} i_q \\ i_d \end{bmatrix}^* + K_2 \begin{bmatrix} i_q \\ i_d \end{bmatrix} + \omega_e \begin{bmatrix} \lambda'_{mpm} \\ 0 \end{bmatrix} \tag{5.13}$$

In this case the results change between 1-period and 2-period version due to the different gains used. The derivation for the 2-period version is the same but noting the following changes. In K_1 and K_2, the term T_s should be changed into $2T_s$. On the other hand in K_1 the term $1/2$ is changed into $2/3$ and the term $1/2$ into $1/3$ in K_2. This is because the predictive controller estimates the current reference in the average terms, which without saturation will be the same as the reference. In steady-state the voltage command will also be the same in every period so $\frac{u_k + u_{k+1}}{2} = u = f(i^*_{qd}, i_{qd}, \omega_e, 2T_s)$.

5.3.3 *Equations of steady-state errors*

With the models defined in the two previous subsections the steady-state errors can be calculated by solving the currents as a function of the reference and the machine parameters and working condition by imposing that the voltage applied in controller and machine must be the same. The solution for i_{qd} is then found in Equation 5.14.

$$i'_{qd} = (M_{SS} - K_2)^{-1} \left(K_1 i^*_{qd} + \lambda' - \lambda \right) \tag{5.14}$$

And the error then can be defined as $\epsilon_{qd} = i'_{qd} - i^*_{qd}$. As there is a matrix inversion with all the elements defined it can be inferred that the analytical solution will not be simple. Therefore to interpret the results the formulas derived will be used to plot the errors in different conditions.

5.3.4 *Results*

The effect of changing individually any parameter is shown in Figure 5.10 for a reference of $i_q^* = 15$ and $i_d^* = 3$ as setting $i_d^* = 0$ removes any error due to L_d. The graphs are done also for 4 different speeds of $0, 33, 67, 100Hz$ with $T_s = 3000Hz$ and the result plotted is the absolute error for the q and d current given by $i_{qd} - i_{qd}^*$.

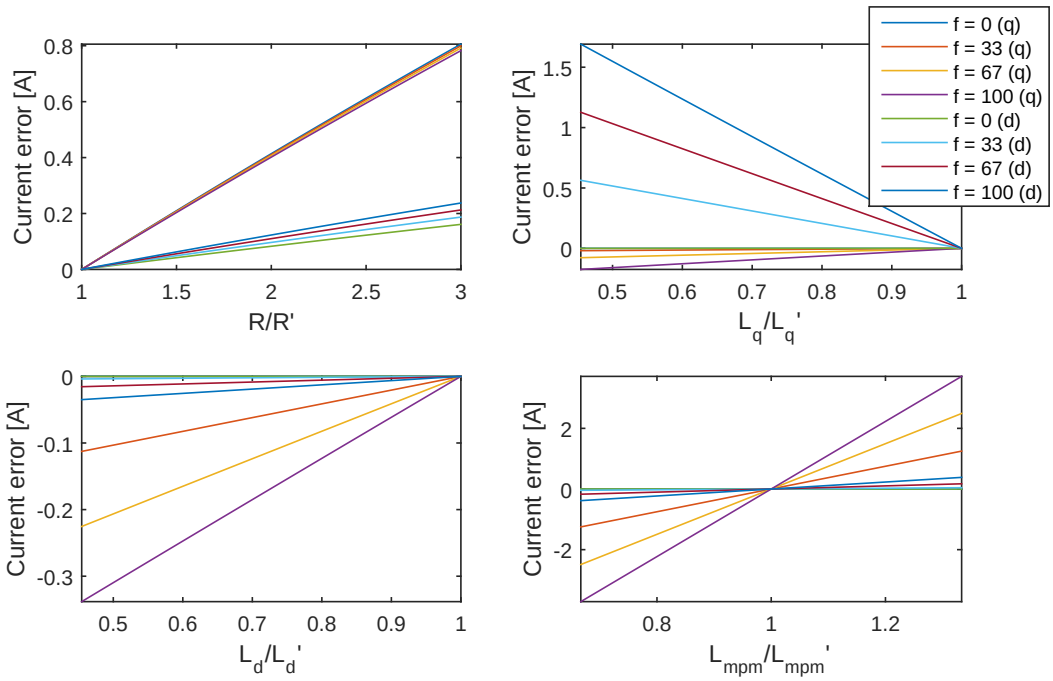


Figure 5.10.: Steady-state error due to parameter error one parameter at a time

As it seems logical the absolute error is bigger in the q axes as the reference is set 5 times bigger. It can be seen that L_d does not play a big role as long as i_d is kept low. Theoretically if $i_d^* = 0$ errors in L_d do not affect the steady-state errors. However the transient phenomena will be affected and as was proved before if L_d dropped more than half the system would become unstable.

5.3.5 *Steady-state errors approximation*

It can be seen in Figure 5.10 that even though the equation solutions are not linear due to the matrix inversion the error produced when only one parameter is changed at a time looks linear. This suggests that to characterize the effect of any combination of parameters at a time and working condition (load, speed, more than one parameter change) the equations obtained may be approximated with Taylor expansion in the 4 machine parameters and this way they will be easier to interpret as there are 8 parameters into play and a plot with every combination is not possible.

Derivations are omitted here and attached in Section B.2 where it is also shown in Figure B.3 that the approximation tends to produce an error in the error of less than

5% so it can be considered adequate and used to infer the relations between states and parameters in the global error. The result of the approximation is that the error can be estimated as Equation 5.15 where K_{ss} is given by Equation 5.16.

$$e(i_{qd}) = i_{qd} - i^*_{qd} \approx K_{ss} \begin{bmatrix} \Delta R \\ \Delta L_q \\ \Delta L_d \\ \Delta \lambda_{mpm} \end{bmatrix} \tag{5.15}$$

$$K_{ss} = \begin{bmatrix} -\frac{T_s}{L_q} i^*_q & -\frac{{T_s}^2 i^*_q {\omega_e}^2}{2 L_q} & -\frac{T_s i^*_d \omega_e}{L_q} & -\frac{T_s \omega_e}{L_q} \\ -\frac{T_s}{L_d} i^*_d & \frac{T_s i^*_q \omega_e}{L_d} & -\frac{{T_s}^2 i^*_d {\omega_e}^2}{2 L_d} & -\frac{{T_s}^2 {\omega_e}^2}{2 L_d} \end{bmatrix} \tag{5.16}$$

Difference between 1 and 2-period controller consist in the constants that multiply the coefficients of the Taylor approximation (for example $1/2$ may become $2/3$ and so on) so the relations observed inferred from Taylor (what variables affect what) are the same. From the Taylor approximation the effect of each parameter and state in the global error can be understood in simple terms and will be discussed in the following sections.

Parameter error analysis

Since the full taylor approximation of first order has been validated to produce an error moderately low of typically less than 5% it may be considered valid to use as a way to study the effect of each parameter variation in the steady-state error. Moreover if the speed is kept below half the nominal Equation B.16 can be used. At higher speeds then it may be modified by multiplying it by $\frac{d_2}{d_2+d_5}$. This comes from the approximation taken in Equation B.13.

The advantage of this approximation is that it shows that the effect of each parameter to the error in each axis is independent of each other and is only function of two parameters only: the speed, and the current reference in one axis only. Because of this a plot for each parameter can be developed and the total error can be approximated as the sum of each parameter contribution to the error in each plot. From the taylor approximation of which formulas were shown in Equation 5.16 the following dependencies in Figure 5.11 between errors and system states can be inferred with a good approximation. The diagram can be interpreted as follows. An error in L_q for example produces an error in both the q and d current, $e(i_q)$ and $e(i_d)$ as a function of i^*_q. The reference i^*_d does not influence the error significantly. The errors are speed-dependent (yellow). The resistor for example produces an error on each axis as function of both i^*_q and i^*_d, being i^*_q the command that affects the error in q and i^*_d the error in d. Because of the green color the speed does not play a big role. Finally the flux is current independent and produces a constant error that is greatly influenced by the speed (red). The approximated errors can then be calculated applying superposition to Figure 5.12 or Figure 5.13 for the system parameters obtained.

If the switching frequency also wants to be taken into account then the speed used to look in the plot may be modified into $\omega_e T'_s / T_s$ as the speed is always multiplied by T_s in the taylor approximation except the resistor where the current should be modified as $i T'_s / T_s$. Results are depicted in Figure 5.12 for the 1-period deadbeat version using the relations inferred from the Taylor approximation but the exact error as given by

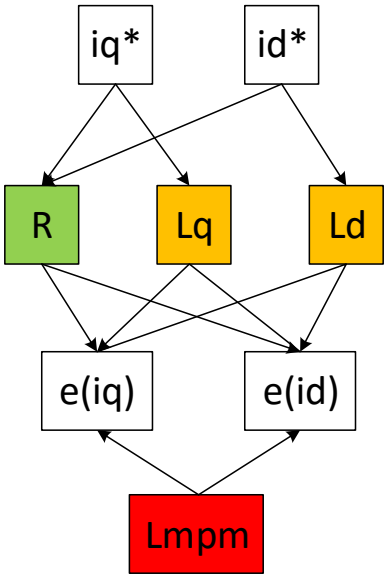


Figure 5.11.: Error dependencies between system parameters and states

Equation 5.14. The same calculations can be redone for the 2-period version and the results are depicted in Figure 5.13 since the errors change considerably.

The graphs can be interpreted as follows. The error in the y-axis is the absolute error produced either in the q (blue) or d axis (red) current due to a change of $+10\%$ in the parameter. The error in this case was defined as $i_{qd} - i^*_{qd}$ so for example a positive value means the current generated by the controller will be bigger than the reference. Since the formulas are approximately linear to see the effect of any other change the proportion which respect to the initial parameter may be multiplied. For example to see the effect of a 20% drop in L_q the value obtained from the graph may be multiplied by $-0.2/0.1$ as the values are done for $+10\%$ change.

The error is plotted as a function of current reference and speed as those are the only two factors that affect it within each parameter without making a significant error. The change due to current is linear and due to speed is linear in some parameters only. The speed is varied from 0 to rated speed. The brighter the color the bigger the speed which goes from 0 to rated $2\pi 300$.

To validate the results the parameters in the machine may be changed under simulations and the error for a given state estimated with Figure 5.13 since the 2-period controller is implemented in simulations. Let for example $R = 1.3R_0, L_q = 0.8L_{q0}, L_d = 0.75L_{d0}, L_{mpm} = 0.9L_{mpm0}$ and the states $i^*_q = 10, i^*_d = 6, \omega_e = 2\pi 100$ with $f_s = 5$ Khz. The error can be estimated from Figure 5.13 and the procedure is depicted in Table 5.1. The values can be read directly from the graph and multiplied by $0.3/0.1, -0.2/0.1, -0.25/0.1, -0.1/0.1$ for each parameter as the graph is made for changes of $+10\%$. To gain precision the values are actually calculated from Equation 5.14 but maintaining the approximations taken. For example in L_q the error due to

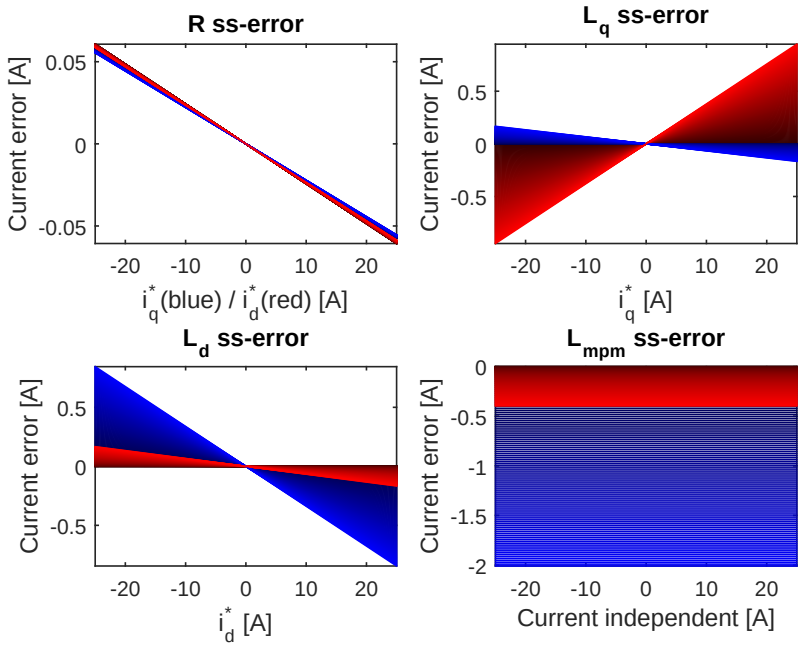


Figure 5.12.: Error in i_{qd} due to each parameter, i^*_{qd}, and speed for 1-period deadbeat controller for $f_s = 5$ kHz

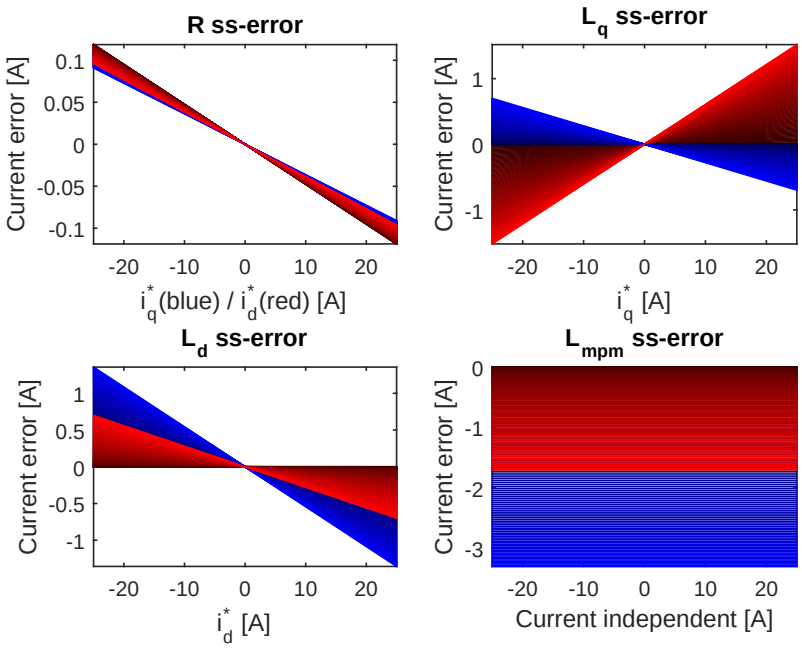


Figure 5.13.: Error in i_{qd} due to each parameter, i^*_{qd}, and speed for 2-period deadbeat controller for $f_s = 5$ kHz

i_d^* is removed and it will also calculate the error for a 10% positive change in L_q and then multiply the result by $-0.2/0.1$.

Parameter	$e(i_q)$	$e(i_d)$
R	-0.13	-0.08
L_q	0.08	-0.50
L_d	0.34	0.06
λ_{mpm}	1.35	0.23
Total error	1.60	-0.30

Table 5.1.: Random parameters variation for Taylor approximation validation

The errors have been estimated with the approximations presented. Now the machine is set at the desired speed and the inertia is set high so the speed does not change during the simulation. The references are set as commented and the parameters in the machine modified according to the numbers set. The current response is shown in Figure 5.14. The error may be calculated then from Figure 5.14 as $e(i_q) = 11.6 - 10 = 1.60$ and $e(i_d) = 5.6 - 6 = -0.40$. It can be seen good concordance between the approximation $(-1.60, 0.30)$ and the simulation. If the formula in Equation 5.14 is used directly instead the error would be estimated as 1.58 and -0.40 showing that Equation 5.14 is valid and the origin of the errors are attributed to the approximations taken but it can be seen that they are still kept low and can be used to infer good intuitions about how each parameter and system state affect the steady-state error.

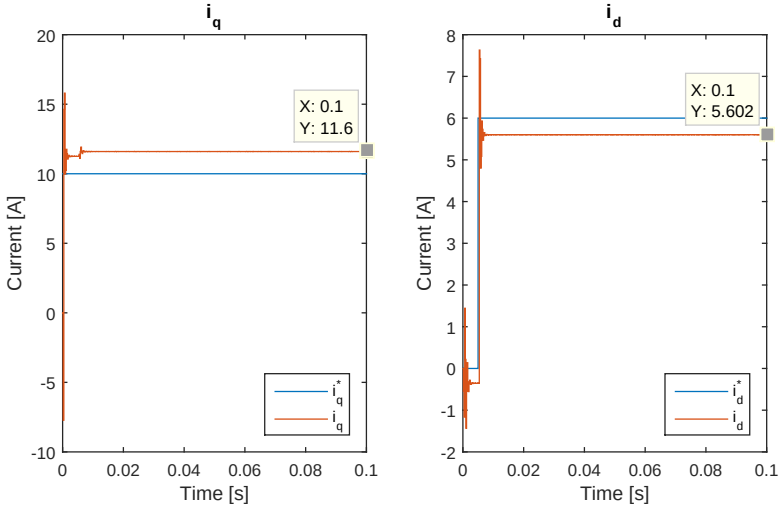


Figure 5.14.: Error in i_{qd} with the states set in simulation

Conclusion

The effect of each parameter in the steady-state error can be calculated accurately with Equation 5.14. Due to the dependency of so many parameters an approximation have been taken with relatively low errors and the dependency between parameters is shown

in Figure 5.11. The errors can be estimated with good precission applying superposition with the graphs from Figure 5.12 - 5.13 which show how each parameter and state affect the global steady-state error. Both current and speed can change the steady-state errors dramatically and the worst case is always with the biggest of the two. The error in the resistor estimate plays a small role in the steady-state error when compared with the other three parameters. If $i_d^* = 0$ then any error in L_d will not produce an error in any current. Switching frequency also plays an important role similar to speed in the steady-state errors since the effect of a different frequency f_s' is equivalent as the error running the machine at $\omega_e' \approx \omega_e \frac{f_s}{f_s'}$ in all parameters but the resistor which on the other hand tends to produce small errors.

5.4 TRANSIENT RESPONSE

The effect of the change in any machine parameter in the performance of the controller is studied in this section.

The resistor effect in the transient is neglected as it only affects at lower loads and speeds since its voltage contribution is usually low compared to the inductors and flux parts. On the other hand from Figure 5.8 it can be seen that errors in the flux linkage can be understood as a DC perturbation to the voltage command since changes in the speed are slow compared to changes in the current. This does not necessarily means that they will not affect the transient as the controller will work differently when trying to compensate them (the differential terms for example will never be zero as there will always be an error under changes in the flux) it was seen under simulations and experiments that the effect to the transient when it is wrongly estimated is usually small.

For this reason the transient response will only take into account the effect of changes in the inductors as they are the main contributors to it. Later the DC claim will be validated under some approximations.

5.4.1 *Overshoot*

If the resistor and flux terms are neglected in the transient they can be removed from the system and controller equations as they are considered compensated. Then, the following relations can be derived in Equation 5.17 - 5.18 to estimate what happens when a step is sent to the controller in the 1-period version.

$$\frac{L_q}{T_s}\Delta q + \omega_e L_d < i_d >= \frac{L_q'}{T_s}\Delta q^* + \omega_e L_d' < i_d >^* \tag{5.17}$$

$$\frac{L_d}{T_s}\Delta d - \omega_e L_q < i_q >= \frac{L_d'}{T_s}\Delta d^* - \omega_e L_q' < i_q >^* \tag{5.18}$$

The inductor L_x (where x is any axis, q or d) is the real inductor in the machine but the controller has the estimate L_x'. The value of $< i_x >^* = \frac{i_x^* + i_x}{2}$ is the average current in the period as estimated by the controller where i_x^* is the current command and i_x the current at the beginning of the period. The real one will be $< i_x >= \frac{i_x' + i_x}{2}$ where i_x' is the actual current at the end of the period. The value of Δi_x^* is the commanded change given

by $i_x^* - i_x$ that can be understood as the step in the x axis. The actual step however will be $\Delta i_x = i_x' - i_x$.

At low speeds the coupling terms are not significant either in comparison with the differential part and may also be neglected and since then the steady-state error will be small as is the speed then the overshoot can be approximated as Equation 5.19. Under those approximations the overshoot in the 2-period version must be the same also as T_s is removed from the equations. Negative values indicate the system is over-damped. This approximation is at first sight only valid when steady-state errors are small but gives an idea about the evolution of the overshoot. When the coupling terms are considered they will modify the steady-state error and also the transient response making the overshoot speed dependent also. Moreover since the equations are coupled errors in L_q will affect the overshoot in L_d and the other way around. However it has been seen that as long as the errors are constant (for example flux errors, or the current reference is set constant in the other axis and does barely change so introduces a constant error) then the equation Equation 5.19 seems to be quite accurate understanding Δx^* as $y_{ss} - y_0$ and Δx as $y_p - y_0$ based on simulations. Those variables are shown in Figure 5.16. A simple proof of why Equation 5.19 works for cases with constant steady-state errors and dc perturbations is conducted in Section B.3.

$$ov = \Phi \approx \frac{\Delta x}{\Delta x^*} - 1 = \frac{L_x'}{L_x} - 1 \tag{5.19}$$

To validate the claims as only 2 parameters are the main responsible of the overshoot (inductor error and speed) then the following procedure can be done. A model in PLECS can be developed where those parameters are varied and a step in the q axis is given and the overshoot calculated from a script. It has been seen that the overshoot is not affected in great value by errors in the other parameters, current references or previous current states in d, and step size in q based on simulations as they produce dc perturbations. Same happens for d. As predicted the overshoot is speed dependent but the relation is not that important either. The speed ω_e and the coefficient between the real inductor and controller, L_q/L_q', may be varied in the model run from a Matlab script trough two loops. The results are depicted in Figure 5.15. Additionally, in red the approximation from Equation 5.19 has been plotted.

The speeds used have been $2\pi[0, 75, 150, 225, 300]$ where $2\pi 300$ is the rated speed. It can be seen that the speed does not play such a significant role in the overshoot. The errors at high speeds when the parameters are correct are due to the coupling effects influencing the results due to the discretization of the controller and having a continuous plant. An step in one axis changes the other current slightly which is not predicted by the controller. This is due to the average approximation used which is worse as the speed increases. Increasing f_s removes any coupling effect. Therefore this current that appears in the other axis that is not accounted for modifies slightly the initial step making a small overshoot.

It can also be seen that the approximation from Equation 5.19 is quite accurate and as expected tends to be better as the error in the inductor is increased and the speed is decreased as that increases the weight of the differential term in the overshoot contribution.

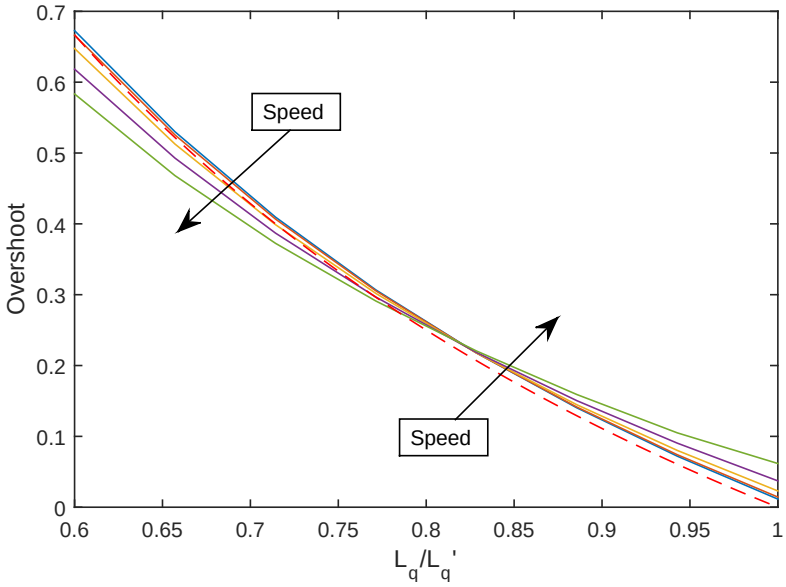


Figure 5.15.: Overshoot in i_q as a function of inductor error and speed

5.4.2 *Settling time*

The settling time defined as the time it takes for a signal to stabilize within a tolerance band can be estimated under some approximations in the 1-period controller. The overshoot, Φ, is mostly function of the speed ω_e and more importantly the relation L/L'. Therefore, since the 1-period deadbeat controller has no delays there is no reason to think the proportion Φ will change in any period after the first step. Then it must be true that the response of the system must follow with good approximation Figure 5.16.

When a step is given the signal will go to $(1+\Phi)s$ where s is the real step of the signal as given by $s = y_{ss} - y_0$ and Φ the real overshoot as $\frac{y_p - y_{ss}}{y_{ss} - y_0}$. This means the error with respect to the steady-state value is given by $(1+\Phi)s - 1 = \Phi s$. In the next period the controller will try to compensate and the signal will go below the steady-state value a proportion lower than the original overshoot. In absolute value since the step now is Φs then the signal will go to $(1+\Phi)(\Phi s)$ so the signal will have now an error with respect to Φs as given by $(1+\Phi)(\Phi s) - \Phi s = \Phi^2 s$. This may be generalized for any period k as $\Phi^n \times (-1)^{(2k+2)} s$. The term -1^{2k+2} may be removed as it only gives information about the side of the error which is not needed for the settling time. On the other hand it seems sensible to use settling time in an relative sense, therefore the value may be scaled by the original step size s. The final formula is then simply Φ^n. The settling time then may be calculated as the time it takes to reach a relative tolerance of p. The number of periods it takes to settle, n_{ss}, may then be solved from $\Phi^{n_{ss}} = p$ and it is given by Equation 5.20. The value of Φ may be estimated from Equation 5.19 or taken from Figure 5.15. If the real inductor increases and there is an over damping response then Φ may be replaced

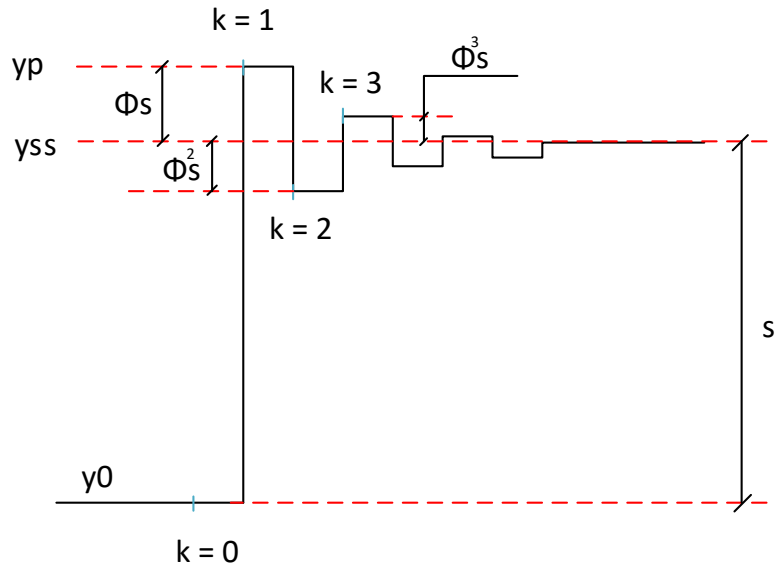


Figure 5.16.: Settling time with approximation taken for 1-period controller

by $-\Phi$ in Equation 5.20 as the same behavior as with overshoot is expected. Therefore to generalize Φ may be replaced by $abs(\Phi)$ in Equation 5.20 to account for both cases.

$$n_{ss}(1) = 1 + \frac{\log p}{\log \Phi} \tag{5.20}$$

As a way to validate the simplifications taken a step response in the q axis at $\omega_e = 2\pi 50$ and with $L_q/L'_q = 0.7$ is performed and the results are depicted in Figure 5.17. It can be seen great concordance with the approximation from Figure 5.16.

For the 2-period controller since there is a delay present the approximation of constant overshoot and Φ^n is not valid anymore as the previous state influences the next overshoot.

The response to a step in q in both axis for $L_q/L'_q = 0.7$ and two speeds, $\omega_e = 2\pi 20$ and $\omega_e = 2\pi 200$ is depicted in Figure 5.18. The error in the inductor yields an overshoot following the same derivations as in 1-period.

It can be seen from Figure 5.18 at low speeds the oscillations tend to be at $2T_s$ while at higher speeds they are faster at T_s. The settling time takes also more duration that in the 1-period version due to the delay in the command. As a rule of thumb from simulation results the algorithm seems twice as slow as the 1-period version, so the settling time may be approximated as $n_{ss}(2 - period) \approx 2n_{ss}(1 - period)$ where $n_{ss}(1 - period)$ was approximated in Equation 5.20. So the settling time for the 2-period version may be approximated as Equation 5.21.

$$n_{ss}(2) = 2\left(1 + \frac{\log p}{\log abs(\Phi)}\right) \tag{5.21}$$

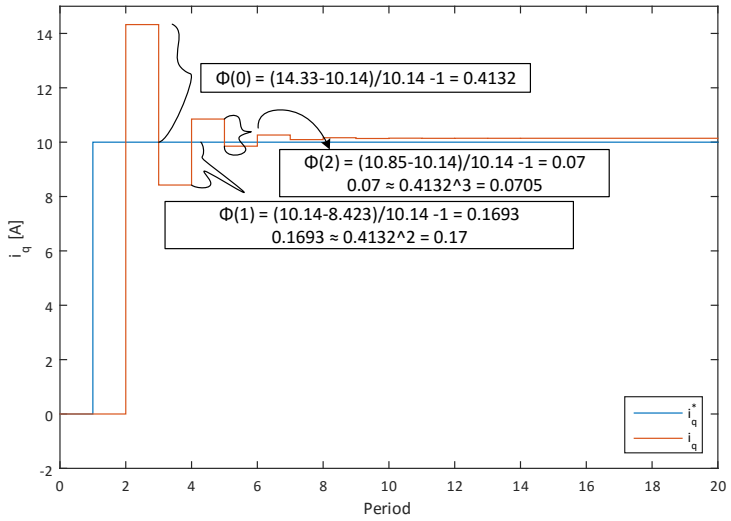


Figure 5.17.: Settling time with approximation taken for 1-period controller for $L_q/L'_q = 0.7$ and $\omega_e = 2\pi 50$

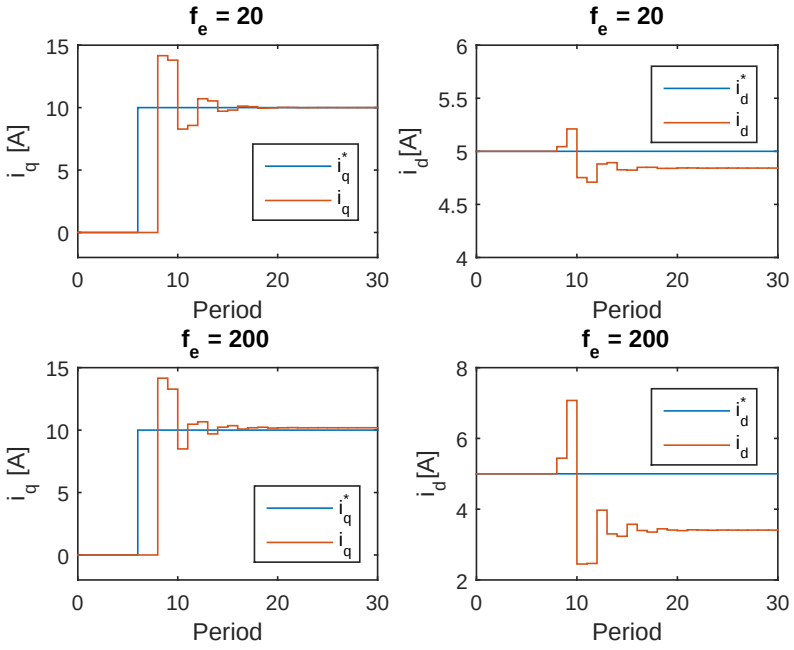


Figure 5.18.: Step response for 2-period controller for $L_q/L'_q = 0.7$ and $\omega_e = 2\pi 20$ and $\omega_e = 2\pi 200$

Finally the settling time may be calculated from simulations with two nested for loops varying L_q/L'_q and ω_e. The model is set running for different combinations and the settling time it is estimated as the time it takes for the signal to settle within 5% of the final value. The results are depicted in Figure 5.19 for the 1-period controller and in Figure 5.20 for the 2-period version. In dashed red the approximation from Equation 5.20 and Equation 5.21 is shown. It can be seen good concordance between the approximation and the simulation results.

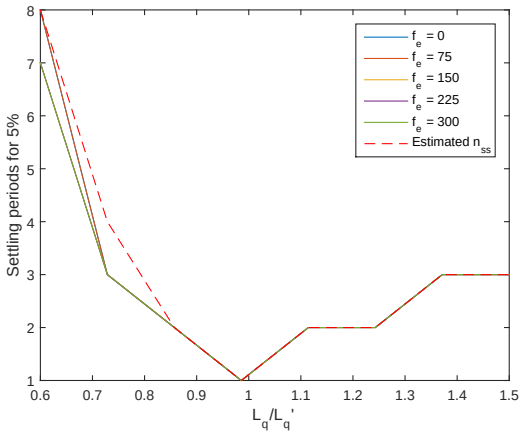


Figure 5.19.: Settling time (5%) for different L_q/L'_q and ω_e at $f_s = 5$ kHz for 1-period controller

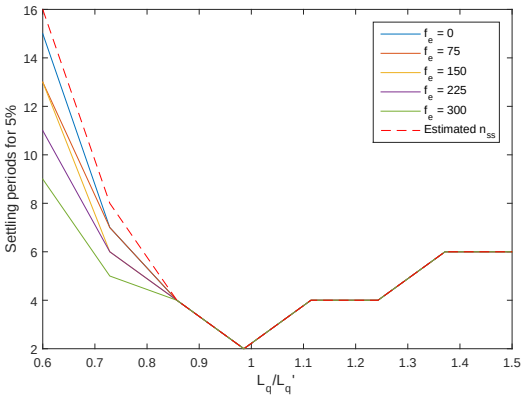


Figure 5.20.: Settling time (5%) for different L_q/L'_q and ω_e at $f_s = 5$ kHz for 2-period controller

5.4.3 *Conclusion*

The transient response evolution due to parameter errors in the estimates has been studied. The resistor effect is neglected due to the typically small contribution to the voltage equations. It has been found that with good approximation the overshoot and

settling time is mostly function of only the inductor error as DC perturbations do not influence the overshoot greatly and settling time is dependent of overshoot mostly.

The speed also influences the overshoot and settling time and helps the transient response in certain conditions as evidenced by the graphs. Overshoot and settling time are asymmetrical with respect to the inductor error as can be seen by the approximation in the formulas derived in and also in the simulation results presented. Increases in the real inductors make the system slower and more robust and decreases also produce slower responses with oscillations. Both the settling time and overshoot/overdamping is more affected when the inductors drop. For example a drop of the inductor by half produces an overshoot of 100% and a marginally stable system but an increase of the inductor to double produces an initial error in the over-damping system of 50% and a settling time around 5 times slower than having no error.

6 NON-LINEARITIES COMPENSATION

In this chapter different non-linearities that affect the converter are presented and compensated as they affect the voltage command given by the controller and decrease performance and can make machine parameter estimations difficult at certain conditions. Each non-linearity is studied independently and at the end simulation and experimental results are presented. Apart from the main discrete method a resonant compensator implementation is also analyzed with experimental validation which provide good results at low speeds.

6.1 DEADTIME

The converter IGBTs need some time to commute from one state to another. This time has some variations depending of the temperature and other factors. Because of this another time t_d bigger than those two is added to guaranty enough time for the current to change paths. Each time the PWM pulse change it is delayed by t_d as explained in [9].

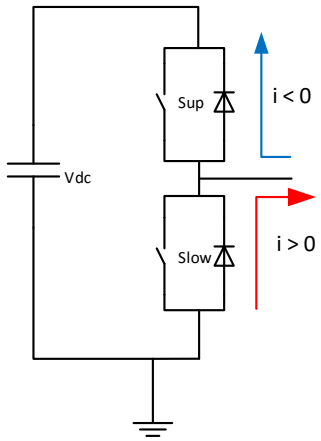


Figure 6.1.: Converter leg with the current paths

On this period of transition no pulses are given and the effect in the voltage change depends on the current sign as can be seen in the circuit in Figure 6.1 for one leg for one period of the triangular. If no pulses are given depending of the current sign it will flow one way or another inducing either v_{dc} or 0. The effect in the other legs is the same and independent.

The effect on the signals is shown in Figure 6.2. When the current is positive and no pulses are given the lower part will conduct as the current can flow trough the

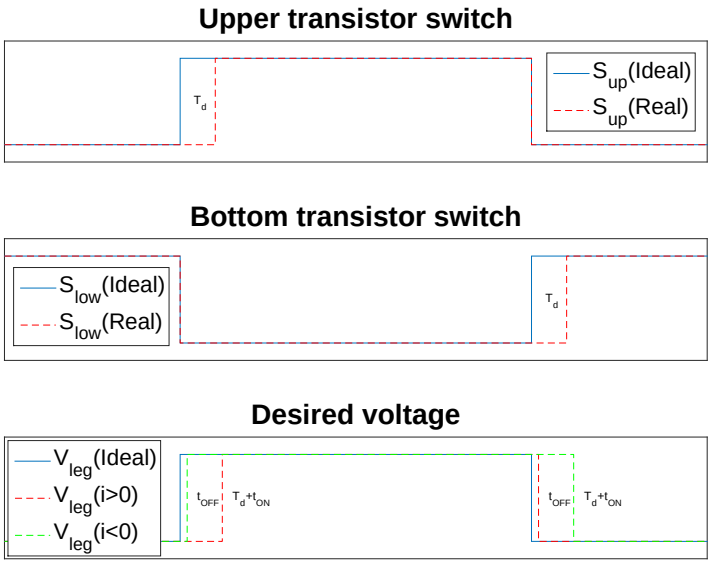


Figure 6.2.: Deadtime effect in the voltage command

bottom diode. After the delayed t_d and t_{ON} it will switch to the igbt that is commanded. Therefore there will be a voltage drop in the transition from down to top igbt. On the contrary, when it switches from the lower igbt it will do so almost instantly at the time t_{OFF} which induces a small voltage gain. When the current is negative something similar happen. On the switch from the top part to the bottom it will be delayed by t_d and t_{ON} as the current can travel freely trough the top diode and the delay will be t_{OFF} only when it goes back to the positive state and thus induces a voltage gain.

Moreover the voltage drop in the components of the converter will produce more changes in the voltage command. All those modifications can be predicted and compensated.

During the development of the compensator several simplifications will be done in order to be able to develop a compromise with simple analytical solutions and good performance. The results of the simulations will show that the simplifications are adequate and the results of the experiments will evidence that the modeled non-linearities have been done correctly.

To compensate for this time delay t_d one can calculate the voltage difference that is produced and add it directly to the reference. This can be done due to the linearity of the system and the independence of the time delays of the input as will be explained. If the voltage drop given by the time delays was a function also of the reference then the voltage drop could not be directly added to the reference and more calculations may be done to get a better result.

On a fundamental level, the converter voltage (which respect to its neutral placed in the negative side of u_{dc}) can be written as Equation 6.1 where the duty times can be rewritten as Equation 6.2, that is the reference duty time d^* coming from the modulator (which

receives the original voltage command from the controller) followed by the undesired dead time duty cycle d_d.

$$\begin{bmatrix} u_{an} \\ u_{bn} \\ u_{cn} \end{bmatrix} = u_{dc} \begin{bmatrix} d_a \\ d_b \\ d_c \end{bmatrix} \tag{6.1}$$

$$d_i = d_i^* + d_d \tag{6.2}$$

Because d_d is only a function of the current direction as explained before it is independent of the reference d^* and therefore if d^* is transformed in $d^* - d_d$ the final result will not have the deadtime. So to compensate the deadtime this d_d can be calculated at any moment and added directly to the duty cycle that is given to the PWM hardware. The value of d_i is on average $(-t_{ON} - t_d + t_{OFF})/T_s$ when $i_i > 0$ and $(t_{ON} + t_d - t_{OFF})/T_s$ when $i_i < 0$. The values of t_{ON} and t_{OFF} will be neglected.

To see the effect of the deadtime in dq as the control is done in this reference frame the duty cycles can be transformed into Line-Line voltages in the converter and this to line-neutral machine voltages using Equation 2.6. For a balanced system, for a fundamental period, the injected voltage v_d is calculated and shown in Figure 6.3 for the values of Table 6.1. For reference the distortion in phase-a voltage is also presented.

Name	Value
$u_{dc}(V)$	325
$f_{sw}(Hz)$	5000
$t_d(\mu s)$	4

Table 6.1.: Parameters for the deadtime voltages

This voltage can be transformed to $dq0$ and is shown on the right. It can be seen that the effect in q is a DC offset and a sinusoidal and in d there is not such offset but rather a triangular shape.

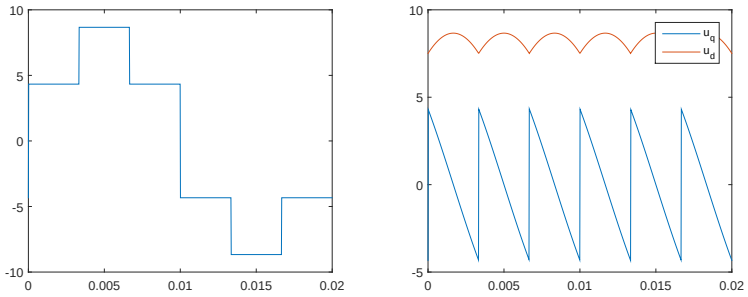


Figure 6.3.: Voltage injected due to the deadtime in abc for one phase and dq respectively

If this voltage distorsion is then injected as duty cycles to the PWM with reverse sign the deadtime can be considered to be compensated. Therefore the voltage of the controller is still being used directly in the equations to predict parameters. The flowchart describing the algorithm for one leg is shown in Figure 6.4.

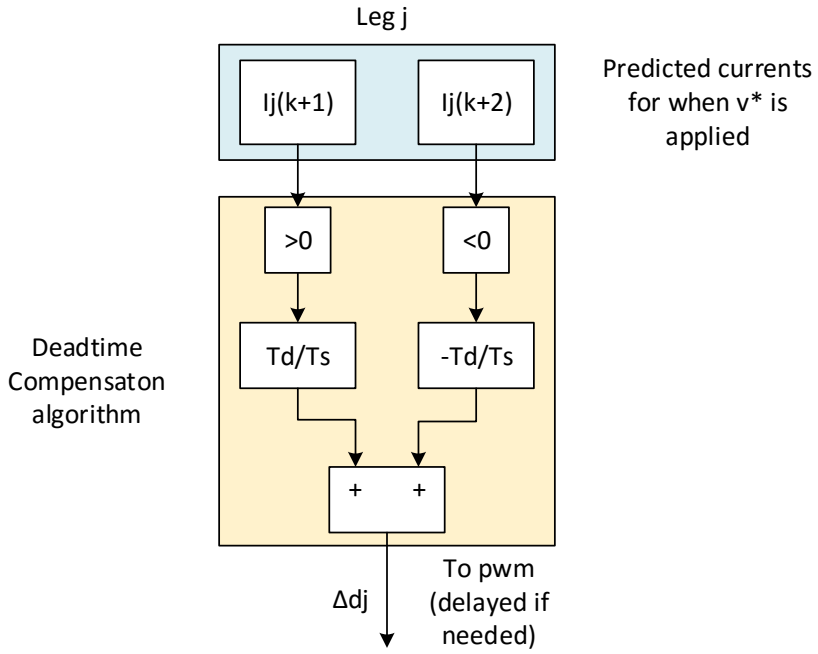


Figure 6.4.: Steps performed in one leg to compensate deadtime

6.2 SNUBBER COMPENSATION

Some converters have snubbers in parallel with the diodes to try to minimize the effect of the dead-time in the voltage change. As explained in [10] the snubbers may be modeled as just capacitors. Under this assumption a compensator that has into consideration the change in the voltage due to these capacitors can be added to the dead-time.

The effect in the signals is shown in Figure 6.5. When the current is positive and needs to change from the top to the bottom part it is delayed t_d and on this case due to the snubbers it will have two paths to follow trough both capacitors. The capacitor on the top is discharged as the top part was conducting and the voltage on it is the same as in the transistor that can be neglected. On the bottom, on the other hand the capacitor is charged to V_{dc} as when the top transistor was on it was directly connected to the source. In this transition, half the current will go trough the top capacitor and half trough the bottom as both capacitors must discharge at the same rate and are considered to have the same capacitance. Therefore it can be seen that the voltage in the terminal will not be zero instantly but will have a slope due to the voltage in the capacitors. This will produce a voltage gain that can be calculated.

On the other hand, when the current is positive and switches back to the top part, it is delayed t_d but during this part it can continue to flow freely trough the bottom diode as that capacitor is not charged. Therefore, there will be the same voltage drop as there was with the dead-time described before. The same can be done for the case when the current is negative. There will be a slope that produces a voltage drop in the transition

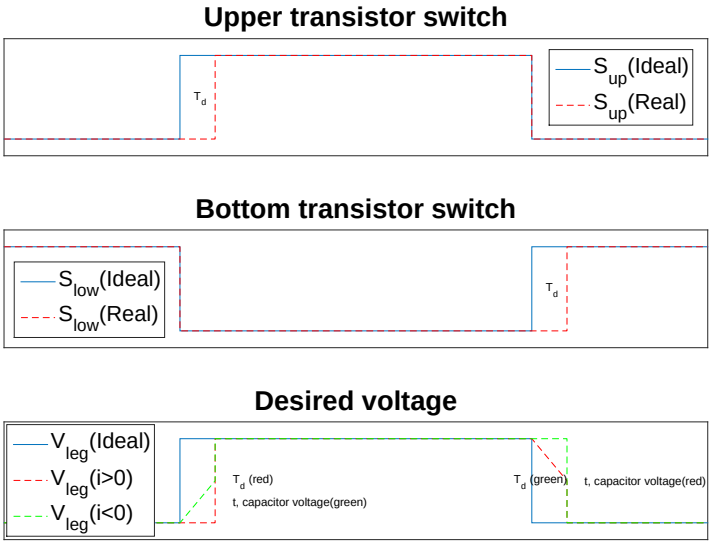


Figure 6.5.: Deadtime effect in the voltage command

to the bottom to the top part and the same voltage gain as in the previous dead-time in the transition to the bottom part again.

With this considerations an extra voltage change due to the capacitors can be calculated. Depending of the slope, the voltage may go to zero before or at t_d or after. There are therefore two equations depending on the case. The time it takes to go to zero can be calculated as shown in Equation 6.3 as the absolute slope is the same independent of the current sign and state.

$$t = \frac{C}{|i|} V_{dc} \tag{6.3}$$

This time t can be compared with t_d. When it is lower, it means the voltage will drop to zero before t_d and therefore the formula used to calculate the average voltage is given by Equation 6.4.

$$v = \frac{1}{2} V_{dc} t \tag{6.4}$$

It will depend of the sign of the current. When it is positive it induces a voltage gain and when it is negative a drop so this voltage can be multiplied by the negative value of the sign of the current to compensate for it correctly. In the case that t is bigger than t_d then the voltage will not drop to zero and the formula used to calculate the average is given by Equation 6.5 where v_2 is the voltage at the end of t_d and can be calculated as shown in Equation 6.6 independent of the current sign.

$$v_{ch} = \frac{1}{2} \left(V_{dc} + v_2 \right) t_d \tag{6.5}$$

$$v_2 = V_{dc} - |i|\frac{t_d}{C} \tag{6.6}$$

To generalize for any current sign, this expression can again be multiplied by the negative value of the sign of the current. All this considerations can be taken into account by creating a function that calculates this voltage as duty cycles that can be added directly to PWM. The diagram of the implementation is depicted in Figure 6.6.

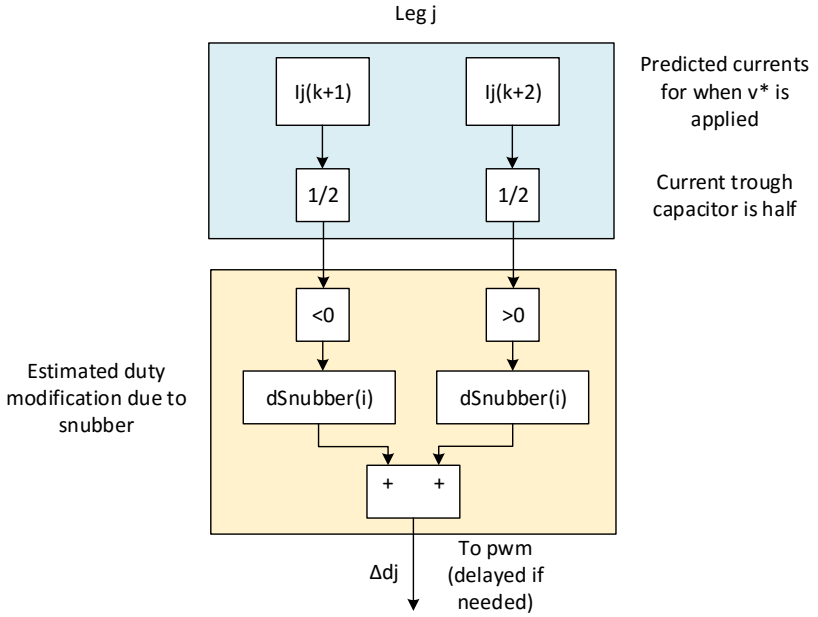


Figure 6.6.: Snubber voltage drop compensation for one leg

The implementation of $dSnubber(i)$ is shown in Figure 6.7 where the functions used are the ones mentioned in Equation 6.4-6.6.

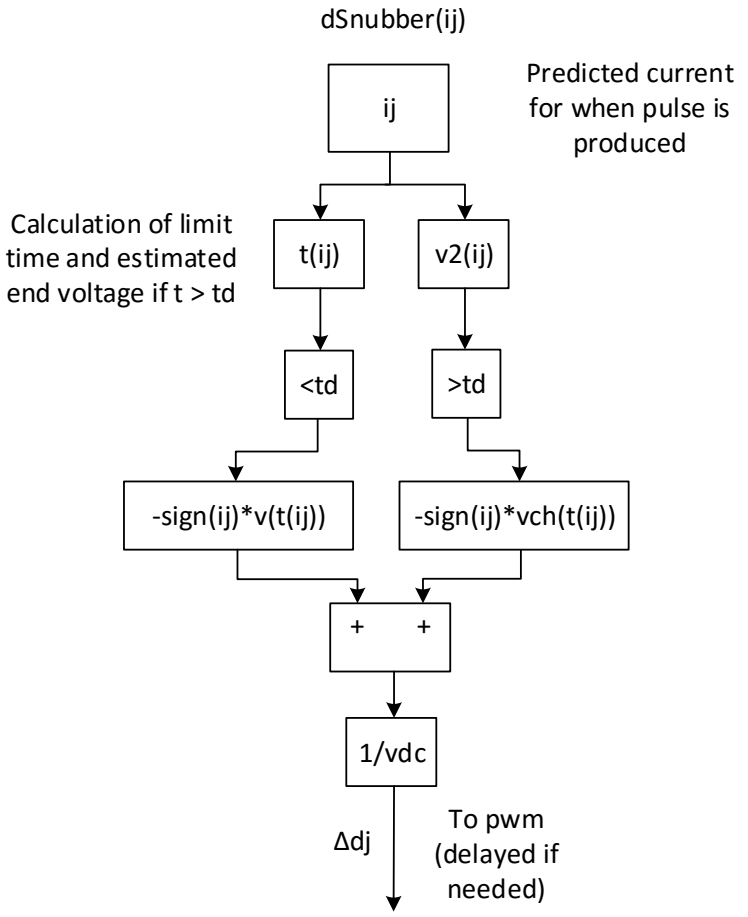


Figure 6.7.: Implementation of $dSnubber(i)$ function

From all this analysis it can be seen that the voltage compensation induced by the capacitors is load dependent as it is function of the current. When the current is small the capacitors have more time to discharge and therefore the voltage change is bigger. When the current is zero the voltage change due to the snubbers is the maximum and is the same as the voltage induced by the dead-time but with different sign so the dead-time is completely compensated on its own.

6.3 COMPENSATING DIODE AND IGBT VOLTAGE DROP

Apart from the snubbers there are more voltage changes in the output due to the voltage drop in the diodes and IGBTs. In this section a method will be deducted to compensate for it.

One can compensate with good precession the IGBT voltage drop easily by neglecting the effect of T_d and the snubber change in the voltage signal. As it will be explained the IGBTs and diodes induce a voltage difference during the on or off state which in

proportion of time is much smaller than T_d/T_s so this seems like a sensible approximation. For $T_d = 2.5\mu s$ and $f_s = 3000$, the proportion is 0.0125. In contrast the on and off varies but on average it will be close to 0.5. Under this approximation the leg j will have a duty cycle d_j and the voltage signal will be perfectly rectangular. The problem that the diode and the IGBT induces is that the upper and lower limit of the voltage will change from V_{DC} and 0 (using as reference the neutral of the converter) to a different value. If the new values are calculated the compensation is then trivial by subtracting the changes. The analysis can be done independently from IGBT and diode and then added together to the final result.

6.3.1 *Diode compensation*

It can be considered the voltage drop in the diode as an always positive function that takes the current in the leg as the only argument in absolute value (the sign can be added later by hand in the derivation). The voltage drop then can be approximated by Equation 6.7.

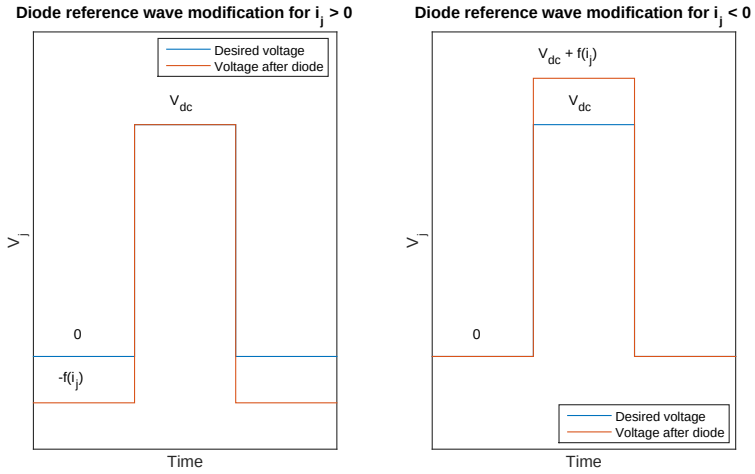


Figure 6.8.: Deadtime effect in the voltage command when considering the diode

$$f(i_j) = V_{on}^d + R_{on}^d \times |i_j| \tag{6.7}$$

One can infer by solving the circuit that there will be 4 different modifications in the voltage in the leg j depending if the upper or lower diode is conducting (positive or negative voltage) and if the current is positive or negative. When the current is positive the voltages at the output will be depending if the $+$ or $-$ is conducting as Equation 6.8 - 6.9 as the IGBT is not considered the drop is only produced when the current travels trough the diode.

$$v_+(i_j > 0) = V_{dc} \tag{6.8}$$

$$v_-(i_j > 0) = 0 - f(i_j) \tag{6.9}$$

For a negative current the voltage is transformed into Equation 6.10 - 6.11.

$$v_+(i_j < 0) = V_{dc} + f(i_j) \tag{6.10}$$

$$v_-(i_j 0) = 0 \tag{6.11}$$

The expected voltage without the diode effect is the same value as in Equation 6.8 - 6.11 by setting $f(i_j) = 0$ and is illustrated under the simplified square voltage wave in Figure 6.8. Therefore the compensation can be done by adding the difference between the expected voltage and the actual one. The steps performed are shown in Figure 6.9.

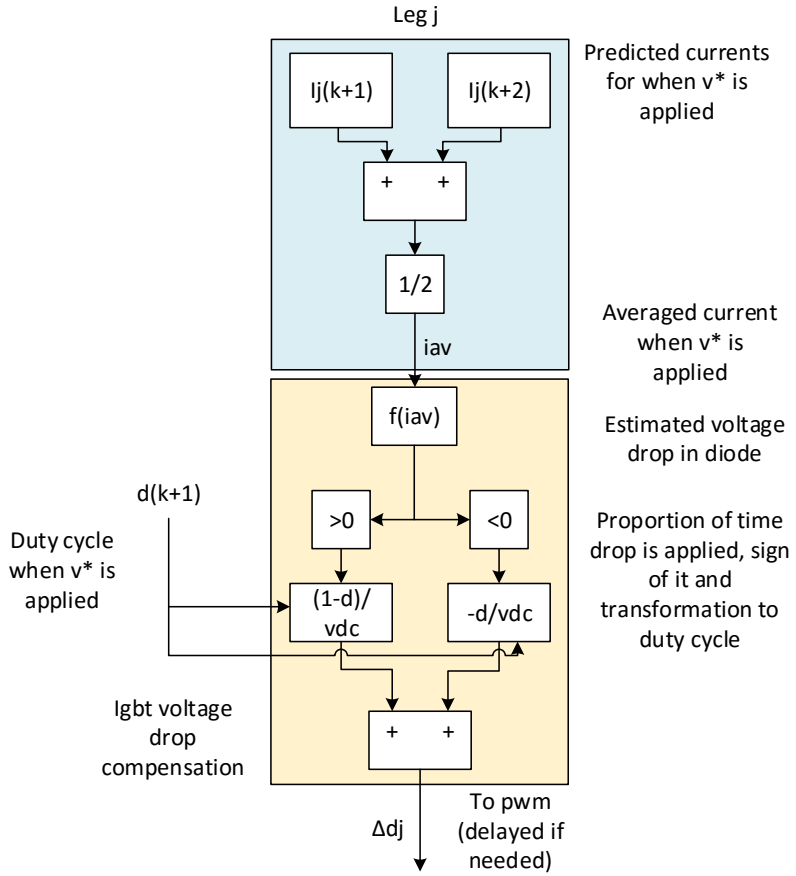


Figure 6.9.: Steps performed in one leg to compensate diode voltage drop

6.3.2 *IGBT compensation*

It can be considered the voltage drop in the IGBT as in the diode as an always positive function that takes the current in the leg as the only argument in absolute value. The voltage drop then can be approximated by Equation 6.12.

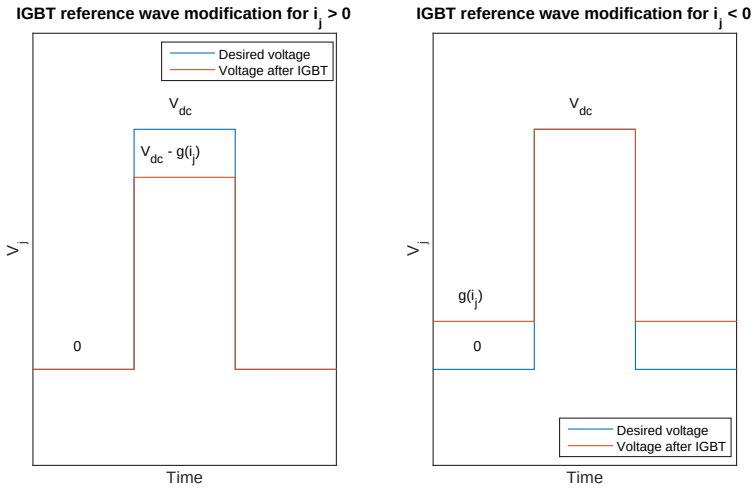


Figure 6.10.: Deadtime effect in the voltage command when considering the IGBT

$$g(i_j) = V_{on}^i + R_{on}^i \times |i_j| \tag{6.12}$$

One can again infer by solving the circuit that there will be 4 different modifications in the voltage in the leg j depending if the upper or lower part is conducting (positive or negative voltage) and if the current is positive or negative. When the current is positive the voltages at the output will be depending if the $+$ or $-$ is conducting as Equation 6.13 - 6.14 as the diode now is not considered in the drop and is only produced when the current travels trough the IGBT.

$$v_+(i_j > 0) = V_{dc} - g(i_j) \tag{6.13}$$

$$v_-(i_j > 0) = 0 \tag{6.14}$$

For a negative current the voltage is transformed now into Equation 6.15 - 6.16.

$$v_+(i_j < 0) = V_{dc} \tag{6.15}$$

$$v_-(i_j 0) = 0 + g(i_j) \tag{6.16}$$

The expected voltage without the IGBT effect is the same value as in Equation 6.13 - 6.16 by setting $g(i_j) = 0$ and is illustrated under the simplified square voltage wave in Figure 6.10. Therefore the compensation can be done by adding the difference between the expected voltage and the actual one. The steps performed are presented in Figure 6.11 which is very similar to the ones for the diode.

6.4 FINAL COMPENSATOR

The final code for the compensator proposed is presented in Code D.7. As was previously stated each consideration is simply added as another extra time to the total time to add

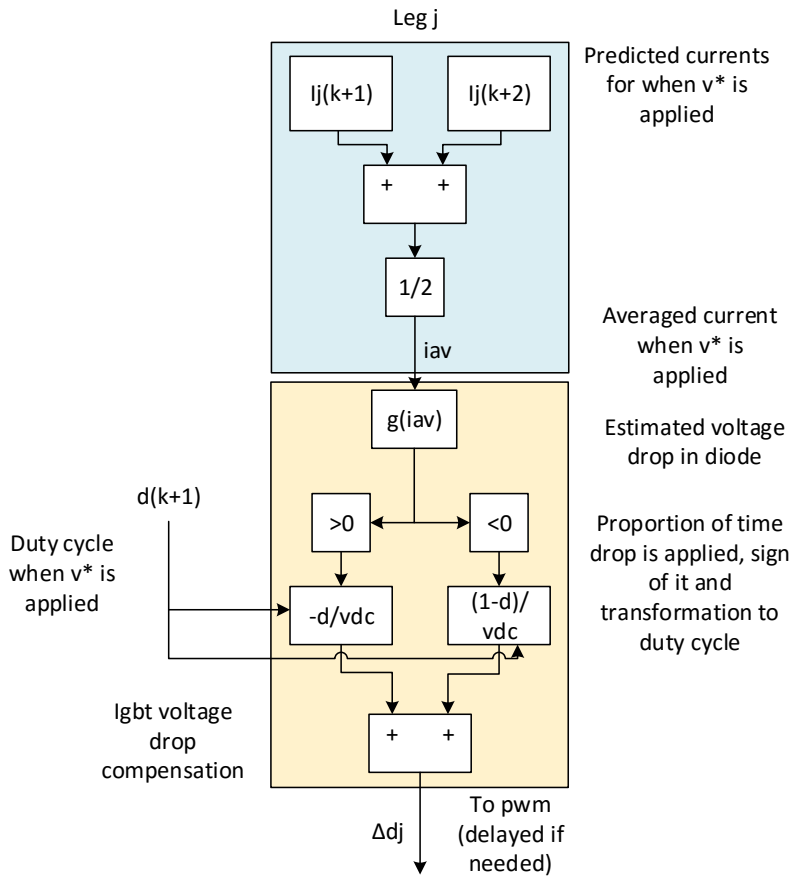


Figure 6.11.: Steps performed in one leg to compensate IGBT voltage drop

and finally transformed into duty cycles to add to the PWM. Some small modifications have been done for better testing of the algorithm in the experiment such as the possibility of changing the deadtime parameters and the type of compensation as inputs to the deadtime compensator function. It has also been tested to neglect the zero component and add the voltage to the dq reference directly. The result changes slightly.

6.5 RESONANT CONTROLLER COMPENSATION

It can be seen in Figure 6.3 that the voltage error due to the deadtime presents a periodic waveform. From the image it can be seen that in abc there is a main component at the electrical speed and harmonics since the shape is not purely sinusoidal. Further analysis of the effect of the other non-linearities present similar results as they are all affected by the sign of the current. The current magnitude is also present in the snubber however which may affect the overall voltage error shape slightly. In the IGBT and diode the resistor part is typically small and therefore the shape is similar to Figure 6.3 also. Testing with different load conditions and typical IGBT/diode voltage drops values suggest the shape is simiar. This suggests that the total effect of the non-linearities present a period waveform that has the same shape (and therefore same harmonics) and a resonant controller may be used to compensate the non-linearities.

The current error in the predictive-controller of 1-period delay can be calculated as the current reference delayed one period minus the actual current, that is $z^{-1}i^*_{abc} - i_{abc}$ if there is no saturation. For the 2-period version it would be $z^{-2}i^*_{abc} - i_{abc}$. With a PI instead of the deadbeat or if saturation is met the current may be predicted based on Equation 4.4 with the machine equations and voltage used. If the machine parameters are properly adjusted and the non-linearities not compensated it is expected that this error will also present a main component in the electrical speed and harmonics as suggested by Figure 6.3. Therefore this error can be used in a PR controller to compensate the non-linearities by placing a set of PR in parallel for all the harmonics. The diagram for the incorporation of the resonant controller for all the harmonics in cooperation with deadbeat is shown in Figure 6.12.

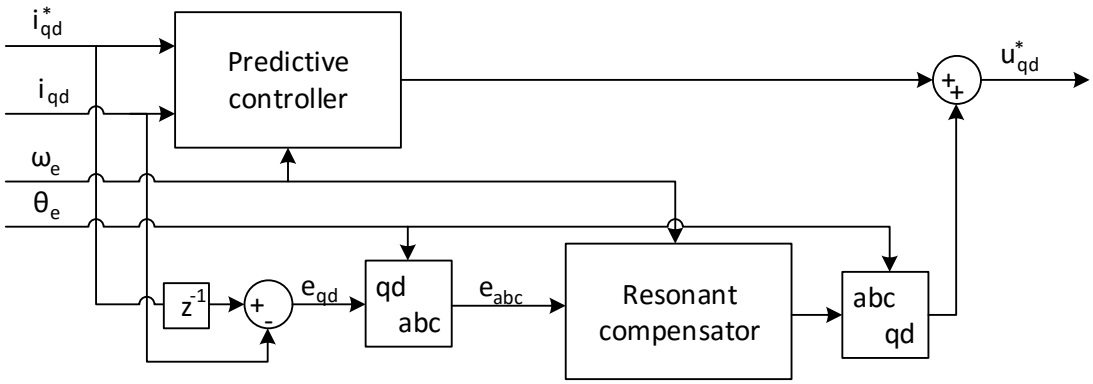


Figure 6.12.: Resonant controller with predictive-controller

The resonant controller for the compensation of the h harmonic in the error function of the i coordinate is depicted in Figure 6.13. The error fed is the current error for each coordinate in a, b, c and the resonant controller is the blue box in the diagram where outside K_r is the resonant gain to be tuned.

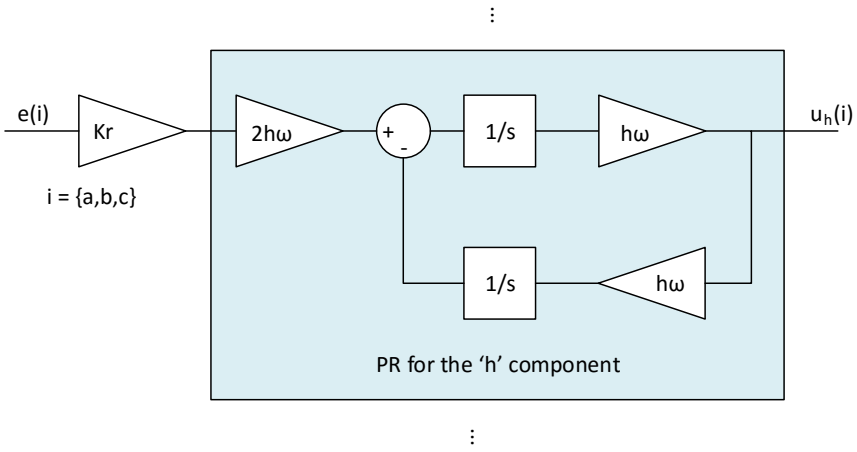


Figure 6.13.: Resonant controller for h component compensator of coordinate i

Common used PI controllers follow the form $K_p + K_i/s$ and the integral term K_i/s presents infinite gain at zero frequency. This can be interpreted as that any DC error will be compensated if enough time is left as the bode at this frequency is infinite. In the case of the resonant controller the same idea is used but the gain is made infinite at the desired frequency to compensate, that is $\omega' = h\omega$. The formulas can be derived by creating a transfer function of which the gain is infinite at the positive and negative $\omega' = h\omega$ and it is shown in Equation 6.17.

$$Res(s) = \frac{1}{s + j\omega'} + \frac{1}{s - j\omega'} = \frac{2s}{s^2 + \omega'^2} \tag{6.17}$$

If this transfer function $Res(s)$ is used any component at $\pm\omega'$ will be eventually compensated as the error produced will be integrated continuously until it is completely removed. This can be seen as this. The close-loop transfer function of a system plant and controller has the shape $\frac{G_c(s)G(s)}{1+G_c(s)G(s)}$ where $G_c(s)$ is the controller and $G(s)$ the plant. If a reference is given in AC at ω' and $G_c(s)$ has a generalized integrator as presented in Equation 6.17 the system will present a bode close to $0db$ at ω' and 0 degrees phase-lag as $G_c(s)$ as an input is a sinusoidal at ω' tends to infinite and the limits yields one. This means from the bode in steady-state the reference will be followed. The term $Res(s)$ is the blue square in Figure 6.13 as can be validated by operating the blocks. In the experimental implementation the first integrator may be discretized as forward-euler and the second as backward-euler as explained in [11].

This non-linearity compensator has the advantage of the simplicity and the adaptive process that adjusts itself without the need of knowing any non-linearity parameter

beforehand. As a disadvantage when used with the predictive controller is that it can be argued that it will couple with the machine prediction algorithm. For example since the non-linearities present a component in the fundamental it may affect the resistor and coupling terms as this component can be compensated by a virtual resistor and inductor that varies with the load used. Also there will always be one period delay due to the error used in the compensation.

6.5.1 *Main components involved in the non-linearities*

A fourier transform of the non-linearities compensator provided before in this chapter can be done to detect the main harmonics content and it is found that the main components are the 1, 5 and 7^{th} harmonic. Also a experimental test can be performed to obtain them and validate the analytical result. The machine can be set running in steady-sate with the predictive-controller. If the machine parameters are determined beforehand offline then any error in the current is due to the non-linearities.

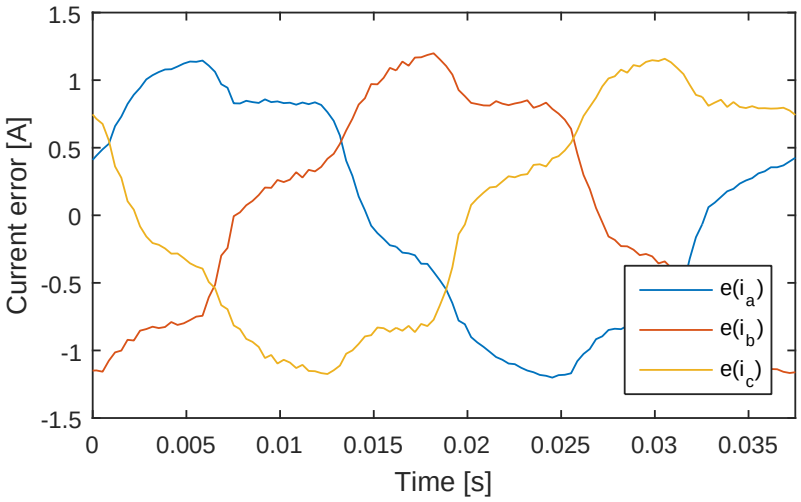


Figure 6.14.: i_{abc} error without non-linearity compensation

The current error $z^{-1}i^*abc - i_{abc}$ is recorded for several seconds and is plotted in Figure 6.14 for 400 rpm. Since the current follows the voltage it is interesting to see that the shape is similar to Figure 6.3. With this data a fourier analysis can be performed and the results are depicted in Figure 6.15. It can be seen in Figure 6.15 that the main components are placed at 1, 5 and 7 times the electrical speed. Therefore the PR can be used at those frequencies that can be updated in real-time in the PR.

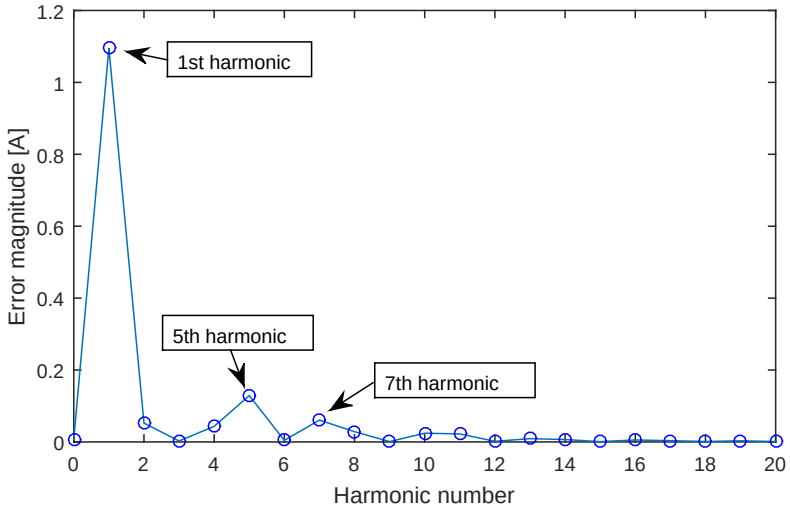


Figure 6.15.: Fourier analysis of error in i_{abc} at 400 rpm

6.6 SIMULATION RESULTS

In this part simulation results for each compensator are shown to show the validity of the simplification formulas derived. The parameters used in the non-linearities are either from the experiment or educated guesses since the point is to show the validity of the formulas. Since the predictive controller is used without estimating the machine parameters it cannot adapt itself and any error in the compensation will be apparent as a current error so if the compensation is adequate the error should be close to zero.

6.6.1 *Blanking time*

Simulation results for the blanking time alone are depicted in Figure 6.16 for one period of operation where both the current with and without the compensation is shown.

The figure shows that the compensation is almost perfect. The only origin of error is the current zero crossings. Since the blanking time changes with the current sigh each time there is a current sign change if it is not estimated correctly there will be an error in the compensation. Moreover due to the use of PWM the current that needs to be taken into account is not the current at the begging nor end of the period but rather the one where the PWM changes its state. Estimating correctly this value would require to take into consideration the load connected to the converter and while it is possible it would make the solution presented more complicated and less general.

6.6.2 *IGBT and diode compensation*

Simulation results for the IGBT and diode voltage drop alone are depicted in Figure 6.17 for one period of operation.

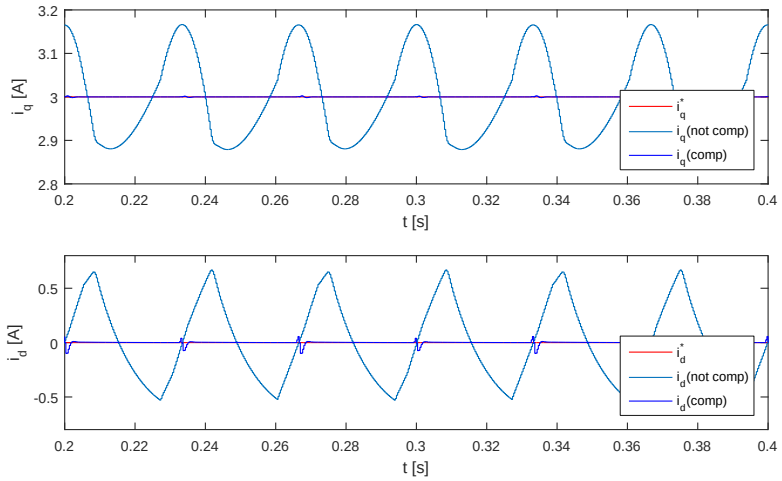


Figure 6.16.: Blanking time compensation results

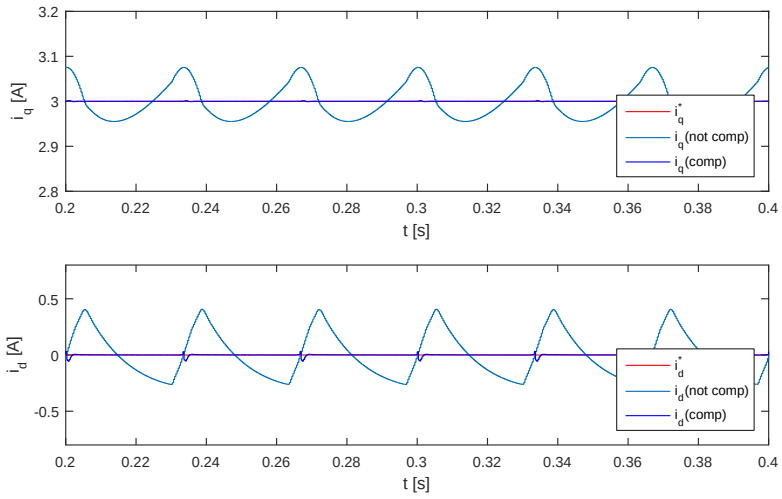


Figure 6.17.: IGBT voltage drop compensation results

It can be seen that the compensation is almost perfect and there are only problems in the current zero crossings. The blanking time has been removed from the simulation to see the compensation of IGBT/diode voltage drop alone. However the compensation of these elements is still dependent of the current sign and now also absolute value. Therefore when the zero changes signs there will be an error in the estimation similar to the one in the blanking time case.

6.6.3 *Altogether results*

The compensation in the simulations can be used to show the validity of the simplifications and formulas derived. Each compensator has been derived separately and tested on its own to ensure each component is working. After that all the parts have been combined and the total results are presented here.

In this section the response will be compared with and without compensations. Because of the formulas derived it can be inferred that the dead-time and non-linearities effect is function of the current and the components used (delay signal, snubber capacitor, voltage drop in igbts/diodes). While the speed does not play a role at first glance as the number of zero crossing is increased which are not easy to compensate it produces errors in forms of spikes. Therefore the following procedure is proposed. Non-linearity parameters are set from Table 2.3. Three different speeds will be used to see the effect due to zero-crossing errors. The reference for i_d is set to zero and i_q^* is changed in a dc sweep. The THD levels are recorded in continuous time including the switching.

To analyze the compensator the following will be recorded:

- SS-error of the currents in dq: The predictive controller will not eliminate SS-errors unless the system parameters are correct. In the event of non-linearities the controller will not be able to correct the SS-error, therefore a graph showing the SS-error can be used to assure the validity of the compensation.
- THD of the current (abc): Non-linearities will produce errors in the currents that are not constant and therefore the THD of the currents will increase as the non-linearities effect is increased. A lower value means a better compensation.

Steady-state error

The results for the steady-state error simulations without compensation (red) and with compensation (blue) are presented for the q axis (solid line) and d axis (dashed line) in Figure 6.18 - 6.21. For the uncompensated system it can be seen that the steady-state errors tend to saturate without compensation as i_q^* is increased. On the other hand as the speed is increased the steady-state errors decrease.

For the compensated response it can be seen that at lower i_q^* the errors produced are bigger but apart from Figure 6.20 they are always kept below the uncompensated system. At speeds close to rated as in Figure 6.20 the performance of the compensator decreases. This can be attributed to the fact that $f_s = 3$ kHz may be a small frequency. At $f_e = 250$ Hz, f_s is only 12 times bigger than f_e. If the same test is repeated at $f_s = 5$ kHz as in Figure 6.21 it can be seen a great improvement with the error always smaller in the compensated system.

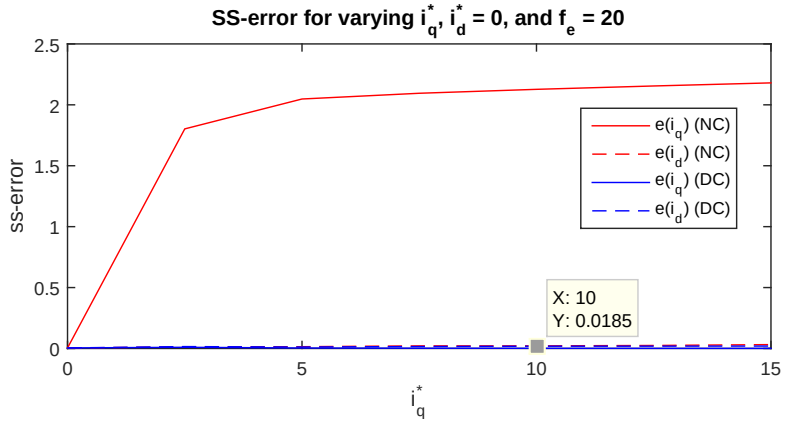


Figure 6.18.: Steady-state error at $f_e = 20$ Hz

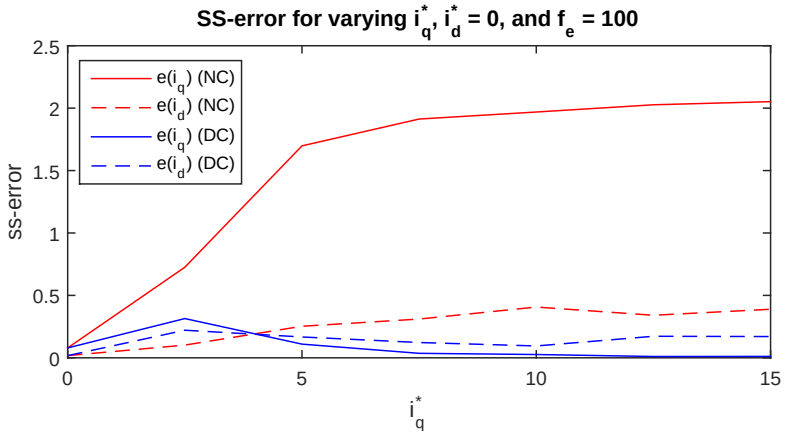


Figure 6.19.: Steady-state error at $f_e = 100$ Hz

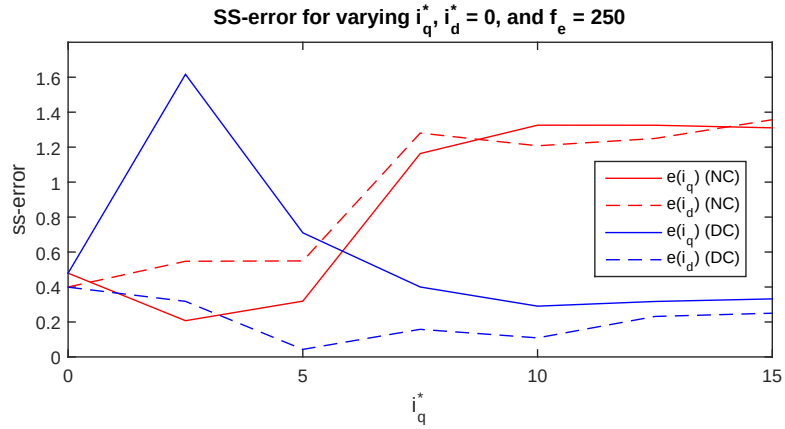


Figure 6.20.: Steady-state error at $f_e = 250$ at $f_s = 3$ kHz

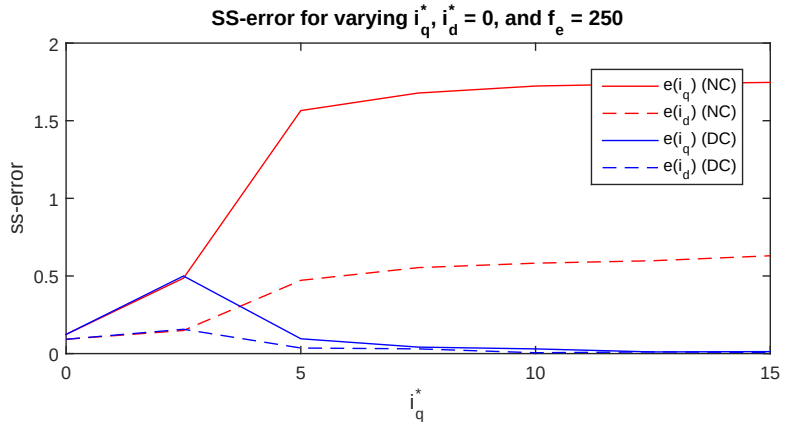


Figure 6.21.: Steady-state error at $f_e = 250$ at $f_s = 5$ kHz

THD

The THD levels before (red) and after the discrete compensation (blue) are shown in Figure 6.23 - 6.26. For reference the THD levels for an ideal converter (due to current ripple) at 3 kHz are depicted in Figure 6.22.

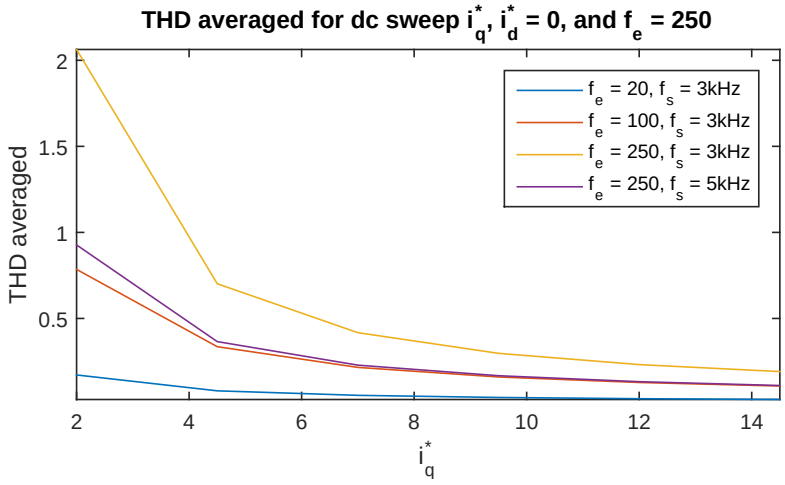


Figure 6.22.: THD for ideal converter at the same tests performed

It can be seen in Figure 6.23 - 6.26 that THD levels are always better in the compensated case than it is when compared to the uncompensated response. The effect of the compensation is more important at lower loads. However one may be cautious with the results at lower i_q^* that evidence better THD levels in the compensated system with non-linearities than in the ideal case. It was shown before the steady-state errors in the compensated systems and they are typically bigger at lower loads. If they increase the current then the THD levels will drop as the ripple is smaller in comparison with the fundamental. Comparing with Figure 6.22 it can be seen the THD levels are close to the ones from the ideal case so according to the modeled cases from a distortion point of view the compensation can be considered adequate.

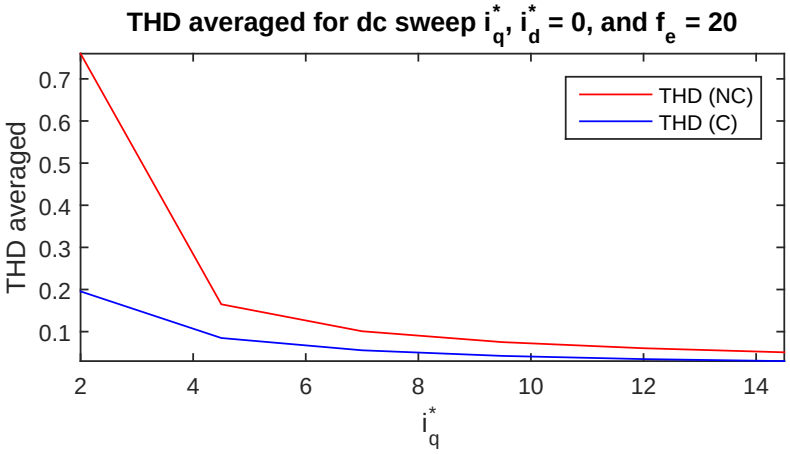


Figure 6.23.: THD at $f_e = 20$ Hz

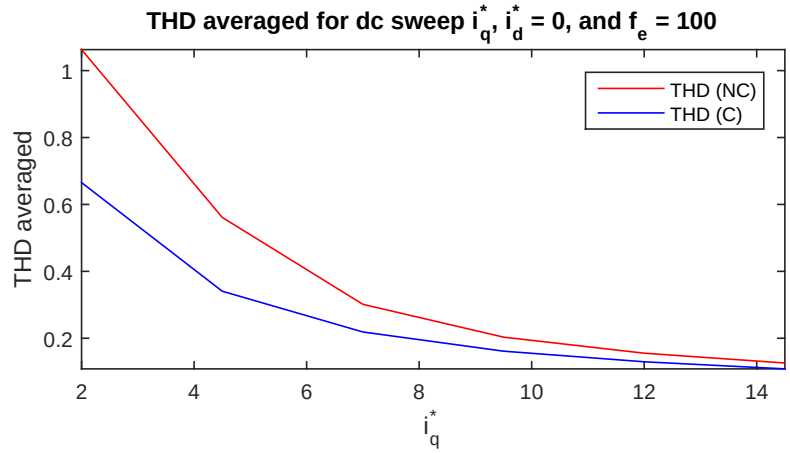


Figure 6.24.: THD at $f_e = 100$ Hz

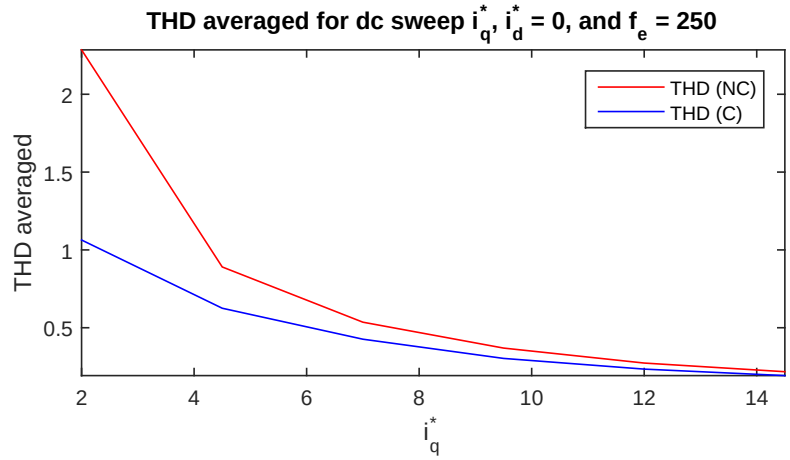


Figure 6.25.: THD at $f_e = 250$ at $f_s = 3$ kHz

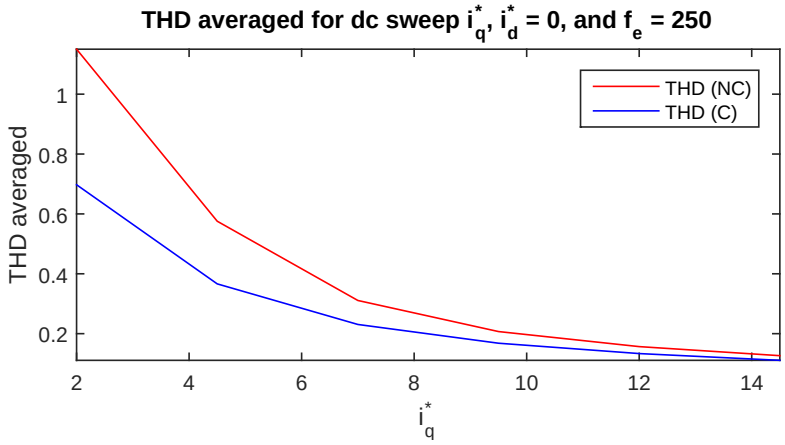


Figure 6.26.: THD at $f_e = 250$ at $f_s = 3$ kHz

6.7 EXPERIMENTAL RESULTS

Experimental results of the discrete non-linearity compensation and the resonant controller are shown here. The parameters used in the compensations in the discrete solution have been the ones from Appendix A but have been slightly modified to improve the response based on the ones that provided better experimental results at different working conditions.

The current error in *abc* before the compensation, with the discrete proposed compensation and the resonant controller is depicted in Figure 6.27. The speed used is 150 rpm. As for the currents $i_d^* = 4$ and $i_q^* = 5$. It can be seen how the error is decreased in both compensators.

While the resonant controller seems to have a better response its effect is worse as the speed of the machine is increased and may only be used at lower speeds. As the zero crossings which are the main responsible of the errors are more present and the sampling frequency is constant so as the speed is increased the proportion of zero crossings is increased. This produces a big distortion in the current error due to the relative low f_s. The discrete compensator becomes also worse at higher speeds. However when used the error is still decreased. The resonant controller on the other has been observed to unstabilize the system at higher speeds so as the speed is increased it is proposed to use only the discrete solution.

The effect of the speed is explored in different experiments with and without the proposed discrete compensation at 100, 500 and 1000 rpm. In Figure 6.28 - 6.30 the current response during two periods are depicted for those speeds with $i_d^* = 0$ and a constant torque with the speed controller setting i_q^*.

It can be seen an almost perfect compensation at 100 rpm in Figure 6.28. As the speed is increased the compensation is worse. At 500 rpm as shown in Figure 6.29 it is still quite adequate but there is a small steady-state error in i_d and the spikes are more present and i_q is still ok. This evidences a problem in the compensation that is closely related with the zero crossings as they introduce a high uncertainty. Finally at 1000 rpm the error becomes even worse as shown in Figure 6.30. The error in i_q is much worse

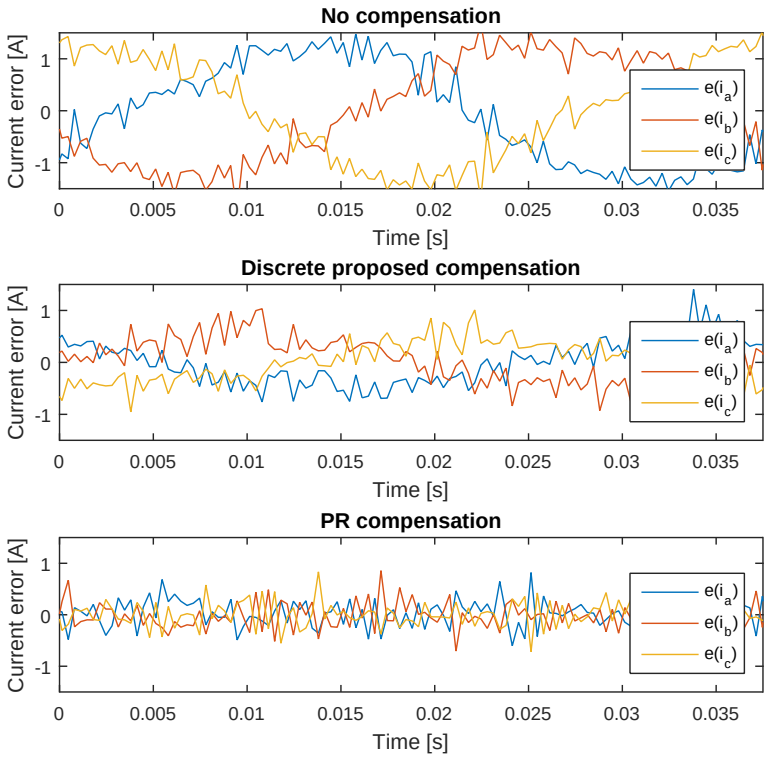


Figure 6.27.: i_{abc} error before and after PR compensation

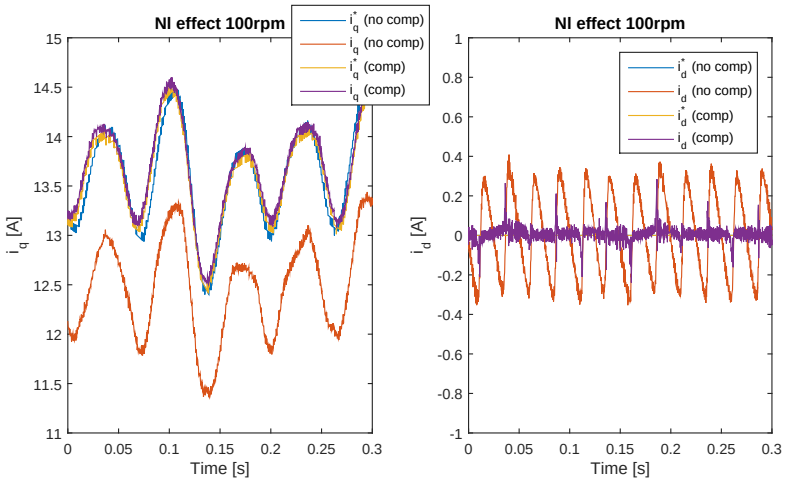


Figure 6.28.: Non-linearities before and after compensation effect in current error at 100 rpm

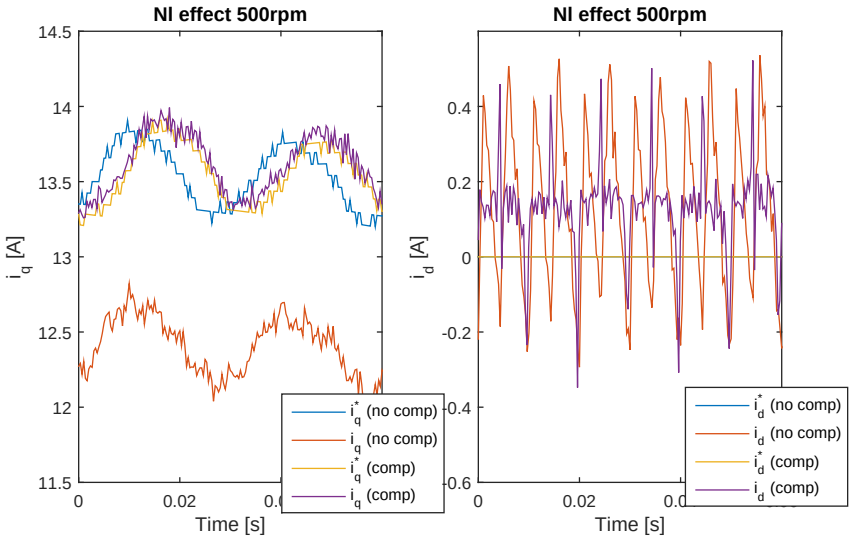


Figure 6.29.: Non-linearities before and after compensation effect in current error at 500 rpm

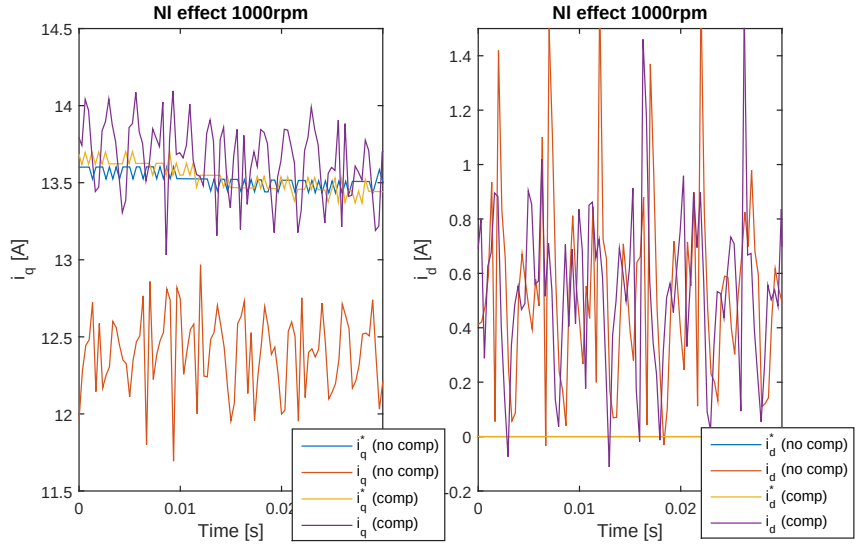


Figure 6.30.: Non-linearities before and after compensation effect in current error at 1000 rpm

and in i_d the spikes worse. There is a clear improvement with the compensation at any speed but the effect can be seen worse as the speed is increased.

Even without compensation the peaks grow and so they cannot be attributed to errors in the compensation. Use of PI controller instead of deadbeat present similar problems with the spikes (the *DC* offset however is compensated). It may be concluded that as the speed is increased and f_s is kept constant there are too many zero-crossing which are not correctly compensated as the current is estimated at the beginning and end of the period and not when the PWM is changed.

7 PARAMETER DETERMINATION

As machine parameters can change they can affect the performance of the predictive algorithm. The effect of the different parameters variations was previously discussed in Chapter 5. In this chapter different parameter estimation algorithms are presented. After each method both simulation and experimental results are provided.

7.1 INTRODUCTION

As it has been commented before in Section 1.2 and Chapter 4 the machine parameters may need to be estimated online if a predictive controller is used. In Figure 7.1 the machine parameters are shown with the main component that affects the change as taken from [12].

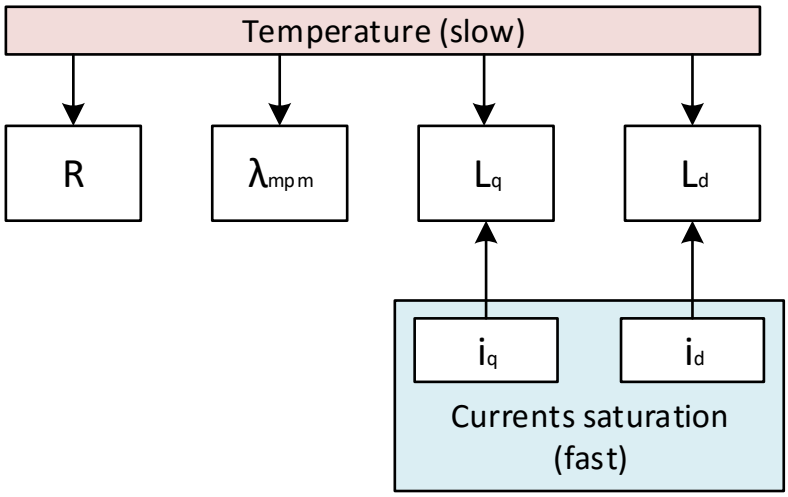


Figure 7.1.: Machine parameters and changes nature

It can be seen from [12] that all the parameters are affected by the temperature that changes their values slowly in comparison with the current loop. The value of the resistor can change +40% (copper, 100 degrees change). Depending of the flux material the change may be as low as −3% for a 100 degree change for SmCo or −11.1% in NdFeB. As for the inductors the effect of the temperature in their value can make then drop 25% for 80 degrees change.

Magnetic saturation is much faster and affects mostly the inductors [12] as a function of the current decreasing their value past some limit. Changes in the values due to this phenomena are hard to model and depend of several parameters. Some experimental

tests have been performed in the machine in the lab and assuming the online parameters later described are accurate it was shown almost constant L_q for $0-15$ A range and a change in L_d past 10 A of -1% per amp as depicted in Figure 7.6.

As it has been described before in Chapter 5 inductors are the only component compromising stability of the predictive controller. From the values read in [12] the system would continue being stable due to temperature as the value does not drop to half or less. However at 25% the overshoot will be close to 33.3% from Equation 5.19 and the settling time at 5% would increase from 1 to 3 periods in the 1-period controller and from 2 to 6 in the 2-period version. If a similar behavior due to saturation happens in L_q and if the almost linear tendency is kept then it is also unlikely that the saturation will make the system unstable from the values read.

From the steady-state error in Figure 5.13, in the 2-period version the resistor will introduce at 100 degrees temperature change around 0.02 A/A^* (current error per commanded current). As for the inductor, for a value of 30% change it would be as big as 0.16 A/A^* in one axis and 0.06 A/A^* in the other at rated speed. The flux in comparison would be at rated speed 1.65/3.3 A/A^* in each axis if it is made of SmCo. For NdFeB, this number would be 0.45/0.9 A/A^* at rated speed. Those are values at $f_s = 5$ kHz as they vary with f_s. For example running at $f_s = 10$ kHz would decrease an error of 1.65/3.3 A/A^* due to the flux to around 0.45/1.6 A/A^*. Errors in inductors are also expected to drop and the resistor will error will almost not be affected. While the global error will be approximately (as from Taylor approximation) the sum of all of them and some of it may be canceled out it can be seen the effect in the steady-state error can be important.

From those quick calculations it can be seen that from a stability point of view it is unlikely the predictive controller will become unstable. Its transient however will be affected making it around 3 times slower and the steady-state errors depend of many conditions but can be seen to be important. In definitive it seems that online parameter determination can be an important step needed in the implementation of a reliable predictive controller algorithm.

The following algorithms have been tested to estimate the parameters online:

- Recursive Least Square
- Gradient descent method

7.2 RECURSIVE LEAST SQUARE (RLS)

The motor equations can be rewritten in matrix form as Equation 7.1 - 7.4. With this notation it can be seen the system is linear in the parameters and a linear regression method may be applied.

$$\vec{y} = H\vec{\theta} \tag{7.1}$$

$$\vec{\theta} = \begin{bmatrix} R \\ L_q \\ L_d \\ \lambda_{mpm} \end{bmatrix} \tag{7.2}$$

$$\vec{y} = \begin{bmatrix} u_q \\ u_d \end{bmatrix} \tag{7.3}$$

$$H = \begin{bmatrix} \frac{1}{2}(i'_q + i_q) & \frac{1}{Ts}(i'_q - i_q) & \frac{\omega_e}{2}(i'_d + i_d) & \omega_e \\ \frac{1}{2}(i'_d + i_d) & -\frac{\omega_e}{2}(i'_q + i_q) & \frac{1}{Ts}(i'_d - i_d) & 0 \end{bmatrix} \tag{7.4}$$

Since the system is linear the parameters could be obtained by measuring 4 different points and inverting H. However in the real experiment noises my affect the measurements and this may not be such a good idea. Therefore a linear regression with more points may be done so even though there are noises in the measurements on average things will tend to go on the right direction. For more measurements that number of parameters, the parameters may then be estimated as Equation 7.5.

$$\theta = \left(H'H\right)^{-1} H'Y \tag{7.5}$$

The problem with this method arises in that usually micro controllers cannot perform big matrix inversions at a reasonable speed and the algorithm may not be implemented in such small devices. Another more common solution is to use a recursive least square solution. As parameters can change over time a forgetting factor λ may be added. The forgetting factor λ between 0 and 1 gives λ^k weight to the error in the k past measurement. Previous work from [12] - [13] show the approach is feasible and discuss the method in detail for the use with both equations. While the q equation is enough to determine machine parameters as it contains all the parameters it seems more sensible to use both axis. Otherwise errors in the d axis would not affect parameter determination. The formulas for this algorithm are shown in Equation 7.6 - 7.9.

$$\vec{e} = \vec{y} - H\vec{\theta} \tag{7.6}$$

$$K = PH' \left(\lambda + HPH'\right)^{-1} \tag{7.7}$$

$$\vec{\theta} = \vec{\theta} + K\vec{e} \tag{7.8}$$

$$P = \frac{P - KHP}{\lambda} \tag{7.9}$$

The value of λ can be chosen to regulate how fast predictions change. A lower value will give more importance to new values but will also be less immune to noise. On the other hand an initial value of P must be set and is usually done to kI where I is the unity matrix and k a small value. Setting k high will mean less confidence in the initial estimations and faster initial change while setting it low does the opposite.

7.2.1 *Simulation results*

The initial machine parameters are set the same as in the experiment but the initial values in the parameter estimator are modified to see the response when the parameters differ.

Depending of the working condition some parameters can play a small role and the determination cannot be made precisely. Typically in low speeds the resistor is more dominant and the flux may be hard to estimate while at high speeds it is the other way around. In simulations however if there is no added noise in the measurements any parameter may be determined correctly at any working condition.

Presented in Figure 7.2 is the evolution of the parameter estimations over time for different values of λ to see the effect where $P_0 = 10^{-4}I$ has been chosen and no noise is added.

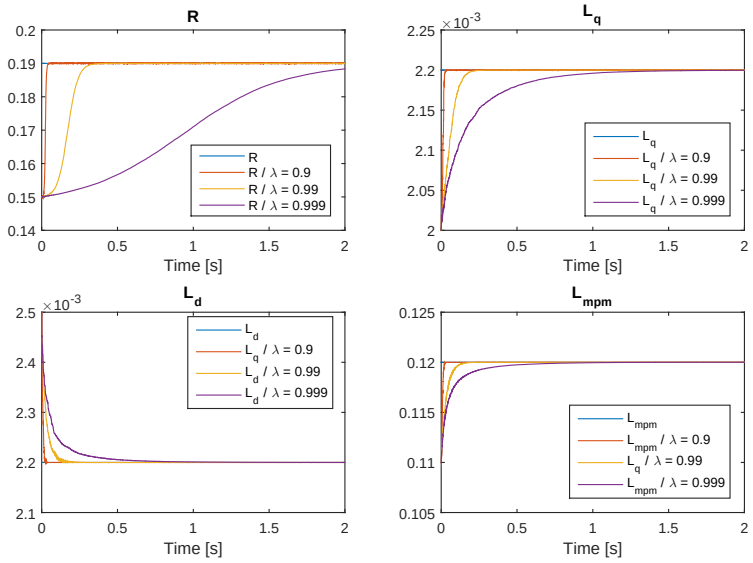


Figure 7.2.: RLS algorithm with different λ

The algorithm can be made much faster decreasing the value of λ but when the noise is considered small values of lambda produce oscillating results. Since the resistor plays a smaller role it is typically estimated slower. The current is perturbed randomly to generate new independent point as in steady-state with 2 equations and 4 parameters they cannot be estimated. Changing the amplitude of this perturbation changes the weight of the inductors in the error and thus increasing the amplitude makes the inductors be estimated faster at the expense of bigger errors in the other parameters (when noise is considered) or slower convergence of the later as has been seen from both simulation and experiments.

As for the initial value of P some considerations can be made. A bigger P means low confidence in the initial estimations and the opposite is also true. However as this matrix will adapt itself it seems sensible to use a relative small initial value and let the algorithm find the best value adaptively. Setting it too high may cause instabilities at first or even

divergence. Following the recommendation of setting it to kI seen in previous work different values of k have been tested experimentally and it has been concluded that $k = 10^{-4}$ seems like a good initial value. After the algorithm has been launched and has found the parameters the initial value of P is not relevant.

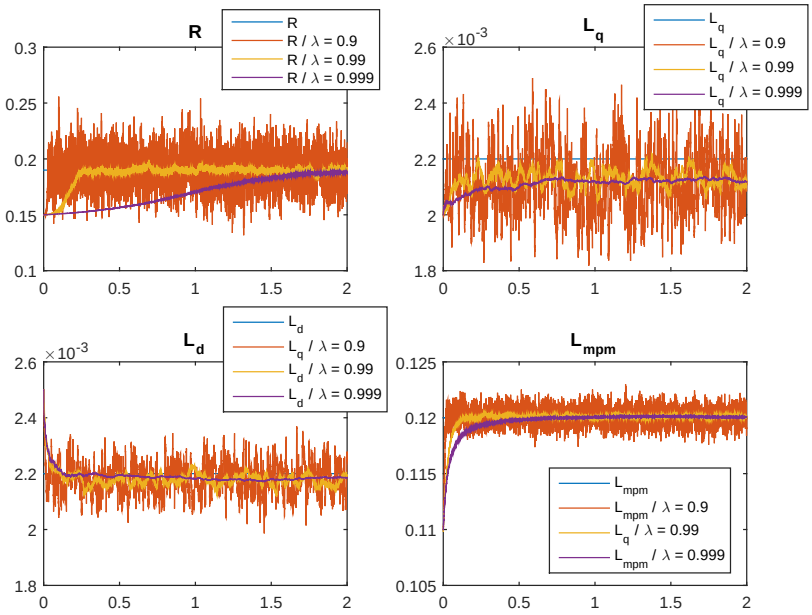


Figure 7.3.: RLS algorithm with different λ for a random noise between -0.2 and 0.2A

It is depicted in Figure 7.3 the same results done with an added random noise between -0.2 and $0.2A$. The current sensors in the lab have a smaller error but the effect of not fully compensating the non-linearities can be seen as an added noise (not random) which at high speeds can be as high as $1A$ (with spikes around the zero crossing and otherwise close to zero) and is not symmetrical. This is the reason why estimating machine parameters at high speeds is complicated. However as long as the disturbance is kept within reasonable limits it can be seen in Figure 7.3 that the machine parameters can still be estimated. In this case with the considered noise it does not seem sensible to use a value of λ as low as 0.9 and 0.99 or even 0.999 seem a better choice.

Another way to minimize the noise issue in predictions can be to use bigger perturbations in the current so the differential terms have a big weight and inductors can still be predicted. This has the disadvantage of making flux estimation harder and after some point it may require too big pulses. Setting perturbations in the reference in estimation too big has the drawback of maximizing the weight of the inductors and in different conditions parameters such as the resistor and flux may be hard to be estimated correctly due to the low weight in the voltage equation.

7.2.2 *Experimental results*

The algorithm has been split in different parts in the experimental implementation since some parameters can only be estimated at certain conditions due to their variable weight in the voltage. The code used is given in Code D.8

At low speeds the resistor and inductors can be estimated and the flux considered constant with the previous value used. The results for $w = 100rpm$ are presented in Figure 7.4.

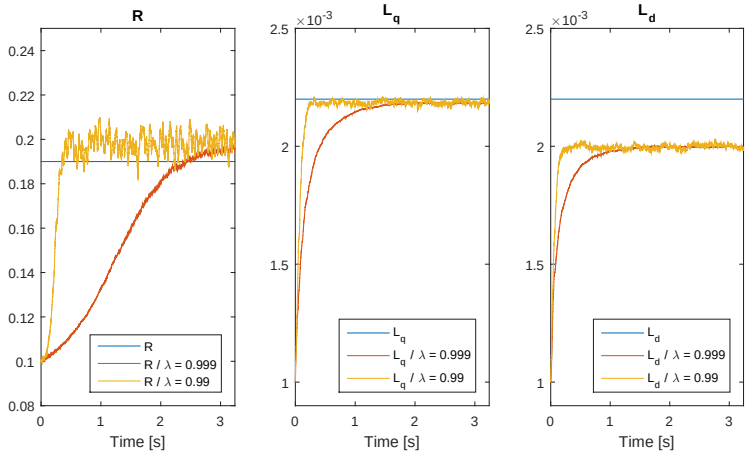


Figure 7.4.: RLS algorithm with different λ for RL estimation

The results seem adequate and are similar to the ones obtained offline in Figure A.4. Even though the non-linearity compensation is not perfect it helps in machine parameter determination. Failing to compensate the non-linearities at low speed tend to produce big errors in the resistor estimation mainly.

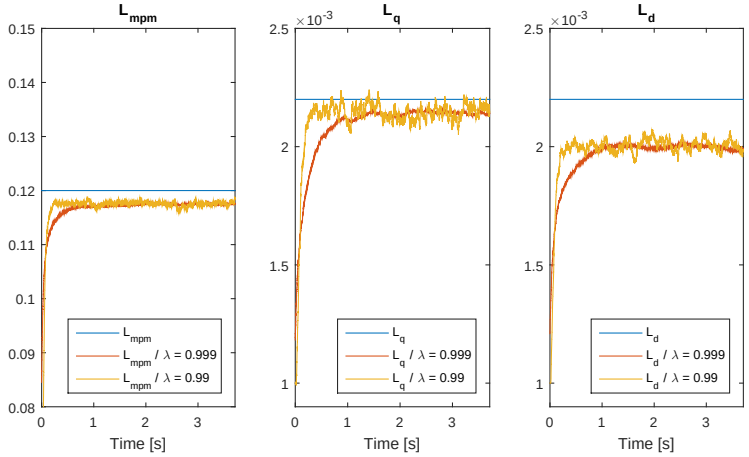


Figure 7.5.: RLS algorithm with different λ for LF estimation

Finally at higher speeds the flux is more dominant and the resistor stops playing a big role so another estimation considering the resistor constant is done to estimate the flux and inductors. Results are shown in Figure 7.5.

The algorithm seems to be able to estimate the machine parameters with good accuracy. Another test can be performed to see if the algorithm is able to estimate inductor values under saturation. For this purpose i_q has been increased (by increasing the load) and has been seen that the estimation of L_q is more or less constant. This however can be attributed to the fact that the maximum current that can be provided due to the torque limits is around $15A$. At this level the value of L_q seems not to drop. With higher currents it would be expected to drop after saturation were found.

In the d axes however the current can be increased much more as the limit now is the converter of $32A$ from Table 2.3. For this purpose several steps of $1A$ have been performed in i_d from 0 to up to $28A$ every second which is enough time for the algorithm to settle. Then the result of the RLS estimation at the end of each step plus some previous points is used to create an average of the prediction (to remove ripple as much as possible) where $L_d = f(i_d)$ can be found. The results for L_q and L_d estimation as function of i_d are shown in Figure 7.6.

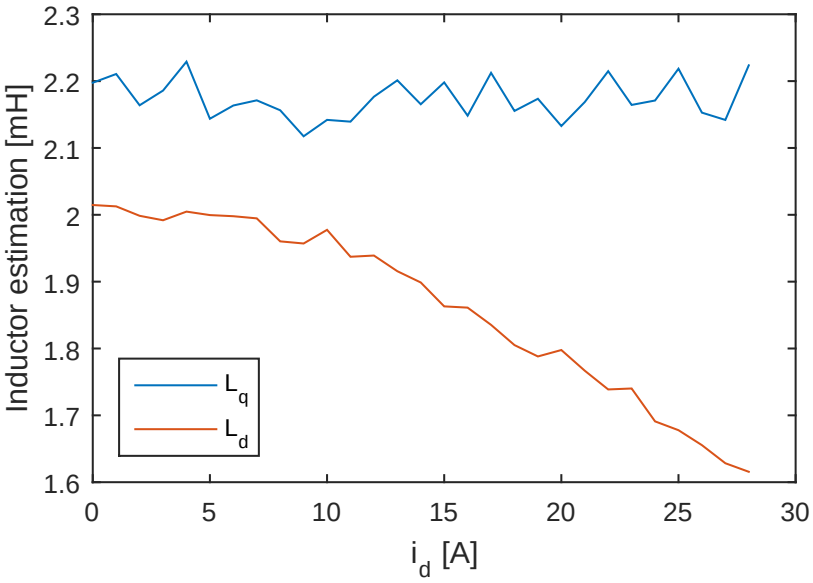


Figure 7.6.: RLS algorithm predicting inductor saturation

It can be seen in Figure 7.6 that the value of L_q can be considered constant as i_d is increased while the value of L_d is constant up to about $10A$. After that point the estimation drops so it can be inferred that saturation in the d axis occurs after $10A$.

7.3 GRADIENT DESCENT METHOD

Inspired by the gradient descent algorithm the parameters can be modified in the direction given by the opposite of the gradient of an error function over parameter change. A model can be run in parallel that tries to predict either the current or voltage by using the machine parameters and measured currents and voltage commands. An error function can then be created as Equation 7.10 by comparing for example the applied voltage (as given by the controller) to the voltage that is expected to have been

applied using measurements and the estimated machine parameters. There are more choices for the cost functions and the currents could also be used instead of voltages as a measurement of the error. The choice of the voltage is made because using the current as a measurement of the error would require to perform matrix inversions. For the choice of the cost function it can be proved its global convergence in Section B.4.

$$e(u_q, u_d) = |u_q(command) - u_q(model)| + |u_d(command) - u_d(model)| \tag{7.10}$$

The gradient of the error to respect to the parameters can be either calculated analytically or more easily estimated with small perturbations in the parameters in the running models. Once the gradient is obtained the parameters, $\vec{p}$, can be updated by using Equation 7.11 where they are moved in the contrary direction to a normalized vector gradient (each component measures the error difference which respect to each parameter) that is multiplied by λ.

$$\vec{p} = \vec{p} - \lambda \frac{\vec{grad}}{\sqrt{\sum \vec{grad}(j)^2}} \tag{7.11}$$

As there are more equations than parameters with only one point it is impossible to predict the parameters. However, if the current reference is perturbated with random values (to ensure new points) and λ is small, on average the algorithm will converge to the true value of the parameters without oscillations and small ripples.

A compromise for λ can be made between speed and ripple in the estimations and its effect is similar to λ in the RLS method. Smaller values of λ will filter noise in the measurements better but will yield slower convergence. Bigger values means faster convergence but more noises in the predictions. Alternatively a low pass filter could be applied to the estimation to smooth the ripple but this will also make the predictions slower. However a compromise between λ and the filter may give better results than using λ alone. It may also be studied the use of a variable λ as a function of the error.

The code implemented in the lab is presented in Code D.9. The parameters are scaled by the initial values and some modifications are done for simplicity of the testing. The gradient is estimated with a small perturbation over the model.

7.3.1 *Simulation results*

The results for the estimation of the resistor and inductors at a lower speed of $50rpm$ is depicted in Figure 7.7 for different values of λ.

The algorithm is also tested for the estimation of the flux and inductors at a higher speed of $400rpm$ and is depicted in for different value of λ in Figure 7.8.

It can be seen that in both cases the algorithm performs worse than the RLS method. The time to settle tends to be longer for the similar degrees of noise in the predictions. The results could be improved by either using a variable λ that changes as a function of the error or by making a compromise between λ and the use of a low-pass filter to smooth the predictions.

It is concluded that while the results of this algorithm could be further improved the RLS method is more suitable as the system is linear and the gradient method is a general

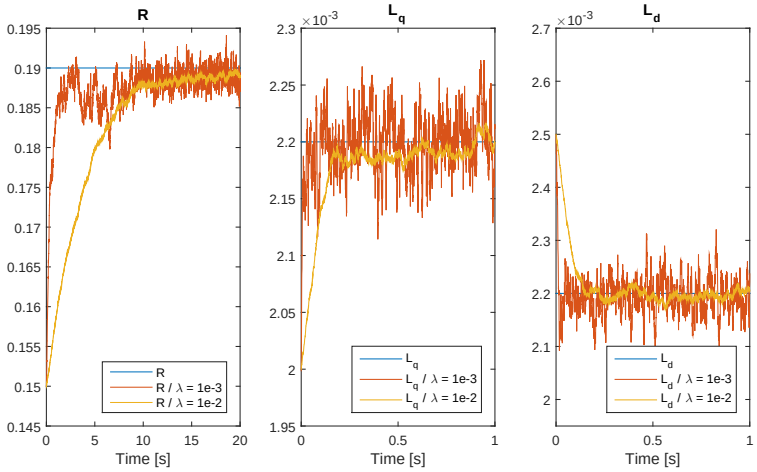


Figure 7.7.: Gradient-based method RF estimation under simulations

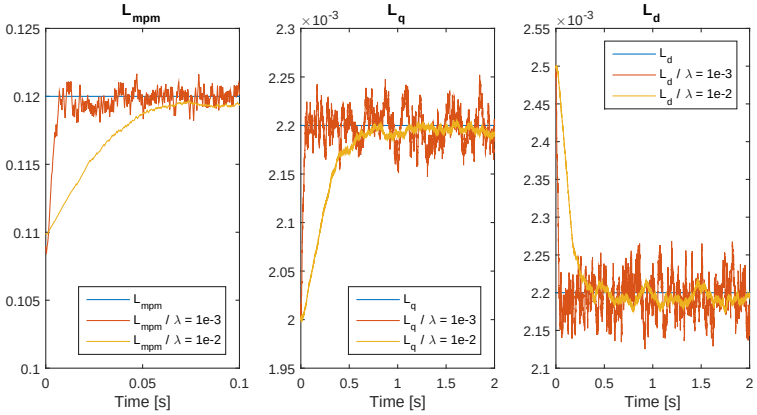


Figure 7.8.: Gradient-based method RF estimation under simulations

purpose algorithm. The use of the later may then be left for non-linear systems where the RLS method cannot be applied.

7.3.2 *Experimental results*

The results of the algorithm are presented in Figure 7.9. The code used is given in Code D.9. In the figure, the parameters are estimated from the initial value used at the beginning of the graph 2 times. The blue lines are the estimations and the red expected values as given in the datasheet. First the estimation is done with $\lambda = 0.001$ and secondly with $\lambda = 0.01$. As expected the first gives slower results but less oscillations in contrary to the second. For the estimation both i_q and i_d were randomly perturbated with values between $\pm 2.5A$. In the experiment, the speed was $200rpm$ and the torque $4Nm$ with the average $i_q = 6$ and the average value of $id = 5$. At this speed the estimation of the resistor may not be accurate and this is the reason why its value is wrongly overestimated. The same procedure as with the RLS method may be done then and two separate estimations developed, one for the resistor and inductors at lower speed and one for the flux and inductors at higher ones.

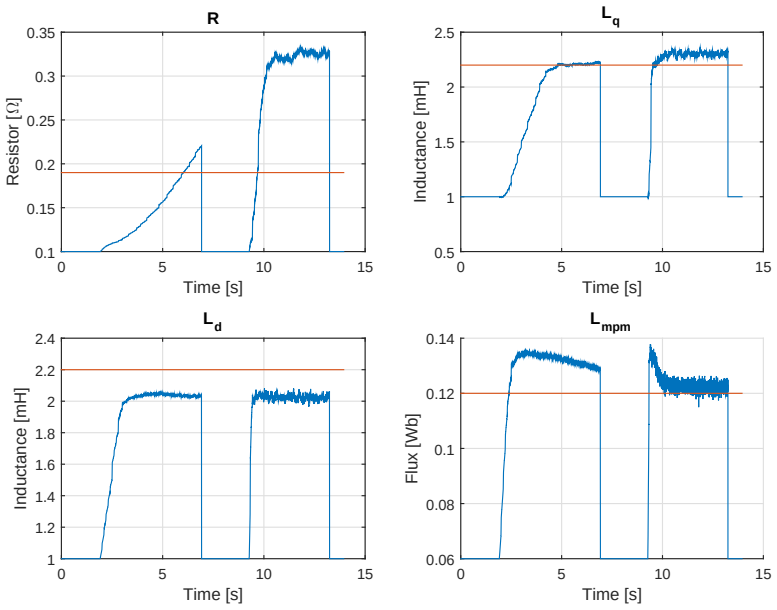


Figure 7.9.: Evolution of parameters estimations using far initial values in gradient-method in the experiment

As expected the algorithm prioritizes changes in the parameters that affect most the system. In this respect it tends to find the true value of the inductors much faster than the resistor and flux linkage. This is adequate since this goes in concordance to what it is expected as inductors are more crucial for stability and performance and also vary much faster than the other two parameters. The estimations shown in Figure 7.9 seem adequate and the bigger difference in R may be due to the fact that at that condition it may play a small role and may be better to leave out of the estimations.

7.3.3 *Experimental results for saturation*

By increasing i_d the machine can be saturated and this would be seen as smaller values in L_d. The algorithm can also be tested by forcing a saturation on L_d and seeing if it shows the decrease of it as i_d is increased. Results are presented in Figure 7.10.

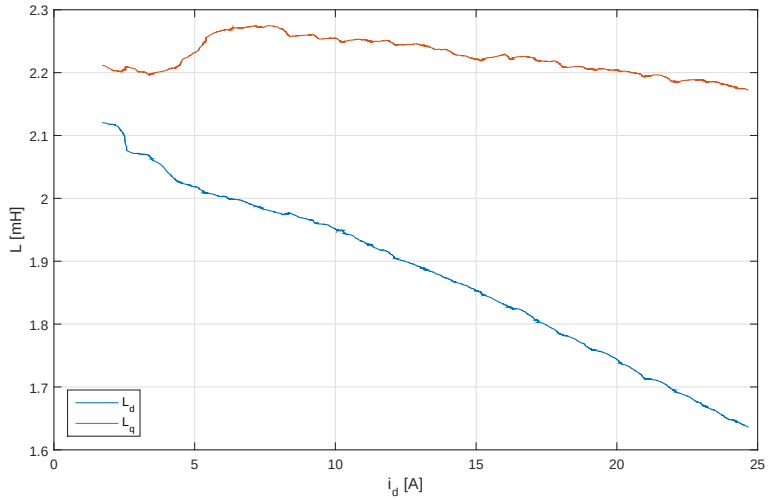


Figure 7.10.: Evolution of estimated L_q and L_d as i_d increases

The test has been performed as follows. The current in i_d has been increased in steps of $1A$. To make the predictions both i_q and i_d references have been perturbed with random values bewteen $\pm 2.5A$. To save time in the data process two lower pass filters have been applied to both the inductors and i_d resulting in Figure 7.10. This can be considered adequate as the graph point is to show the tendencies and that the algorithm is able to predict the saturated inductors and the filter will remove the transient in the estimations and the perturbations added to the currents presenting still a good accuracy as the steps have been small.

8 CONCLUSIONS

The objective of the project was to design a predictive controller, analyze and solve the different problems present with its implementation.

The classical FOC with PIs was revised in Chapter 3 as it is a widely used method to control this type of machines. It was found possible under simulations and experiments to design current controllers with settling times of 6 - 8 times the sampling time, little to no overshoot and no steady-state errors when the feed-forward terms are added and ideal converters are considered or non-linearities compensated. It was found that at higher speeds the transient response is weakened due to the simplified coupling terms used.

In Chapter 4 the predictive controller was introduced. Due to limitations in DSpace two versions were developed. The most widely used version where the command is delayed one period due to the DSP computation delay was implemented with the different compensations added. It was found under simulations to match the 2 sampling time settling time with little to no overshoot when all the delays are compensated, ideal converters are used (or non-linearities compensated) and machine parameters are known. In the experiment implementation the 1-period version without the added delay in the voltage command was developed as it was not found possible to ensure the voltage command to enter the PWM in the next period. As the experimental results show with the non-linearity compensations it was possible to reach a settling time of 1-period.

In Chapter 5 the effect of changes in machine parameters were analyzed. First the stability limits were studied based on MIMO analysis with a pole map. It was found that only the inductors compromise the system stability when their value drop by half or less. The limit was validated under simulation and experiments. The steady-state errors were calculated and later approximated with a first-order Taylor series which showed the main effect of each parameter in the error. It was possible to match each state and parameter to the error in a simplified form. It was found as the speed is increased inductors and flux are the main responsible to the error. Finally the effect of parameter changes in the transient were also analyzed under some approximations validated under simulations which showed the inductors as the components with the biggest role in it. The results presented in this chapter can be used to analyze the need for an online parameter estimation algorithm.

The non-linearities in the converter were studied in Chapter 6. Deadtime, snubbers, igbt and diode voltage drops were considered. For each one of them some approximations were taken to yield solutions that were validated under simulations and experiments. It was found that the difficulty of a non-linearity compensation lies in the zero-crossing of the currents as they introduce a high uncertainty and error in the compensation. For this reason it was found that for a given load as the speed is increased the steady-state errors and spikes in the currents increase with the compensation. This was found both under simulation and experiments.

Finally in Chapter 7 two online-parameter estimations, RLS and a gradient method were implemented in simulations and experiments. Both of them were tested and showed good accordance with the offline parameters determined. Moreover it was possible to see the drop in L_d as i_d was increased. The results show the approach is feasible if non-linearities are correctly compensated.

8.1 FUTURE WORK

From the results of this project it can be seen that the predictive deadbeat controller is a simple and effective solution with great transient performance when the different delays and system disturbances are compensated.

Online parameter estimations were developed and implemented successfully to account for parameter modifications. They are affected by the non-linearities that were compensated in Chapter 6. However at higher speeds as the number of zero-crossing increases non-linearity compensations become worse and parameter determination can become difficult. A proposal for future work could be to develop better compensations. One solution proposed would be to predict the currents at the instants in which the PWM changes its state (not at the beginning/end of the period) as this may play a bit role as a source of the errors. A better non-linearity compensator would improve the overall performance and also make online parameter identification easier.

BIBLIOGRAPHY

[1] R Krishnan. *Permanent Magnet Synchronous and Brushless DC Motor Drives*. 2009.

[2] P. Cortes, M.P. Kazmierkowski, R.M. Kennel, D.E. Quevedo, and J. Rodriguez. Predictive control in power electronics and drives. *Industrial Electronics, IEEE Transactions on*, 55(12):4312–4324, Dec 2008. ISSN 0278-0046. doi: 10.1109/TIE.2008.2007480.

[3] Jesper Moos. Predictive deadbeat control for pmsm drive. Master's thesis, Aalborg Universitet, 2014.

[4] Sonny Quillo Judit Banos Garcia. Predictive controller for pmsm drive. Master's thesis, Aalborg Universitet, 2013.

[5] Kaiyuan Lu. Dynamic modeling of electrical machines: A first step to motor modeling, 2015.

[6] Donghua Pan, Xinbo Ruan, Chenlei Bao, Weiwei Li, and Xuehua Wang. Capacitor-current-feedback active damping with reduced computation delay for improving robustness of lcl-type grid-connected inverter. *Power Electronics, IEEE Transactions on*, 29(7):3414–3427, July 2014. ISSN 0885-8993. doi: 10.1109/TPEL.2013.2279206.

[7] Xiao Xi Wang Weihua. Research on predictive control for pmsm based on online parameter identification. *Electric Machines & Drives Conference (IEMDC), IEEE International*, pages 1249–1254, 2013.

[8] S.G. Parker, B.P. McGrath, and D.G. Holmes. Regions of active damping control for lcl filters. In *Energy Conversion Congress and Exposition (ECCE), 2012 IEEE*, pages 53–60, Sept 2012. doi: 10.1109/ECCE.2012.6342412.

[9] N. Urasaki, T. Senjyu, K. Uezato, and T. Funabashi. Adaptive dead-time compensation strategy for permanent magnet synchronous motor drive. *Energy Conversion, IEEE Transactions on*, 22(2):271–280, June 2007. ISSN 0885-8969. doi: 10.1109/TEC.2006.875469.

[10] Zhendong Zhang and Longya Xu. Dead-time compensation of inverters considering snubber and parasitic capacitance. *Power Electronics, IEEE Transactions on*, 29(6): 3179–3187, June 2014. ISSN 0885-8993. doi: 10.1109/TPEL.2013.2275551.

[11] Laszlo Mathe. Control of grid connected photovoltaic and wind turbine systems: Grid synchronization in single-phase and three-phase power converters, 2015.

[12] S.J. Underwood and I. Husain. Online parameter estimation and adaptive control of permanent-magnet synchronous machines. *Industrial Electronics, IEEE Transactions on*, 57(7):2435–2443, July 2010. ISSN 0278-0046. doi: 10.1109/TIE.2009.2036029.

[13] Henrik Neugebauer. Parameter identification of a permanent magnet synchronous motor. Master's thesis, Chalmers University of Technology, 2012.

[14] Mbpi - igbtmodel identification tool. URL http://www.igbtmodel.org/index.php?title=Main_Page.

[15] Infineon. Fp75r12kt3 technical information.

[16] Determination of inverter nonlinear characteristics for voltage error compensation used in sensorless drives. Giorgos Tsolaridis, Vasilios Dimitris Karaventzas, Dipen Dalal, Lorenzo Ceccarelli, 2015.

[17] Kaiyuan Lu. Dynamic modeling of electrical machines: Reference frame theory, 2015.

Appendices

A

OFFLINE PARAMETER DETERMINATION

In this appendix it is described the methods used to determine different system parameters offline.

A.1 RESISTOR ESTIMATION

The machine resistor plus all the parasitic resistors present in the wires connecting it to the converter can be estimated with a DC test. By inspecting Equation 2.24 it can be seen that if the machine is still and the $dq0$ current is constant the machine equations reduce to Equation A.1.

$$u_{qd0} = Ri_{qd0} \tag{A.1}$$

On the other hand from Equation 2.31 it can be seen that if $i_q = 0$ there will be no electrical torque and therefore if no other external torque is applied the machine will remain still. Therefore i_q can be set to zero and i_d to an arbitrary value and the resistor can be estimated as v_d/i_d. To eliminate non-linearity effects from the converter two points can be used and the two equations subtracted to remove them and therefore the resistor can be estimated using Equation A.2.

$$R = \frac{v_d(2) - v_d(1)}{i_d(2) - i_d(1)} \tag{A.2}$$

If the current does not change the sign most of the non-linearities will remain the same and will be canceled out. In this test the PI is used since the predictive-controller requires to know the machine parameters beforehand. With the PI then the average v_d and i_d is recorded and the data is shown in Table A.1 also for different temperatures as given by the sensor placed in the setup.

$< v_d >$ (V)	$< i_d >$ (A)	T (deg)
10.80	15	24
14.45	30	24
10.84	15	30
14.58	30	30

Table A.1.: Voltage needed in d axis for given i_d and no i_q with no speed

The resistor is estimated as $R(24deg) = 0.2433\Omega$ and $R(30deg) = 0.2493\Omega$. As predicted the resistor increases with temperature but it can be seen that the change is

quite small. On the other hand the resistor change as a function of temperature can be approximated as Equation A.3.

$$R(T) = R(T_0)\,(1 + \alpha(T - T_0)) \tag{A.3}$$

The value of α for copper is $3.9e-3$ and placing $T_0 = 24deg$ and $R(T_0) = R(24deg)$ the value for 30 degrees is estimated as 0.2510Ω. It must be noted that the temperature sensor has an uncertainty of $+-1deg$ however. It must also be noted that part of the resistor is inside the machine and another part is outside so only the inside one will be heated and a more valid formula would be Equation A.4.

$$R(T) = R_{wire} + R_{machine}(T_0)\,(1 + \alpha(T - T_0)) \tag{A.4}$$

A.2 ELECTRICAL MACHINE PARAMETERS AND NON-LINEARITIES

The electrical machine parameters $R, L_q, L_d, \lambda_{mpm}$ can be determined offline with a fitting application developed for a student job called MBPI which can be found in [14]. The idea is that if a set of experiments are performed and are replicated under simulation then the parameters of a system can be estimated by modifying them with a tool that minimizes the error between the simulation and the experiments as is illustrated in Figure A.1. Ideally, if the model is good in the true parameters there will be a global minimum. On reality the global minimum may be shifted slightly if the model is not accurate.

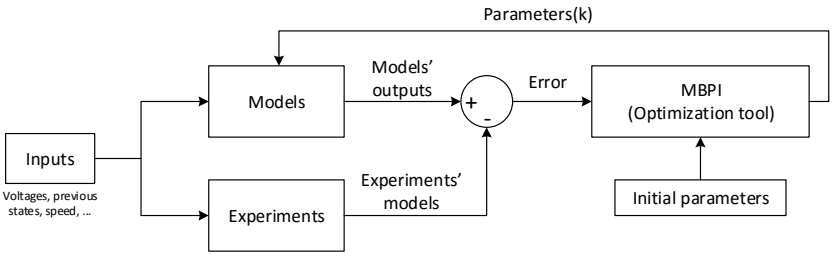


Figure A.1.: Offline parameter determination with MBPI optimization tool

Another consideration to be made about this method is that depending of the algorithm used global convergence is not guaranteed. This means that the tool may find a set of parameters that minimize the error between experiment and simulation but to a minimum which is not the smallest one. There is a trade-off between optimization speed and accuracy of the methods and while there are some methods that can guarantee global convergence from a theoretical point of view the time needed for the convergence would probably make them not feasible. For this reason in this solution fmincon which uses gradient methods is used which will typically converge to the closest local minimum. If the results obtained however seem adequate (the final error is small) they can be considered valid.

The method proposed is as follows. The resistor has been estimated in Section A.1 by canceling out most of the non-linearities so the result can be considered adequate but

can be refined with this method by only allowing a small change. Then there are three extra machine parameters to determine. At any speed the currents can be perturbed randomly to obtain different points to use in predictions. The following variables are recorded in the experiment: $u_q, u_d, i_q, i_d, \omega_e$ as those 5 are all the variables that play a role in the electrical part. Then the current in the next period can be predicted using a model where the parameters are modified with MBPI.

Those recorded variables are placed into the model using repeating sequence stair block in Simulink and the simulation is set to fixed-time with the same step, T_s, used in the experiment. If the simulation is run then those variables are updated each T_s providing the value recorded in the experiment. Using Equation 4.14 - 4.15 the predicted current can be solved which is equivalent to the reference on those formulas. The previous current, voltage and speed are the ones from the experiment given by the repeating sequence stair block. This is illustrated in Figure A.2. This predicted current in the next period can then be compared with the measured one to create an error function to minimize. The comparison between simulation and experiment data as well as the parameter variation following an optimization method is done automatically in MBPI.

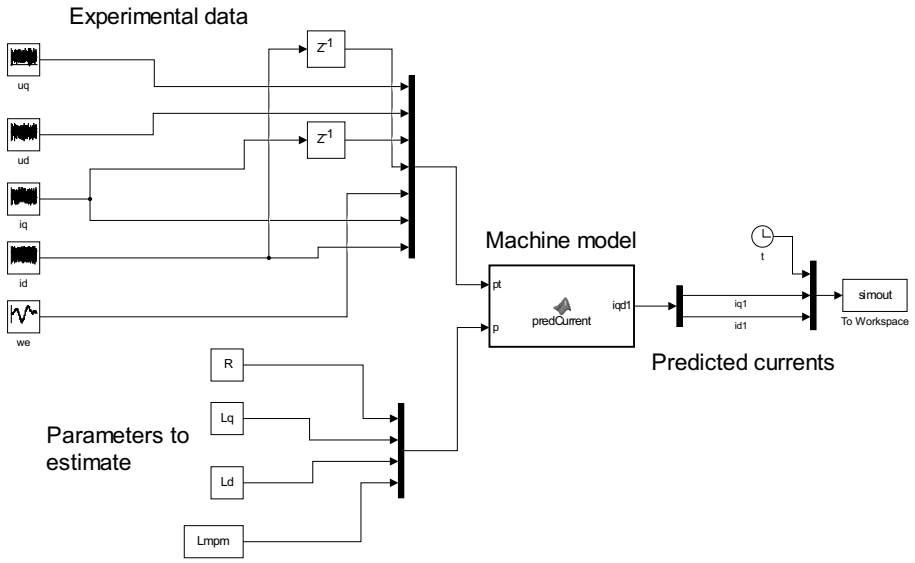


Figure A.2.: Machine side model used in MBPI for parameter estimation

Since the non-linearities may influence the result the compensator proposed is also added to the offline model and the non-linearities parameters are also estimated with MBPI. This part of the model is depicted in Figure A.3. The non-linearity block contains the code in Code D.5.

As a gradient method is used a set of initial parameters is needed. The initial values used in the estimations for the machine parameters are close to the ones seen in previous work in the same machine but still far away to show the validity of the solution. As for the non-linearities educated guesses about the diode and igbt voltage and resistive part can be made. From the datasheet in [15] which corresponds to the module used in the setup, for the igbt v_{ON} is around $0.5 - 1V$ and R_{ON} lies around 0.02Ω as can be estimated from the $I_C = f(V_{CE})$ graph. As for the snubber the initial value used in [16]

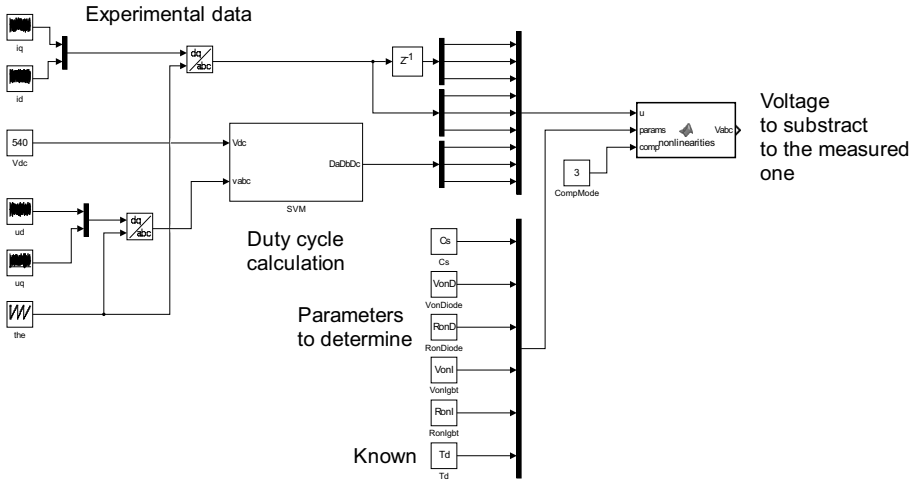


Figure A.3.: Non-linearity side model used in MBPI for parameter estimation

has been used. For that value the effect in the voltage drop can be neglected as it can be estimated in the mV range using Equation 6.6 with the currents used in the experiment so it will not play a role in the results. Finally T_d is known and it is $2.5\mu s$.

Parameter	Initial Value	Final value
$R(\Omega)$	0.24	0.258
$L_q(mH)$	1	2.151
$L_d(mH)$	1	2.033
$\lambda_{mpm}(Wb)$	0.1	0.1204
$C_s(nF)$	1	1.267
$V_{on}(IGBT)(V)$	0.5	1.241
$R_{on}(IGBT)(\Omega)$	0.01	0.0336
$V_{on}(Diode)(V)$	0.5	1.239
$R_{on}(Diode)(\Omega)$	0.01	0.0336

Table A.2.: Initial and final parameters as estimated by MBPI

The results of the iterative process are depicted in Table A.2 and the error has dropped from 46% to 3.3%. The evolution of the parameters as well as the error function is shown in Figure A.4. If those parameters are used in the setup in the predictive controller and the proposed non-linearity compensator the average error in the current is then expected to be similar and therefore the values obtained can be considered valid.

Finally even though not much can be seen in the graph the estimation of the currents vs the real one over time is presented in Figure A.5. It can be seen great concordance between experiment and prediction. The point by point error is also added in the lower part of the graph.

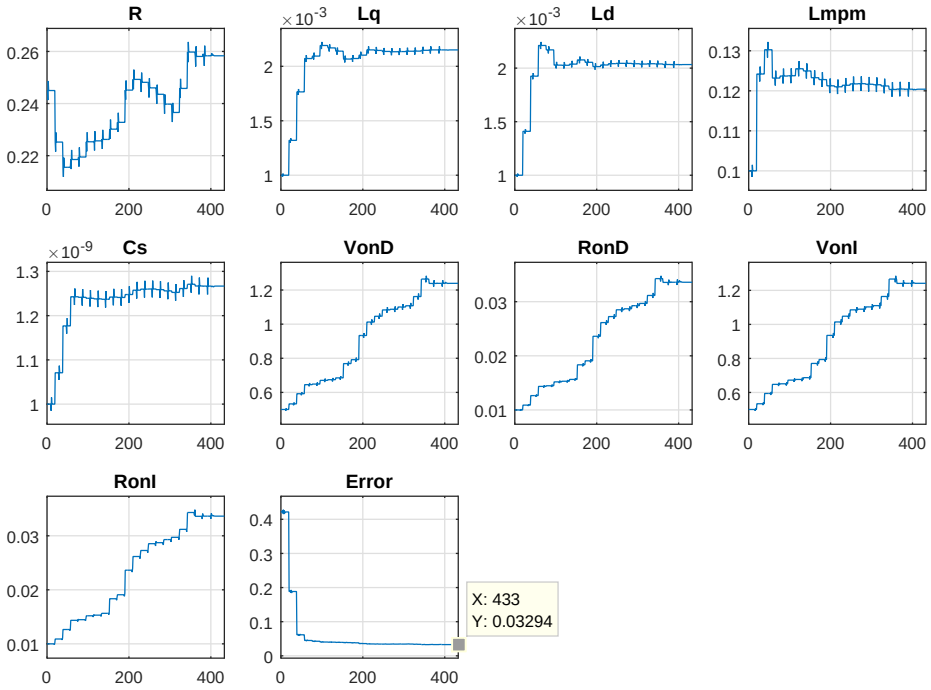


Figure A.4.: Parameter estimation evolution in MBPI

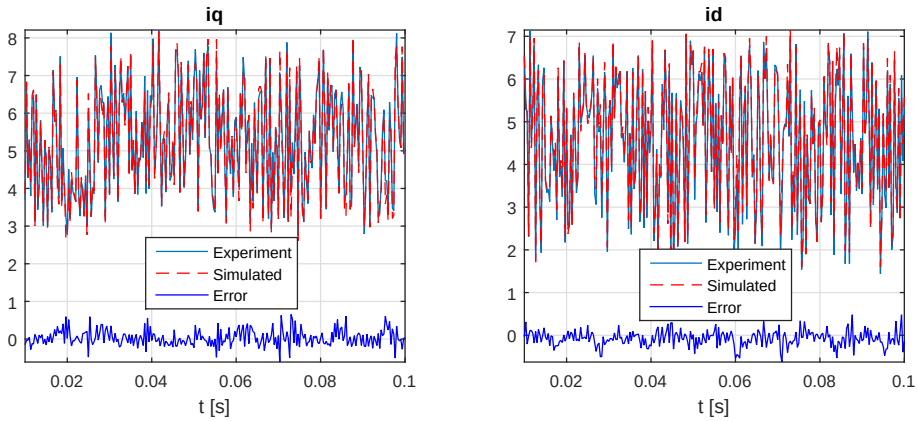


Figure A.5.: Parameter estimation evolution in MBPI

A.3 MECHANICAL PARAMETERS

The mechanical parameters J, J_0 and B can be determined with the following procedure which was also conducted in [3]. The PMSM is set running in steady-state without load (induction machine is plugged off) and then the voltage applied is stopped. The machine will stop on its own and the evolution of speed vs time can be used to determine those parameters since it follows Equation A.5.

$$-J_0 = J\frac{d}{dt}\omega_{mec}(t) + B\omega_{mec}(t) \tag{A.5}$$

If the equation is solved for a given initial speed $\omega_{mec}(0)$ the evolution of speed over time $\omega_{mec}(t)$ is given by Equation A.6.

$$\omega_{mec}(t) = -\frac{J_0}{B} + \left(\frac{J_0}{B} + \omega_{mec}(0)\right) e^{-\frac{B}{J_m}t} \tag{A.6}$$

It can be seen that using this technique only two parameters can be obtained since there are infinite combination of parameters that yield the same response. As only two different coefficients between parameters appear in the solution and there are three parameters setting one parameter arbitrary will always produce a valid solution. Therefore another equation needs to be added.

In steady-state, without load the equation governing the speed is Equation A.7. The value of T_{el} can be calculated using the formula given in Equation 2.31. Therefore the value of i_q and i_d can be recorded when the speed is set constant at two different references and using the machine parameters estimated the torque can be calculated and J_0 and B solved.

$$T_{el} - J_0 = B\omega_{mec}(t) \tag{A.7}$$

The parameters J_0 and B can then be calculated by solving Equation A.7 using 2 points and the solution is given by Equation A.8 - A.9.

$$J_0 = \frac{T_{el}(1)\omega_{mec}(2) - T_{el}(2)\omega_{mec}(1)}{\omega_{mec}(2) - \omega_{mec}(1)} \tag{A.8}$$

$$B = \frac{T_{el}(1) - T_{el}(2)}{\omega_{mec}(1) - \omega_{mec}(2)} \tag{A.9}$$

Two tests at 500 and 1000 rpm have been conducted. The machine is set running in steady-state for a few seconds and then the average value of i_d and i_q is calculated. The results are depicted in Table A.3 and using those points in Equation A.8 - A.9 it is found $J = 0.1941$ Nm and $B = 4.56e-4$ Nms.

$<\omega_{mec}>$ (rpm)	$<i_q>$ (A)	$<i_d>$ (A)	T_{el} (Nm)
500	0.310	0.056	0.218
1000	0.344	0.139	0.242

Table A.3.: Steady-state currents and estimated torque at different speeds

Finally the value of J_m can be estimated using Equation A.6 in a fitting application as J_0 and B are now known and it can be solved. For simplicity MBPI has been used again since it makes easy to extract parameters by providing a toolbox which uses experimental data to tune parameters in simulations and both continuous and discrete models can be used with experiment data. The Simulink model provided to MBPI is shown in Figure A.6.

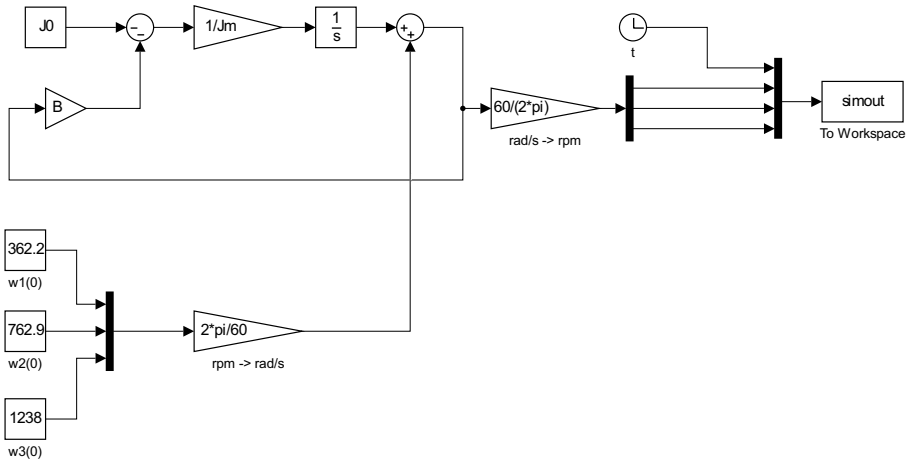


Figure A.6.: Simulink model of mechanical behavior without load for MBPI

The model has the parameter J_m as a variable parameter that MBPI can vary. By running the simulation with different J_m the speed over time can be estimated and saved in the workpsace with the simout block. Then it is compared with the experimental data. The tool will modify the parameter(s) and find a minimum. Convergence to global minimum is not guaranteed but as long as the results are adequate it can be concluded that the value estimated is valid. The fmincon function is used to estimate J_m in MBPI since it provides the fastest convergence.

It is important to synchronize times and initial speeds with experimental data. The times of each experiment are shifted so the initial time is always zero and this speed is used in the initial speed in the Simulink model. Once this is carefully done the parameter J_m can be determined automatically in MBPI by setting it running until it stops automatically the iterative process.

Three experiments have been performed starting at 500, 1000 and 1500 rpm. Only the first two are used and MBPI finds $J_m = 0.1128 Nms^2$ and the final error is 2.8% showing a good concordance between model and experiment. The third result can be used then with the parameters obtained to validate them. The error for the new curve starting at $1500 rpm$ that has not been used in the fitting is 1.9% showing the validity of the results obtained. The comparison between experiment and simulation is depicted in Figure A.7.

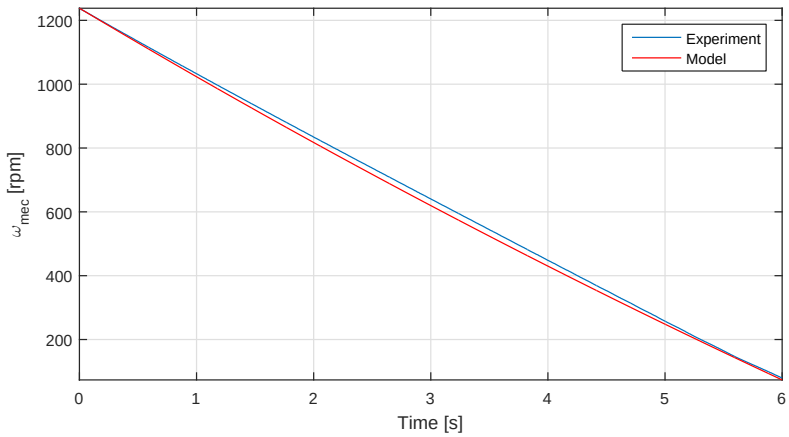


Figure A.7.: Comparison between speed change in experiment and model using the estimated mechanical parameters

B

DERIVATIONS

In this appendix some derivations are attached to some of the calculations or approximations done.

B.1 REFERENCE FRAME DERIVATION

Equations of the PMSM were rewritten in a $qd0$ reference frame since they become simpler. On one hand the position-variant inductor and permanent-magnet is made constant and on the other the AC control used is transformed into DC simplifying the control. A basic derivation of the formulas used to transform coordinates is presented here based on the projection method used in [17].

In many power electronics applications there are three phases that are typically balanced. In a PMSM this three phases combined produce a rotating flux that moves the machine. If the three phases are sinusoidal and separated 120 degrees then they will produce a constant flux that rotates at the speed of the sinusoidal (in one pole-pair machine and a multiple of N_{pp} with more pair-poles). If the system is balanced then there are only two independent variables as one of the phases can always be obtained with the other two. Under this condition then it is possible to create a new coordinate system that rotates with the flux and only uses two variables (the flux projections) and is able to represent the 3 phases without any loss of information. If the phases are not balanced then a third coordinate will be needed as will be evidenced in the derivation. The advantage of a rotating coordinate system fixed in the flux is that any pure sinusoidal variable that rotates in the same direction and speed of the coordinate system (independent of the dephase) will be transformed into DC and this can simplify the equations and control.

The reference frame theory is based in the projection of one coordinate system into another. A complex vector can be rotated by multiplying it by $e^{i\theta}$ which will rotate it θ. This is based in Euler's formula $e^{j\theta} = \cos\theta + i\sin\theta$ which can be obtained by derivating $f(\theta) = e^{-i\theta}(\cos t + i\sin t)$ and showing $f'(\theta)$ is 0 for all θ as the terms are identical. Therefore the original $f(\theta)$ must be a constant. Placing $\theta = 0$ for example in $f(\theta)$ yields 1 and Euler's formula is derived by solving $e^{i\theta}$. With this formula a vector represented in complex notation can be transformed into an exponential with $j\theta$ as the argument. Based on exponential rules multiplying two exponentials yields a new one with the sum of the arguments and this is equivalent to rotating the vector as when Euler's formula is used in the result it can be seen that the vector has been rotated.

The projection of a $2D$ vector represented with imaginary coordinates $\vec{u}$ into $\vec{v}$ can be done as follows. From the image in Figure B.1 it can be seen that the projection of a vector into another will not change if both vectors are rotated the same angle. Then both vectors can be rotated $-\theta_v$ so $\vec{v}$ is placed in the real axis. Then it can be seen that the

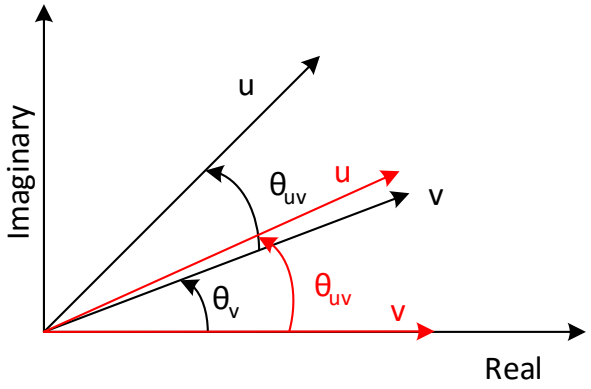


Figure B.1.: Real part as projection of u in v

real component of $\vec{u}$ is the projection as the real axis is place in $\vec{v}$. Mathematically this is shown in Equation B.1.

$$proj(\vec{u})_{\vec{v}} = Real\left(\vec{u}e^{-i\theta_{uv}}\right) \tag{B.1}$$

$$Real\left(\vec{u}e^{-i\theta_v}\right) = \frac{1}{2}\left(\vec{u}e^{-i\theta_v} + \vec{u}^* e^{i\theta_v}\right) = proj(\vec{u})_{\vec{v}} \tag{B.2}$$

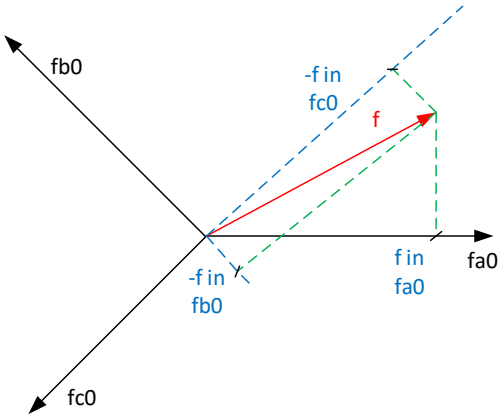


Figure B.2.: Projection of f in the f_{a0}, f_{b0}, f_{c0} axis

It may be seen the relation Equation B.2. Based on those projections formulas a new rotating vector can be created and later used to project into the new reference frame $qd0$. Let f_{a0}, f_{b0}, f_{c0} be three variables of a balanced system of which axes are represented in Figure B.2, then each component in each phase f_{a0}, f_{b0}, f_{c0} can be obtained by projecting the new rotating vector f into each axes. This is shown in Equation B.3 - B.5. Each equation can be multiplied in both sides by $e^{-i\theta}$ being θ the angle of each phase. Then

both three equations can be summed and simplified to obtain the special vector $\vec{f}$ in Equation B.6.

$$f_{a0} = \frac{1}{2}\left(\vec{f}e^{-i0} + \vec{f}^{*}e^{i0}\right) \tag{B.3}$$

$$f_{b} = \frac{1}{2}\left(\vec{f}e^{-i2\pi/3} + \vec{f}^{*}e^{i2\pi/3}\right) \tag{B.4}$$

$$f_{c} = \frac{1}{2}\left(\vec{f}e^{i2\pi/3} + \vec{f}^{*}e^{-i2\pi/3}\right) \tag{B.5}$$

$$\vec{f} = \frac{2}{3}\left(f_{a0}e^{j0} + f_{b0}e^{j2\pi/3} + f_{c0}e^{-j2\pi/3}\right) \tag{B.6}$$

It is important to note that f_{a0}, f_{b0}, f_{c0} must be balanced, that is $f_{a0} + f_{b0} + f_{c0} = 0$. If not this derivation is not valid as if the equations Equation B.3 - B.5 are summed on the left side zero is obtained and on the right it is $f_{a0} + f_{b0} + f_{c0}$ which then must be zero meaning f_{a0}, f_{b0}, f_{c0} must be balanced. This is logical as it is impossible to use a 2 independent variables to represent 3 independent variables. If the system is not balanced with f_a, f_b, f_c then the zero component $f_0 = 1/3(f_a + f_b + f_c)$ can be removed from each phase creating $f_{i0} = fi - f_0$. When this is done the sum of Equation B.3 - B.5 produces zero in both sides so it is valid and the new component f_0 is created as the third variable needed.

$$\vec{f} = fe^{j\omega t} \tag{B.7}$$

Three cosines inputs displaced $2\pi/3$ can be placed as f_{a0}, f_{b0}, f_{c0} in Equation B.6. The formula for f_{k0} can be $1/2(e^{i(\theta - 2\pi/3(k-1))} + e^{-i(\theta - 2\pi/3(k-1))})$. This can be operated and the result is given by Equation B.7. This results shows that if three sinusoidal phases delayed $2\pi/3$ are applied then $\vec{f}$ is a rotating vector. Therefore a new coordinate system can be created following $\vec{f}$ which will transform it to a constant value.

B.1.1 *Abc to qdo*

Once the vector in Equation B.6 is obtained the transformation from *abc* to $qd0$ can be made by projecting $\vec{f}$ into the new coordinate system rotating in the same speed and direction of *abc*. The q axis can be placed $\pi/2$ before d and the angle between the real axis and q denoted by θ_q. Then in complex form the position of the q axis is given by $e^{j\theta_q}$ and the position of the d by $e^{j(\theta_q - \pi/2)}$. Then to obtain the qd components from $\vec{f}$ it can be projected into those axis using Equation B.2 and the result is given in Equation B.8 and B.9 where $\theta_d = \theta_q - \pi/2$ and $\cos(x - \pi/2) = \sin(x)$ and the zero component f_0 is the same in every coordinate system.

$$f_q = Real(\vec{f}e^{-j\theta_q}) = \frac{2}{3}\left(f_{a0}e^{j(0-\theta_q)} + f_{b0}e^{j(2\pi/3-\theta_q)} + f_{c0}e^{j(-2\pi/3-\theta_q)}\right) = \frac{2}{3}\left(\cos(\theta_q) + \cos(\theta_q - \frac{2\pi}{3}) + \cos(\theta_q + \frac{2\pi}{3})\right) \tag{B.8}$$

$$f_d = Real(\vec{f}e^{-j\theta_d}) = \frac{2}{3}\left(f_{a0}e^{j(0-\theta_d)} + f_{b0}e^{j(2\pi/3-\theta_d)} + f_{c0}e^{j(-2\pi/3-\theta_d)}\right) = \frac{2}{3}\left(\sin(\theta_q) + \sin(\theta_q - \frac{2\pi}{3}) + \sin(\theta_q + \frac{2\pi}{3})\right) \quad \text{(B.9)}$$

This can be represented by a matrix as K_{abc}^{qd0} and the transformation matrix from $qd0$ to abc can be done by using the projection method again or by operating $K_{qd0}^{abc} = (K_{abc}^{qd0})^{-1}$ as since $f_{qd0} = K_{abc}^{qd0} f_{abc}$ it can be seen then $f_{abc} = (K_{abc}^{qd0})^{-1} f_{qd0} = K_{qd0}^{abc} f_{qd0}$.

B.2 TAYLOR DERIVATION TO APPROXIMATE STEADY-STATE ERRORS

The graph in Figure 5.10 offers some information for a typical point but results may change abruptly at any other. The first terms of the Taylor expansion of a function $f(x_1, x_2, ..., x_n)$ is given by Equation B.10.

$$f(x = x_1, x_2, ..., x_n) \equiv f(x(0)) + \sum_{i=1}^{n} \frac{d}{dx_i} f(x(0))\Delta x_i \quad \text{(B.10)}$$

In this problem there are two functions, the prediction of the final value of the current in the q axis and in the d denoted as i'_q and i'_d. The variables are four, each parameter. For each function a taylor expansion can be done. For the q axis the Taylor series will look like Equation B.11 and for the d as Equation B.12.

$$i'_q \approx i^*_q + \sum_{i=1}^{4} \frac{d}{dp_i} i'_q(p(0))\Delta p_i \quad \text{(B.11)}$$

$$i'_d \approx i^*_d + \sum_{i=1}^{4} \frac{d}{dp_i} i'_d(p(0))\Delta p_i \quad \text{(B.12)}$$

The coefficients of the derivatives can be calculated analytically be differentiating Equation 5.14 which respect to each parameter and evaluating at the point where the controller and system parameters are the same. The initial Taylor expansion is still too complicated and may be reduced even more by removing terms with a small weight in the equation.

It can be seen that the denominator is common in all the derivatives, both in the q and d axis and looks like Equation B.13.

$$den_t = d_1 + d_2 + d_3 + d_4 + d_5 = R^2T_s^2 + 4L_dL_q + 2L_dRT_s + 2L_qRT_s + L_dL_qT_s^2\omega_e^2 \quad \text{(B.13)}$$

If the values from Section 2.4 are substituted it can be seen that d_1 is less than 1000 times than d_2. For d_3 and d_4 this number is close to 100 times less. Finally d_5 varies with speed and at nominal speed is about 10 times less than d_2. For this reason the denominator may be simplified as d_2 alone although at high speeds d_5 may be taken into account.

The numerator of each derivative may also be simplified by removing the term RT_s present in the resistor, flux and coupling terms as it is about 10 times less than the other term $2L_i$. Moreover the resistive term due to i_q in the d axis and due to q in the d axis may also be removed. It is speed dependent but less than 10 times less at nominal speed. Under this 2 assumptions the final equations can be reduced as shown in Equation B.14 in matrix form:

$$i'_{qd} \approx i^*_{qd} + K_{ss} \begin{bmatrix} \Delta R \\ \Delta L_q \\ \Delta L_d \\ \Delta \lambda_{mpm} \end{bmatrix} \tag{B.14}$$

The matrix K_{ss} takes the expression depicted in Equation B.15 under the simplifications stated. Under the parameters stated in Section 2.4 then K_{ss} may be evaluated as Equation B.16.

$$K_{ss} = \begin{bmatrix} -\frac{T_s}{L_q} i^*_q & -\frac{{T_s}^2 i^*_q {\omega_e}^2}{2 L_q} & -\frac{T_s i^*_d \omega_e}{L_q} & -\frac{T_s \omega_e}{L_q} \\ -\frac{T_s}{L_d} i^*_d & \frac{T_s i^*_q \omega_e}{L_d} & -\frac{{T_s}^2 i^*_d {\omega_e}^2}{2 L_d} & -\frac{{T_s}^2 {\omega_e}^2}{2 L_d} \end{bmatrix} \tag{B.15}$$

$$K_{ss} = \begin{bmatrix} -0.155 i^*_q & -2.58 \times 10^{-5} \omega_e^2 i^*_q & -0.155 \omega_e i^*_d & -0.155 \omega_e \\ -0.1642 i^*_d & 0.1642 \omega_e i^*_q & -2.75 \times 10^{-5} \omega_e^2 i^*_d & -2.75 \times 10^{-5} \omega_e^2 \end{bmatrix} \tag{B.16}$$

The matrix K_{ss} evaluated may also be scaled by each parameter it is affected by as $K'_{ss} = K_{ss}/[p_0; p_0]$ where $p_o = [R_0, L_{q0}, L_{d0}, \lambda_{mpm0}]$ are the initial estimated values from Section 2.4. The new values are shown in Equation B.17. This way the contribution of each element to the global error can be seen better as the difference in the parameters is now in relative terms and K'_{ss} may be interpreted as the current error due to a change in each parameter relative to the initial estimation.

$$K'_{ss} = \begin{bmatrix} -0.039 i^*_q & -5.55 \times 10^{-8} \omega_e^2 i^*_q & -3.15 \times 1^{-4} \omega_e i^*_d & -0.0186 \omega_e \\ -0.041 i^*_d & 3.53 \times 10^{-4} \omega_e i^*_q & -5.58 \times 10^{-8} \omega_e^2 i^*_d & -3.30 \times 10^{-6} \omega_e^2 \end{bmatrix} \tag{B.17}$$

The approximation derived can be validated as follows. Assuming the derivation in Equation 5.14 is correct then a set of working conditions can be generated at random for cases that could happen in the experiment. Also the frequency f_s is varied to show the validity of the formulas if different switching frequencies are used. Then both formulas are used and the error estimated by both compared. A total of 1000 random points are generated following an uniform distribution with the values in Table B.1.

The results for the approximation used are depicted in Figure B.3. On blue it is presented the equations as shown in Equation B.15. Because some terms presenting the speed are only important at speeds close to nominal for this machine in red it is shown the approximation where d_5 in Equation B.13 is considered by multiplying K_{ss} by $\frac{d_2}{d_2+d_5}$. On the left the speed has been randomized up to half the nominal speed so the approximation without this term is still good and on the right it can be seen the error may become important at certain conditions (with high speed and small parameter changes).

Parameter	Min value	Max value
R	R_0	$1.5R_0$
$L_q(mH)$	$0.5L_{q0}$	L_{q0}
$L_d(mH)$	$0.5L_{d0}$	L_{q0}
$\lambda_{mpm}(Wb)$	$0.85\lambda_{mpm0}$	$1.15\lambda_{mpm0}$
f_s	$0.5f_{s0}$	$2f_{s0}$
$i_q^*(A)$	5	15
$i_d^*(A)$	-5	5

Table B.1.: Random parameters variation for Taylor approximation validation

It can be seen that in general the results of the approximation can be considered an adequate way to show and understand the importance of each term in the steady-state error in any condition as the error is usually kept below 5%.

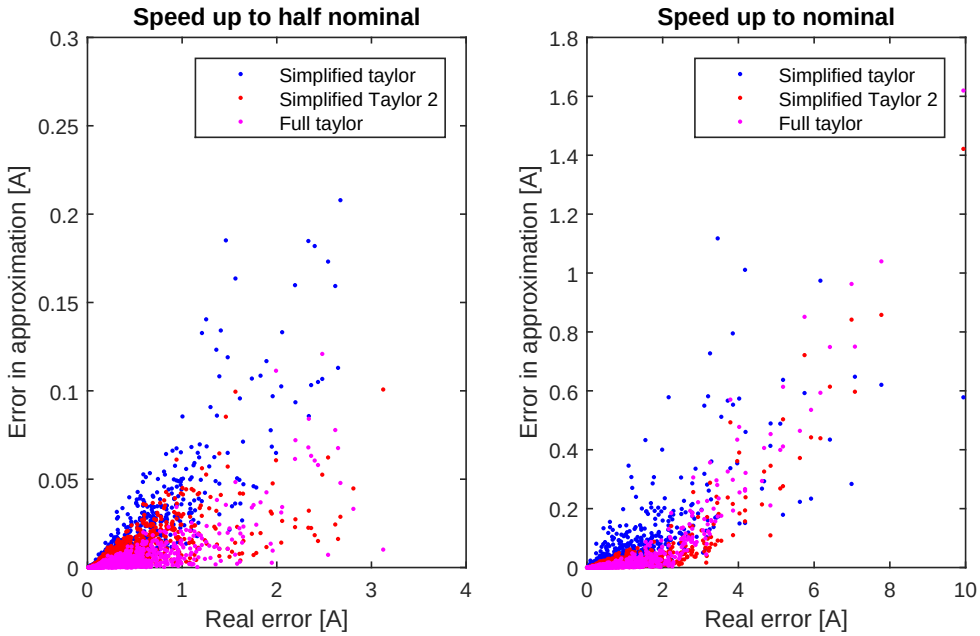


Figure B.3.: Taylor approximation vs real formulas error for different random points

As the errors due to big changes in the parameters are produced the error due to the taylor expansion is increased but the proportion seems constant so the approximation may be used to understand the contribution of each parameter to the steady-state error in the predictive controller.

B.3 DC PERTURBATIONS INDEPENDENCE IN THE OVERSHOOT

An interesting observation seen in simulations is that DC perturbations (understanding the steady-state error before and after is constant and their voltage contribution also) did not seem to affect the overshoot. Moreover the formula Equation 5.19 initially derived

to estimate the overshoot when only one inductor is varied seemed to work in many more conditions such as when there is high speeds, with errors in the flux and also with errors in the other inductor axis with good accuracy if the DC condition is met. This two phenomenas can be proved. They will be proved in the q axis as it will be evident the same can be conducted without any change in the d.

The machine equation in the q axis equating controller and system in the transient with some modifications is shown in Equation B.18. This formula is valid for the 1-period controller at any state and for the 2-period controller if the system was in steady-state prior to the step. The value L_q is the real inductor and L'_q the estimate from the controller. The term δ_v represents the voltage-error due to the DC terms. The current in the next period is i'_q and the reference i^*_q while i_q is the current at the begging of the period.

$$\frac{L_q}{T_s}(i'_q - i_q) = \frac{L'_q}{T_s}(i^*_q - i_q) + \delta_v \tag{B.18}$$

From Figure 5.12 the following can be observed. If i^*_d is not changed the current error due to L_d in the q axis will be constant before and after the step. If i_d does not change then the voltage error will also be constant. Since the speed changes slowly compared with the current, flux errors will also produce constant current and voltage errors in i_q before and after the step. Finally errors due to L_q in the steady-state current in i_q are typically small and will be also speed-dependent. The resistor is neglected.

From simulations and experiments it can be seen there is a coupling effect between axes. When i^*_q is changed then i_d changes but the changes are much smaller than the changes in i_q which is the current of which the reference is modified. If there are errors in L_q then the steady-state value of i_d will change when i^*_q is changed too. The changes are speed dependent.

Therefore it can be seen that if the resistor contribution is neglected when i^*_q is changed in the q axis voltage equation, Equation B.18 is quite accurate. The only source of the error are the changes in d due to coupling effects. Let δ_i be the current error in the q axis. The value of δ_i is approximately constant for the steady-state before and after the step as taken from Figure 5.12 since the speed is approximately the same, i^*_d has not been changed and L_q errors do not affect that much the steady-state errors in i_q.

The step response of the system can be described in Figure B.4 with the approximations described. The overshoot of the system is given by $\frac{y_p - y_f}{y_f - y_0}$. Based on the graph in Figure B.4 some substitutions can be placed in Equation B.18. The objective is to remove the currents i_q, i^*_q, i'_q and use the variables y_p, y_f, y_0 so the overshoot can be estimated. The formula may be rewritten as Equation B.19 where $i'_q = y_p$, $i_q = y_0 = y_f - s$ and $i^*_q = y'_f \approx y_f + \delta'_i$. It must be noted from the previous paragraphs that $\delta'_i \approx \delta_i$ and $s \approx s'$. Then Equation B.18 may be rewritten following:

$$\frac{L_q}{T_s}(y_p - y_0) = \frac{L'_q}{T_s}(y'_f - y_0) + \delta_v \tag{B.19}$$

$$\frac{L_q}{T_s}(y_p - (y_f - s)) = \frac{L'_q}{T_s}((y_f + \delta_i) - y_0) + \delta_v \tag{B.20}$$

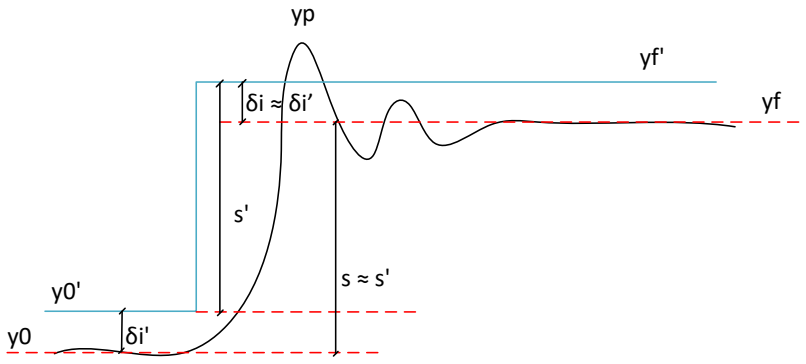


Figure B.4.: Overshoot variables and approximations

$$\frac{L_q}{T_s}(y_p - y_f + s) = \frac{L'_q}{T_s}(y_f - y_0 + \delta_i) + \delta_v \tag{B.21}$$

$$\frac{L_q}{T_s}\left((y_p - y_f) + (y_f - y_0)\right) = \frac{L'_q}{T_s}\left((y_f - y_0) + \delta_i\right) + \delta_v \tag{B.22}$$

$$\frac{L_q}{T_s}\left(\frac{y_p - y_f}{y_f - y_0} + 1\right) = \frac{L'_q}{T_s}\left(1 + \frac{\delta_i}{y_f - y_0}\right) + \frac{\delta_v}{y_f - y_0} \tag{B.23}$$

$$\Phi = \frac{y_p - y_f}{y_f - y_0} = \frac{L'_q}{L_q}\left(1 + \frac{\delta_i}{s}\right) + \left(\frac{T_s\delta_v}{L_q(y_f - y_0)} - 1\right) \tag{B.24}$$

$$\Phi = \frac{L'_q}{L_q} - 1 + \frac{1}{s}\left(\delta_i\frac{L'_q}{L_q} + \frac{T_s\delta_v}{L_q}\right) \tag{B.25}$$

It can be seen that if there are no steady-state errors then the formula derived in Equation B.25 is directly Equation 5.19. The formula may be rewritten as Equation B.26.

$$\Phi = \frac{L'_q}{L_q} - 1 + \frac{T_s}{sL_q}\left(\frac{\delta_i L'_q}{T_s} + \delta_v\right) \tag{B.26}$$

It may be observed that if the DC perturbations are constant then it must be true that $\frac{\delta_i L'_q}{T_s} + \delta_v = 0$ since $\delta_i \approx \delta'_i$ and δ_v is also constant. In steady-state the system does not have the differential term unlike the controller. In δ_v it is recorded the voltage difference between system and controller without the differential term of the controller. In steady-state the differential term of the controller before was given by $L'_q/T_s\delta'_i$ as $\delta'_i = i^*_q - i_q$. Therefore it can be concluded that any dc error does not influence the overshoot and the overshoot (neglecting coupling errors in the transient which make it speed-dependent) is function of L_q/L'_q.

B.4 PROOF OF CONVERGENCE OF GRADIENT METHOD IN PMSM

The cost function used in this solution is the sum of the absolute value of the error between applied voltage and estimated voltage using the model. Others choice could be taken. For instance another common cost function is the square error. Alternatively instead of using the voltage error the current could be predicted and this be used as a way to estimate the machine parameters.

When using a gradient method the algorithm will converge typically to the closest minimum. Depending of the system and cost function chosen there may be more than one and this could be a problem as the algorithm could find different estimates depending of the starting point.

With the cost function defined in Equation 7.10 the gradient can be calculated. For simplicity of the solution the following definitions are made in Equation B.27 - B.28 for the error in the voltage in q and d where the current has two states. At the beginning of the period it appears alone and at the end of it is denoted with an apostrophe.

$$\epsilon_q = u_q - \left(\frac{R}{2}(i'_q + i_q) + \frac{L_q}{T_s}(i'_q - i_q) + \omega_e\left(\lambda_{mpm} + \frac{L_d}{2}(i'_d + i_d)\right)\right) \tag{B.27}$$

$$\epsilon_d = u_d - \left(\frac{R}{2}(i'_d + i_d) + \frac{L_d}{T_s}(i'_d - i_d) - \omega_e\frac{L_q}{2}(i'_q + i_q)\right) \tag{B.28}$$

The gradient for each parameter can then be written as Equation B.29 - B.32 by differentiating Equation 7.10 to respect to each parameter. The σ represents the sign function which is defined as 1 when the input is positive, 0 when it is zero and -1 when it is negative.

$$\frac{\nabla\epsilon}{\nabla R} = -\sigma(\epsilon_q)\frac{i'_q + i_q}{2} - \sigma(\epsilon_d)\frac{i'_d + i_d}{2} \tag{B.29}$$

$$\frac{\nabla\epsilon}{\nabla L_q} = -\sigma(\epsilon_q)\frac{i'_q - i_q}{T_s} + \sigma(\epsilon_d)\frac{\omega_e(i'_q + i_q)}{2} \tag{B.30}$$

$$\frac{\nabla\epsilon}{\nabla L_d} = -\sigma(\epsilon_q)\frac{\omega_e(i'_d + i_d)}{2} - \sigma(\epsilon_d)\frac{i'_d - i_d}{T_s} \tag{B.31}$$

$$\frac{\nabla\epsilon}{\nabla L_{mpm}} = -\omega_e\sigma(\epsilon_q) \tag{B.32}$$

What the gradient equations show is that there is only one minimum when the voltage is used as an estimator and the cost function is the sum of the absolute values of the errors. This minimum is taken when the voltage error in both axes is zero and can only be achieved for the true parameters as the model cannot be simplified more. Similar derivation can be made with the square cost function and while the different would change the same principle is conserved.

The gradient equations also tell us more information. If the system is in steady-state as it has been said before that it is impossible to estimate the parameters as there are more unknowns than equations. This is also shown in the gradient as in steady-state there will be infinite values that make ϵ_q and ϵ_d equal to zero. As we are perturbing the current

randomly it is expected however that there will only be one set of values that ensure this as the model cannot be further simplified. Moreover there may be (although very much may not) combinations of currents and speeds that create additional minimums (by setting to zero all the gradient equations) without making ϵ zero but if the current is perturbed randomly on average this will not matter.

The gradient equations are also logical. For instance in Equation B.29 the resistor will be decreased when the error in the q equation is negative and i_q is also positive (remember we move in the direction contrary to the gradient). If all the other parameters are correctly estimated a higher estimated voltage means effectively that the resistor estimation is too big and should be decreased. The same happens in the d equation. The same can be seen in the rest of the equations. Further inspection shows that the gradient is just the sign scaled for how much the parameter contributes to the voltage error so it prioritizes changes over the parameters that affect the error the most.

Finally what it cannot be studied in the gradient equations is the effect of λ or errors in the sensors or model. However if λ is not chosen too big the algorithm is expected to converge and the ripple on the estimations can be minimized at the expense of slower responses. As for errors in the sensors or the models both simulations and experiments results show that the effect is a bigger ripple in the estimation without affecting the convergence as long as they are random and λ is small enough.

C

MODEL'S DIAGRAMS

In this appendix different models schematics used are shown.

C.1 SYSTEM MODEL

The system is modeled as depicted in Figure C.1 as a DC source connected to a 2-level conveter that feeds the PMSM. The torque can be modified by connecting a controlled torque source to the mechanical port of the PMSM model.

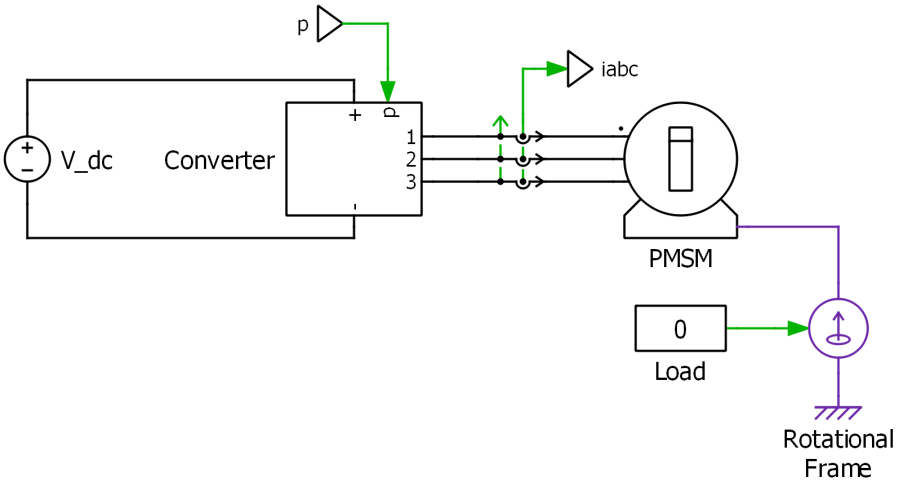


Figure C.1.: System model in PLECS

The converter in Figure C.1 is the IGBT Conveter (3ph) from PLECs modified into a configurable system to test the effect of the non-linearities. The snubbers are modeled as capacitors in parallel with the converter's diodes. The IGBT and diode voltage drops are implemented as a constant voltage V_{on} and a resistor R_{on}.

C.2 CONTROL SCHEME

The main parts of the inner-loop control scheme are shown in Figure C.2. The controller receives the reference i^*_{dq} and generates the voltage command. The non-linearities are compensated with the compensator block below the controller. The final voltage is passed to the modular which generates the switches states. Those states are passed to the configurable deadtime that if chosen adds the signal delay t_d to the switches states.

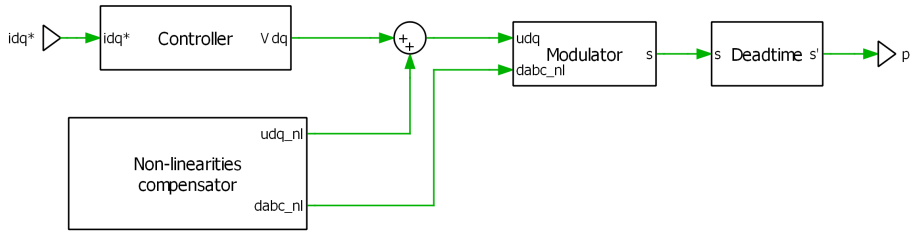


Figure C.2.: Inner-loop control scheme

C.2.1 *Controller*

The controller is a configurable system with the deadbeat and continuous and discrete PIs. The deadbeat implementation is shown in Figure C.3. The controller is implemented in a C-script using Code D.2.

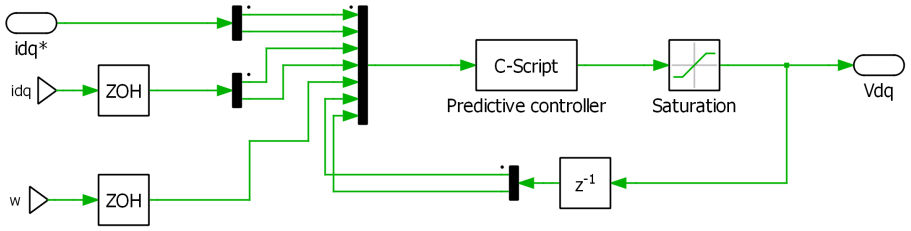


Figure C.3.: Deadbeat controller

The PI is shown for the continuous case in Figure C.4. The discrete case is the same but with a integrator using Tustin. The controller presents the decoupling down and the anti-windup after the voltage command.

C.2.2 *Non-linearity compensation*

The discrete non-linearity compensation with the code from Code D.7 is shown in Figure C.5.

The resonant version is shown in Figure C.6 with the resonants in parallel to compensate each harmonic.

C.2.3 *Modulator*

The modulator block is shown in Figure C.7. It uses the Space Vector Modulation block from PLECS modified to output the duty cycles which are then generated trough a comparaison with a carrier. The voltage command is delayed one sample to simulate the DSP computation delay.

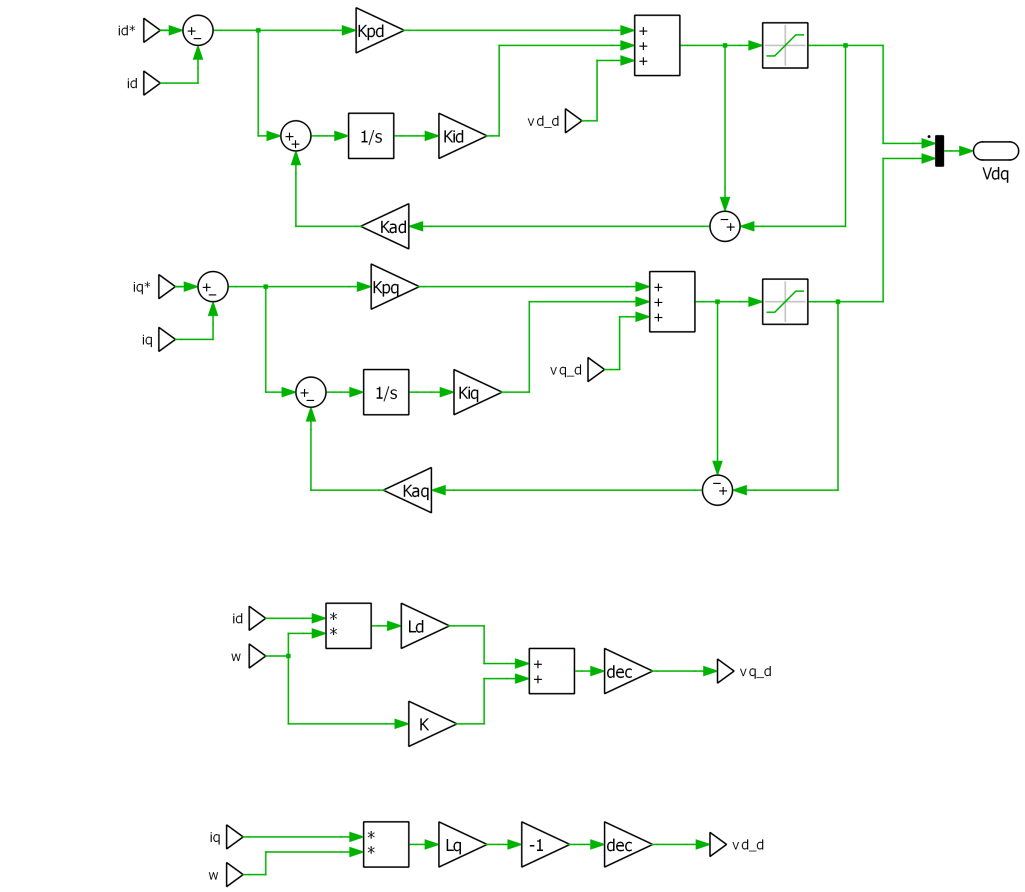


Figure C.4.: PI current controller

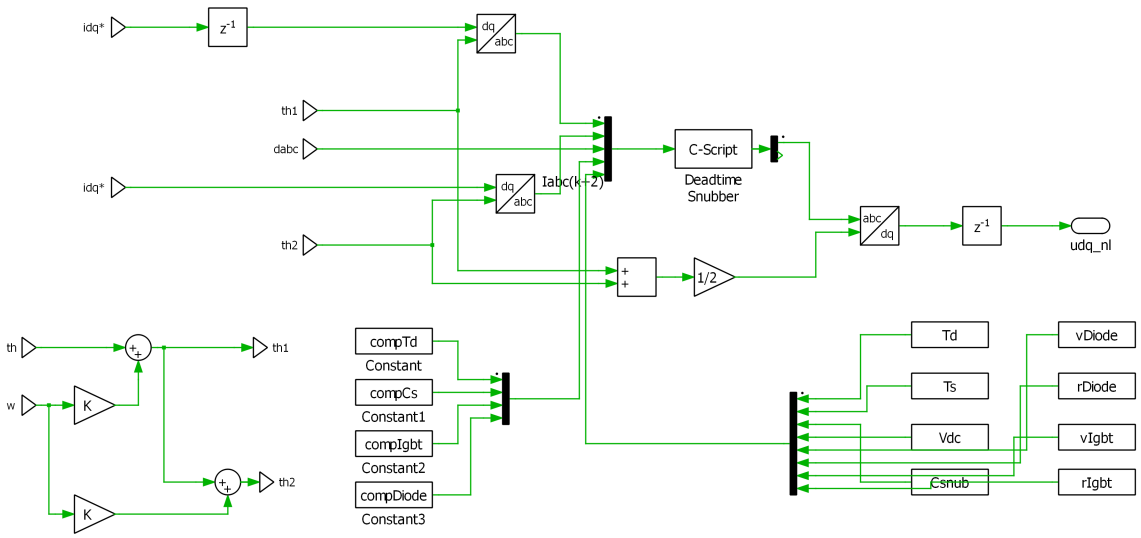


Figure C.5.: Discrete non-linearity compensation

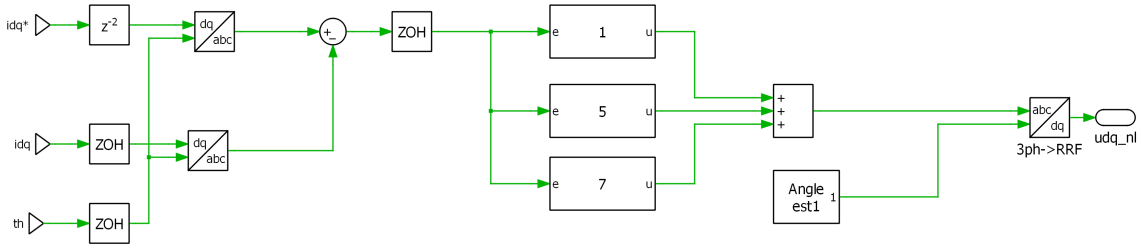


Figure C.6.: PR non-linearity compensator

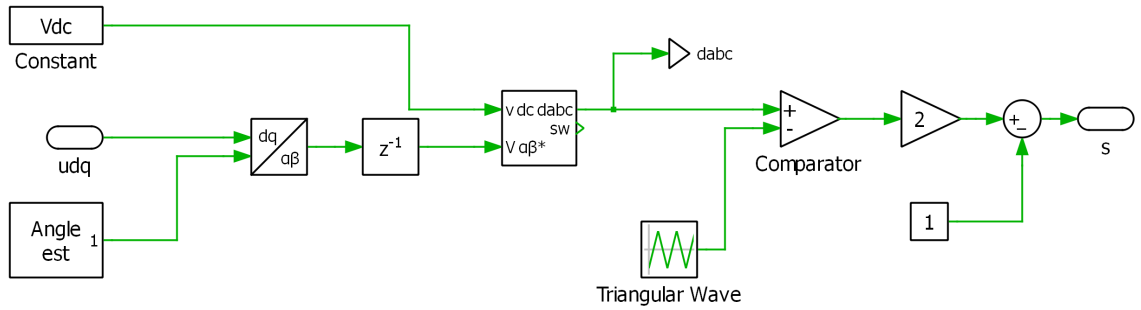


Figure C.7.: Modulator

C.3 DEADTIME

The deadtime is implemented as a configurable system to be able to see the effect with and without it. The deadtime implementation uses the blanking time from PLECS as depiceted in Figure C.8.

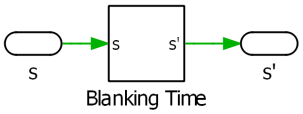


Figure C.8.: Deadtime model

C.4 PARAMETER ESTIMATION

Most simulations have been conducted in PLECS standalone for speed. There is also a Simulink model that runs the model presented in Figure C.1 with some of the blocks used in the experiment. For the parameter estimation the model is presented in Figure C.9. The codes used are Code D.8 for the RLS method and Code D.9 for the gradient one.

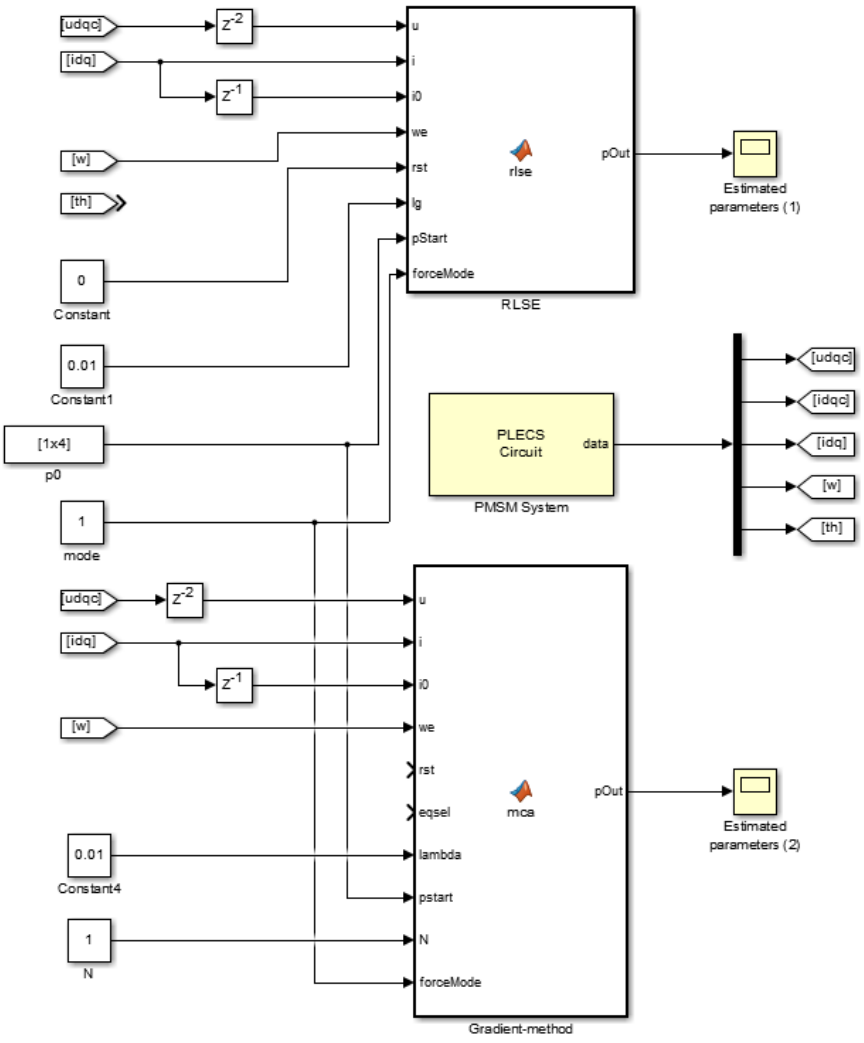


Figure C.9.: Parameter estimation model

D

LIST OF CODES

In this appendix the different set of codes provided are listed.

D.1 PREDICTIVE CONTROLLER

D.1.1 *1-period version*

```
function [uqc,udc] = deadbeat(iqr,idr,iq,id,w,dataIn,pred)
  %Offline data
  Ts = 1/3000;
  Ld = 2.2e-3;
  Lq = 2e-3;
  R = 0.24;
  Lmpm = 0.1175;

  %If use online data is set
  if(pred == 1)
    R = dataIn(1);
    Lq = dataIn(2);
    Ld = dataIn(3);
    Lmpm = dataIn(4);
  end

  %Estimating differences and averages
  iqdiff = (iqr-iq)/Ts;
  iddiff = (idr-id)/Ts;
  iqav = (iqr+iq)/2;
  idav = (idr+id)/2;

  %Calculating voltage commands
  uqc = (R*iqav+Lq*iqdiff + w*(Ld*idav+Lmpm));
  udc = (R*idav+Ld*iddiff - w*(Lq*iqav));
end
```

Code D.1: 1-period predictive controller Simulink code

D.1.2 *2-period*

```
function [uqc,udc] = deadbeat(iqr,idr,iq,id,iqro,idro,uqo,udo,w,dataIn,pred)
  %Offline data
  Ts = 1/3000;
  Ld = 2.2e-3;
  Lq = 2e-3;
  R = 0.24;
  Lmpm = 0.1175;

  %If use online data is set
```

```
10   if(pred == 1)
11     R = dataIn(1);
12     Lq = dataIn(2);
13     Ld = dataIn(3);
14     Lmpm = dataIn(4);
15   end
16
17   %Estimating differences and averages
18   iqdiff = (iqr-iq)/(2*Ts);
19   iddiff = (idr-id)/(2*Ts);
20   iqav = (iqr+iqro+iq)/3;
21   idav = (idr+idro+id)/3;
22
23   %Calculating voltage commands
24   uqc = 2*(R*iqav+Lq*iqdiff + w*(Ld*idav+Lmpm))-uqo;
25   udc = 2*(R*idav+Ld*iddiff - w*(Lq*iqav))-udo;
26 end
```

Code D.2: 2-period predictive controller Simulink code

```
1  #define idr Input(0)
2  #define iqr Input(1)
3  #define id Input(2)
4  #define iq Input(3)
5  #define w Input(4)
6  #define vdo Input(5)
7  #define vqo Input(6)
8  #define iq1 Input(7)
9  #define id1 Input(8)
10
11 #define vd Output(0)
12 #define vq Output(1)
13
14 float R = 0.19;
15 float Lq = 2.2e-3;
16 float Ld = 2.2e-3;
17 float Lmpm = 0.12;
18 float Ts;
```

Code D.3: 2-period predictive controller PLECS C code (Code declarations)

```
1  Ts = ParamRealData(0,0);
2  float iqp = -(R*Ts*iq - Ts*vqo - Lq*iq + Lmpm*Ts*w + Ld*Ts*id*w)/Lq;
3  float idp = (Ld*id + Ts*vdo - R*Ts*id + Lq*Ts*iq*w)/Ld;
4
5  float iqav = (iqr+iq+iqp)/3.0;
6  float idav = (idr+id+idp)/3.0;
7  float iqdiff = (iqr-iq)/(2*Ts);
8  float iddiff = (idr-id)/(2*Ts);
9
10 float uqc = 2*(R*iqav+Lq*iqdiff + w*(Ld*idav+Lmpm))-vqo;
11 float udc = 2*(R*idav+Ld*iddiff - w*(Lq*iqav))-vdo;
12
13 vd = udc;
14 vq = uqc;
```

Code D.4: 2-period predictive controller PLECS C code (Output function code)

D.2 NON-LINEARITY COMPENSATION (ANALYTICAL)

```
function [Vabc,dabcC] = fcn(u,params,comp)
  %Returns a voltage compensation in abc coordinates for the deadtime of
  %a system with a converter that is assumed to have the following
  %conditions:
  %   - Signals are delayed Td to avoid shortcircuits
  %   - There are capacitors in paralell with the diodes
  %   - Diodes and IGBTs have a Von and Ron
  %
  % comp:
  %        0 - Td
  %        1 - Td zero crossing fix
  %        2 - Cs compensation
  %        3 - Diode and IGBT compensation


  %Input data
  ia = u(1);
  ib = u(2);
  ic = u(3);
  ia1 = u(4);
  ib1 = u(5);
  ic1 = u(6);
  tao = u(7);
  tbo = u(8);
  tco = u(9);

  %System data
  Td = params(6);
  Ts = 1/3000;
  Vdc = 520;
  Cs = params(1);
  VonDiode = params(2);
  RonDiode = params(3);
  VonIgbt = params(4);
  RonIgbt = params(5);

  ta = 0; tb = 0; tc = 0;
  Vabc = [0 0 0];
  %Time compensation (normal one)
  if(comp >= 0)
    if(ia>0.0)
      ta = Td;
    else
      ta = -Td;
    end
    if(ib>0.0)
      tb = Td;
    else
      tb = -Td;
    end
    if(ic>0.0)
      tc = Td;
    else
```

```
54        tc = -Td;
55      end
56    end
57
58    %Improving of zero crossings
59    if(comp >= 1)
60      if(ia1*ia < 0.0)
61        ta = 0.0;
62      end
63
64      if(ib1*ib < 0.0)
65        tb = 0.0;
66      end
67
68      if(ic1*ic < 0.0)
69        tc = 0.0;
70      end
71    end
72
73    if(comp >= 2)
74      %Snubber compensation
75      if(ia < 0.0)
76        ta = ta + timeSnubber(ia/2.0, Cs, Vdc, Td);
77      end
78      if(ib < 0.0)
79        tb = tb + timeSnubber(ib/2.0, Cs, Vdc, Td);
80      end
81      if(ic < 0.0)
82        tc = tc + timeSnubber(ic/2.0, Cs, Vdc, Td);
83      end
84
85      if(ia1 > 0.0)
86        ta = ta + timeSnubber(ia1/2.0, Cs, Vdc, Td);
87      end
88      if(ib1 > 0.0)
89        tb = tb + timeSnubber(ib1/2.0, Cs, Vdc, Td);
90      end
91      if(ic1 > 0.0)
92        tc = tc + timeSnubber(ic1/2.0, Cs, Vdc, Td);
93      end
94    end
95
96    if(comp >= 3)
97      %Diode Von and Ron compensation
98      ta = ta + timeDiode((ia+ia1)/2.0, VonDiode, RonDiode, Vdc, Ts, tao);
99      tb = tb + timeDiode((ib+ib1)/2.0, VonDiode, RonDiode, Vdc, Ts, tbo);
100     tc = tc + timeDiode((ic+ic1)/2.0, VonDiode, RonDiode, Vdc, Ts, tco);
101
102     %Igbt Von and Ron compensation
103     ta = ta + timeIgbt((ia+ia1)/2.0, VonIgbt, RonIgbt, Vdc, Ts, tao);
104     tb = tb + timeIgbt((ib+ib1)/2.0, VonIgbt, RonIgbt, Vdc, Ts, tbo);
105     tc = tc + timeIgbt((ic+ic1)/2.0, VonIgbt, RonIgbt, Vdc, Ts, tco);
106   end
107
108   %Transformation Time -> Voltage
109   vab = (ta-tb)*Vdc/Ts;
```

```
110   vbc = (tb-tc)*Vdc/Ts;
111   vca = (tc-ta)*Vdc/Ts;
112
113   dabcC = [ta tb tc]*1/Ts;
114
115   %Transformation LL-LN(motor side) voltages
116   va = 1/3.0*(vab-vca);
117   vb = 1/3.0*(-vab+vbc);
118   vc = 1/3.0*(-vbc+vca);
119
120   Vabc = [va vb vc];
121 end
122
123 function tc = timeSnubber(i,C,Vdc,Td)
124   t = C/abs(i)*Vdc;
125   v2 = Vdc-abs(i)*Td/C;
126
127   if(t<Td)
128     tc = -sign(i)*1/2.0*Vdc*t;
129   else
130     tc = -sign(i)*1/2.0*(Vdc+v2)*Td;
131   end
132   tc = tc/Vdc;
133 end
134
135 function tc = timeDiode(i, Von, Ron, Vdc, Ts, t)
136   vDiode = Von+Ron*abs(i);
137
138   if(i<0)
139     tc = vDiode*t;
140   else
141     tc = -vDiode*(1-t);
142   end
143   tc = -tc/Vdc*Ts;
144 end
145
146 function tc = timeIgbt(i, Von, Ron, Vdc, Ts, t)
147   vIgbt = Von+Ron*abs(i);
148
149   if(i<0)
150     tc = -vIgbt*(1-t);
151   else
152     tc = vIgbt*t;
153   end
154   tc = tc/Vdc*Ts;
155 end
```

Code D.5: Discrete non-linearity compensation Simulink code (the duty cycles, dabcC, are added to the PWM block directly)

```
1 #define ich (-0.5*0)
2 #define Ia (Input(0))
3 #define Ib (Input(1))
4 #define Ic (Input(2))
5 #define Ia1 (Input(3))
6 #define Ib1 (Input(4))
```

```
7  #define Ic1 (Input(5))
8  #define tao Input(6)
9  #define tbo Input(7)
10 #define tco Input(8)
11 #define compTd Input(9)
12 #define compCs Input(10)
13 #define compIgbt Input(11)
14 #define compDiode Input(12)
15
16 #define Va Output(0)
17 #define Vb Output(1)
18 #define Vc Output(2)
19 #define Da Output(3)
20 #define Db Output(4)
21 #define Dc Output(5)
22
23 #define Td Input(13)
24 #define Ts Input(14)
25 #define Vdc Input(15)
26 #define Cs Input(16)
27 #define VonDiode Input(17)
28 #define RonDiode Input(18)
29 #define VonIgbt Input(19)
30 #define RonIgbt Input(20)
31
32 float abs(float val){
33   if(val > 0.0)
34     return val;
35   return -val;
36 }
37
38 float sign(float val){
39   if(val >= 0.0)
40     return 1.0;
41   return -1.0;
42 }
43
44 float timeSnubber(float i, float C, float vdc, float td){
45   float t = C/abs(i)*vdc;
46   float v2 = vdc-abs(i)*td/C;
47   float tc = 0.0;
48
49   if(t<td)
50     tc = -sign(i)*1/2.0*vdc*t;
51   else
52     tc = -sign(i)*1/2.0*(vdc+v2)*td;
53
54   return tc/vdc;
55 }
56
57 float timeDiode(float i, float Von, float Ron, float vdc, float td, float t,
       float ts){
58   float vDiode = Von+Ron*abs(i);
59   float tc = 0;
60
61   if(i<0)
```

```
62      tc = vDiode*t;
63    else
64      tc = -vDiode*(1-t);
65    return -tc/vdc*ts;
66  }
67
68  float timeIgbt(float i, float Von, float Ron, float vdc, float td, float t,
        float ts){
69    float vIgbt = Von+Ron*abs(i);
70    float tc = 0;
71    if(i<0)
72      tc = -vIgbt*(1-t);
73    else
74      tc = vIgbt*(t);
75    return tc/vdc*ts;
76  }
```

Code D.6: Discrete non-linearity compensation Plecs C code (Code declarations)

```
1  float ta = 0;
2  float tb = 0;
3  float tc = 0;
4
5  float ia = Ia;
6  float ib = Ib;
7  float ic = Ic;
8  float ia1 = Ia1;
9  float ib1 = Ib1;
10 float ic1 = Ic1;
11
12 //Time compensation
13 if(compTd == 1){
14   if(ia>0.0)
15     ta = Td;
16   else
17     ta = -Td;
18   if(ib>0.0)
19     tb = Td;
20   else
21     tb = -Td;
22   if(ic>0.0)
23     tc = Td;
24   else
25     tc = -Td;
26
27   //Zero fix
28   if(ia1*ia < 0.0)
29     ta = 0.0;
30   if(ib1*ib < 0.0)
31     tb = 0.0;
32   if(ic1*ic < 0.0)
33     tc = 0.0;
34 }
35
36 if(compCs == 1){
37   if(ia < 0.0)
```

```
38      ta += timeSnubber(ia/2.0, Cs, Vdc, Td);
39    if(ib < 0.0)
40      tb += timeSnubber(ib/2.0, Cs, Vdc, Td);
41    if(ic < 0.0)
42      tc += timeSnubber(ic/2.0, Cs, Vdc, Td);
43
44    if(ia1 > 0.0)
45      ta += timeSnubber(ia1/2.0, Cs, Vdc, Td);
46    if(ib1 > 0.0)
47      tb += timeSnubber(ib1/2.0, Cs, Vdc, Td);
48    if(ic1 > 0.0)
49      tc += timeSnubber(ic1/2.0, Cs, Vdc, Td);
50  }
51
52  if(compDiode){
53    ta += timeDiode((ia+ia1)/2.0, VonDiode, 1*RonDiode, Vdc, Td, tao,Ts);
54    tb += timeDiode((ib+ib1)/2.0, VonDiode, 1*RonDiode, Vdc, Td, tbo,Ts);
55    tc += timeDiode((ic+ic1)/2.0, VonDiode, 1*RonDiode, Vdc, Td, tco,Ts);
56  }
57
58  if(compIgbt){
59    ta += timeIgbt((ia+ia1)/2.0, VonIgbt, 1*RonIgbt, Vdc, Td, tao,Ts);
60    tb += timeIgbt((ib+ib1)/2.0, VonIgbt, 1*RonIgbt, Vdc, Td, tbo,Ts);
61    tc += timeIgbt((ic+ic1)/2.0, VonIgbt, 1*RonIgbt, Vdc, Td, tco,Ts);
62  }
63
64  //Transformation of frames
65  float vab = (ta-tb)*Vdc/Ts;
66  float vbc = (tb-tc)*Vdc/Ts;
67  float vca = (tc-ta)*Vdc/Ts;
68
69  float va = 1/3.0*(vab-vca);
70  float vb = 1/3.0*(-vab+vbc);
71  float vc = 1/3.0*(-vbc+vca);
72
73  Va = va;
74  Vb = vb;
75  Vc = vc;
76
77  Da = ta/Ts;
78  Db = tb/Ts;
79  Dc = tc/Ts;
```

Code D.7: Discrete non-linearity compensation Plecs C code (Output function code)

D.3 PARAMETER DETERMINATION

D.3.1 *RLS*

```
1  function pOut    = rlse(u,i,io,we,rst,lg,pStart,forceMode)
2    persistent pp mode;
3
4    uq = u(1);
5    ud = u(2);
6    iq1 = i(1);
```

```
7   id1 = i(2);
8   iq = io(1);
9   id = io(2);
10  Ts = 1/3000;
11
12  oS = pStart';
13  rstCh = rst;
14
15  if( isempty(pp) || rst == 1 )
16    pp = pStart;
17    mode = 0;
18  end
19
20  if(forceMode ~= mode)
21    rstCh = 1;
22    mode = forceMode;
23  end
24
25  l = 1-lg;
26  newP = [0 0 0 0];
27  switch(mode)
28    case 1
29      newP = rleRLF(uq,ud,iq,id,iq1,id1,we,Ts,l,pp,rstCh,oS);
30    case 2
31      newP = rleRL(uq,ud,iq,id,iq1,id1,we,Ts,l,pp,rstCh,oS(1:(end-1)));
32    case 3
33      newP = rleLF(uq,ud,iq,id,iq1,id1,we,Ts,l,pp,rstCh,oS(2:end));
34    case 4
35      newP = rleL(uq,ud,iq,id,iq1,id1,we,Ts,l,pp,rstCh,oS(2:(end-1)));
36  end
37  pp = newP;
38  pOut = pp;
39 end
40
41 function pOut = rleRLF(uq,ud,iq,id,iq1,id1,we,Ts,l,pp,rst,oS)
42  persistent o P;
43  oo = pp';
44  newP = pp;
45
46  I = eye(4);
47  Xt = [(iq1+iq)/2 (iq1-iq)/Ts we*(id1+id)/2 we; (id1+id)/2 -we*(iq1+iq)/2 (
      id1-id)/Ts 0].*[oS';oS'];
48  y = [uq;ud];
49
50  if( isempty(o) || rst == 1 )
51    p = oo';
52    o = p'./oS;
53    P = 1e-4*I;
54  end
55
56  if(rst == 0)
57    k = (1^-1*P*Xt')/(1+1^-1*sum(sum(Xt*P.*Xt)));
58    e = y-Xt*o;
59    o = o+k*e;
60    P = 1^-1*(P-k*Xt*P);
61
```

```
62      Rest = o(1)*oS(1);
63      Lqest = o(2)*oS(2);
64      Ldest = o(3)*oS(3);
65      Lmpmest = o(4)*oS(4);
66
67      newP = [Rest Lqest Ldest Lmpmest];
68    end
69
70    pOut = newP;
71  end
72
73  function pOut = rleLF(uq,ud,iq,id,iq1,id1,we,Ts,l,pp,rst,oS)
74    persistent o P;
75    oo = pp(2:end)';
76    newP = pp;
77
78    I = eye(3);
79    Xt = [(iq1-iq)/Ts we*(id1+id)/2 we;-we*(iq1+iq)/2 (id1-id)/Ts o].*[oS';oS
        '];
80    y = [uq-(iq1+iq)/2*pp(1);ud-(id1+id)/2*pp(1)];
81
82    if( isempty(o) || rst == 1 )
83      p = pp(2:end);
84      o = p'./oS;
85      P = 1e-4*I;
86    end
87
88    if(rst == 0)
89      k = (l^-1*P*Xt')/(1+l^-1*sum(sum(Xt*P.*Xt)));
90      e = y-Xt*o;
91      o = o+k*e;
92      P = l^-1*(P-k*Xt*P);
93
94      Lqest = o(1)*oS(1);
95      Ldest = o(2)*oS(2);
96      Lmpmest = o(3)*oS(3);
97
98      newP = [pp(1) Lqest Ldest Lmpmest];
99    end
100
101   pOut = newP;
102 end
103
104 function pOut = rleRL(uq,ud,iq,id,iq1,id1,we,Ts,l,pp,rst,oS)
105   persistent o P;
106   oo = pp(1:(end-1))';
107   newP = pp;
108
109   I = eye(3);
110   Xt = [(iq1+iq)/2 (iq1-iq)/Ts we*(id1+id)/2;(id1+id)/2 -we*(iq1+iq)/2 (id1-
        id)/Ts].*[oS';oS'];
111   y = [uq-we*pp(4);ud];
112
113   if( isempty(o) || rst == 1 )
114     p = pp(1:(end-1));
115     o = p'./oS;
```

```
116     P = 1e-4*I;
117   end
118
119   if(rst == 0)
120     k = (1^-1*P*Xt')/(1+1^-1*sum(sum(Xt*P.*Xt)));
121     e = y-Xt*o;
122     o = o+k*e;
123     P = 1^-1*(P-k*Xt*P);
124
125     Rest = o(1)*oS(1);
126     Lqest = o(2)*oS(2);
127     Ldest = o(3)*oS(3);
128
129     newP = [Rest Lqest Ldest pp(4)];
130   end
131
132   pOut = newP;
133 end
134
135 function pOut = rleL(uq,ud,iq,id,iq1,id1,we,Ts,l,pp,rst,oS)
136   persistent o P;
137   oo = pp(2:(end-1))';
138   newP = pp;
139
140   I = eye(2);
141   Xt = [(iq1-iq)/Ts we*(id1+id)/2;-we*(iq1+iq)/2 (id1-id)/Ts].*[oS';oS'];
142   y = [uq-(iq1+iq)/2*pp(1)-we*pp(4);ud-(id1+id)/2*pp(1)];
143
144   if( isempty(o) || rst == 1 )
145     p = pp(2:(end-1));
146     o = p'./oS;
147     P = 1e-4*I;
148   end
149
150   if(rst == 0)
151     k = (1^-1*P*Xt')/(1+1^-1*sum(sum(Xt*P.*Xt)));
152     e = y-Xt*o;
153     o = o+k*e;
154     P = 1^-1*(P-k*Xt*P);
155
156     Lqest = o(1)*oS(1);
157     Ldest = o(2)*oS(2);
158
159     newP = [pp(1) Lqest Ldest pp(4)];
160   end
161
162   pOut = newP;
163 end
```

Code D.8: RLS method with mode selector (what parameters to estimate) for Simulink

D.3.2 *Gradient method*

```
1 function pOut = mca(u,i,io,we,rst,eqsel,lambda,pstart,N,forceMode)
2   uq = u(1);
3   ud = u(2);
```

```
4   iq1 = i(1);
5   id1 = i(2);
6   iq = io(1);
7   id = io(2);
8   poR = pstart;
9   po = [1;1;1;1];
10  n = length(po);
11  l = lambda;
12
13  persistent p j grad;
14  if(isempty(p) || rst == 1)
15    p = po;
16    j = 0;
17    grad = zeros(n,1);
18  end
19
20  if(rst == 0)
21    j = j + 1;
22
23    H = [1/2*(iq1+iq),1/Ts*(iq1-iq),we/2*(id1+id),we;
24     1/2*(id1+id),-we/2*(iq1+iq),1/Ts*(id1-id),0];
25    y = u;
26
27    %Calculate error
28    e = error(y,H,p.*poR,eqsel);
29    gradk = zeros(n,1);
30
31    %Calculate gradient
32    change = 0.025;
33    for(k=1:n)
34      pkp = p; pkp(k) = (1+change)*p(k);
35      pkm = p; pkm(k) = (1-change)*p(k);
36      ep = error(y,H,pkp.*poR,eqsel);
37      em = error(y,H,pkm.*poR,eqsel);
38
39      gradk(k) = (ep-e)/(change) + (em-e)/(-change);
40    end
41    %Calculate average gradient with N points, N can be 1 for no
42    %average
43    grad = grad+gradk;
44    if(j >= N)
45      grad = l*(grad./sum(sqrt(grad.^2)));
46      switch(forceMode)
47        case 2
48          grad(4) = 0;
49        case 3
50          grad(1) = 0;
51      end
52      p = p - grad;
53      grad = zeros(n,1);
54      j = 0;
55    end
56  end
57  pOut = p.*poR;
58 end
59
```

```
60 function e = error(y,H,p,eqsel)
61   %Constraints (min,max values parameters can take)
62   pmin = [0.15;0.5e-3;0.5e-3;0.03];
63   pmax = [0.30;3.5e-3;3.5e-3;0.25];
64
65   %Constraints functions
66   bmax = (pmax-p)./abs(pmax);
67   bmin = (p-pmin)./abs(pmin);
68
69   bmax = (bmax<0).*((100*bmax).^2);
70   bmax = sum(bmax);
71
72   bmin = (bmin<0).*((100*bmin).^2);
73   bmin = sum(bmin);
74
75   e0 = y-H*p;
76
77   %1: q, 2: d, else: all
78   if(eqsel == 1)
79     e = e0(1);
80   elseif(eqsel == 2)
81     e = e0(2);
82   else
83     e = e0;
84   end
85
86   %Error function, including constraints
87   e = sum(e.^2+(bmax+bmin));
88 end
```

Code D.9: Gradient method with parameter constraints and mode selector (what parameters to estimate) for Simulink