UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA

Corso di Laurea magistrale in Informatica e dell'Automazione

"Progetto e sviluppo di un sistema di controllo model based e sensorless per un motore sincrono a magneti permanenti"

"Design and development of a model based and sensorless control system for a permanent magnet synchronous motor"

Relatore:

Prof. **Giuseppe Orlando**

Tesi di Laurea di:

Rumeysa Nur Gulesin

Correlatori:

Prof. **Gianluca Ippoliti**

Alessio Beato

A.A. **2022** / **2023**

Sommario

Questa tesi intende affrontare il problema della progettazione di un controllo sensorless e model based di un motore sincrono a magneti permanenti. Dapprima verrà descritto lo strumento di cui ci si serve per la simulazione del processo e l'implementazione del controllo; tale strumento è un progetto MATLAB/Simulink fornito dalla Whirlpool di Fabriano. All'interno del simulatore si può simulare il comportamento del processo dipendentemente dal controllo applicato. La Whirlpool impiega dei regolatori PI che si intende sostituire con dei regolatori sliding mode, perciò è riportata la formulazione matematica dello Sliding Mode Control (SMC). A seguito dello sviluppo dei regolatori di velocità e corrente in Simulink, sono presenti dei risultati grafici per il confronto delle prestazioni dello SMC con quelle dei regolatori PI. Il confronto è stato effettuato sia in condizioni nominali, sia saturando la tensione e/o la coppia. Si nota che controllori sliding mode e controllori PI dimostrano prestazioni simili. In particolare, per quanto riguarda i controllori sliding mode, si osserva elevata prontezza, transitori privi di oscillazioni e eccellente inseguimento dei segnali di riferimento. Successivamente sono riportati i valori assunti dagli indici di prestazione Integral Absolute Error (IAE) e Mean Squared Error (MSE) in presenza di variazioni parametriche. I valori assunti dagli indici dimostrano che lo SMC gode di robustezza. Gli esiti delle simulazioni e dei test di robustezza permettono di ritenere lo SMC adatto ad un'applicazione reale. Per quanto riguarda possibili sviluppi futuri, non si esclude che i risultati finora ottenuti possano migliorare ulteriormente grazie all'introduzione di uno Sliding Mode Observer (SMO).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis aims to solve the challenge of creating a model-based and sensorless control for a permanent magnet synchronous motor. The goal is to suggest an alternative control technique that performs better than the standard PI regulators in terms of quality and efficiency. Sliding Mode Control (SMC) is an effective method that is known for its ability to provide robustness and invariance to system uncertainties. To demonstrate the efficacy of SMC, it was crucial to use an environment that could simulate the behavior of the engine being studied. Whirlpool has implemented a simulator able to emulate the behavior of the real process and comes with a speed regulator and a current regulator. The controllers used in this process are standard PIs. The thesis aims to incorporate controllers regulated by sliding mode control laws to demonstrate their effectiveness and evaluate their performance. First, the structure of the simulator will be described in detail, followed by preliminary notions for the creation of a control law with the sliding mode technique. Subsequently, the design of sliding mode speed and current regulators in MATLAB/Simulink is described. Following the virtual implementation of the control system, the results obtained in nominal conditions are shown. Afterward, the thesis describes the system behavior when electrical or mechanical parameters' variation occurs. The robustness of the system will be evaluated considering these variations. Additionally, this discussion will present some interesting results obtained by applying sliding mode control to a real system. Finally, it will anticipate possible future developments of the topic.

Chapter 2

Thesis Problem and Objective

This thesis aims to design and develop a model based and sensorless control system for a dishwasher permanent magnet drain pump (figure 2.1). To improve the performance and efficiency of the drive system, Sliding Mode Control (SMC) is used to implement speed and current regulators. The final objective is to demonstrates that the considered solution, compared with a PID control policy, achieves effective speed trajectory tracking and maintains robustness in the presence of system disturbances. Thanks to the Software-in-the-Loop (SIL) methodology, the control software to be verified can directly interact with the emulation of the system implemented by Whirlpool. The Whirlpool company of Fabriano has, in fact, designed a simulator that consist of a MATLAB/Simulink project containing the PI controllers. The PI controllers are S-functions executed starting from the C code inside them, which is the same imported on the dishwasher's microcontroller. The company has also set up the simulator for the C code's auto-generation of the sliding mode controllers subsystems: this will allow to import these controllers C code onto the actual system and make a comparison outside the simulated environment.

Figure 2.1: Dishwasher permanent magnet drain pump

Chapter 3

The Whirlpool Simulator

The objective of this chapter is to introduce the tool through which it was possible to simulate the behavior of the process that concerns the discussion. The Whirlpool simulator permits the observation of the control action of the regulators in a simulated environment. Therefore the chapter describes the general structure of the simulator by outlining the context in which the sliding mode controllers are inserted.

3.1 The simulator's structure

The Whirlpool simulator is a MATLAB/Simulink project consisting of 3 blocks shown in figure 3.1.

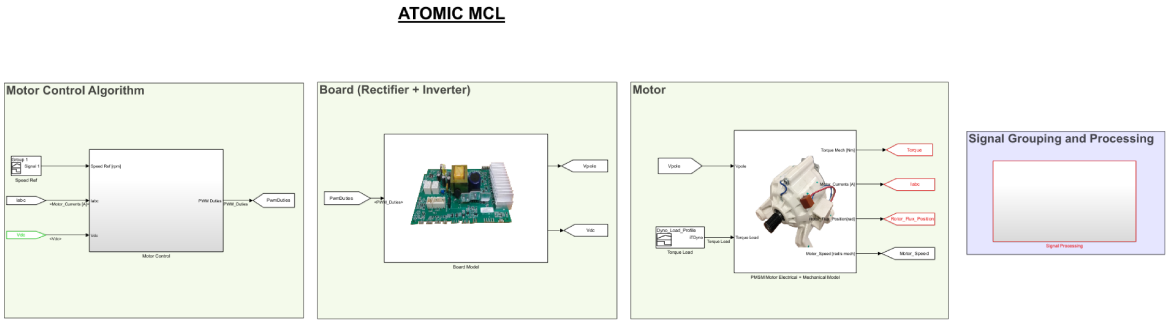


Figure 3.1: Whirlpool Simulator

The electronic board and mechanical model of the electric motor, represented by "Board (Rectifier + inverter)" and "Motor" blocks respectively, are outside the scope of this thesis work. The block that is of interest for this work is the one depicted in the figure 3.2. These are the necessary steps for the motor control, made with Simulink blocks containing s-functions, which are protected by industrial secrecy. First of all, simulation can be performed in two modes:

1. Sensorless

2. Sensored

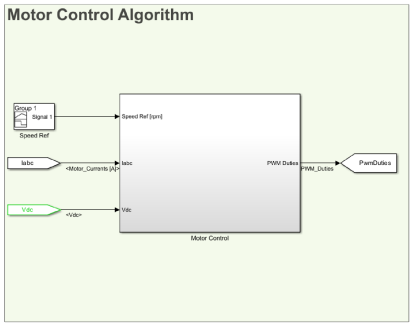


Figure 3.2: Whirlpool Motor Control

To select the desired mode, the value in the block in figure 3.3 has to be set to 1 or 0; if "Sensored" mode is active, blocks use the measurements obtained from the sensors. Conversely, if "Sensorless" is active, blocks consider the estimates obtained by the observer.

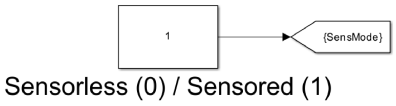


Figure 3.3: SensMode

The diagram representing the control scheme implemented by Whirlpool is shown below (figure 3.4). The function of the individual blocks is explained in detail in the following sections.

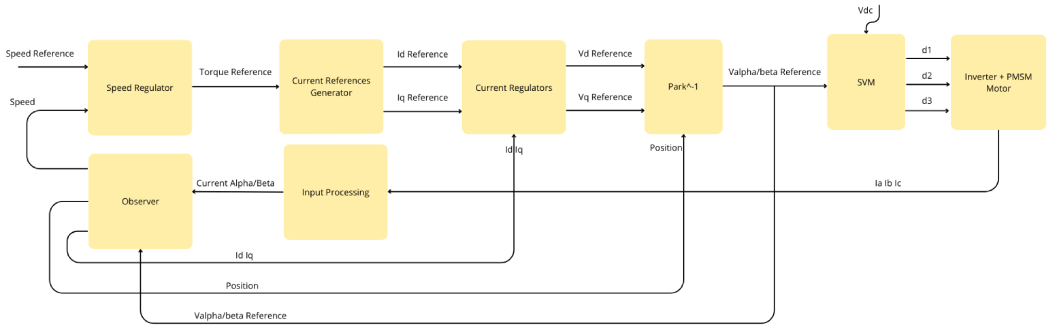


Figure 3.4: Control scheme

3.1.1 Speed Regulator

Whirlpool uses a speed controller that is based on a standard PI controller. This controller receives the speed error as input, which is the difference between the reference speed and the measured speed, and outputs the reference torque (figure 3.5). The measured speed can either be estimated by the observer or obtained by the

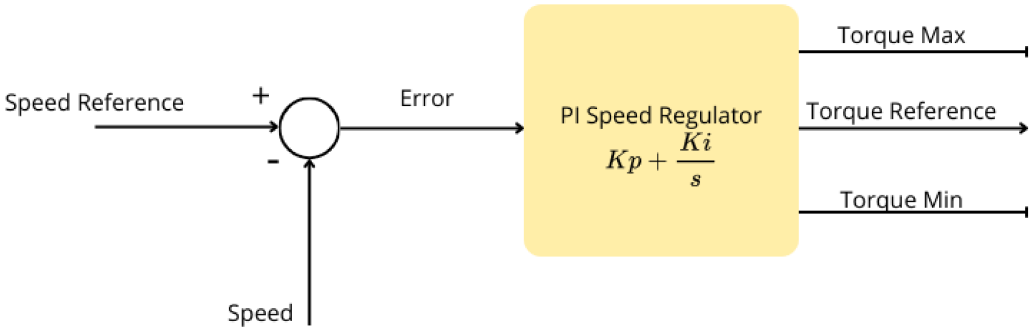


Figure 3.5: Whirlpool Speed Controller

sensor. Both speeds are expressed in mechanical radians per second. The involved parameters are:

1. the proportional gain K_p;
2. the integrative gain K_i.

In addition, the output of the PI regulator is upperly and lowerly limited to avoid exceeding the physical limits of the motor and board.

3.1.2 Reference Currents generator

The reference's torque is divided by K_t (the torque constant), to obtain the reference current I_q to be supplied as input to the current regulator (figure 3.6).

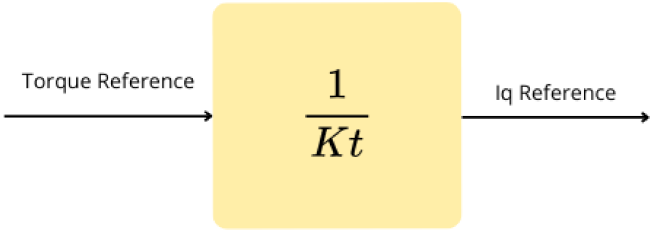


Figure 3.6: Whirlpool Current References Generator

3.1.3 Current Regulator

The current regulator made by Whirlpool is implemented by a standard PI regulator. This regulator takes as input the error obtained from the difference between the

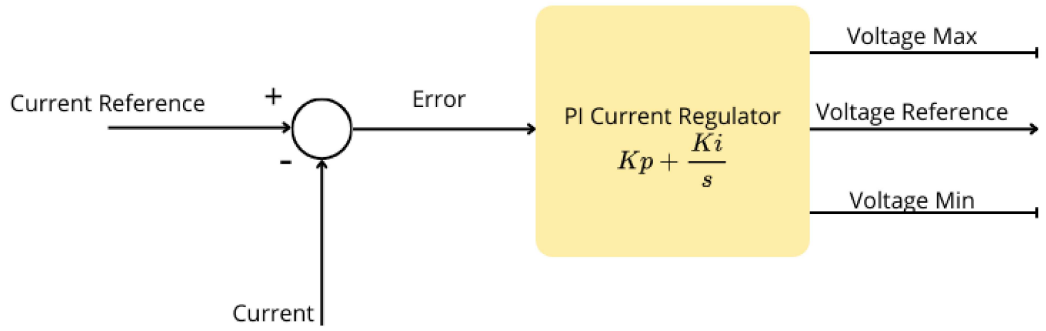


Figure 3.7: Whirlpool Currents Controller

reference current and the measured current and provides as output the reference voltage in d-q coordinates (figure 3.7). The involved parameters are::

1. Kp: proportional gain
2. Ki: integrative gain

In addition, it is important to limit the output of the PI regulator above and below the available voltage on the board.

3.1.4 Transformations

After acquiring the reference voltages in the d-q coordinates, they are transformed into the α-β coordinates by performing an inverse Park transform. These voltages are then used for the Space Vector Modulation, which takes the input voltages in α-β coordinates along with the voltage V_{dc} and generates the PWM pulses to be supplied as input to the inverter. Finally, there is an inverter connected to a synchronous permanent magnet motor (figure 3.8).

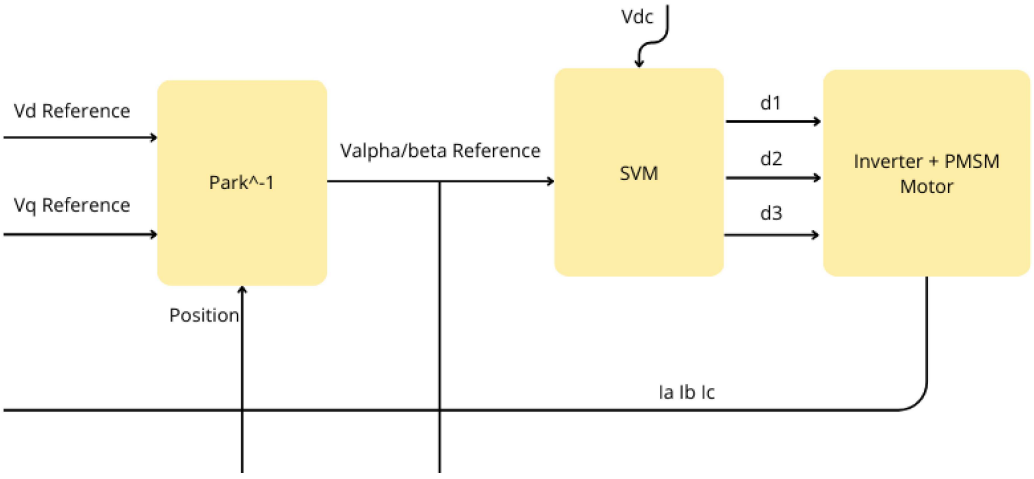


Figure 3.8: Inverse Park Transformation

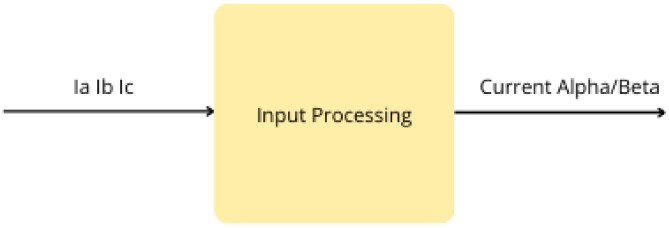


Figure 3.9: Input Processing

These steps are essential for obtaining the currents in the ABC coordinates.

3.1.5 Input Processing

This module converts currents from ABC coordinates to α-β coordinates using a direct Clarke transform.

3.1.6 Observer

The Whirlpool observer utilizes a standard Luenberger observer, where the error between the estimated and measured quantity is multiplied by the gain G (figure 3.10).

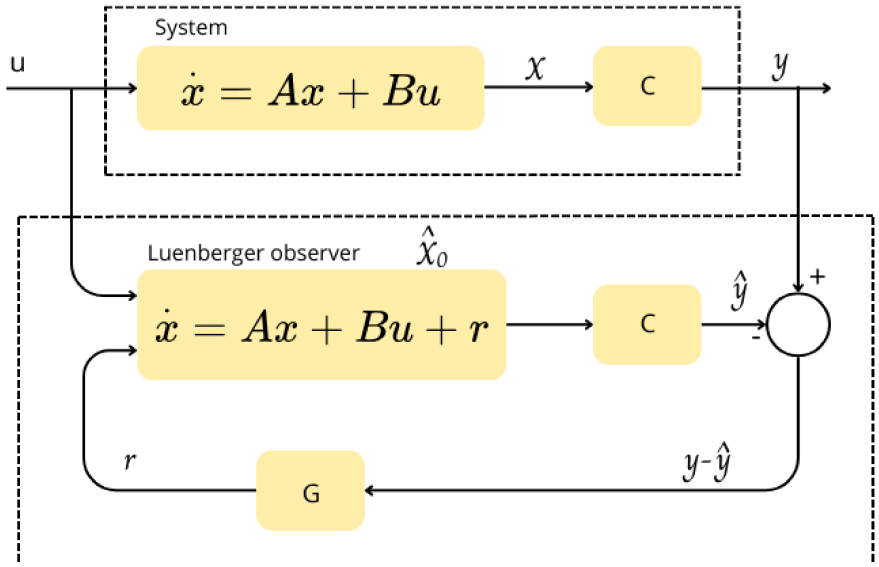


Figure 3.10: Whirlpool Observer

The input includes the motor's currents and voltages. The output provides the estimated speed, position, and force against the electromotive. The involved parameters are:

1. inductances L_d and L_q;
2. magnetic flux;
3. phase resistance;
4. gain G.

3.2 Motor Control

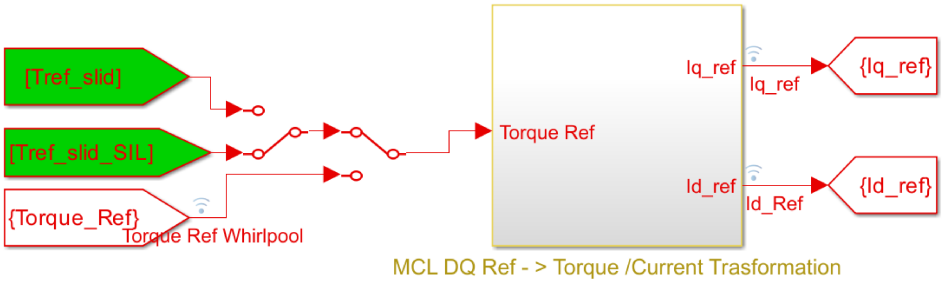


Figure 3.11: Selection of the torque value for calculating the reference currents

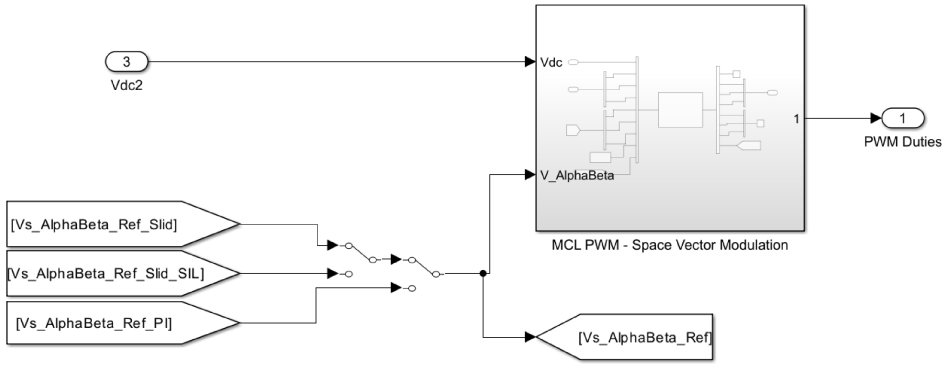


Figure 3.12: Selection of the value of the α-β voltages for the SVM

As anticipated in the first section of the chapter, the "Motor Control" block contains the speed controller and the current controllers. In version 22 of the software, there are:

1. the regulators implemented with Sliding Mode technique;
2. needed block for the software in the loop (SIL) of the regulator's autogenerated code;
3. the regulators implemented by Whirlpool, whose performance will be compared with that of the sliding mode controllers in chapter 8.

The blocks designed for the self-generation of the C code are described in detail in "Sensorless control of a permanent magnet synchronous motor: experimental validation on an appliance" by Laura Moretti [1].

The simulation is designed to switch from one controller to another for both the speed and current regulator (figures 3.11 and 3.12). The controllers implemented with the sliding mode technique, which concern the discussion, are contained in two subsystems whose description is given below.

3.3 Design of the speed controller subsystem in Simulink

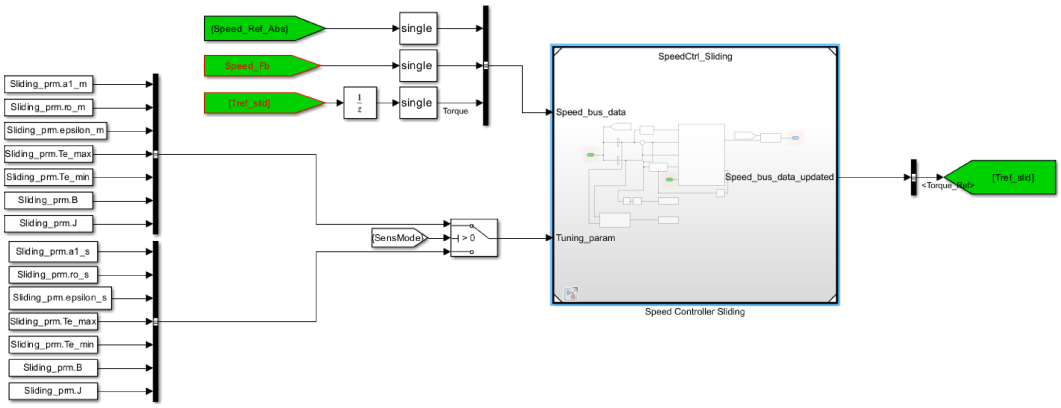


Figure 3.13: Speed controller subsystem

The speed regulator is contained within the "Speed Controller Sliding" subsystem (figure 3.13) which has the following inputs:

- **"Speed_bus_data"**: it is a bus consisting of $Speed\_Ref\_Abs$, $Speed\_Fb$ and $Tref\_slid$ that are respectively reference speed, measured or estimated speed and the reference torque calculated with the sliding mode technique;
- **"Tuning_param"**: it is the result of selection between two buses which contain speed control law's tuning parameters (a_1, ρ and ε), the reference torque upper bound Te_max, its lower bound Te_min, the coefficient of viscous friction B and the mechanical inertia of the motor and load J. The first bus starting from the top contains sensored mode tuning parameters, while the second bus the sensorless mode ones; these buses are identified and selected by the switch before the subsystem. The switch selects the top bus when $SensMode$ is equal to 1 and vice versa if it is equal to 0.

"Speed_bus_data_updated" is the output of the subsystem; the reference torque calculated can be extracted from this bus through a Bus Selector.
In the subsystem it is easy to find the MATLAB Function "Speed Controller"

which contains the sliding mode control law; the needed mathematical steps for its synthesizing are in chapter 5.

3.4 Design of the current regulator subsystem in Simulink

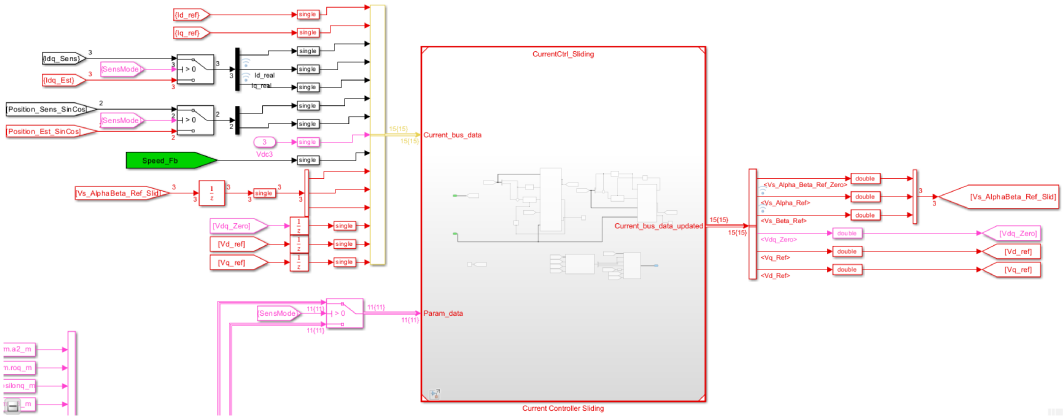


Figure 3.14: Current regulator subsystem

Current I_q and I_d controllers are inside the subsystem "Current Controller Sliding (figura 3.14) and it has the following inputs:

- **"Current_bus_data"**: it is a bus composed by the reference currents Id_ref and Iq_ref, the measured or estimated currents I_d and I_q, measured or estimated rotor position ($Position\_Sens\_SinCos$ and $Position\_Est\_SinCos$), the voltage Vdc, the measured or estimated speed $Speed\_Fb$, the reference voltages α-β, the sequence zero voltage Vdq_Zero and lastly the reference voltages Vd_ref and Vq_ref.

- **"Param_data"**: it is the result of the selection between two buses which contain both currents control laws tuning parameters ($a2$, $a3$, roq, rod, $espilonq$, $epsilond$) and electrical control parameters ($lambda0$, R, Lq, Ld). The bus is shown in figure 3.15.

"Current_bus_data_updated" is the output of the subsystem; the reference voltages α-β, Vd_ref, Vq_ref and their sequence zero voltages can be extracted from the above-mentioned bus.
In the subsystem it is easy to find the MATLAB Functions "Iq Controller" and "Id Controller" which contain the sliding mode control laws; the needed mathematical steps for their synthesizing are in chapter 6.

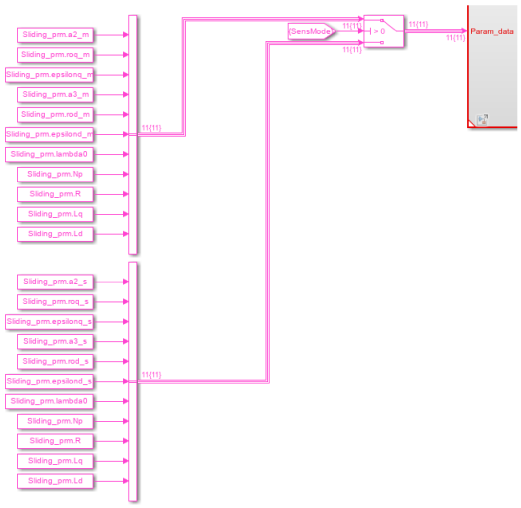


Figure 3.15: *Param_data* buses with switch

3.5 MATLAB Files

Before proceeding to the discussion, it is useful to briefly describe the MATLAB scripts that allowed the simulator to be adapted to control needs using the sliding mode technique. The scripts that will be reported in this section provide the structure of the aforementioned buses, as well as their initialization through tuning parameters and the electrical and mechanical motor control parameters. The simulator consists of a group of folders, in particular, the "main_model" folder contains all Simulink files and the scripts of interest (figure 3.16).

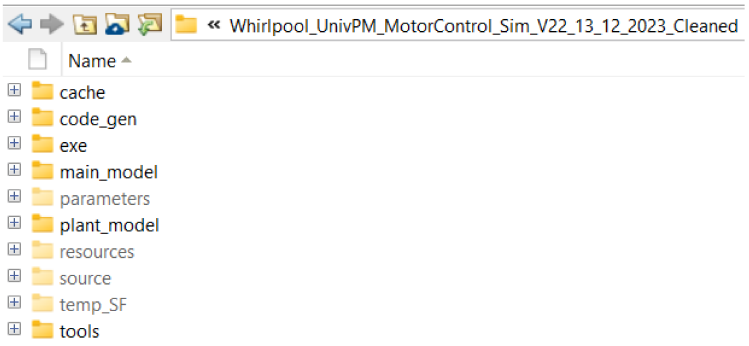


Figure 3.16: Project directory

3.5.1 BusInit_CodeGen.m

This script defines the structure of the buses used for the speed regulator and the current regulators. Once the structure of the buses is defined, the script "parameters_sliding_init.m" initializes them as shown in the next subsection. The speed controller has two buses:

- "MCL_SPEED_SMC_IO_F_TYPE": it contains the reference speed $Speed\_Rot\_Ref$, the measured or estimated speed $Speed\_Rot$ and the reference torque $Torque\_Ref$;
- "MCL_SPEED_SMC_PARAMS_TYPE": it contains the tuning parameters $a1$, ro and $epsilon$, the reference torque saturation bounds Te_max and Te_min, the the coefficient of viscous friction B and the mechanical inertia of the motor and load J.

These buses are both inputs of the subsystem "Speed_Controller_Sliding" described previously (section 3.3); the first one has three signals and the second one has seven signals. The current controller has also two buses:

- "MCL_CURRENT_SMC_IO_F_TYPE": it contains the reference currents Id_ref and Iq_ref, the zero sequence current Idq_zero, Id and Iq currents, sine and cosine values to apply the Park transform, the voltage Vdc, the measured or estimated speed, the triple of voltages obtained from the inverse Park transform ($Vs\_Alpha\_Beta\_Ref\_Zero$, Vs_Alpha_Ref, Vs_Beta_Ref) and the triple of voltages in Park domain (Vdq_Zero, Vd_Ref, Vq_Ref);
- "MCL_CURRENT_SMC_PARAMS_TYPE": it contains the tuning parameters of both current controllers (respectively $a2$, roq, $epsilonq$ and $a3$, rod, $epsilond$), the electrical parameters as the flux linkage of the permanent magnet $lambda0$, the number of pole pairs Np, the winding resistance R and the winding inductances Ld and Lq.

The first bus has fifteen signals, while the second one has eleven of them. These buses are the inputs of the subsystem "Current_Controller_Sliding".

```
%%%%%%%%%%%%%%%%%%%%%%% Speed COntroller %%%%%%%%%%%%%%%%%%%%%
speed_ctrl_io.Speed_Rot_Ref = single(0);
speed_ctrl_io.Speed_Rot =single(0);
speed_ctrl_io.Torque_Ref = single(0);

Simulink.Bus.createObject(speed_ctrl_io)
MCL_SPEED_SMC_IO_F_TYPE = slBus1;
clear slBus1
clear speed_ctrl_io

speed_ctrl_prm.a1= single(0);
speed_ctrl_prm.ro=single(0);
speed_ctrl_prm.epsilon =single(0);
speed_ctrl_prm.Te_max =single(0);
speed_ctrl_prm.Te_min = single(0);
speed_ctrl_prm.B = single(0);
speed_ctrl_prm.J = single(0);

Simulink.Bus.createObject(speed_ctrl_prm)
```

```
MCL_SPEED_SMC_PARAMS_TYPE = slBus1;
clear slBus1
clear speed_ctrl_prm

%%%%%%%%%%%%%%%% Current Controller %%%%%%%%%%%%%%%%%%

current_ctrl_io.Id_ref = single(0);
current_ctrl_io.Iq_ref = single(0);
current_ctrl_io.Idq_zero =single(0);
current_ctrl_io.Id =single(0);
current_ctrl_io.Iq =single(0);
current_ctrl_io.Sin =single(0);
current_ctrl_io.Cos = single(0);
current_ctrl_io.Vdc = single(0);
current_ctrl_io.Speed = single(0);

current_ctrl_io.Vs_Alpha_Beta_Ref_Zero = single(0);
current_ctrl_io.Vs_Alpha_Ref = single(0);
current_ctrl_io.Vs_Beta_Ref = single(0);
current_ctrl_io.Vdq_Zero= single(0);
current_ctrl_io.Vd_Ref= single(0);
current_ctrl_io.Vq_Ref=single(0);

Simulink.Bus.createObject(current_ctrl_io)
MCL_CURRENT_SMC_IO_F_TYPE = slBus1;
clear slBus1
clear current_ctrl_io


current_ctrl_prm.a2= single(0);
current_ctrl_prm.roq =single(0);
current_ctrl_prm.epsilonq =single(0);
current_ctrl_prm.a3 =single(0);
current_ctrl_prm.rod =single(0);
current_ctrl_prm.epsilond =single(0);
current_ctrl_prm.lambda0 =single(0);
current_ctrl_prm.Np =single(0);
current_ctrl_prm.R =single(0);
current_ctrl_prm.Lq =single(0);
current_ctrl_prm.Ld =single(0);

Simulink.Bus.createObject(current_ctrl_prm)
MCL_CURRENT_SMC_PARAMS_TYPE = slBus1;
clear slBus1
clear current_ctrl_prm
```

3.5.2 parameters_sliding_init.m

This script's role is to initialize all the parameters contained in the buses previously described. Tuning parameters are empirically selected to obtain satisfying performances of the speed and currents controllers. Signals behavior is also very sensitive to the flux linkage and the torque constant's values in fact, the values assigned to them are slightly different from those obtained by applying the formula that links them. The last part of the script can be uncommented for the robustness testing; this topic will be discussed in detail in the chapter 8.

```
%%%%%%%%%%%%%%%%% Speed Parameters %%%%%%%%%%%%%%%%%%%
% tuning parameters SENSORLESS
Sliding_prm.a1_s= 90;
Sliding_prm.ro_s = 0.08;
Sliding_prm.epsilon_s = 400;

% tuning parameters SENSORED
Sliding_prm.a1_m= 10;
Sliding_prm.ro_m = 0.5;
Sliding_prm.epsilon_m = 200;

% mechanical parameters
Sliding_prm.Te_max = 0.070000000298023;
Sliding_prm.Te_min = -0.009999999776483;
Sliding_prm.B=7.40e-05;
Sliding_prm.J=2.130e-06;

%%%%%%%%%%%%%%%% Current Parameters%%%%%%%%%%%%%%%%%%%%
% tuning parameters SENSORLESS
%Iq
Sliding_prm.a2_s = 500;
Sliding_prm.roq_s = 5;
Sliding_prm.epsilonq_s = 10;
%Id
Sliding_prm.a3_s = 500; %500
Sliding_prm.rod_s = 5; %5
Sliding_prm.epsilond_s = 3000; %3000

% tuning parameters SENSORED
%Iq
Sliding_prm.a2_m = 100;
Sliding_prm.roq_m= 20;
Sliding_prm.epsilonq_m = 10;
%Id
Sliding_prm.a3_m = 1000;
Sliding_prm.rod_m = 136;
Sliding_prm.epsilond_m = 3000;
```

```
38
39 % electrical parameters
40 Sliding_prm.lambda0 = 0.0857;
41 Sliding_prm.Np=1;
42 Sliding_prm.R=45.5;
43 Sliding_prm.Lq=0.120;
44 Sliding_prm.Ld=0.120;
45 Motorparams.K_Torque = 0.128;
46
47 % Motorparams.K_Torque = (3/2)*(Sliding_prm.lambda0)*(
       Sliding_prm.Np);
48
49
50 Sliding_prm.prm0 = 0;
51 Sliding_prm.prm1 = 0;
52
53 %%%%%%%%%%%% Automatic Robustness Tests %%%%%%%%%%%%%%%%%
54 %tableData = table('Size', [0, 2],'VariableTypes', {'double', '
       double'}, 'VariableNames', {'IAE', 'MSE'});
55
56 %for i = 1:32
57 %     [Rs,Ld,Lq,Phim,J,B] = robustness_test(i);
58 %     Motor_prm.Electrical.Rs_0= Rs;
59 %     Motor_prm.Electrical.Ld_0= Ld;
60 %     Motor_prm.Electrical.Lq_0= Lq;
61 %     Motor_prm.Electrical.PM_flux_h1_0= Phim;
62 %     Motor_prm.Mechanical.B= B;
63 %     Motor_prm.Mechanical.Jrotor= J;
64 % sim('MCU_Simulation_Architecture_Sliding_CodeGen_13_12_2023.
       slx',10);
65
66 %     newRow = table(IAE, MSE, 'VariableNames', {'IAE', 'MSE'});
67 %     tableData = [tableData; newRow];
68 %end
69 % excelFileName = 'IAE_MSE.xlsx';
70 %writetable(tableData, excelFileName, 'Sheet', 'Sheet1');
```

Chapter 4

Theory and preliminary notions

This chapter's purpose is to describe the needed knowledge for the realization of the permanent magnet synchronous motor control with the sliding mode technique. The first section defines electrical and mechanical equations which are fundamentals to design the model-based control. The second section describes the sliding mode control in detail: its mathematical formulation, some of its characteristics and method's issues. There is also a useful example to follow step by step to easily implement the control laws of the speed and the currents regulators.

4.1 Motor model and dynamics

This section contains the model of a permanent magnet synchronous motor. [2]:

> In the (d, q) reference frame, synchronously rotating with the motor rotor, the electrical equations of motion of a PMSM can be written as:
>
> $$\begin{aligned} \frac{di_d}{dt} &= -\frac{R}{L}i_d + \omega_e i_q + \frac{1}{L}u_d \\ \frac{di_q}{dt} &= -\frac{R}{L}i_q + \omega_e i_d - \frac{1}{L}\lambda_0\omega_e + \frac{1}{L}u_q \end{aligned} \tag{4.1}$$
>
> where i_d and i_q are the d-axis and q-axis stator currents, respectively; u_d and u_q are the d-axis and q-axis stator voltages, respectively; R is the winding resistance and $L = L_d = L_q$ is the winding inductance on axis d and q; λ_0 is the flux linkage of the permanent magnet and ω_e is the electrical angular speed of the motor rotor. The electrical torque τ_e and the mechanical power P of the motor are given by
>
> $$\tau_e = K_t i_q; \quad P = \tau_e \omega_r \tag{4.2}$$
>
> in which $K_t = (3/2)\lambda_0 N_r$ is the torque constant with N_r the number of pole pairs and ω_r is the mechanical angular speed of the motor rotor. [...] The mechanical motion of the motor is described by

$$J\frac{d\omega_r}{dt} + B\omega_r = \tau_e - \tau_l$$
$$\frac{d\theta_r}{dt} = \omega_r \tag{4.3}$$

where J is the mechanical inertia of the motor and load, B is the coefficient of viscous friction, τ_l is the load torque and θ_r denotes the mechanical angular position of the motor rotor. For the electrical angular position/speed and the mechanical angular position/speed, the following relations hold:

$$\omega_e = N_r\omega_r; \quad \theta_e = N_r\theta_r. \tag{4.4}$$

4.2 Sliding Mode Control (SMC)

The Sliding Mode Control (SMC) is a non linear and robust control technique. Sliding mode control laws allow to preserve performances regardless of system disturbances presence. It is important to specify that model's uncertainties invariance should not be taken for granted: differences between the real and the ideal system's behavior are source of undesirable phenomena in real life application; nevertheless, some control law's adjustments are able to reduce the occurring of these problems, preserving a substantial robustness. The SMC is a Variable Structure Control (VSC) [3]:

> Variable Structure Control (VSC) is a viable high-speed switching feedback control [...]. VSC utilizes a high-speed switching control law to drive the nonlinear plant's state trajectory onto a specified and user-chosen surface in the state space (called the sliding or switching surface), and to maintain the plant's state trajectory on this surface for all subsequent time. [...] By proper design of the sliding surface, VSC attains the conventional goals of control such as stabilization, tracking, regulation, etc.

Aforesaid concepts can be formalized as follows [4].

> Given the system described by this vector differential equation:
>
> $$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{t}) \tag{4.5}$$
>
> where $\mathbf{x}$, $\mathbf{f} \in \mathbb{R}^{\mathbf{n}}$, $\mathbf{u} \in \mathbb{R}^{\mathbf{m}}$, $\mathbf{t} \in \mathbb{R}_+$. It is also given the vector equation:
>
> $$\mathbf{s}(\mathbf{x}) = \mathbf{0} \tag{4.6}$$
>
> with $\mathbf{s}(\mathbf{x}) : \mathbb{R}^{\mathrm{n}} \longrightarrow \mathbb{R}^{\mathrm{m}}$, that is $\mathbf{s}(\mathbf{x}) = [\ \mathrm{s}_1(\mathbf{x})\ ....\ \mathrm{s}_{\mathrm{m}}(\mathbf{x})\ ]^{\mathrm{T}}$; it is assumed that the m scalar equations $\mathbf{s_i}(\mathbf{x}) = 0$ are linearly independent. The prob-

lem consist of searching for a control law $\mathbf{u}(\mathrm{x},\mathrm{t}) = [\mathrm{u}_1(\mathbf{x},\mathrm{t}) \dots \mathrm{u}_\mathrm{m}(\mathbf{x},\mathrm{t})]^\mathrm{T}$ of this type:

$$\mathrm{u_i}(\mathbf{x},\mathrm{t}) = \begin{cases} \mathrm{u_i^+}(\mathbf{x};\mathrm{t}) & \text{if} \quad \mathrm{s_i}(\mathbf{x}) > 0 \\ \\ \mathrm{u_i^-}(\mathbf{x};\mathrm{t}) & \text{if} \quad \mathrm{s_i}(\mathbf{x}) < 0 \end{cases} \tag{4.7}$$

($\mathrm{i} = 1 \dots \mathrm{m}, \mathrm{u_i^+}(\mathbf{x};\mathrm{t}), \mathrm{u_i^-}(\mathbf{x};\mathrm{t})$ continous functions, $\mathrm{u_j^+}(\mathbf{x};\mathrm{t}) \neq \mathrm{u_j^-}(\mathbf{x};t)$ on $\mathrm{s_i}(\mathbf{x}) = 0$) so that, from a certain istant t_s onwards, the condition $\mathbf{s}(\mathbf{x}) = 0$ is verified, namely the plant's state is moving along the m surfaces intersection. The control law's synthesis occurs in two distinct phases: first, the equation $\mathbf{s}(\mathbf{x}) = 0$ is chosen, thus, a region of dimension (n-m) is selected, where the sliding mode is desired to occur; this region's choice is related to the system's requirements. Then, the control functions $\mathrm{u_i}(\mathbf{x},\mathrm{t})$ with $i = 1...\mathrm{m}$, must be selected to satisfy the equation 4.6 from t_s onwards.

4.2.1 Sliding Mode conditions for existence and equations

Considering a system with a scalar control input (m=1) with $\mathbf{s}(\mathbf{x}) : \mathbb{R}^\mathrm{n} \longrightarrow \mathbb{R}$, these are the local sliding mode conditions for existence:

$$\lim_{s\to 0^-} \dot{s} > 0, \quad \lim_{s\to 0^+} \dot{s} < 0 \quad \Leftrightarrow \quad \lim_{s\to 0}(s\dot{s}) < 0 \tag{4.8}$$

They ensure the sliding mode occurring when initial states are in the vicinity of the sliding surface. The Lyapunov method is a valid alternative to define the scalar conditions for existence: an equivalent expression for the condition in equation 4.8 is the definite positive function $\frac{1}{2}\mathrm{s}^2(\mathbf{x})$ and its definite negative derivative. In general, when $\mathbf{u} \in \mathbb{R}^\mathrm{m}$, the following definite positive Lyapunov function is used:

$$\mathrm{V}(\mathbf{s}(\mathbf{x})) = \frac{1}{2}\mathbf{s}^\mathrm{T}(\mathbf{x})\mathbf{s}(\mathbf{x}) \tag{4.9}$$

with its definite negative derivative

$$\dot{\mathrm{V}}(\mathbf{s}(\mathbf{x})) = \mathbf{s}^\mathrm{T}(\mathbf{x})\dot{\mathbf{s}}(\mathbf{x}) < 0 \tag{4.10}$$

and a control function in the same form of the equation 4.7. The control function must fullfil the inequality 4.10 for each x [4]. To define the sliding mode equation, Filippov's theory of differential equations with discontinuous right-hand sides is used [5]:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathrm{t}) + \mathbf{B}(\mathbf{x},\mathrm{t})\mathbf{u}(\mathbf{x},\mathrm{t}) \tag{4.11}$$

but this form is only proper for system with linear control. Then the equivalent

control method is embraced:

$$\frac{\partial \mathbf{s}}{\partial \mathbf{x}}\dot{\mathbf{x}} = 0 \tag{4.12}$$

where $\frac{\partial \mathbf{s}}{\partial \mathbf{x}}$ is the Jacobian of $s(x)$. Replacing the 4.11 into the 4.12:

$$\frac{\partial \mathbf{s}}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x},\mathrm{t}) + \frac{\partial \mathbf{s}}{\partial \mathbf{x}}\mathbf{B}(\mathbf{x},\mathrm{t})\mathbf{u}(\mathbf{x},\mathrm{t}) = 0 \tag{4.13}$$

explicating $\mathbf{u}$ and assuming the matrix $\left[\frac{\partial s}{\partial x}\mathbf{B}(\mathbf{x},\mathrm{t})\right]$ inverse exists,

$$\mathbf{u_{eq}} = -\left[\frac{\partial \mathbf{s}}{\partial \mathbf{x}}\mathbf{B}\left(\mathbf{x},\mathrm{t}\right)\right]^{-1}\frac{\partial \mathbf{s}}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x},\mathrm{t}) \tag{4.14}$$

The $\mathbf{u_{eq}}$ function is the equivalent continuous control and replacing it in 4.11, it returns the ideal sliding mode equation:

$$\dot{\mathbf{x}} = \left\{\mathbf{I} - \mathbf{B}\left(\mathbf{x},t\right)\left[\frac{\partial \mathbf{s}}{\partial \mathbf{x}}\mathbf{B}\left(\mathbf{x},\mathrm{t}\right)\right]^{-1}\frac{\partial \mathbf{s}}{\partial \mathbf{x}}\right\}\mathbf{f}(\mathbf{x},\mathrm{t}) \tag{4.15}$$

where $\mathbf{I}$ is the identity matrix of order n. If $\frac{\partial \mathbf{s}}{\partial \mathbf{x}}\mathbf{B}(\mathbf{x},\mathrm{t})$ is singular:

- the equivalent control is not unique and but the sliding mode equation continues being unique;
- or the equivalent control does not exist so no sliding motions of the state occur.

The sliding mode is invariant to non singular transformations of the sliding surface or of the control vector; given this transformation

$$\mathbf{s}^*(\mathbf{x}) = \mathbf{H}_\mathrm{s}(\mathbf{x},\mathrm{t})\mathbf{s}(\mathbf{x}) \tag{4.16}$$

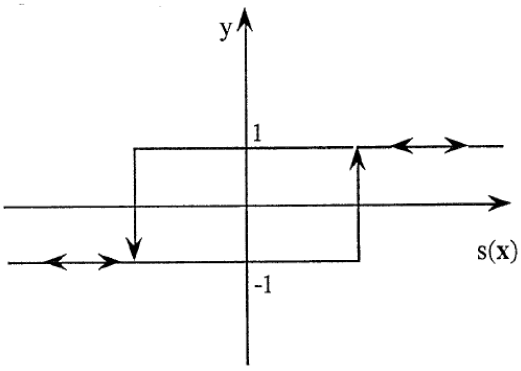
where $\mathbf{H}_\mathrm{s}(\mathbf{x},\mathrm{t})$ is a mxm matrix with $\det[\mathbf{H_s}(\mathbf{x},\mathrm{t})] \neq 0$, the sliding mode equation 4.15 is still valid if the m control vector components are discontinuous on the surfaces $\mathrm{s}_\mathrm{i}^*(\mathbf{x}) = 0$, with $i = 1, ..., m$. Also defining

$$\mathbf{u}^*(\mathbf{x}) = \mathbf{H_u}(\mathbf{x},\mathrm{t})\mathbf{u}(\mathbf{x}) \tag{4.17}$$

where $\mathbf{H_u}(\mathbf{x},\mathrm{t})$ is a mxm matrix with $\det[\mathbf{H_u}(\mathbf{x},\mathrm{t})] \neq 0$, the sliding equation does not change if the m components of the new control vector $\mathrm{u}_\mathrm{i}^*(\mathbf{x})$ are discontinuous on the surfaces $\mathrm{s}_\mathrm{i}^*(\mathbf{x}) = 0$. These properties simplify control function synthesis when condition like the 4.10 are set. In fact the last two equations (4.16 and 4.17) allow to choose control vector m components independently. thus the problem can be solved m monodimensional problems [4].

4.2.2 VSC issues and characteristics

The sliding motion is an ideal condition; actually there are trajectory oscillations in the neighborhood of $\mathbf{s}(\mathbf{x}) = \mathbf{0}$ because of the non ideal switching mechanism, which is not physically feasible due to the infinite switching frequency it would entail. The delay introduced by the switching frequency also causes the state to not align promptly with the sliding surface, resulting in chattering: the generation of high-frequency oscillations caused by switching (figure 4.2). The switching mechanism can be characterized by a hysteresis cycle of amplitude Δ (figure 4.1) where $s(x) = 0$ is supposed scalar. Because of u(x.t) behavior described in equation 4.18, trajectories that start from a point where $s(x) < 0$, do not stay in the place of points $s(x) = 0$: trajectories continue to $s(x) = +\Delta$ then go back to $s(x) = -\Delta$ and so on, generating oscillations. It is evident that for smaller Δ values, there are higher frequence and lower chattering amplitude. When $\Delta = 0$ (ideal switching), frequence would be infinite and chattering amplitude would be zero; it means the state would perfeclty "slide" on the surface $s(x) = 0$.



$$u(x,t) = \begin{cases} u^{+}(x,t) & \text{if} \quad y = 1 \\ u^{-}(x,t) & \text{if} \quad y = -1 \end{cases} \tag{4.18}$$

Figure 4.1: Hysteresis cycle [4]

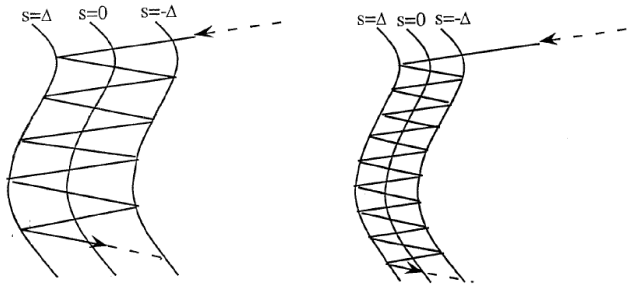


Figure 4.2: Chattering [4]

Requiring the state proximity to the sliding surface rather than its perfect sliding motion, chattering phenomena can be reduced [6]. Considering the scalar case:

$$|s(\mathbf{x})| \leq \varepsilon \tag{4.19}$$

an area called "boundery layer" is defined, with $\varepsilon > 0$ (figure 4.3). Outside the boundary layer the control function is defined as:

$$\mathbf{u}(\mathbf{x},\mathrm{t}) = \begin{cases} \mathbf{u}^{+}(\mathbf{x};\mathrm{t}) & \text{if} \quad \mathrm{s}(\mathbf{x}) > \varepsilon \\ \\ \mathbf{u}^{-}(\mathbf{x};\mathrm{t}) & \text{if} \quad \mathrm{s}(\mathbf{x}) < -\varepsilon \end{cases} \tag{4.20}$$

so that the state reaches the boundary layer and there remains. Inside the boundary layer, the control function is defined as:

$$\mathrm{u}(\mathbf{x},\mathrm{t}) = \begin{cases} \mathrm{k} \cdot \mathrm{sign}[\mathrm{s}(\mathbf{x})] & \text{if} \quad |\mathrm{s}(\mathbf{x})| > \varepsilon \\ \\ \mathrm{k} \cdot \frac{\mathrm{s}(\mathbf{x})}{\varepsilon} & \text{if} \quad |\mathrm{s}(\mathbf{x})| \leq \varepsilon \end{cases} \tag{4.21}$$

Using this method, the control law depends on ε value that should be selected searching for the best trade-off between reducing chattering and a fullfilling control requirements [4].

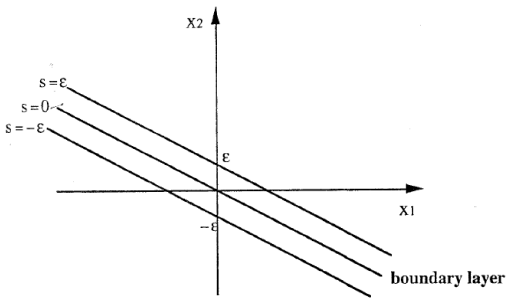


Figure 4.3: Sliding surface and its Boundary Layer

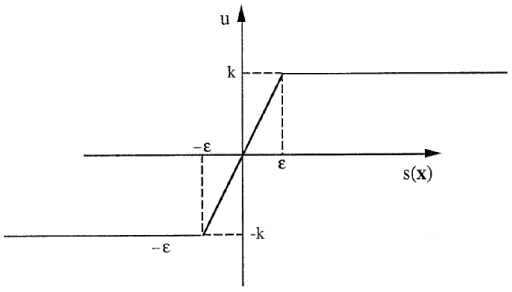


Figure 4.4: Control law with boundary layer

4.2.3 Choosing a sliding surface

The sliding surface is time dependent even thought it was not specified in the previous sections. The sliding surface selection geometrically defines control requirements. Given a tracking problem, it is assumed that the system output is $\mathbf{y} = \mathbf{h}(\mathbf{x})$, with

$\mathbf{y}, \mathbf{h} \in \mathbb{R}^{\mathrm{m}}$. Indicating the reference output as $\mathbf{y_d}$ and the tracking error as $\mathbf{e} = \mathbf{y} - \mathbf{y_d}$ the sliding surface is defined as follows:

$$\mathbf{s}(\mathbf{x}, \mathrm{t}) = \dot{\mathbf{e}} + \Lambda \mathbf{e} = \mathbf{0} \quad \text{with} \quad \Lambda = \mathrm{diag}(\lambda_\mathbf{i}), \quad \lambda_\mathbf{i} > 0, \quad \mathrm{i} = 1 \ldots \mathrm{m} \tag{4.22}$$

When the abovementioned equation is verified, each tracking error component $\mathbf{e_i}$ exponentially tends to zero with the time constant $\frac{1}{\lambda_\mathrm{i}}$ [4].

4.2.4 Sliding Mode application: a scalar case study

Considering this bidimensional system

$$\begin{cases} \dot{x}_1(t) = x_2 \\ \dot{x}_2(t) = f(x) + g(x)u \end{cases} \quad \text{with} \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^2 \tag{4.23}$$

where $x_1 \in \mathbb{R},\ x_2 \in \mathbb{R},\ u \in \mathbb{R}$ and $f : D \to \mathbb{R},\ g : D \to \mathbb{R}$ continuous functions with $D \subseteq \mathbb{R}^2$ as domain. The system is, for hypothesis, uncertain so the functions are defined as follows:

$$\begin{aligned} f(x) &= \underbrace{\hat{f}(x)}_{\text{nominal value}} + \underbrace{\Delta f(x)}_{\text{uncertainty}} \\ g(x) &= \underbrace{\hat{g}(x)}_{\text{nominal value}} + \underbrace{\Delta g(x)}_{\text{uncertainty}} \end{aligned} \tag{4.24}$$

The objective is to design a control law that can stabilize the state and make it reach the straight line $s = x_2(t) + a x_1(t) = 0 \quad \forall\ t \geq t_s$; this straight line is the chosen sliding surface and t_s is the instant of time in which the state reaches it. To guarantee that $s = 0$, a Lyapunov function $V(s)$ that always decrease in time ($\dot{V}(s) < 0$) is used:

$$V(s) = \frac{1}{2} s^2 \tag{4.25}$$

then

$$\dot{V}(s) = \frac{1}{2} s \cdot \dot{s} \cdot 2 = s \cdot \dot{s} \tag{4.26}$$

where $\dot{s}$ is the time derivative of $s = x_2 + a x_1$, consequently

$$\begin{aligned} \dot{V}(s) = s \cdot \dot{s} = s \cdot \frac{d}{dt}[x_2 + a x_1] = s[\dot{x}_2 + a\dot{x}_1] = \\ = s[f(x) + g(x)u + a x_2] = s g(x)[u + \frac{f(x) + a x_2}{g(x)}] \end{aligned} \tag{4.27}$$

since $f(x)$ and $g(x)$ are uncertain quantities, it is assumed that

$$\left|\frac{f(x)+ax_2}{g(x)}\right| \leq \rho \tag{4.28}$$

where ρ is a constant (but could be a known function $\rho(x)$). Thus from the equation 4.27 it is obtained the following expression

$$\begin{aligned}\dot{V}(s) &= sg(x)[u+\frac{f(x)+ax_2}{g(x)}] \leq g(x)u+|s|\left|\frac{f(x)+ax_2}{g(x)}\right|g(x) \\ &\leq sg(x)u+|s|g(x)\rho\end{aligned} \tag{4.29}$$

this control law is chosen

$$u=-\beta sign(s) \quad \text{with} \quad \beta=\rho+\beta_0 \quad \text{where} \quad \beta_0>0,\ \rho>0 \tag{4.30}$$

and replaced in the equation 4.29

$$\dot{V}(s) \leq sg(x)\{-[\rho(x)+\beta_0]sign(s)\}+|s|g(x)\rho=-|s|g(x)\beta_0 \tag{4.31}$$

then

$$s\cdot\dot{s} \leq -|s|g(x)\beta_0 \quad \text{con} \quad 0<g_0\leq g(x) \tag{4.32}$$

consequently

$$s\cdot\dot{s} \leq -g_0\beta_0|s| = -\alpha_0|s| \tag{4.33}$$

guaranteeing that $|s|$ always decreases and thus $s=0$ [7].

4.2.5 Equivalent Control Law

It is supposed that $f(x)$ is subject to changes $f(x)=\hat{f}(x)+\Delta f(x)$, while the function $g(x)$ is known $g(x)=\hat{g}(x)$. The equation 4.27 has the following form

$$\dot{V}(s)=sg(x)[u+\frac{\hat{f}(x)+\Delta f(x)+ax_2}{g(x)}] \tag{4.34}$$

The input has two components:

$$u=u_{eq}+v \tag{4.35}$$

- $u_{eq}=-\frac{\hat{f}(x)+ax_2}{g(x)}$ component with continuous and known dynamic;
- v discontinuous component.

Replacing u in the equation 4.34

$$\dot{V}(s) = sg(x)[-\frac{\hat{f}(x) + ax_2}{g(x)} + v + \frac{\hat{f}(x) + \Delta f(x) + ax_2}{g(x)}] = sg(x)[v + \frac{\Delta f(x)}{g(x)}] \quad (4.36)$$

it is assumed that

$$|\frac{\Delta f(x)}{g(x)}| \leq \rho_\Delta \quad (4.37)$$

and given that usually $|\Delta f(x)| < |f(x)|$, it is generally true that $\rho_\Delta(x) < \rho(x)$. Thus oscillations' amplitude is sufficiently reduced. While the discontinuous component is implemented as follows:

$$v = -\rho_\Delta \cdot sat(s) \quad (4.38)$$

Chapter 5

The speed controller design

In the first part of this chapter, the focus is on the calculation of the speed regulator's control law. The control law synthesis adheres to the theory presented in chapter 4. The second part of the chapter will cover the design of the controller in Simulink. Lastly, the third part provides the MATLAB Function's code which carries out the speed control action.

5.1 The speed control law

Considering the following system

$$\begin{cases} \dot{x}_1(t) = x_2 = \omega_{ref} - \omega_{mis} \\ \dot{x}_2(t) = f(x) + g(x)u = \dot{\omega}_{ref} - \dot{\omega}_{mis} \end{cases} \tag{5.1}$$

while the sliding surface is defined as

$$s = e + a_1 \int e \Longrightarrow s = x_2 + a_1 x_1 \tag{5.2}$$

where $e = \omega_{ref} - \omega_{mis}$ is the speed error and its integration is the position error. The control law represents the electrical torque.

$$u = \tau_e = J\dot{\omega}_{mis} + B\omega_{mis} + \tau_L \quad [2] \tag{5.3}$$

Explicating $\dot{\omega}_{mis}$

$$\dot{\omega}_{mis} = \frac{u - B\omega_{mis} - \tau_L}{J} + \delta(\omega) \tag{5.4}$$

the system is

$$\begin{cases} \dot{x}_1(t) = \omega_{ref} - \omega_{mis} \\ \dot{x}_2(t) = \dot{\omega}_{ref} - \frac{u}{J} + \frac{B}{J}\omega_{mis} + \frac{\tau_L}{J} - \delta(\omega) \end{cases} \tag{5.5}$$

so the functions $f(x)$ and $g(x)$ can be determined.

$$\begin{aligned} f(x) &= \dot{\omega}_{ref} + \frac{B}{J}\omega_{mis} + \frac{\tau_L}{J} - \delta(\omega) \\ g(x) &= -\frac{1}{J} \end{aligned} \tag{5.6}$$

Consequently this is the Lyapunov function derivative (from the equation 4.27)

$$\dot{V}(s) = s(-\frac{1}{J})[u - J\dot{\omega}_{ref} - B\omega_{mis} - \tau_L + J\delta(\omega) - Ja_1(\omega_{ref} - \omega_{mis})] \tag{5.7}$$

and the control law is obtained in the same form as equation 4.35.

$$u = \underbrace{B\omega_{mis} + J[\dot{\omega}_{ref} + a_1(\omega_{ref} - \omega_{mis})]}_{u_{eq}} - \underbrace{\rho \cdot sat(s)}_{v} \tag{5.8}$$

It removes all the known dynamic from the Lyapunov function derivative.

$$\dot{V}(s) = s(-\frac{1}{J})(v - \tau_L + J\delta(\omega)) \tag{5.9}$$

By adjusting the values of ε, a_1, and ρ, a tuning can be carried out.

5.1.1 The speed sliding surface

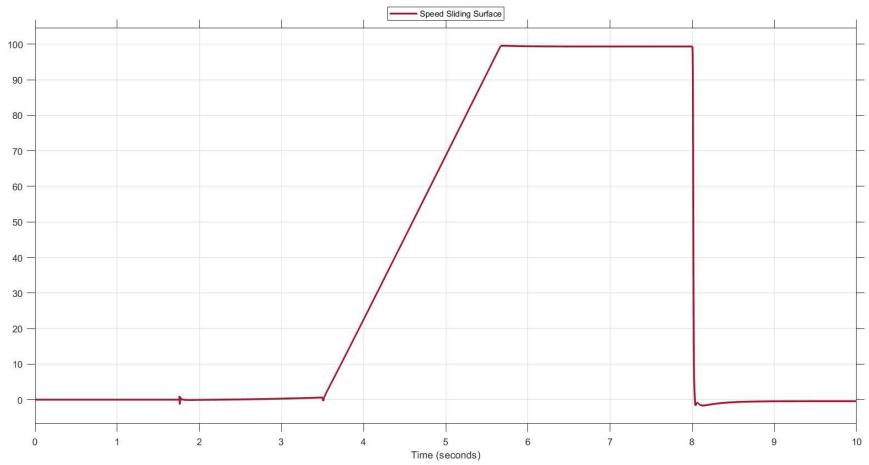


Figure 5.1: The speed sliding surface

The speed sliding surface is convergent: this means that a sliding mode motion of the speed error is correctly occurring (figure 5.1). In general, the sliding surface behavior depends on the chosen tuning and can improve the surface convergence's speed.

5.2 Speed Controller Sliding

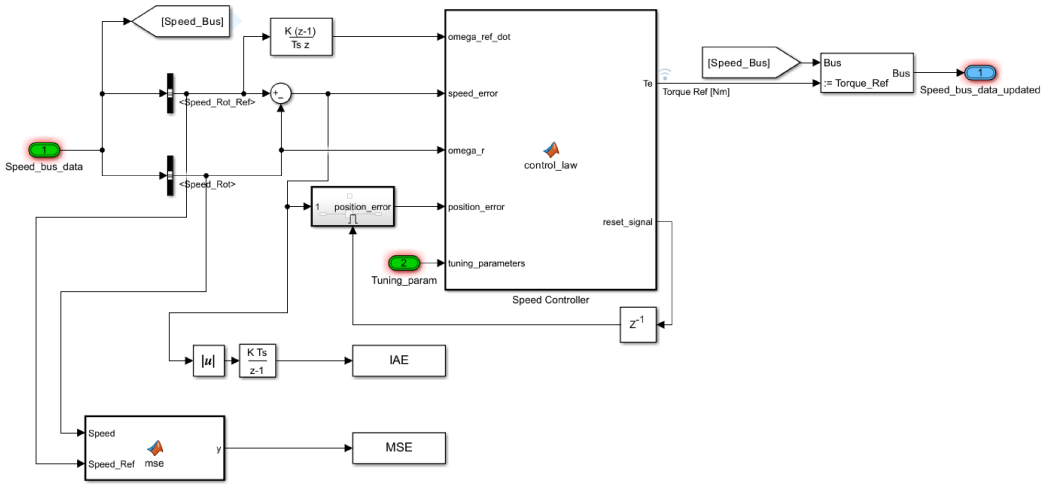


Figure 5.2: Speed regulator Simulink scheme

As anticipated in chapter 3, there is a MATLAB Function inside the subsystem "Speed Controller Sliding"; Inside the MATLAB Function, there is the control law previously implemented (figura 5.2). These are the block's inputs:

- *omega_ref_dot*: the motor's reference angular acceleration, calculated as the reference angular speed derivative provided by the "Speed_Bus";
- *speed_error*: the speed error, calculated as the difference between the reference speed and the measured/estimated speed (both extracted from the "Speed_Bus");
- *omega_r*: the motor's measured or estimated angular speed;
- *position_error*: the position error calculated as the integration of the speed error inside an Enabled Subsystem;
- *tuning_parameters*: a bus that contains the tuning parameters.

The position error is calculated inside an Enabled Subsystem, through which the integration is reset when the block receives the reset signal called *reset_signal* as input; it notifies the saturation of the torque preventing the windup of the integral action (see the next subsection 5.2.1). The *reset_signal* and the reference torque *Torque_Ref* are the outputs of the Speed Controller block. The reference torque value updates the torque value inside the "Speed_Bus" through the Bus Assignment block, then the updated bus is returned as output by the subsystem. Furthermore, within the subsystem, the **IAE** and **MSE** indices were calculated to carry out robustness tests. The robustness tests' methods and results are reported in chapter 8.

5.2.1 Integrator Windup

The controlled variable of every regulator always has upper and lower bounds. During the permanent scheme, the controlled variable's value is significantly distant from saturation's limits, but it could reach them during wide and rapid transients. When the controlled variable is saturated, the process evolves with constant input, without a proper regulation as if the process was not in a closed loop. The regulator also evolves as it was in an open loop, without a feedback effect between the output and the error. Furthermore, the integrator is a dynamic system not asymptotically stable; thus when the controlled variable's saturation occurs, the range of values the integrator takes could be uneffective as control action. It takes time before the integrator's value ceases to increase; this phenomena is known as the integrator windup. The conditional integration is an effective method to prevent the wind-up: the integration is switched off when the control is far from steady state. Integral action is thus only used when certain conditions are fullfilled, otherwise the integral term is kept constant [8].

5.3 The Speed Controller's code

"Speed Controller" contains the code in this section. The notation used in the code respects the one adopted for the mathematical calculation of the control law. Among the inputs of the MATLAB Function, there is the bus that contains the struct *tuning_parameters*, necessary for the initialization of the tuning parameters. Subsequently, the sliding surface s is defined and the saturation is calculated so that v can be set. It is useful to remember that v is one of the two components of the control law and note that this component depends on the sliding surface itself and the parameters ρ and ε. Immediately afterward the component u_{eq} and the control law u itself are defined. Finally, the saturation limits of the reference torque Te are established. The reference torque is the controller's output. Whenever the torque saturates, *reset_signal* takes the value -1 to disable the integral's action, otherwise it takes the value 1.

```
function [Te,reset_signal] = control_law(omega_ref_dot,
    speed_error,omega_r,position_error,tuning_parameters)

a1 = tuning_parameters.a1;
ro = tuning_parameters.ro;
epsilon = tuning_parameters.epsilon;
Te_max=tuning_parameters.Te_max;
Te_min=tuning_parameters.Te_min;
B=tuning_parameters.B;
J=tuning_parameters.J;

%sliding surface
```

```
12 s = speed_error + a1*position_error;
13
14 %sat(s)
15 if abs(s) > epsilon
16     v = ro*sign(s);
17 else
18     v = ro*(s/epsilon);
19 end
20
21 %control law
22 u_eq = B*omega_r + J*(omega_ref_dot+a1*(speed_error));
23 u = u_eq + v;
24
25 %sat(Te)
26 if u >= Te_max
27     u = Te_max;
28     reset_signal=-1;
29 elseif u <= Te_min
30     u = Te_min;
31     reset_signal=-1;
32 else
33     reset_signal=1;
34 end
35 Te=u;
```

Chapter 6

The currents controller design

The first part of this chapter is dedicated to calculating the control law of current regulator I_d and current regulator I_q, the synthesis of which follows the theory reported in chapter 4. The second part describes the design of the controllers in Simulink. Finally, the third part of the chapter provides detailed comments and reports on the MATLAB Functions code that executes the control action on the currents.

6.1 The current I_q control law

Considering the following system

$$\begin{cases} \dot{x}_1(t) = x_2 = I_{q_{ref}} - I_{q_{mis}} \\ \dot{x}_2(t) = f(x) + g(x)u = \dot{I}_{q_{ref}} - \dot{I}_{q_{mis}} \end{cases} \tag{6.1}$$

while the sliding surface is defined as follows

$$s = e + a_2 \int e \Longrightarrow s = x_2 + a_2 x_1 \tag{6.2}$$

where $e = I_{q_{ref}} - I_{q_{mis}}$. The control law coincides with the reference voltage $V_{q_{ref}}$.

$$u = V_{q_{ref}} = RI_{q_{mis}} + L\dot{I}_{q_{mis}} + \omega_e \varphi + \omega_e L I_{d_{mis}} \quad [2] \tag{6.3}$$

Explicating $\dot{I}_{q_{mis}}$

$$\dot{I}_{q_{mis}} = \frac{V_{q_{ref}}}{L} - \frac{RI_{q_{mis}}}{L} - \frac{\omega_e \varphi}{L} - \omega_e I_{d_{mis}} + \delta(I_q) \tag{6.4}$$

the system is

$$\begin{cases} \dot{x}_1(t) = I_{q_{ref}} - I_{q_{mis}} \\ \dot{x}_2(t) = \dot{I}_{q_{ref}} - \frac{V_{q_{ref}}}{L} + \frac{RI_{q_{mis}}}{L} + \frac{\omega_e \varphi}{L} + \omega_e I_{d_{mis}} - \delta(I_q) \end{cases} \tag{6.5}$$

so the functions $f(x)$ and $g(x)$ can be determined.

$$\begin{aligned} f(x) &= \dot{I}_{q_{ref}} + \frac{RI_{q_{mis}}}{L} - \omega_e I_{d_{mis}} - \delta(I_d) \\ g(x) &= -\frac{1}{L} \end{aligned} \tag{6.6}$$

Consequently this is the Lyapunov function derivative (from the equation 4.27).

$$\begin{aligned} \dot{V}(s) = s(-\frac{1}{L})[V_{q_{ref}} - L\dot{I}_{q_{ref}} - \omega_e\varphi - L\omega_e I_{d_{mis}} + \\ -RI_{q_{mis}} - a_2 L(I_{q_{ref}} - I_{q-mis}) + L\delta(I_q)] \end{aligned} \tag{6.7}$$

and the control law is obtained in the same form as equation 4.35

$$u = \underbrace{L\dot{I}_{q_{ref}} + \omega_e\varphi + \omega_e I_{d_{mis}} L + RI_{q_{mis}} + a_2 L(I_{q_{ref}} - I_{q_{mis}})}_{} + \underbrace{\rho \cdot sat(s)}_{v} \tag{6.8}$$

By adjusting the values of ϵ, a_3 and ρ, a tuning can be carried out.

6.2 The current I_d control law

Considering the following system

$$\begin{cases} \dot{x}_1(t) = x_2 = I_{d_{ref}} - I_{d_{mis}} \\ \dot{x}_2(t) = f(x) + g(x)u = \dot{I}_{d_{ref}} - \dot{I}_{d_{mis}} \end{cases} \tag{6.9}$$

while the sliding surface is defined as follows

$$s = e + a_3 \int e \Longrightarrow s = x_2 + a_3 x_1 \tag{6.10}$$

where $e = I_{d_{ref}} - I_{d_{mis}}$. The control law coincides with the reference voltage $V_{d_{ref}}$.

$$u = V_{d_{ref}} = RI_{d_{mis}} + L\dot{I}_{d_{mis}} - \omega_e L I_{q_{mis}} \quad [2] \tag{6.11}$$

Explicating $\dot{I}_{d_{mis}}$

$$\dot{I}_{d_{mis}} = \frac{V_{d_{ref}}}{L} - \frac{RI_{d_{mis}}}{L} + \omega_e I_{q_{mis}} + \delta(I_d) \tag{6.12}$$

the system is

$$\begin{cases} \dot{x}_1(t) = I_{d_{ref}} - I_{d_{mis}} \\ \dot{x}_2(t) = \dot{I}_{d_{ref}} - \frac{V_{d_{ref}}}{L} + \frac{RI_{d_{mis}}}{L} - \omega_e I_{q_{mis}} - \delta(I_d) \end{cases} \tag{6.13}$$

so the functions $f(x)$ and $g(x)$ can be determined.

$$\begin{aligned} f(x) &= \dot{I}_{d_{ref}} + \frac{RI_{d_{mis}}}{L} - \omega_e I_{q_{mis}} - \delta(I_d) \\ g(x) &= -\frac{1}{L} \end{aligned} \tag{6.14}$$

Consequently this is the Lyapunov function derivative (from the equation 4.27).

$$\dot{V}(s) = s(-\frac{1}{L})[V_{d_{ref}} - L\dot{I}_{d_{ref}} + I_{d_{mis}}(a_3 L - R) + L\omega_e I_{q_{mis}} - a_3 L I_{d_{ref}} + L\delta(I_d)] \tag{6.15}$$

and the control law is obtained in the same form as equation 4.35

$$u = \underbrace{L\dot{I}_{d_{ref}} - I_{d_{mis}}(a_3 L - R) - L\omega_e I_{q_{mis}} + a_3 L I_{d_{ref}}}_{u_{eq}} + \underbrace{\rho \cdot sat(s)}_{v} \tag{6.16}$$

6.2.1 The currents sliding surfaces

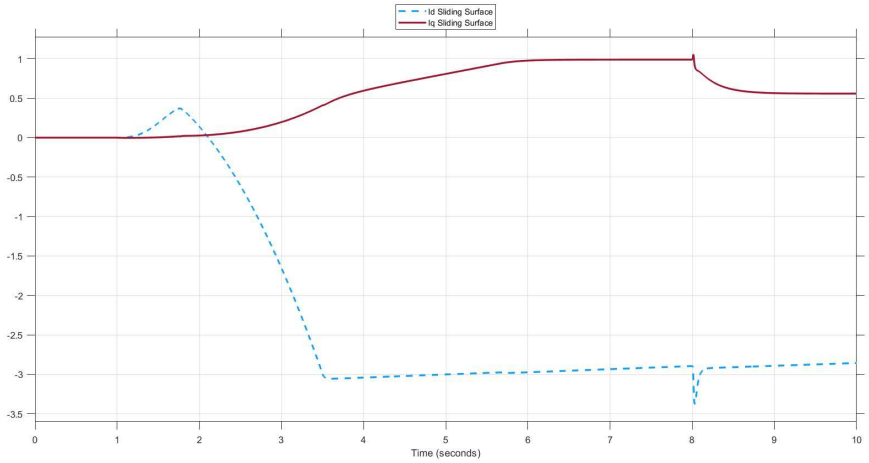


Figure 6.1: The currents sliding surfaces

The sliding surface for the current I_q is currently converging, which indicates that the sliding mode motion of the I_q current error is happening correctly. On the other hand, the sliding surface for the current I_d is increasing so slowly that it can be considered constant. Therefore, by extending the simulation time, the drifting settles down, and the sliding motion is guaranteed by the I_d current error (figure 6.1). The behavior of the sliding surfaces, in general, depends on the chosen tuning that can improve the speed of surface's convergence.

6.3 Current Controller Sliding

As anticipated in chapter 3, there are two MATLAB Functions inside the subsystem "Current Controller Sliding"; Inside the MATLAB Functions "Iq Controller" and "Id Controller", there are the control laws previously implemented (figures 6.2 e 6.3).

6.3.1 Iq Controller

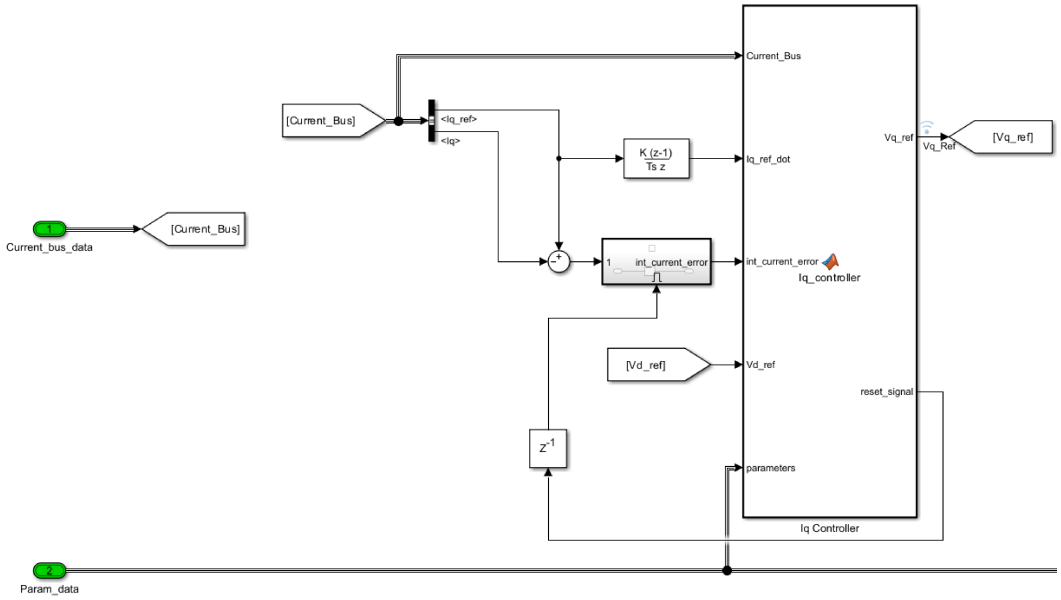


Figure 6.2: The current I_q regulator Simulink scheme

The MATLAB Function "Iq Controller" (figure 6.2) has the following inputs:

- *Current_Bus*: a bus containing useful signals for the control law calculation;
- Iq_ref_dot: reference current I_q derivative;
- *int_current_error*: current error integration, that is calculated as the difference between the reference current $I_{q_{ref}}$ and the measured or estimated I_q current. The Enabled Subsystem that realizes the anti windup logic returns *int_current_error* as output;
- Vd_ref: reference V_d voltage;
- *parameters*: contains the parameters provided by the bus "Param_data", described in chapter 3.

The reference voltage Vq_ref and *reset_signal* are the outputs.

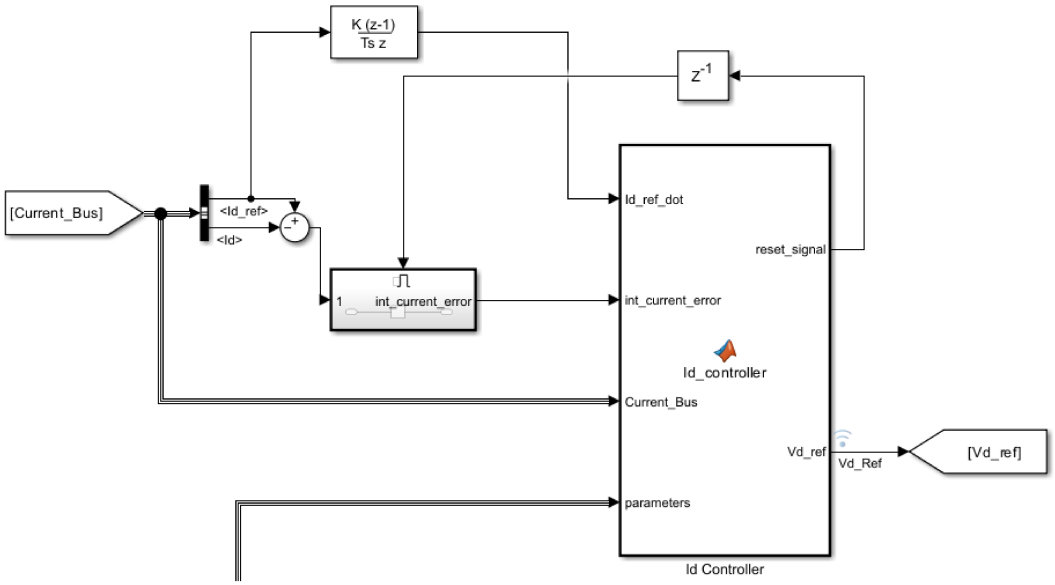


Figure 6.3: The current I_d regulator Simulink scheme

6.3.2 Id Controller

The MATLAB Function "Iq Controller" (figure 6.3) has the following inputs:

- Id_ref_dot: reference current I_d derivative;
- $int\_current\_error$: current error integration, that is calculated as the difference between the reference current $I_{d_{ref}}$ and the measured or estimated current I_d. There is an anti windup logic for this current regulator too;
- $Current\_Bus$ and $parameters$: they are the same input bus of the current I_q controller;

The reference voltage Vd_ref and $reset\_signal$ are the outputs.

6.3.3 Inverse Park Transformation

The "Current_bus_data_updated" bus is populated through a Bus Assignment: the calculated voltage triad Vdq_Zero, Vd_ref, Vq_ref and their inverse Park transform; the latter is calculated by the "InversePark" block (figure 6.4) whose code is shown.

6.4 Iq Controller and Id Controller's code

This section contains the code inside the "Iq Controller" and "Id Controller". They have a structure similar to the "Speed Controller", except for the control laws relating to the currents I_q and I_d. For both controllers:

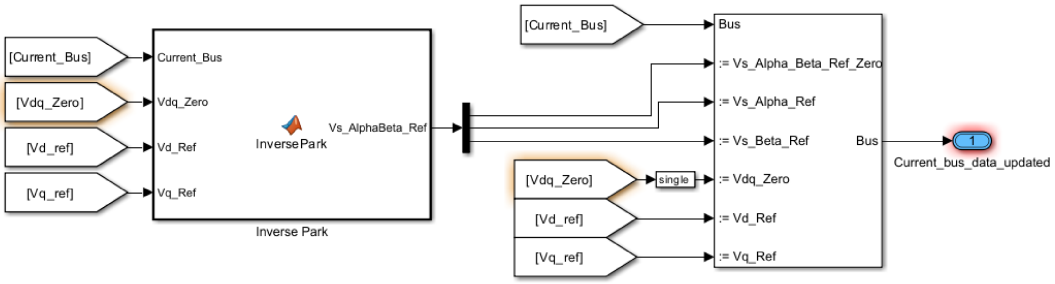


Figure 6.4: Inverse Park Transformation's MATLAB Function

1. the parameters initialization for tuning and electrical parameters happens through, respectively, the struct *parameters* and the bus *Current_Bus*;
2. the respective sliding surface is defined;
3. the control laws are created by combining the components of v and u_{eq}.
4. the saturation limits of the reference voltages are defined. The reference voltages are the regulators outputs;
5. whenever the voltage saturates, *reset_signal* takes on the value -1, otherwise 1.

```
function [Vq_ref, reset_signal] = Iq_controller(Current_Bus,
    Iq_ref_dot,int_current_error,Vd_ref,parameters)

%Iq_ref=Current_Bus.Iq_ref;
omega_r=Current_Bus.Speed;
Iq=Current_Bus.Iq;
Id=Current_Bus.Id;
Vdc=Current_Bus.Vdc;
Iq_ref=Current_Bus.Iq_ref;

lambda0 = parameters.lambda0;
Np = parameters.Np;
omega_e=Np*omega_r;
R=parameters.R;
Lq=parameters.Lq;

a2=parameters.a2;
ro=parameters.roq;
epsilon=parameters.epsilonq;

%sliding surface
s = (Iq_ref - Iq) + a2*int_current_error;

%sat(s)
```

```
if abs(s) > epsilon
    v = ro*sign(s);
else
    v = ro*(s/epsilon);
end

%control law
u_eq = (omega_e*lambda0)+(R*Iq)+Lq*(Iq_ref_dot+(omega_e*Id)+(a2
   *(Iq_ref-Iq)));
u = u_eq + v;

Upper_Limit=sqrt(((Vdc/sqrt(3))^2)-(Vd_ref^2));
Lower_Limit=-sqrt(((Vdc/sqrt(3))^2)-(Vd_ref^2));

%sat(Vq_ref)
if u >= Upper_Limit
    u = Upper_Limit;
    reset_signal=-1;
elseif u <= Lower_Limit
    u = Lower_Limit;
    reset_signal=-1;
else
    reset_signal=1;
end

Vq_ref= u;
```

```
function [reset_signal,Vd_ref] = Id_controller(Id_ref_dot,
   int_current_error,Current_Bus,parameters)

Id=Current_Bus.Id;
Id_ref=Current_Bus.Id_ref;
Iq=Current_Bus.Iq;
omega_r=Current_Bus.Speed;
Vdc=Current_Bus.Vdc;

Np = parameters.Np;
omega_e=Np*omega_r;
R=parameters.R;
Ld=parameters.Ld;

a3=parameters.a3;
ro=parameters.rod;
epsilon=parameters.epsilond;

%sliding surface
s = (Id_ref-Id)+(a3*int_current_error);
```

```
20
21 %sat(s)
22 if abs(s) > epsilon
23     v = ro*sign(s);
24 else
25     v = ro*(s/epsilon);
26 end
27
28 %control law
29 u_eq = (Ld*Id_ref_dot)-(Id*(-R+a3*Ld))-(Ld*omega_e*Iq)+(a3*Ld*
       Id_ref);
30 u = u_eq + v;
31
32 Upper_Limit = Vdc/sqrt(3);
33 Lower_Limit = -Vdc/sqrt(3);
34
35 %sat(Vd_ref)
36 if u >= Upper_Limit
37     u = Upper_Limit;
38     reset_signal=-1;
39 elseif u <= Lower_Limit
40     u = Lower_Limit;
41     reset_signal=-1;
42 else
43     reset_signal=1;
44 end
45
46 Vd_ref=u;
```

6.5 Il codice di InversePark

The "InversePark" code defines the inverse Park transformation matrix and pre-multiplies it to the voltage triad Vdq_Zero, Vd_Ref, Vq_Ref. The sine and cosine of the angle used for the Park transform are extracted from $Current\_Bus$. Finally, the MATLAB Function outputs the new triple, enclosed in the vector $Vs\_AlphaBeta\_Ref$ [9].

```
1 function Vs_AlphaBeta_Ref  = InversePark(Current_Bus,Vdq_Zero,
      Vd_Ref,Vq_Ref)
2 sinTheta=Current_Bus.Sin;
3 cosTheta=Current_Bus.Cos;
4 T=[cosTheta sinTheta;
5     -sinTheta cosTheta];
6 Vs_AlphaBeta=(T')*[Vd_Ref;Vq_Ref];
7 Vs_AlphaBeta_Ref=[Vdq_Zero;Vs_AlphaBeta];
8 end
```

Chapter 7

Comparative Controller Performance: Graphical Results

This chapter contains graphs that represent the behavior of some signals of interest for the analysis. Both the signals produced by the PI regulators and those produced by the sliding mode regulators are present, to make a comparison regarding the performance of the two controllers. In particular, the trend of torque, speed and currents will be reported. The following working conditions for regulators are analyzed: -nominal case: the regulators work without disturbances, variations or saturations; -voltage saturation: the regulators work with saturated voltage -torque saturation: the regulators work with saturated torque -double saturation: the regulators work with saturated voltage and torque at the same time. For each of these cases, appropriate considerations are reported which highlight the characteristics of PI regulators and sliding mode regulators. All graphs were obtained by setting the simulator in sensorless mode, therefore using estimated rather than measured quantities. In all graphs: the reference signals are green and dashed, the signals produced by the PIs are orange and the signals produced by the SMC are purple.

7.1 Results in nominal conditions

The Sliding Mode tuning parameters selected are defined below:

- Speed parameters:
  - a1=90;
  - ρ=0.08;
  - ε=400.
- Current I_d parameters:
  - a2=500;
  - ρ_d=5;
  - ε_d=3000.
- Current I_q parameters:

- a2=500;
- ρ_q=5;
- ε_q=10.

As for the sliding mode controller, thanks to these tuning parameters, the sliding mode torque is able to track its reference trend almost perfectly (figure 7.3), as well as the estimated speed tracks its reference (figure 7.1); the sliding mode estimated speed has just a small spike at second 8, according to the end of the torque load's action. Regarding the PI controller, both torque and speed have a transient characterized by bumps and oscillations (figures 7.1 and 7.2). Regarding the PI controller, both torque and speed have a transient characterized by bumps and oscillations. Except for the spikes in correspondence of torque load changes, such as in seconds 1,2 and 8, both speed and torque realize adequate tracking. PI speed has also a longer reaching time compared to sliding mode speed.

The SM current I_d reaches the reference approximately at second 3.5 and has a considerable spike at second 8. The PI current I_d immediately reaches the reference, but there are a first noticeable spike and a smaller one at second 8 (figure 7.4). The voltage V_d graphs of the two regulators are not present as their trends are very similar and determine the behavior of the I_d currents previously described. The only difference between the two concerns the presence of more evident peaks in the PI voltage V_d, where the torque load changes.

The current I_q is proportional to the torque, so they have the same trend (figures 7.5 and 7.6). The SM voltage V_q has the same behavior of the PI voltage V_q, but the first one has a large spike at the 8th second, while the second one has smoother spikes in the first 2 seconds.

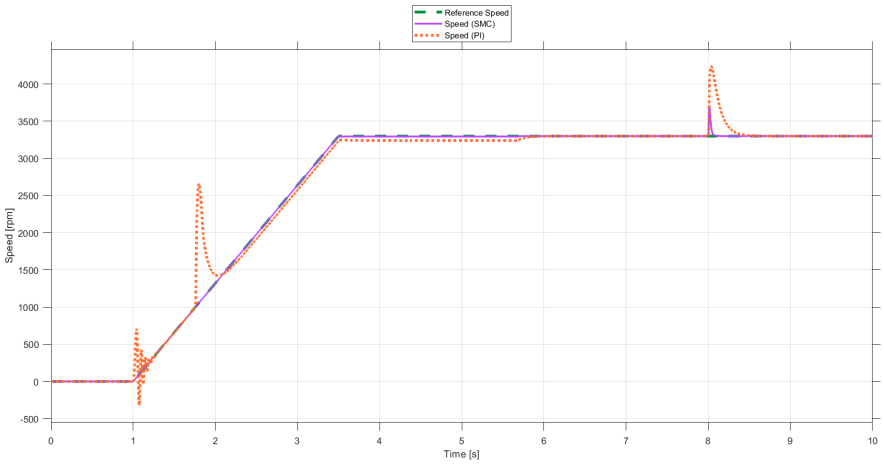


Figure 7.1: Reference speed, PI speed and SMC speed in nominal conditions

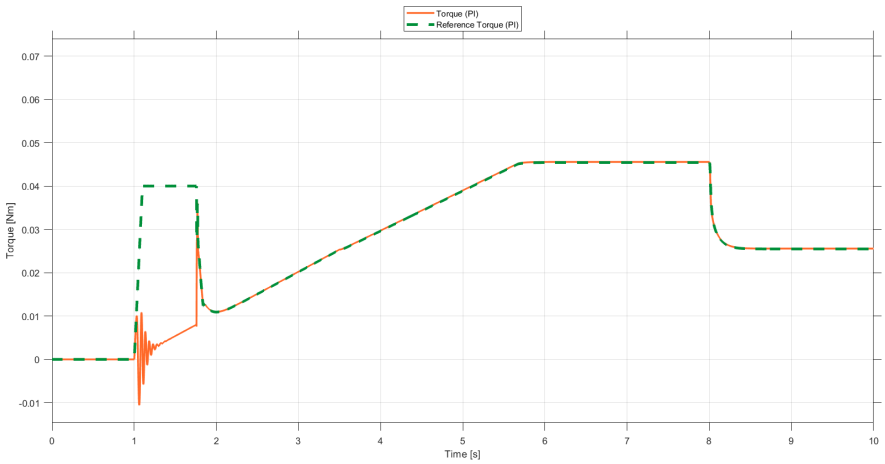


Figure 7.2: PI: reference torque and real torque in nominal conditions

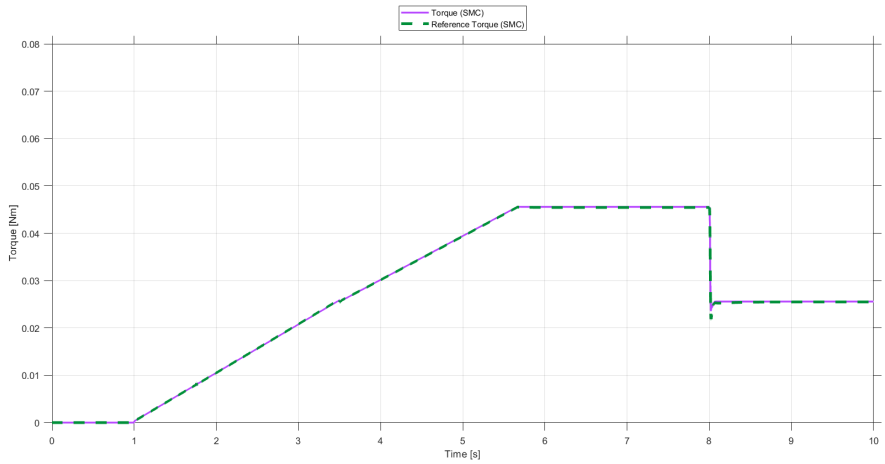


Figure 7.3: SMC: reference torque and real torque in nominal conditions

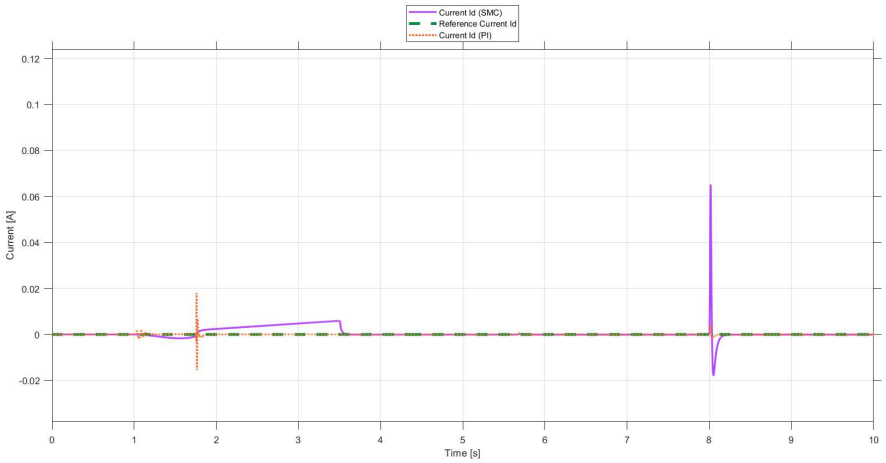


Figure 7.4: Reference current I_d, PI current I_d and SM current I_d in nominal conditions

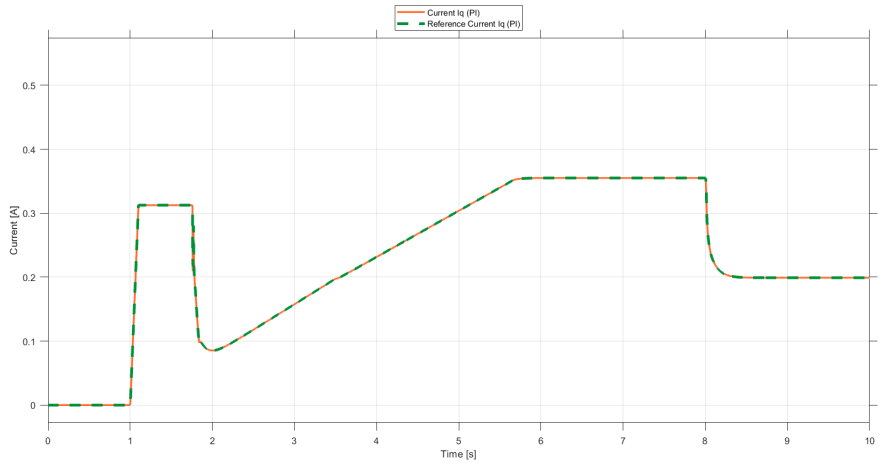


Figure 7.5: PI: reference current I_q and current I_q in nominal conditions

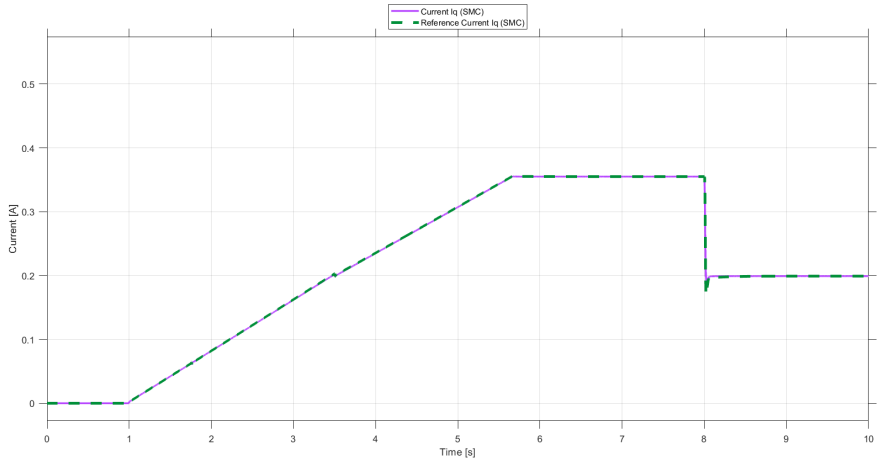


Figure 7.6: SMC: reference current I_q and current I_q in nominal conditions

7.2 Results in presence of voltage saturation

This section contains the graphs obtained in the presence of saturated voltage. Saturation is imposed on the voltage by changing the value of the $DC\_Bus.Const\_Vbus$ variable from 325 V to 60 V; its value can be set within the MATLAB workspace. As expected, all real signals strive to follow the reference unsuccessfully. The Current Id is the only exception because it converge to zero as wanted. Overall, both SM and PI controllers show a satisfactory performances (figures 7.7, 7.8, 7.9, 7.10, 7.11, 7.12).

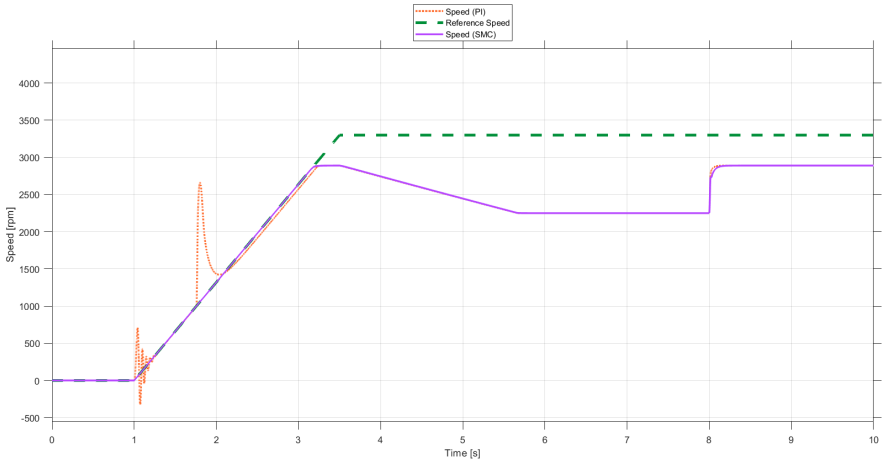


Figure 7.7: Reference speed, PI speed and SMC speed in presence of voltage saturation

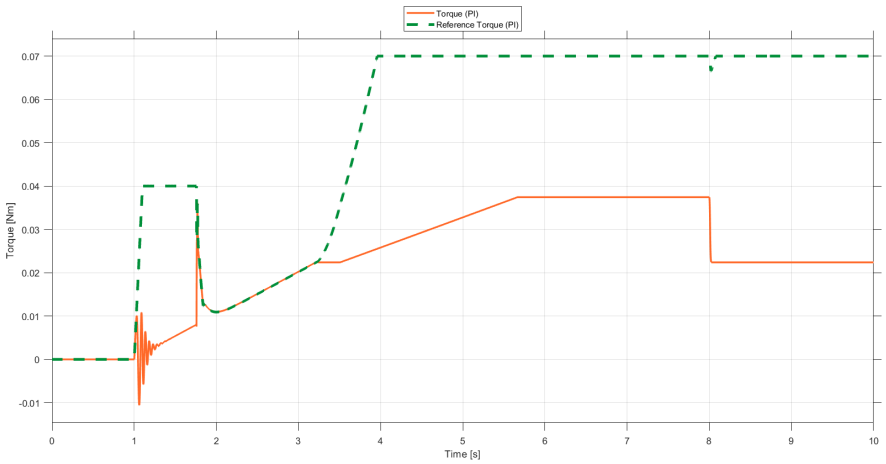


Figure 7.8: PI: reference torque and real torque in presence of voltage saturation

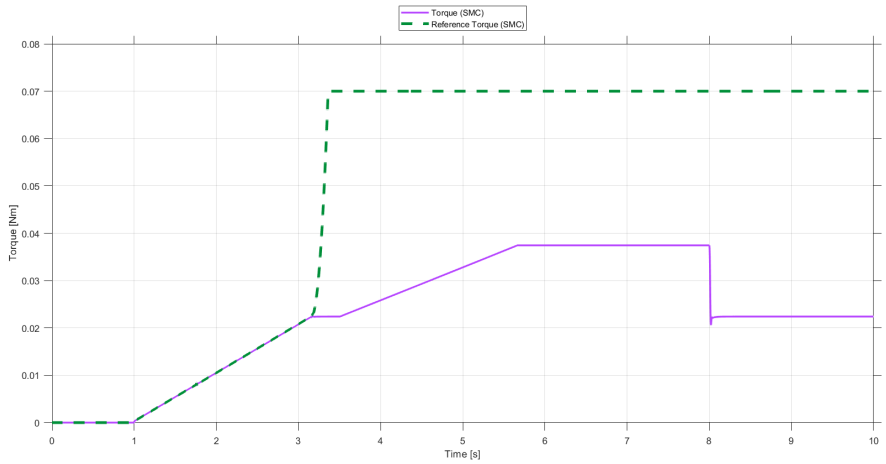


Figure 7.9: SMC: reference torque and real torque in presence of voltage saturation

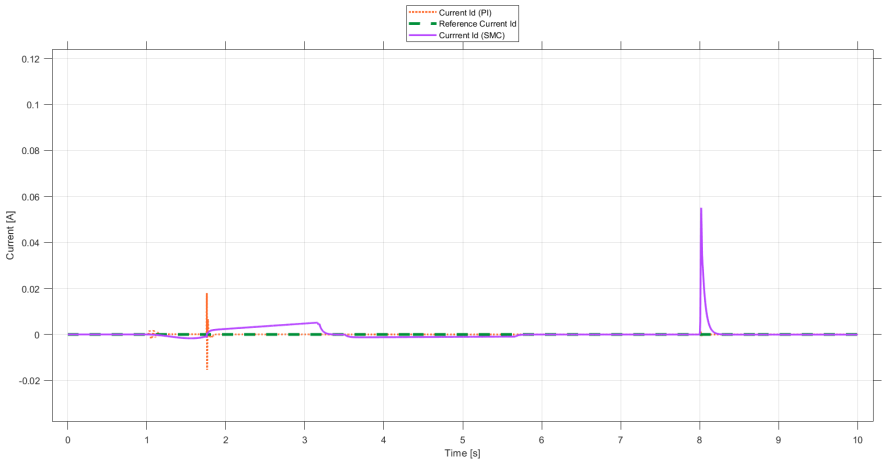


Figure 7.10: Reference current I_d, PI current I_d and SM current I_d in presence of voltage saturation

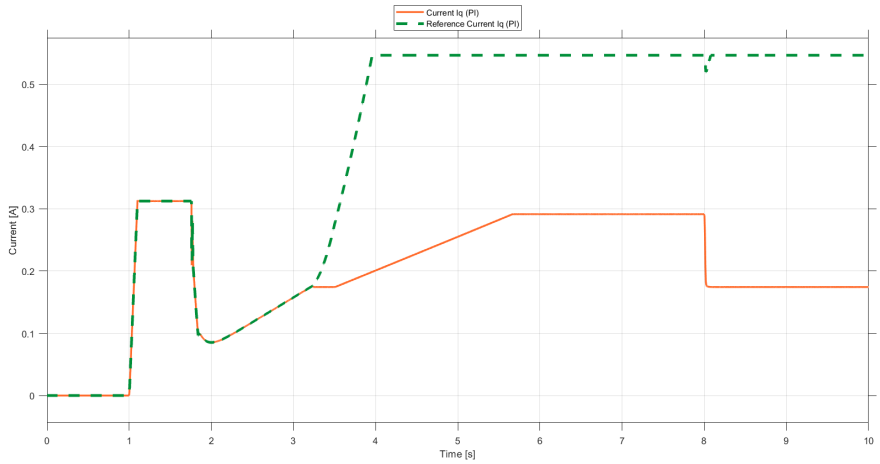


Figure 7.11: PI: reference current I_q and current I_q in presence of voltage saturation

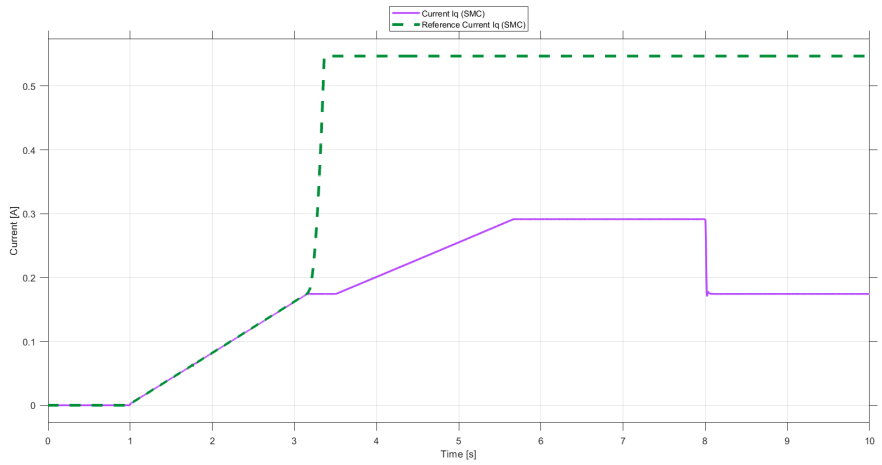


Figure 7.12: SMC: reference current I_q and current I_q in presence of voltage saturation

7.2.1 Controllers reaction to voltage desaturation

This section describes the signals' behavior following voltage desaturation. In particular, saturation is introduced at second 4 and then removed at second 6, setting $DC\_Bus.Const\_Vbus$ value from 325 V to 60 V and finally again to 325 V. Concerning the SMC:

- due to saturation, the speed is unable to follow the reference generating a gap. When saturation is removed, perfect tracking promptly occurs (figure 7.13);
- the torque, as well as the speed, does not follow its reference in presence of saturation; on the other hand, once the voltage desaturates, the torque achieves perfect tracking (figure 7.15);
- the I_d current presents spikes at the moments in which the voltage value changes. However, the overall trend of the current is similar to that which occurs in nominal conditions (figure 7.16);
- the current I_q behaves exactly like the torque (figure 7.18).

While as regards the PI:

- the speed maintains the trend it has in nominal conditions, except for the interval where saturation is active. In second 6, in which the voltage returns to the nominal value, the speed presents a modest bump (figure 7.13);
- the torque behaves as in nominal conditions. It does not follow its reference only in the transient and during saturation (figure 7.14);
- the current I_d behaves as in nominal conditions (figure 7.16);
- the current I_q has perfect tracking except during saturation (figure 7.17).

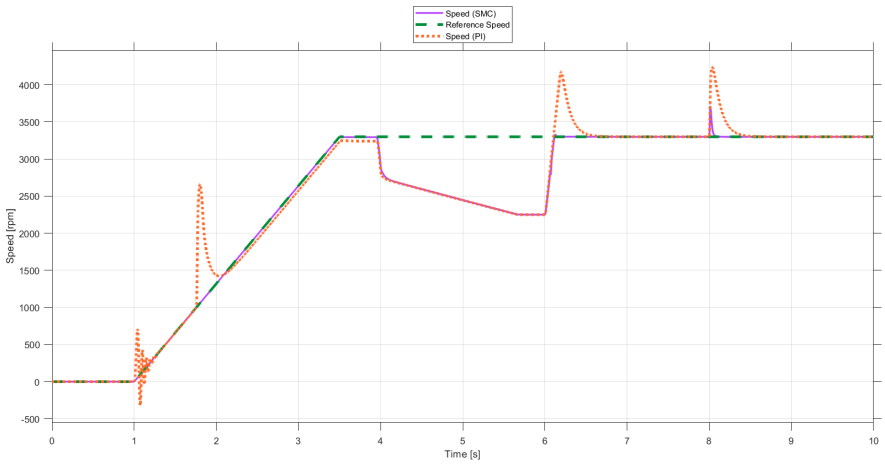


Figure 7.13: Reference speed, PI speed and SMC speed during voltage desaturation

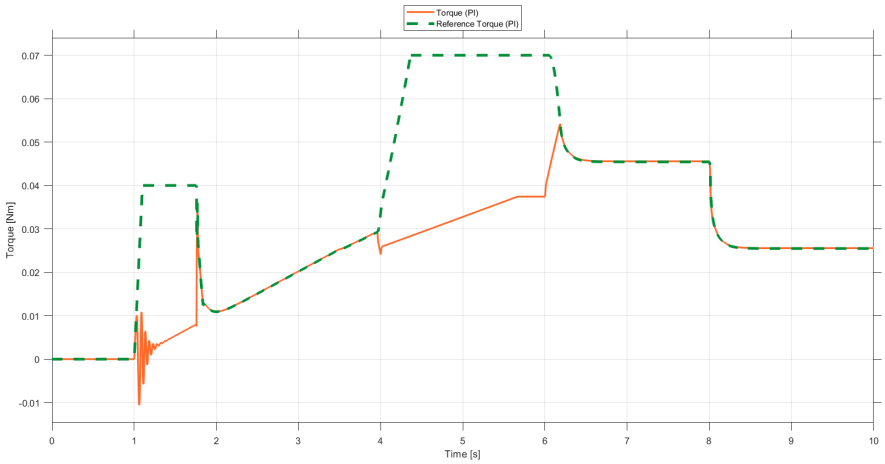


Figure 7.14: PI: reference torque and real torque during voltage desaturation

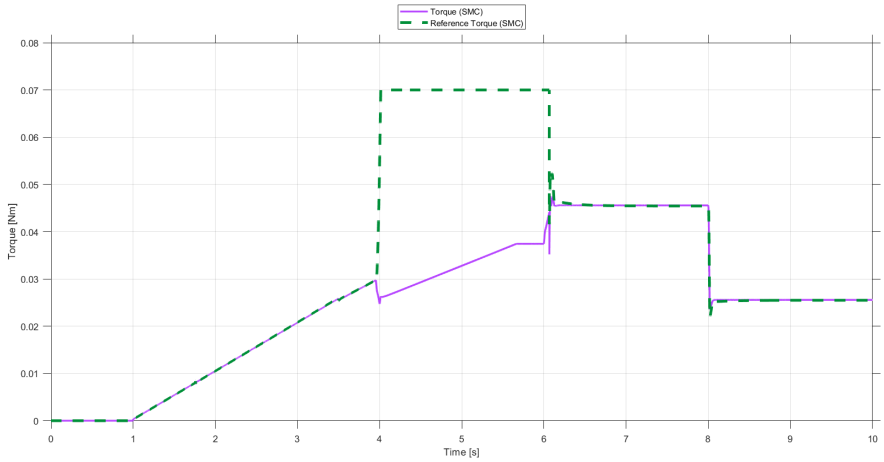


Figure 7.15: SMC: reference torque and real torque during voltage desaturation

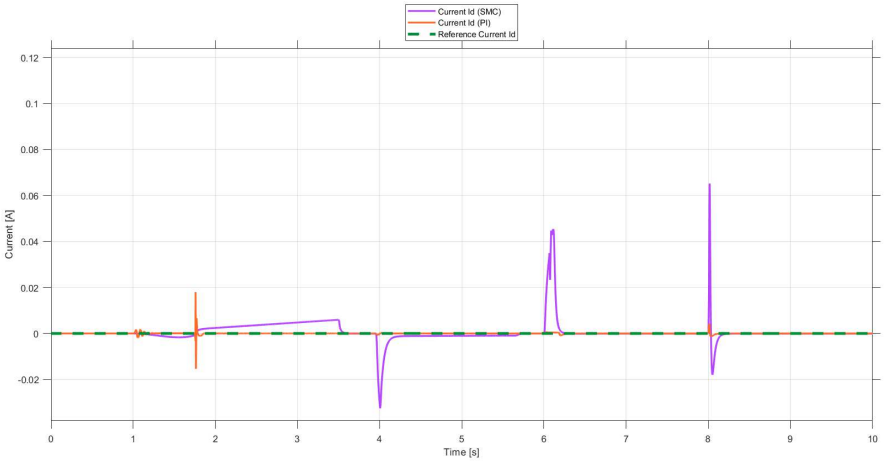


Figure 7.16: Reference current I_d, PI current I_d and SM current I_d during voltage desaturation

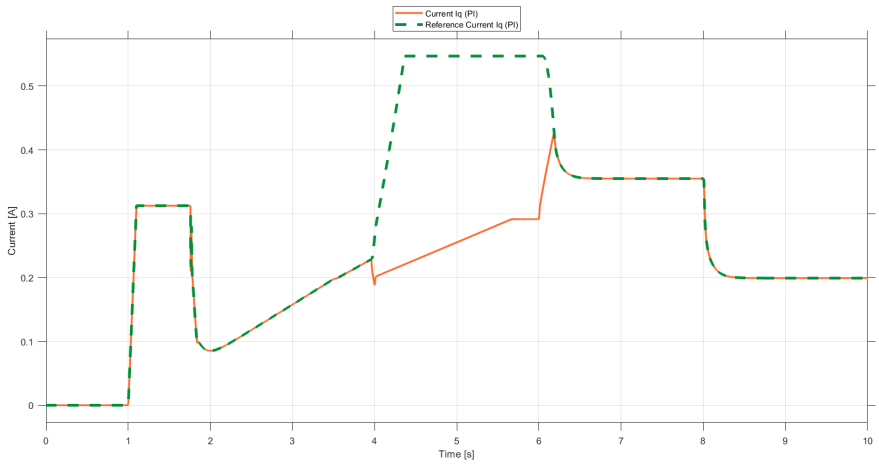


Figure 7.17: PI: reference current I_q and current I_q during voltage desaturation

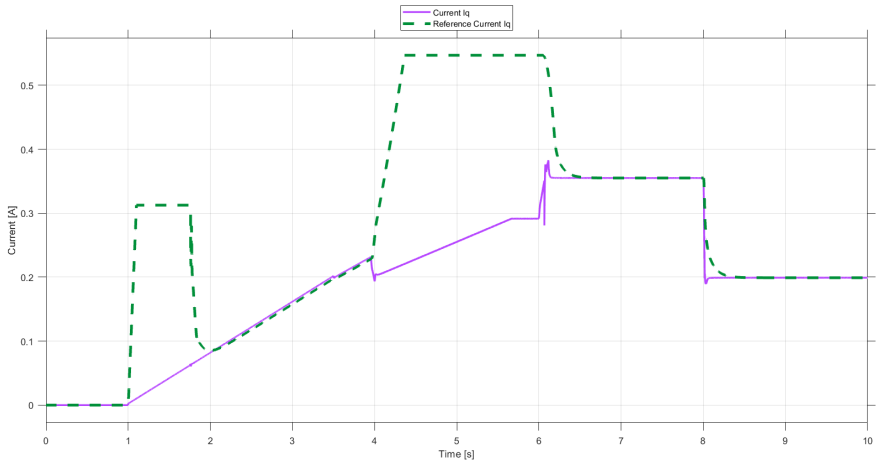


Figure 7.18: SMC: reference current I_q and current I_q during voltage desaturation

7.3 Results in presence of torque saturation

This section describes the effects of torque saturation on sliding mode and PI controllers. Torque saturation is achieved by setting the upper bound of the reference torque to 0.035 Nm instead of 0.07 Nm; in this way, when a speed reference requires more torque than the upper limit, the speed is unable to follow such reference. In fact, the speed of both controllers is unable to follow the reference when the torque takes values equal to its upper bound (figure 7.19). The torque of both regulators follows its reference as in nominal conditions, with slight oscillations during the transient (figures 7.20 and 7.21). There appears to be no change in the I_d current of the PI regulator. The SMC current I_d presents the same trend as the nominal case, with the addition of a spike and a minor offset between the fourth and sixth second (figure 7.22). The I_q currents of both regulators behave like their respective torque, but with perfect tracking even in the transient (figures 7.23 and 7.24).

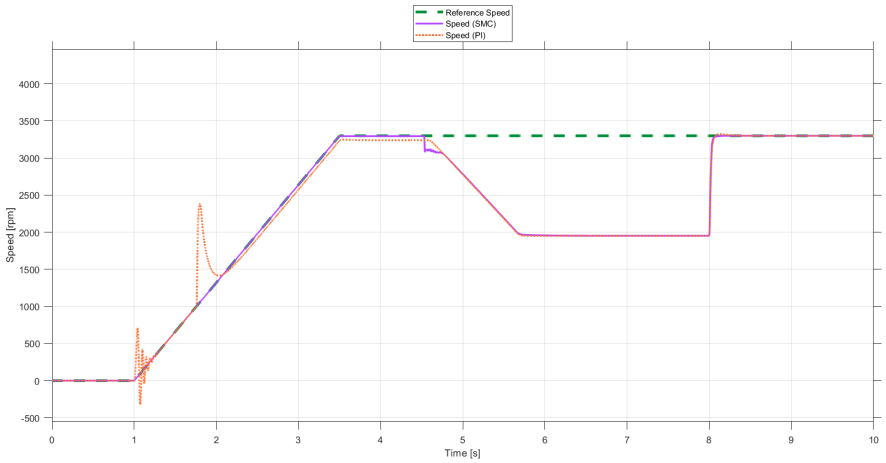


Figure 7.19: Reference speed, PI speed and SMC speed in presence of torque saturation

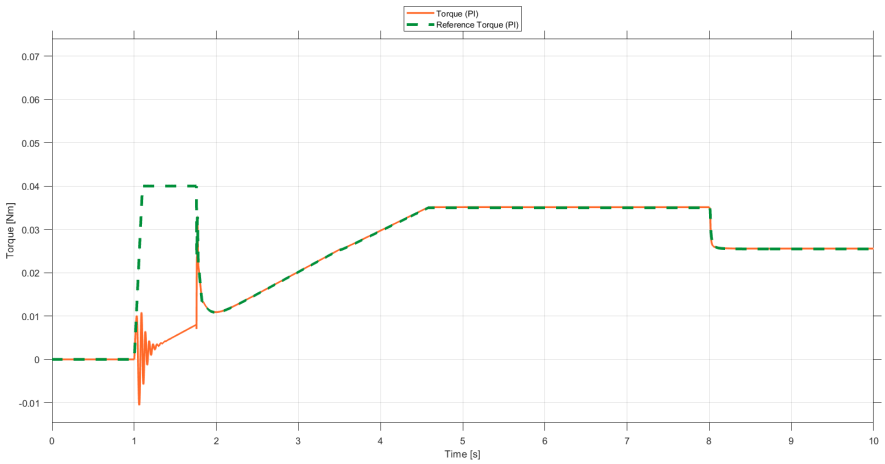


Figure 7.20: PI: reference torque and real torque in presence of torque saturation

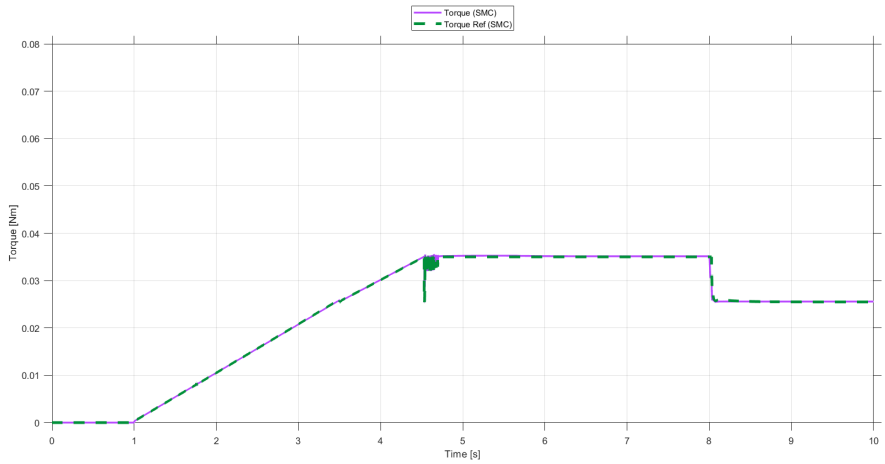


Figure 7.21: SMC: reference torque and real torque in presence of torque saturation

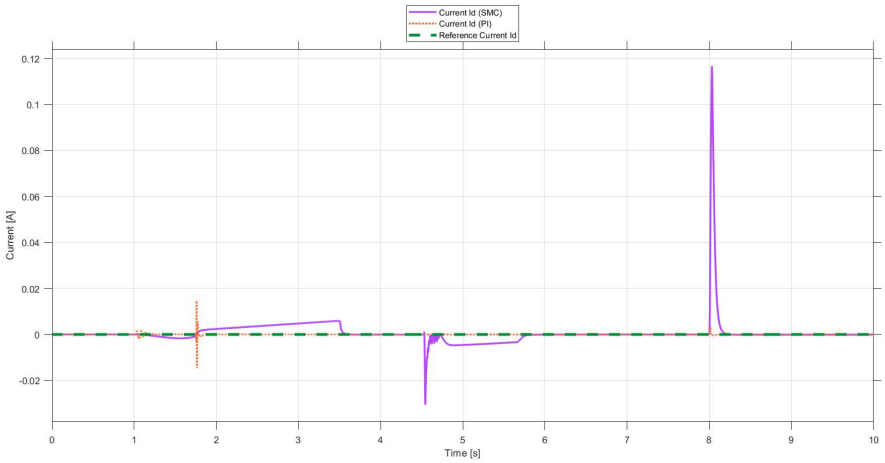


Figure 7.22: Reference current I_d, PI current I_d and SM current I_d in presence of torque saturation

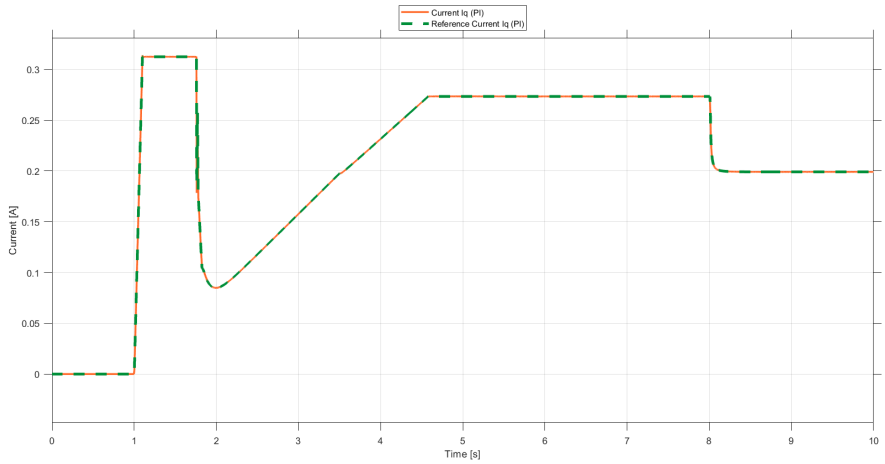


Figure 7.23: PI: reference current I_q and current I_q in presence of torque saturation

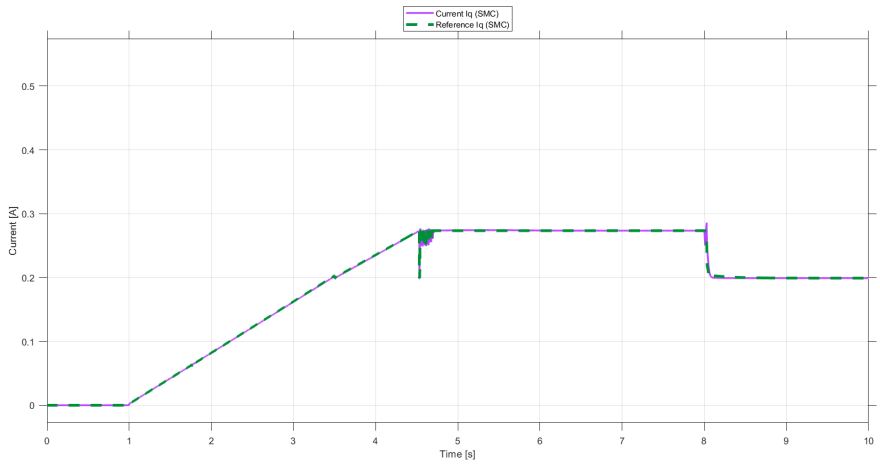


Figure 7.24: SMC: reference current I_q and current I_q in presence of torque saturation

7.4 Results in presence of both torque and voltage saturation

For a complete analysis, this subsection contains the graphics obtained saturating both torque and voltage. These graphics are really close to the ones obtained saturating only the voltage with some negligible differences (figures 7.25, 7.26, 7.27, 7.28, 7.29, 7.30).

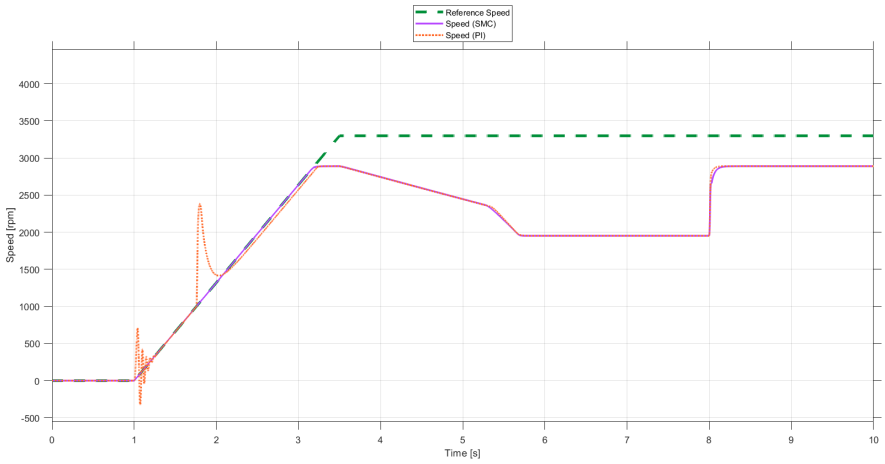


Figure 7.25: Reference speed, PI speed and SMC speed in presence of both torque and voltage saturation

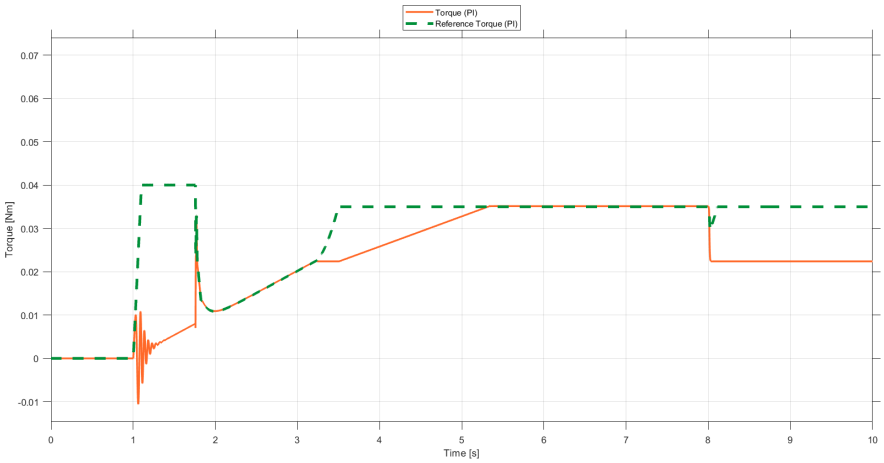


Figure 7.26: PI: reference torque and real torque in presence of both torque and voltage saturation

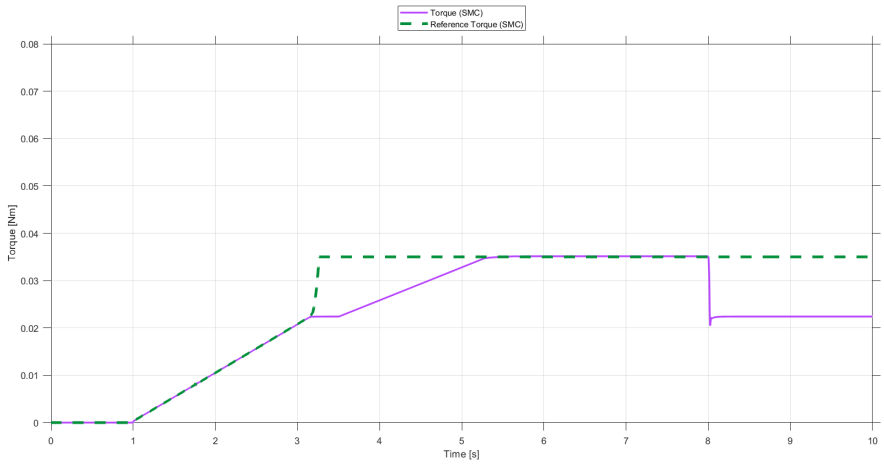


Figure 7.27: SMC: reference torque and real torque in presence of both torque and voltage saturation

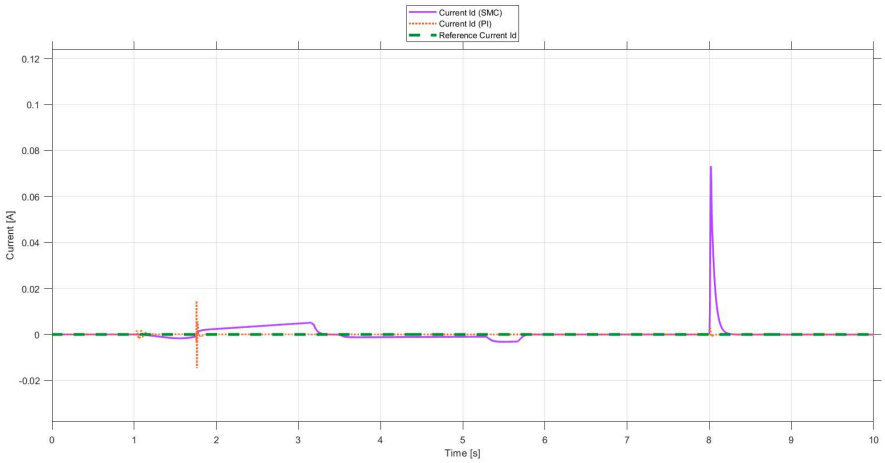


Figure 7.28: Reference current I_d, PI current I_d and SM current I_d in presence of both torque and voltage saturation

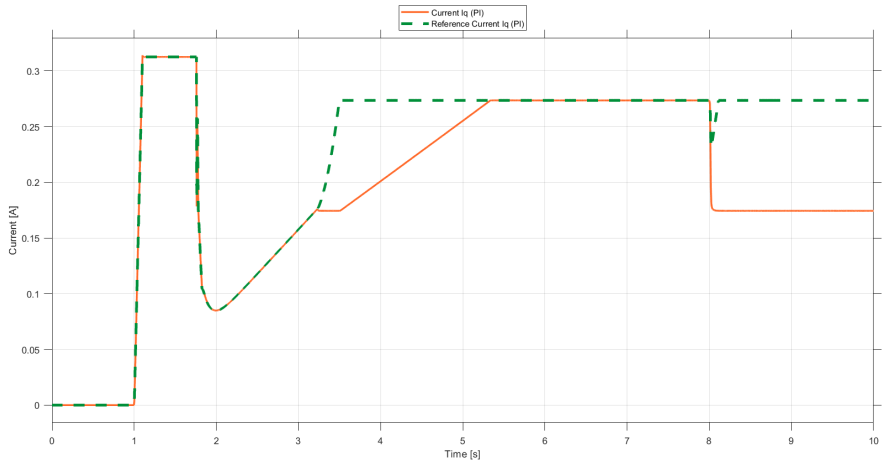


Figure 7.29: PI: reference current I_q and current I_q in presence of both torque and voltage saturation

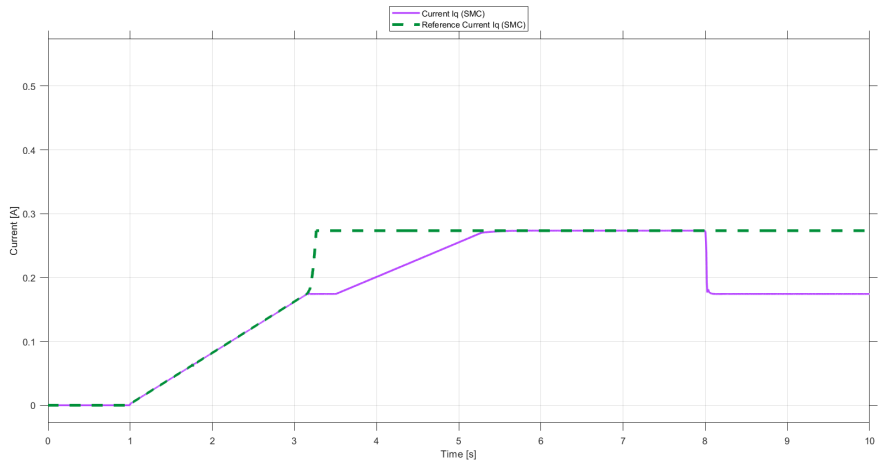


Figure 7.30: SMC: reference current I_q and current I_q in presence of both torque and voltage saturation

7.5 Evaluation and observation of the results

In general, the SMC has demonstrated outstanding responsiveness, minimal oscillations, and nearly perfect tracking. Moreover, the SMC has produced the most accurate tracking between the speed and its reference. Regarding the other signals, it has shown performances comparable to those of the PI controller.

Chapter 8

Robustness Test

Robustness means "the degree to which a system or component can work properly in the presence of invalid or stressful inputs and environmental conditions" [10]. Robustness testing is a type of testing that is used to solicit a system in exceptional situations and to understand how it can react. The purpose of this section is to carry out such tests and report the results. This is followed by the evaluation of the IAE and MSE performance indices to quantify the quality of these tests. These indices were useful for making a quantitative comparison between PI and SM controllers in terms of robustness.

8.1 robustness_test.m

To automatically perform robustness tests, the last part of the script outlined in chapter 3 needs to be uncommented. It creates a table that contains the IAE and MSE values obtained. It then makes the thirty-two tests, populating the table at each iteration, which then is written into an Excel file. The valorization of the parameters for each test is performed by the "robutness_test.m" function: it assigns maximum and minimum values for each parameter by reading from a given matrix.

```
%%%%%%%%%%%%%%% Automatic Robustness Tests %%%%%%%%%%%%%%%%%%
tableData = table('Size', [0, 2],'VariableTypes', {'double', '
    double'}, 'VariableNames', {'IAE', 'MSE'});

for i = 1:32
    [Rs,Ld,Lq,Phim,J,B] = robustness_test(i);
                                                % -1       % 1
    Motor_prm.Electrical.Rs_0= Rs;              %40.2686   61.5965
    Motor_prm.Electrical.Ld_0= Ld;              %0.1116    0.1284
    Motor_prm.Electrical.Lq_0= Lq;              %0.1116    0.1284
    Motor_prm.Electrical.PM_flux_h1_0= Phim;    %0.0673    0.0939
    Motor_prm.Mechanical.B= B;                  %7.03      7.77
    Motor_prm.Mechanical.Jrotor= J;             %2.02      2.24

sim('MCU_Simulation_Architecture_Sliding_CodeGen_13_12_2023.slx
    ',10);

```

```
    newRow = table(IAE, MSE, 'VariableNames', {'IAE', 'MSE'});
    tableData = [tableData; newRow];
end
 excelFileName = 'IAE_MSE.xlsx';
writetable(tableData, excelFileName, 'Sheet', 'Sheet1');
```

```
function [Rs,Ld,Lq,Phim,J,B] = robustness_test(i)

%matrice
range = 'A2:E33';
[matrice]=xlsread('RobustnessTest.xlsx',range);
%disp(matrice);

a=matrice(i,1);
b=matrice(i,2);
c=matrice(i,3);
d=matrice(i,4);
e=matrice(i,5);

if a==1
    Rs= 61.5965;
else
    Rs= 40.2686;
end

if b==1
    Ld= 0.1284;
    Lq= 0.1284;
else
    Ld= 0.1116;
    Lq= 0.1116;
end

if c==1
    Phim=0.0939;
else
    Phim=0.0673;
end

if d==1
    J= 2.24e-06;
else
    J= 2.02e-06;
end

if e==1
    B= 7.77e-05;
```

```
42 else
43     B= 7.03e-05;
44 end
45
46 end
```

8.2 Test Parameters

The performed tests have varied certain parameters' values, which are outlined in the table 8.1. It is important to note that all parameters are tested at their boundary values in a combination called Design of Experiments (DOE). The DOE is a method for designing and executing experiments using statistical analysis to find relationships between input variables and output variables. This means that the robustness of the regulator is tested in extreme scenarios, thus it is unlikely that all values will be at the limit in a real-world situation.

Parameter	Nominal Value	Min (10°C)	Max (100°C)	Unit Measure
Rs	45.5	40.2686	61.5965	Ohm
Ld	120	111.6	128.4	mH
Lq	120	111.6	128.4	mH
Phim	0.0857	0.0673	0.0939	Vpk/rad/s
J	2.13E-06	2.02E-06	2.24E-06	Kg*m$\hat{2}$
B	7.40E-05	7.03E-05	7.77E-05	Nm/ rad/s

Table 8.1: Parameters

These are the involved parameters:

- Rs is the stator resistance;
- Ld and Lq are the inductance;
- Phim is the electromagnetic flux;
- J is the inertia rotor;
- B is the friction coefficient.

Rs and Phim values are related to physics and temperature, not just the process; while others are only tied to the process. Temperature, as showed in the table 8.1, can fluctuate from 10°C to 100°C.

The table 8.2 contains values of either -1 or 1. These values indicate whether the corresponding parameter is at its minimum or maximum value, respectively. Since there are five parameters that vary, there will be 32 tests to conduct, equal to the number of rows in the table.

To quantify the goodness of the tests, two performance indices have been considered:

Rs	Ls	Phim	J	B
1	1	-1	1	1
1	1	1	-1	1
-1	-1	-1	-1	1
1	-1	1	1	-1
-1	-1	-1	1	-1
1	1	-1	-1	1
-1	1	-1	1	1
-1	1	1	1	-1
1	-1	1	1	1
-1	1	-1	1	-1
-1	1	-1	-1	1
-1	-1	-1	1	1
-1	1	-1	-1	-1
-1	1	1	1	1
1	-1	-1	1	1
1	-1	1	-1	1
1	-1	1	-1	-1
1	1	1	1	1
1	-1	-1	-1	-1
1	1	1	1	-1
-1	-1	1	1	-1
-1	1	1	-1	-1
1	1	1	-1	-1
-1	-1	1	-1	1
-1	-1	-1	-1	-1
-1	-1	1	1	1
1	-1	-1	-1	1
-1	1	1	-1	1
-1	-1	1	-1	-1
1	1	-1	1	-1
1	-1	-1	1	-1
1	1	-1	-1	-1

Table 8.2: DOE - Design of Experiment

the Integral of the Absolute Error (IAE) and the Mean Squared Error (MSE). these indices allow to compare PI robustness to that of the SMC.
The tests were performed inside the simulator in sensorless mode.

8.3 Results: 6 KOs on 32 tests

IAE Nominal S	MSE Nominal S	IAE Sensorless	MSE Sensorless
2.269	1.49E-08	4351	2.75E+05
2.269	1.49E-08	4.318	1.13E-07
2.269	1.49E-08	4424	2.91E+05
2.269	1.49E-08	3.835	9.31E-08
2.269	1.49E-08	3.18	2.38E-07
2.269	1.49E-08	4348	2.75E+05
2.269	1.49E-08	5.039	2.38E-07
2.269	1.49E-08	2.583	3.73E-09
2.269	1.49E-08	5.276	5.96E-08
2.269	1.49E-08	3.18	2.38E-07
2.269	1.49E-08	4424	2.91E+05
2.269	1.49E-08	5.039	2.38E-07
2.269	1.49E-08	3.087	3.02E-07
2.269	1.49E-08	3.015	3.36E-07
2.269	1.49E-08	4351	2.75E+05
2.269	1.49E-08	4.318	1.13E-07
2.269	1.49E-08	4.386	9.31E-08
2.269	1.49E-08	5.276	5.96E-08
2.269	1.49E-08	6.146	1.13E-07
2.269	1.49E-08	3.835	9.31E-08
2.269	1.49E-08	2.583	3.73E-09
2.269	1.49E-08	2.608	9.31E-08
2.269	1.49E-08	4.386	9.31E-08
2.269	1.49E-08	4.28	4.51E-07
2.269	1.49E-08	3.087	3.02E-07
2.269	1.49E-08	3.015	3.36E-07
2.269	1.49E-08	4348	2.75E+05
2.269	1.49E-08	4.28	4.51E-07
2.269	1.49E-08	2.608	9.31E-08
2.269	1.49E-08	8.069	7.54E-08
2.269	1.49E-08	8.069	7.54E-08
2.269	1.49E-08	6.146	1.13E-07

Table 8.3: Test results Sliding Mode robustness

The table 8.3 reports the results of the first experiment conducted, which indicate 6 KO.

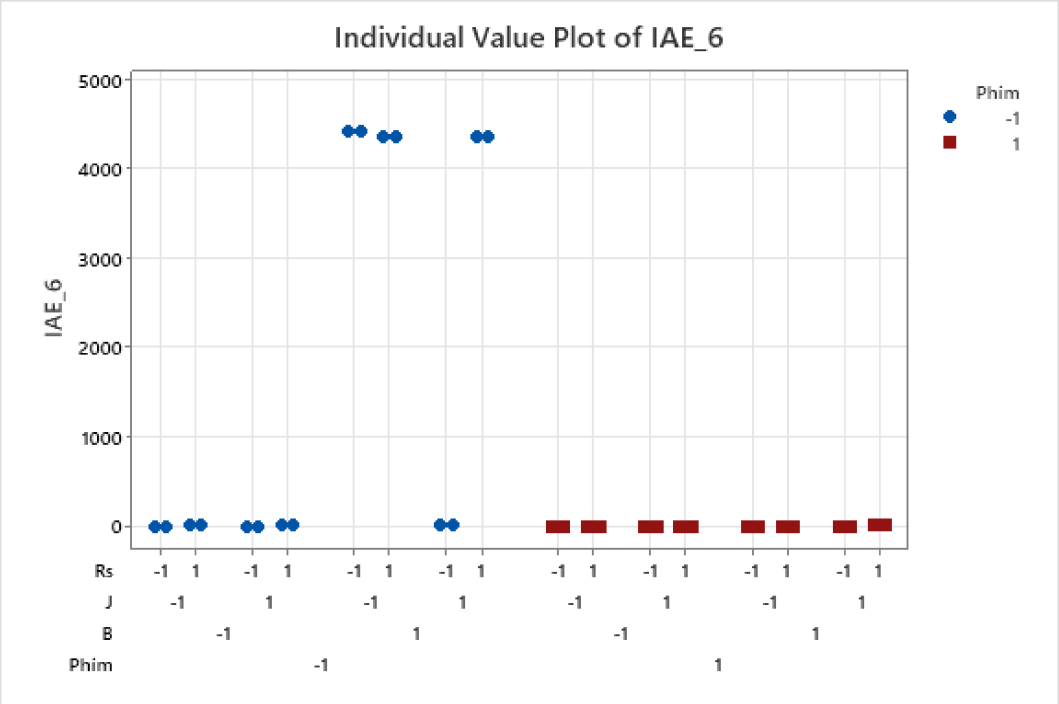


Figure 8.1: IAE - 6 KO

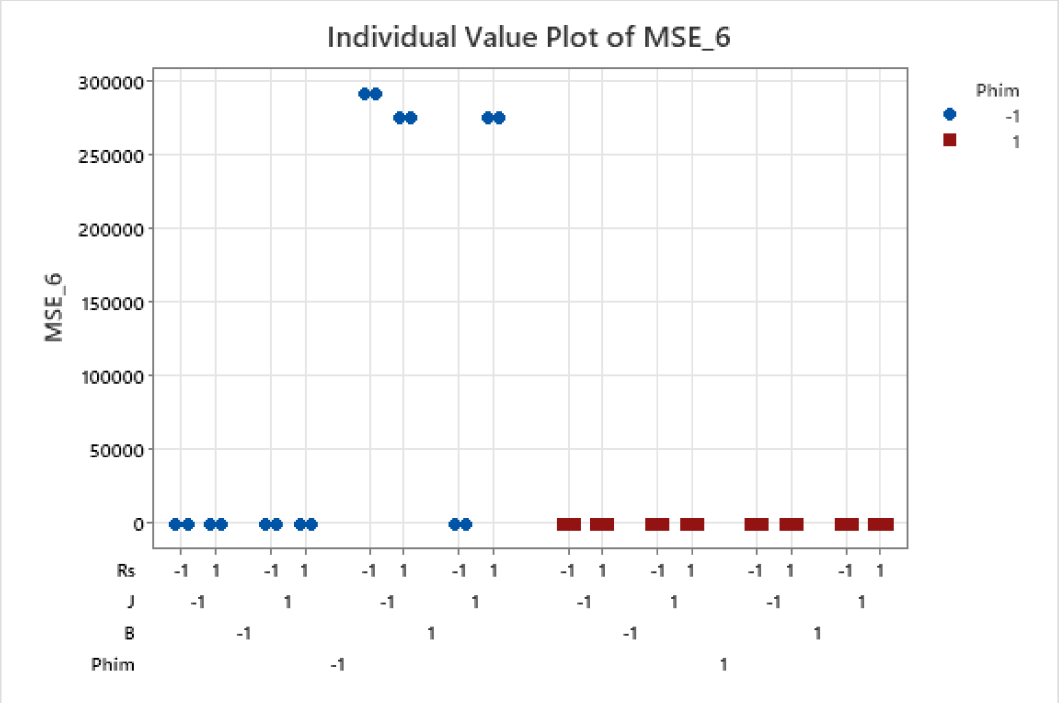


Figure 8.2: MSE - 6 KO

Graphs 8.2 and 8.1 reveals that these knockouts share a common pattern of parameter variation. Specifically, it appears that the regulator is most sensitive to three parameters when they are set to their extremes:

- Phim set to the minimum;
- Rs set to the maximum (in 4/6 KOs);
- B set to the maximum.

On the other hand, the variation of the inductance has no effects overall. Analyzing different negative tests, it was noticed that there was a similarity between them: to address this issue and decrease the likelihood of robustness failure, the parameters Rs and B were set to their maximum value, while the parameter $Phim$ was set to the minimum value, within the sliding mode regulators. This led to a new tuning of the controller's parameters, in order to achieve a balance between control performance and robustness.

8.4 The new tuning outcomes: 6 KOs on 32 tests

IAE Nominal	MSE Nominal	IAE Sensorless	MSE Sensorless
2.295	4.56E-08	6.215	7.21E-04
2.295	4.56E-08	5.891	2.33E-08
2.295	4.56E-08	3.588	2.38E-07
2.295	4.56E-08	4.473	9.31E-08
2.295	4.56E-08	3.819	1.34E-07
2.295	4.56E-08	4.797	6.46E-04
2.295	4.56E-08	3.558	1.83E-07
2.295	4.56E-08	2.309	1.23E-04
2.295	4.56E-08	4619	3.16E+05
2.295	4.56E-08	3.819	1.34E-07
2.295	4.56E-08	3.588	2.38E-07
2.295	4.56E-08	3.558	1.83E-07
2.295	4.56E-08	3.834	7.54E-08
2.295	4.56E-08	2.435	1.30E-04
2.295	4.56E-08	6.215	7.21E-04
2.295	4.56E-08	5.891	2.33E-08
2.295	4.56E-08	3.18	9.31E-08
2.295	4.56E-08	4619	3.16E+05
2.295	4.56E-08	4813	3.29E+05
2.295	4.56E-08	4.473	9.31E-08
2.295	4.56E-08	2.309	1.23E-04
2.295	4.56E-08	2.35	1.19E-04
2.295	4.56E-08	3.18	9.31E-08
2.295	4.56E-08	4681	3.29E+05
2.295	4.56E-08	3.834	7.54E-08
2.295	4.56E-08	2.435	1.30E-04
2.295	4.56E-08	4.797	6.46E-04
2.295	4.56E-08	4681	3.29E+05
2.295	4.56E-08	2.35	1.19E-04
2.295	4.56E-08	5.137	7.61E-04
2.295	4.56E-08	5.137	7.61E-04
2.295	4.56E-08	4813	3.29E+05

Table 8.4: Tests obtained by fixing Rs max, B max and Phim min with new tuning

Table 8.4 indicates that, even after aforementioned implementing changes, the number of KO's remains at 6. It's worth noting that the negative test results obtained are different from those obtained with the previous configuration (8.3). Graphs 8.3 and 8.4 reveal that the "new" KOs exhibit a pattern of parameter variation that is no longer consistent.

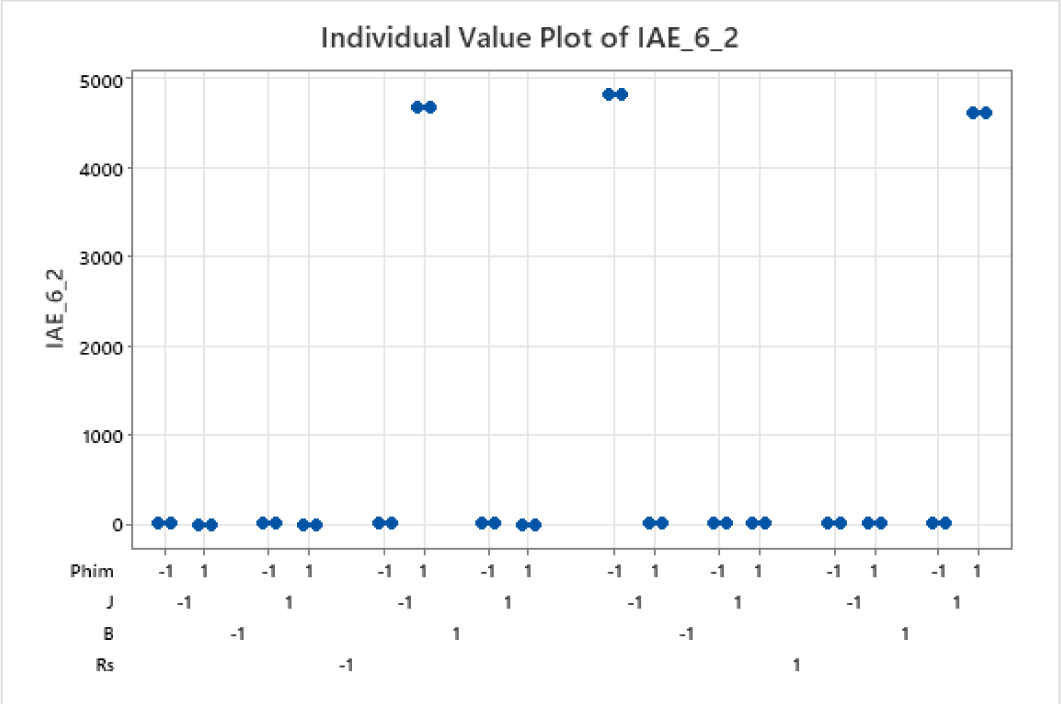


Figure 8.3: IAE - 6 KO

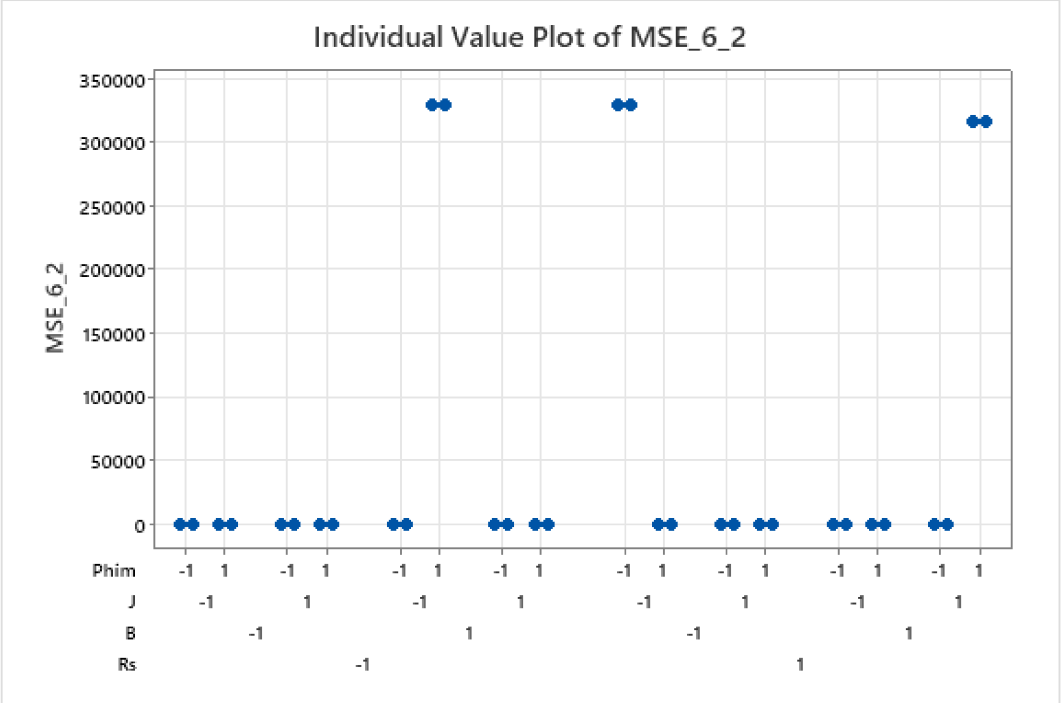


Figure 8.4: MSE - 6 KO

8.5 The K_Torque and Phim modification outcomes: 4 KOs on 32 tests

As the previous optimization has not shown any improvement, the first tuning parameters has been restored. The control variable *Phim* was set to a lower value than the minimum, therefore the *K_Torque* value changes proportionally to *Phim*. In particular, the *Phim* value is fixed at 0.0580 Vpk/rad/s and a *K_Torque* value of 0.087 Nm/A is obtained. Robustness tests will be conducted again by varying the process parameters.

IAE Nominal	MSE Nominal	IAE Sensorless	MSE Sensorless
3.000912	3.36E-07	4.74E+03	3.23E+05
3.000912	3.36E-07	4.65E+00	1.57E-07
3.000912	3.36E-07	2.92E+00	6.79E-07
3.000912	3.36E-07	4.60E+03	3.15E+05
3.000912	3.36E-07	3.10E+00	7.54E-08
3.000912	3.36E-07	5.50E+00	9.31E-08
3.000912	3.36E-07	2.79E+00	1.57E-07
3.000912	3.36E-07	3.59E+00	4.93E-07
3.000912	3.36E-07	2.90E+00	1.34E-07
3.000912	3.36E-07	3.10E+00	7.54E-08
3.000912	3.36E-07	2.92E+00	6.79E-07
3.000912	3.36E-07	2.79E+00	1.57E-07
3.000912	3.36E-07	3.24E+00	1.83E-07
3.000912	3.36E-07	3.49E+00	6.30E-07
3.000912	3.36E-07	4.74E+03	3.23E+05
3.000912	3.36E-07	4.65E+00	1.57E-07
3.000912	3.36E-07	4.25E+00	1.57E-07
3.000912	3.36E-07	2.90E+00	1.34E-07
3.000912	3.36E-07	5.57E+00	7.54E-08
3.000912	3.36E-07	4.60E+03	3.15E+05
3.000912	3.36E-07	3.59E+00	4.93E-07
3.000912	3.36E-07	4.43E+00	4.93E-07
3.000912	3.36E-07	4.25E+00	1.57E-07
3.000912	3.36E-07	4.23E+00	9.54E-07
3.000912	3.36E-07	3.24E+00	1.83E-07
3.000912	3.36E-07	3.49E+00	6.30E-07
3.000912	3.36E-07	5.50E+00	9.31E-08
3.000912	3.36E-07	4.23E+00	9.54E-07
3.000912	3.36E-07	4.43E+00	4.93E-07
3.000912	3.36E-07	4.39E+00	1.83E-07
3.000912	3.36E-07	4.39E+00	1.83E-07
3.000912	3.36E-07	5.57E+00	7.54E-08

Table 8.5: Tests obtained by fixing Phim value inside the regulator

Applied changes produced the results in table 8.5. As displayed KO's number is reduced to 4. In figure 8.5 and 8.6, it is evident that the regulator is sensitive to maximum values of Rs and J.

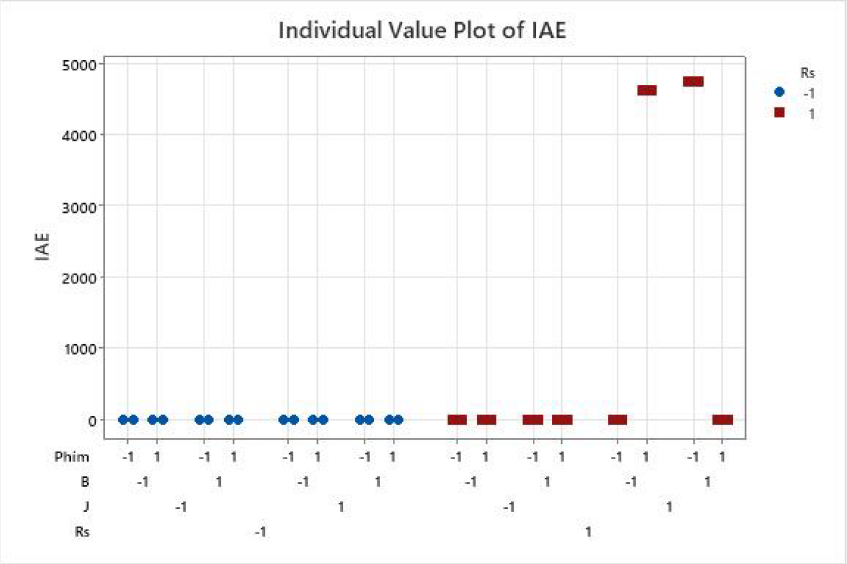


Figure 8.5: IAE - 4 KO

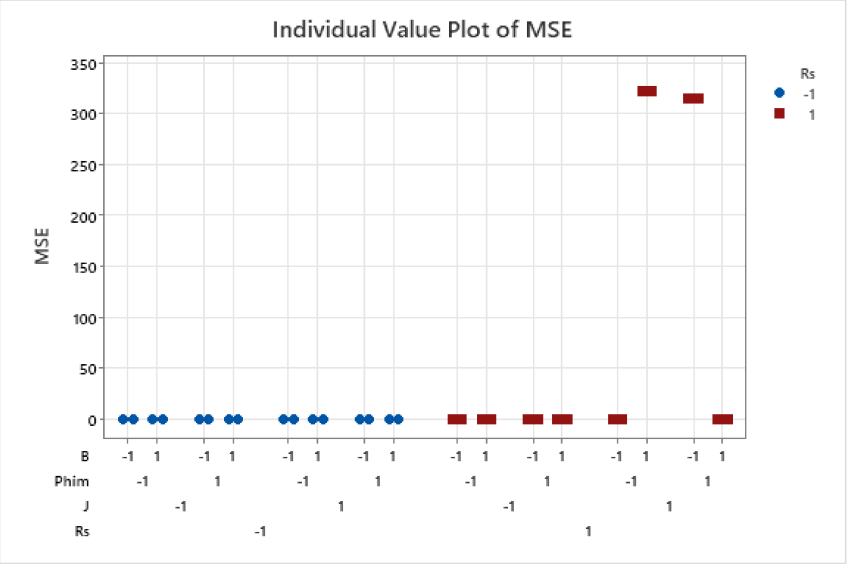


Figure 8.6: MSE - 4 KO

8.6 Sliding Mode Control Robustness: 32/32 OK tests

IAE Nominal	MSE Nominal	IAE	MSE
13.1342268	2.38E-07	13.7096	8.38E-09
13.1342268	2.38E-07	12.3994	2.10E-07
13.1342268	2.38E-07	13.7732	1.49E-08
13.1342268	2.38E-07	12.9721	4.51E-07
13.1342268	2.38E-07	14.6297	9.31E-08
13.1342268	2.38E-07	13.7289	9.31E-10
13.1342268	2.38E-07	13.7498	2.33E-08
13.1342268	2.38E-07	13.128	1.64E-06
13.1342268	2.38E-07	12.3974	1.83E-07
13.1342268	2.38E-07	14.6297	9.31E-08
13.1342268	2.38E-07	13.7732	1.49E-08
13.1342268	2.38E-07	13.7498	2.33E-08
13.1342268	2.38E-07	14.6503	1.57E-07
13.1342268	2.38E-07	12.5269	8.95E-07
13.1342268	2.38E-07	13.7096	8.38E-09
13.1342268	2.38E-07	12.3994	2.10E-07
13.1342268	2.38E-07	12.9722	6.79E-07
13.1342268	2.38E-07	12.3974	1.83E-07
13.1342268	2.38E-07	14.5792	0
13.1342268	2.38E-07	12.9721	4.51E-07
13.1342268	2.38E-07	13.128	1.64E-06
13.1342268	2.38E-07	13.1373	1.57E-06
13.1342268	2.38E-07	12.9722	6.79E-07
13.1342268	2.38E-07	12.5353	8.38E-07
13.1342268	2.38E-07	14.6503	1.57E-07
13.1342268	2.38E-07	12.5269	8.95E-07
13.1342268	2.38E-07	13.7289	9.31E-10
13.1342268	2.38E-07	12.5353	8.38E-07
13.1342268	2.38E-07	13.1373	1.57E-06
13.1342268	2.38E-07	14.5661	8.38E-09
13.1342268	2.38E-07	14.5661	8.38E-09
13.1342268	2.38E-07	14.5792	0

Table 8.6: Tests Robustness passed

The results displayed in table 8.6 indicate that the controller sliding mode has successfully passed all 32 robustness tests. To achieve this, certain control parameters were adjusted, including those of the speed and current regulators, as well as the K_Torque and $Phim$ variables. These adjustments were made through the tuning of:

- Speed regulator
  - a1=20;

  - ρ=0.018;
  - ε=130;
- Current Iq regulator
  - a2=500;
  - ρ_q=100;
  - ε_q=10;
- Current Id regulator
  - a3=500;
  - ρ_d=100;
  - ε_d=3000;
- K_Torque=0.09;
- $Phim$=0.06.

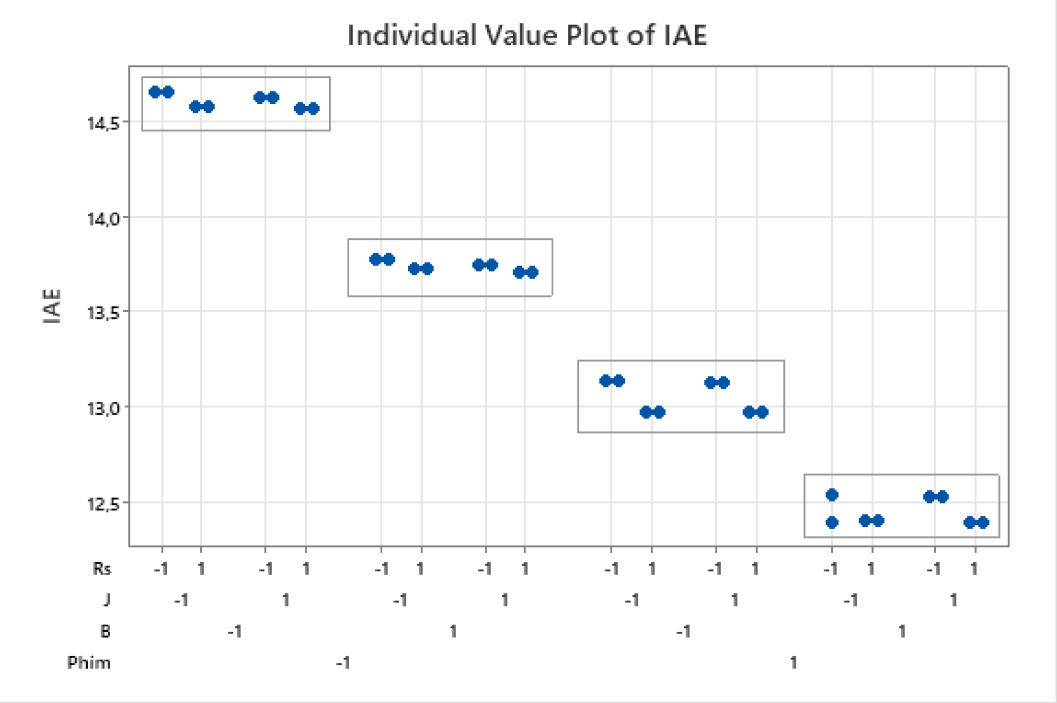


Figure 8.7: IAE - OK

Table 8.7 shows the results of the robustness tests for PI regulators.

IAE Nominal	MSE Nominal	IAE	MSE
56.4062	2.10E-07	64.4127	7.54E-08
56.4062	2.10E-07	53.8872	2.33E-08
56.4062	2.10E-07	64.4751	2.69E-07
56.4062	2.10E-07	53.7957	1.13E-07
56.4062	2.10E-07	64.2792	1.34E-07
56.4062	2.10E-07	64.3975	4.56E-08
56.4062	2.10E-07	64.5113	2.33E-08
56.4062	2.10E-07	53.8351	1.83E-07
56.4062	2.10E-07	53.9064	9.31E-10
56.4062	2.10E-07	64.2792	1.34E-07
56.4062	2.10E-07	64.4751	2.69E-07
56.4062	2.10E-07	64.5113	2.33E-08
56.4062	2.10E-07	64.2402	1.13E-07
56.4062	2.10E-07	53.9458	8.38E-09
56.4062	2.10E-07	64.4127	7.54E-08
56.4062	2.10E-07	53.8877	2.33E-08
56.4062	2.10E-07	53.7716	1.13E-07
56.4062	2.10E-07	53.9064	9.31E-10
56.4062	2.10E-07	64.1643	1.13E-07
56.4062	2.10E-07	53.7957	1.13E-07
56.4062	2.10E-07	53.8351	1.83E-07
56.4062	2.10E-07	53.8111	1.83E-07
56.4062	2.10E-07	53.7716	1.13E-07
56.4062	2.10E-07	53.9276	9.31E-10
56.4062	2.10E-07	64.2402	1.13E-07
56.4062	2.10E-07	53.9458	8.38E-09
56.4062	2.10E-07	64.3975	4.56E-08
56.4062	2.10E-07	53.9276	9.31E-10
56.4062	2.10E-07	53.8111	1.83E-07
56.4062	2.10E-07	64.1839	1.13E-07
56.4062	2.10E-07	64.1839	1.13E-07
56.4062	2.10E-07	64.1643	1.13E-07

Table 8.7: Tests Robustness PI

8.7 Qualitative and quantitative evaluation of the conducted experiments

In the first experiment, most of KOs has Rs set to its maximum value. In both tests with 6 KOs Rs is set to its maximum in 4/6 cases. These observations made it possible to understand which parameters the SMC is most susceptible to. Changing control K_Torque, according to control $Phim$ to lower values then their minimum, robustness failures decreased. The latest tests were conducted by setting $Phim$ to a value smaller than the minimum and adjusting the parameters' tuning. While

the speed does not seem to be influenced by the new tuning, there is a small offset between the reference torque and the real torque. Nevertheless, it is evident that the control preserves its readiness (figure 8.8). The difference between IAE and MSE nominal values and their value in presence of parametric variation is negligible, both using PI and SMC controllers: anyway it is worth to highlight that this difference is ±1 using SMC and is -3 or +8 using PI. Thus, these results demonstrate appreciable SMC robustness.

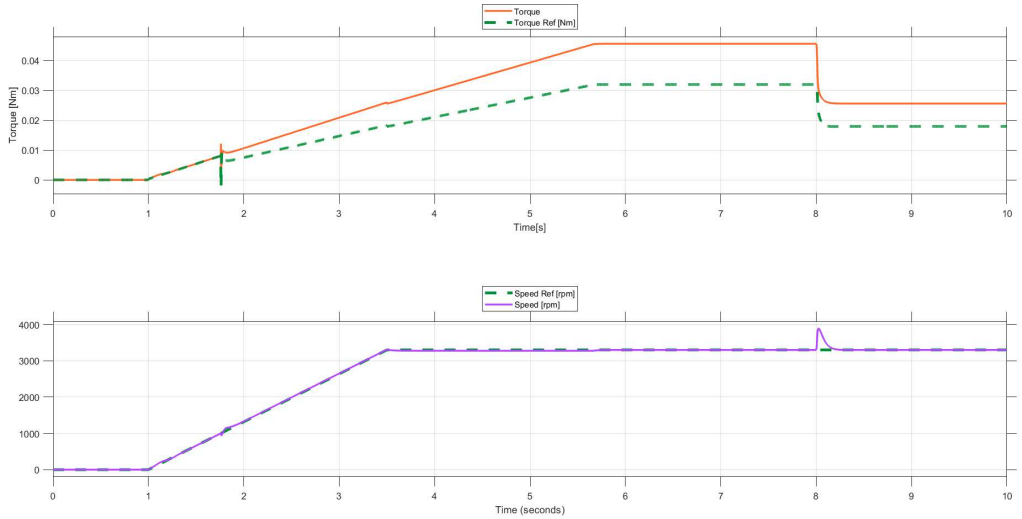


Figure 8.8: Torque and Speed SMC with the new tuning

Chapter 9

Application on the real system and future developments

Thanks to the excellent results obtained in the simulation, the sliding mode control was deemed suitable for testing on the real process. The Simulink blocks containing the regulators are designed for self-generation of the C code. The C code is then imported onto the microcontroller inside the appliance. L. Moretti's thesis [11] deals with the experimental validation of the SMC, collecting some important results in qualitative and quantitative terms regarding performance and robustness; her discussion demonstrates the applicability of a technique that shows excellent qualities already in simulation.

The results obtained are satisfactory, indicating the effectiveness of SMC. However, there is still room for improvement in control performance. In fact, some undesired effects can be reduced by further adjusting the tuning preserving the overall robustness. The analysis for this thesis was conducted using Whirlpool's Observer, which is essentially a Luenberger's Observer. However, it could be worthwhile to consider replacing it with a Sliding Mode Observer (SMO). Several studies have shown that an SMO can provide effective estimates of rotor position and speed, and can achieve good static and dynamic performance. The SMO is popular due to its simple algorithm and robustness, which reduces the observer's dependence on the model [11]. This possibility suggests that a full Sliding Mode approach could be a feasible solution with unexpected but excellent results.

Chapter 10

Conclusions

This thesis objective was to design and develop a model based and sensorless control system for a dishwasher permanent magnet drain pump. This aim has been widely pursued, creating a speed regulator and two current regulators with sliding mode control technique. The design took place in the Simulink simulation environment, in particular through the MATLAB/Simulink project created by the Whirlpool company of Fabriano. By applying the SMC to the simulated process, graphs were obtained representing the trends of some signals of interest; among these are speed, torque and currents. The SMC demonstrated excellent readiness, precision and accurate tracking between real signals and their respective references. The graphs of the SM regulators were compared with those of the PI regulators: the latter present some oscillations in the transients and some spikes in the moments in which the load torque changes; while the SMC does not seem to be affected by these variations in an equally evident way. The behavior of the SM and PI regulators was verified even in the presence of saturations. Both types of controllers react promptly to saturation, maintaining their overall stability. The robustness tests performed also led to extremely positive results regarding the behavior of sliding mode controllers. The positive outcomes of these tests are displayed by the values taken by the IAE and MSE indices. Given the undeniable quality of the control action achieved, sliding mode controllers were deemed suitable for their application on a household appliance. Given the fidelity with which the Whirlpool simulator reproduces the behavior of the real process and controllers, sliding mode controllers are expected to maintain their performance even in reality.

Bibliography

[1] L. Moretti, *Sensorless control of a permanent magnet synchronous motor: experimental validation on an appliance*, UNIVPM, 2024

[2] Maria Letizia Corradini, *Senior Member, IEEE*, Gianluca Ippoliti, Sauro Longhi, *Senior Member, IEEE*, and Giuseppe Orlando: *A Quasi-Sliding Mode Approach for Robust Control and Speed Estimation of PM Synchronous Motors*, IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 59, NO. 2, FEBRUARY, 2012.

[3] R. A. DeCarlo, S. H. Zak and G. P. Matthews, *Variable structure control of nonlinear multivariable systems: a tutorial* in Proceedings of the IEEE, vol. 76, no. 3, pp. 212-232, March 1988

[4] G. Orlando, *Tecniche di controllo a struttura variabile per un veicolo sottomarino comandato a distanza*, 1996.

[5] A. F. Filippov, *Differential Equations with Discontinuous Righthand Sides*, F. M. Arscott, 1988

[6] J.J.E. Slotine, W. Li: *Applied Nonlinear Control*, PRENTICE HALL, 1991.

[7] Hassan K. Khalil: "Nonlinear Design Tools" in *Nonlinear Systems - Third Edition*, PRENTICE HALL, 2002.

[8] K.Astrom, T.Hagglund: "Integrator Windup" in *PID Controllers: Theory, Design and Tuning - Second Edition*, INSTRUMENTS SOCIETY OF AMERICA, 1995

[9] A. E. Fitzgerald, C. Kingsley Jr., A. Kusko, *Macchine elettriche*, FRANCO ANGELI EDITORE, 2006.

[10] IEEE Standard 61012-1990 *IEEE Standard Glossary of Software Engineering Terminology*, Dec. 1990, pp. 1–84.

[11] Zhaowei Qiao, Tingna Shi, Yindong Wang, Yan Yan, Changliang Xia, *Senior Member, IEEE* and Xiangning He, *Fellow, IEEE*, *New Sliding-Mode Observer for Position Sensorless Control of Permanent-Magnet Synchronous Motor*, IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 60, NO. 2, FEBRUARY 2013