UNIVERSITÀ
POLITECNICA
DELLE MARCHE

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE

Controllo Sensorless basato su un Osservatore Sliding Mode per un PMSM: validazione sperimentale su un elettrodomestico

Sliding Mode Observer based Sensorless Control of a PMSM: experimental validation on an appliance

Advisor:

Prof. Gianluca Ippoliti

First Coadvisor:

Prof. Giuseppe Orlando

Second Coadvisor:

Ing. Luigi Fagnano

Candidate:

Eleonora Brasili

Academic Year: 2023-2024

Abstract

Questa tesi si propone di studiare e sviluppare un osservatore basato sulla tecnica *sliding mode* (SMO) per il controllo *sensorless* di un motore sincrono a magnete permanente. L'obiettivo principale è progettare un osservatore in grado di stimare con elevata precisione le variabili di stato del motore, come la posizione e la velocità del rotore, consentendo così di implementare un controllo efficiente senza l'uso di sensori fisici. In particolare è stato implementato un osservatore Super-Twisting Sliding-Mode (STSMO), che a differenza dei tradizionali osservatori sliding-mode risulta meno sensibile al fenomeno del "*chattering*". L'osservatore è integrato in un sistema di controllo completo, che sfrutta a sua volta la tecnica sliding mode (SMC) sia per la regolazione della velocità sia per il controllo della corrente del motore. L'implementazione del progetto è stata realizzata in ambiente MATLAB *Simulink*, sfruttando le potenzialità di un simulatore proprietario messo a disposizione dalla Whirlpool di Fabriano. Tale simulatore ha fornito gli strumenti utili alla modellazione e alla simulazione del PMSM e del sistema di controllo, permettendo di testare e ottimizzare l'osservatore proposto in condizioni operative realistiche. In fase di simulazione, le prestazioni del controllo SMC, in combinazione con lo STSMO, sono state confrontate con quelle del controllo PI, attualmente impiegato nelle produzioni Whirlpool, integrato con un osservatore di Luenberger. La validità e l'efficacia dell'approccio proposto sono state testate mediante simulazioni e sperimentazioni pratiche, al fine di valutare le prestazioni del sistema e la sua capacità di mantenere un controllo stabile e preciso anche in presenza di disturbi. La combinazione di osservatore e controllo sliding mode si è dimostrata una soluzione valida per il controllo di motori PMSM, offrendo prestazioni equivalenti a quelle delle soluzioni tradizionali, ma con il vantaggio di una maggiore robustezza contro le variazioni dei parametri del sistema.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent decades, industry and academic research have shown increasing interest in permanent magnet synchronous motors (PMSM). These motors have excellent performance characteristics, including high power density, energy efficiency and mechanical robustness [1, 2, 3]. These advantages make them ideal for a wide range of applications, from electric traction to industrial automation systems. However, the control of PMSMs requires accurate rotor position and speed information, typically obtained through physical sensors. Although effective, these sensors introduce complexity, additional costs and potential failure points into the system.

Against this backdrop, there is a need to develop '*sensorless*' control techniques that allow the quantities required to control the motor to be estimated, eliminating the dependence on physical sensors.

Sensorless control of PMSM is a particularly active and evolving area of research, driven by the need to reduce the cost, complexity and vulnerability of traditional control systems, which require physical sensors to monitor rotor position and speed. Sensorless techniques aim to estimate these quantities directly from measured electrical signals, such as motor voltages and currents, thus eliminating the need for dedicated sensors. Early techniques developed for sensorless control of PMSMs were based on linear observers, such as the Extended Kalman Filter (EKF) [4] and the Luenberger observer [5], which use a mathematical model of the motor to estimate rotor position and speed. These methods have been widely used due to their ability to provide accurate estimates over a wide range of operating conditions. However, their effectiveness is limited by their sensitivity to variations in motor parameters, such as resistance and inductance, and to external perturbations.

Another class of techniques is based on injecting high-frequency signals into the motor to extract rotor position information from the response currents [6]. These methods are particularly effective at low speeds, where model-based techniques tend to lose accuracy. However, the injection of additional signals can introduce noise and complexity into the system.

In recent years, *sliding mode* technique has emerged as a promising solution for sensorless control of PMSMs due to its robustness against model uncertainties and perturbations. This technique is based on the design of a *sliding surface* such that,

once achieved, the control system can compensate for parameter variations and external perturbations, keeping the system stable. This type of control is particularly suitable for non-linear systems, such as permanent magnet synchronous motors, where load variations and non-linearities can significantly affect performance.
The sliding mode observer (SMO) has emerged as an interesting candidate to estimate the rotor position [7]-[8]. The control loop in an ordinary observer is replaced by a sliding-mode variable structure, and when the system error reaches the sliding mode, the system dynamic performance entirely depends on the sliding surface, which ensures good robustness of the entire system to parameter variations. The discrete switch control can become the cause of the so-called chattering phenomenon. The chattering is an inherent characteristic of the sliding mode technique and as such cannot be totally eliminated, but can be mitigated at the design stage. Indeed, a trade- off between chattering reduction and system robustness is needed.
To attenuate the chattering effect, several methodologies are proposed in sliding mode literature, super-twisting algorithm (STA) is one among them [9] [10]. The effectiveness of this approach has been demonstrated in many works. It has been shown that this approach can lead to a massive reduction in chattering without the use of low-pass filters.
The main contribution of this project is the exploration of the sliding mode technique for both controlling and observing the state variables of a real PMSM. The primary objective is to study and design a Sliding Mode Observer, and then analyze its behavior combined with the control performed by a Sliding Mode Controller, whose implementation is described in [11] and [12]. Given the promising results, the sliding mode approach can be considered a good alternative to traditional approaches, ensuring a good level of accuracy and at the same time an excellent level of robustness. The proposed approach was also applied in practice, further proving its effectiveness on real world scenario. The project was implemented in the MATLAB *Simulink* environment, exploiting the potential of a proprietary simulator made available by Whirlpool in Fabriano. This simulator provided an advanced platform for modelling and simulation of the PMSM and the control system, allowing the proposed observer to be tested and optimised under realistic operating conditions. The validity and effectiveness of the developed approach was verified through an extensive series of simulations, which allowed the performance of the system to be evaluated in terms of estimation accuracy and robustness in the presence of model disturbances. In particular, the proposed approach has found successful application in the control of the drain motor of a dishwasher produced by Whirlpool.
The main sections of this thesis can be summarized as follows: Chapter 2 will be dedicated to an exhaustive description of the Whirlpool Simulator used in the development of the observation algorithm. Chapter 3 aims to review all the theoretical concepts underlying the development of this project, including the Park Transformation and the fundamentals of the sliding-mode theory. In Chapter 4 the design that was carried out to realise the observer is described, including the problem

formulation and the resulting *Simulink/MATLAB* model. In Chapter 5 will show the graphical results of what was obtained from the simulations. The discussion on the results will be further developed and comparisons will be conducted with the performance of the system implemented in Whirlpool devices currently on the market. Chapter 6 will describe the robustness tests that were conducted by varying the system parameters. Chapter 7 will cover the illustration of the steps that led to the generation of the C code implementing the observer, describes the set-up of the experiments and illustrates the obtained results.

Chapter 2

Simulator

This chapter is dedicated to a detailed description of the Whirlpool simulator. It consists of a *Simulink* model which is modularly structured and organised into several subsections, each of which represents an essential part of the engine control system. The start page consists of four main blocks (Fig.2.1), each responsible for a specific function.

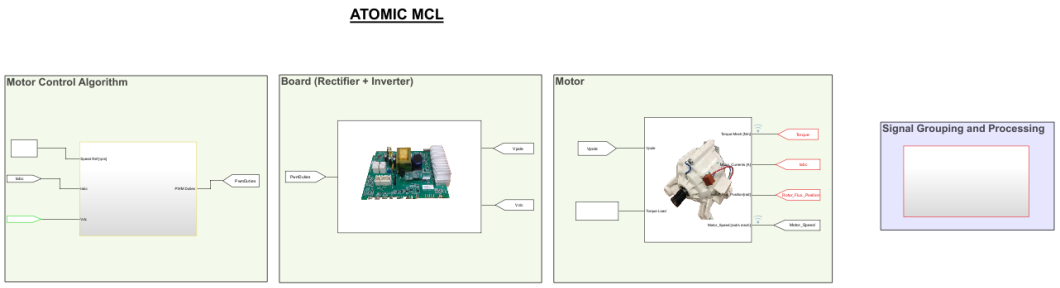


Figure 2.1: Simulator Model

1. **Motor Control Algorithm**: It contains the control logic and algorithms required to manage the motor's performance and dynamics. It is the core of the system, responsible for implementing the control strategies that ensure efficient and stable operation of the motor.

2. **Board (Rectifier + Inverter):** This section models the power electronics system, which includes both the rectifier and the inverter. The rectifier converts AC power from the grid into DC power, and the inverter then converts this DC power back into controlled AC to drive the motor. This part simulates the electrical behavior of the board and how it responds to control signals.

3. **Motor Model:** This block contains the dynamic model of the motor, capturing its electrical and mechanical characteristics. It simulates the motor's response to control inputs and external loads.

4. **Signal Grouping and Processing:** This section gathers and processes the data from the system, showing the results through various plots. It's useful for monitoring and analyzing how the system performs.

For the purposes of this thesis, the focus has been specifically on the Motor Control Algorithm block. The detailed description of this subsystem content is provided in the following section.

2.1 Motor Control Algorithm Block

This subsystem encompasses the entire implementation of the control system, including the design and coding of the control and the observer algorithms. Within it, it is possible, by means of special switches, to switch from one control method to another so that the performance of the various configurations can be easily compared. Below is a summary diagram that concisely describes the logic applied in implementing the system.

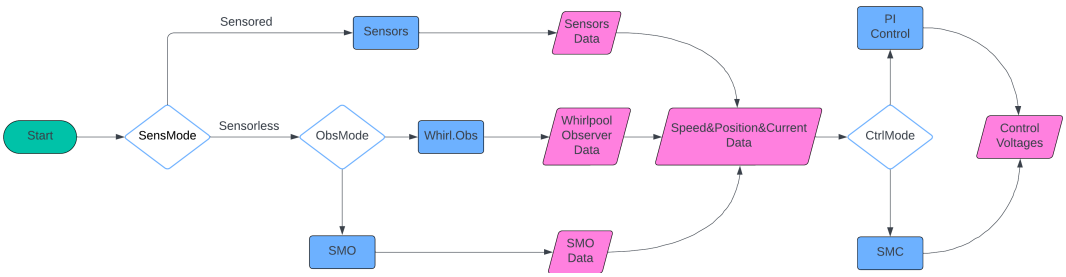


Figure 2.2: Implemented Logic scheme

The process begins by evaluating the value of the *SensMode* variable: the switch in Fig.2.10a determines whether to run the simulation in "sensored" or "sensorless" mode, depending on the numerical value assigned to this variable (Fig.2.11) (1 = Sensored , 0 = Sensorless). If "sensored" mode is chosen, speed, position and current data from dedicated sensor measurements are collected. If, on the other hand, "sensorless" mode was chosen, the *ObsMode* variable is evaluated: based on the value assigned to it, it is determined which observer is to be used for the estimation of speed, position and current (1 = Whirlpool, 3 = Proposed Observer). After that, the value assigned to the variable *CtrlMode* is evaluated: through the value of this variable, one can determine which of the two control methods to use to calculate the voltages to be impressed (0 = Sliding Mode Control, 1 = PI Control). The next sections will review all the functional components integrated in the Motor Algorithm block.

2.1.1 Controllers

This section is dedicated to the description of the blocks containing the implementation of the controllers whose performance is to be compared. The subsystems shown in the figure Fig.(2.3) implement the Sliding Mode Control logic within them. As shown there are two dedicated blocks: one for speed control, one for current control([11],[12]).

Sliding Mode Speed Control The speed control block receives the following signals as input:

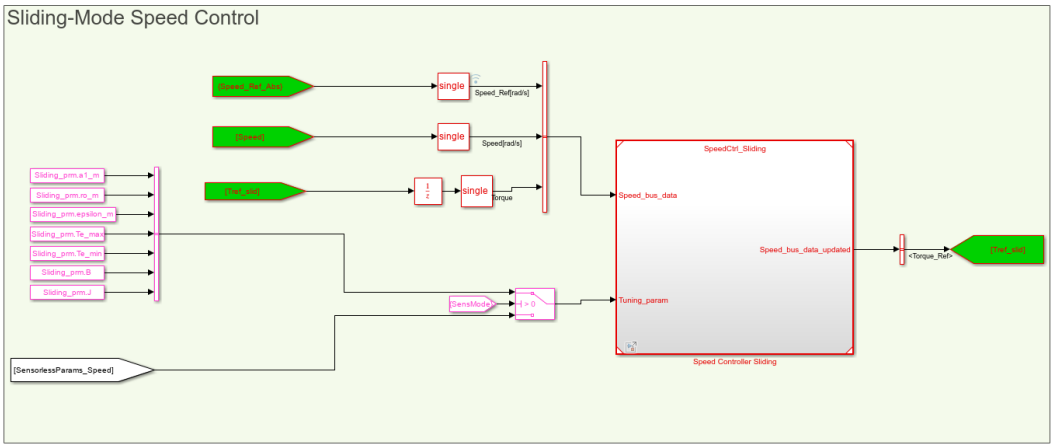
- *Speed_Ref_Abs*: reference speed
- *Speed*: sensor-sensed speed if sensored mode was set or observer-estimated speed if sensorless mode was set
- *Tref_slid*: reference torque generated at the previous sampling instant
- *Tuning_param*: parameter vector valorised according to the chosen simulation mode (sensored/sensorless)(Vedi Appendice B).

The first three signals listed are passed as input to the block through their assignment in the fields of a special bus (*Speed_bus_data*), i.e. a collection of signals that are handled as a single object.
As output we have *Speed_bus_data_updated* , the bus updated with the new value of the reference torque calculated at the current sampling time.

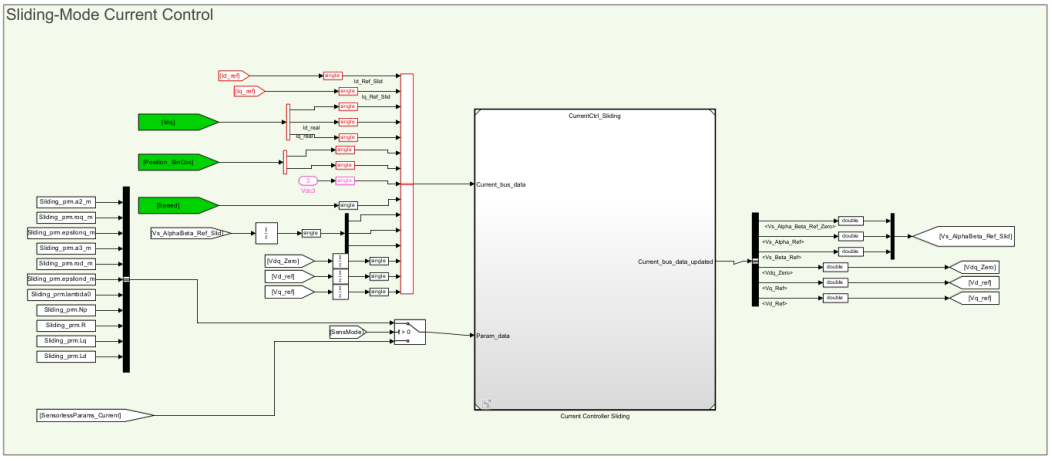
Sliding Mode Current Control The current control block receives the following signals as input:

- *Id_ref,Iq_ref* : current signals with respect to reference axes *d,q* from the torque/current transformation block
- *Izero,Id,Iq* : measured/estimated d,q current signals
- *Position_SinCos* : sine and cosine of the measured/estimated angle
- *Speed* : measured/estimated speed
- *Vs_AlphaBeta_Ref_Slid* : control voltage with respect to axes α, β (at the previous sampling time)
- *Vdq_Zero,Vd_ref,Vq_ref* : control voltage with respect to axes *d,q* (at the previous sampling time)
- *Params_data* : parameters required for the current control law (Vedi Appendice B)

All input signals to the block, except parameters, are grouped in the *Current_bus_data* bus. The latter will be updated with the values calculated by the controller. The output is *Current_bus_data_updated*.



(a) Speed SMC

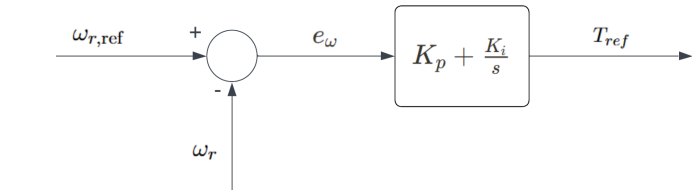


(b) Current SMC

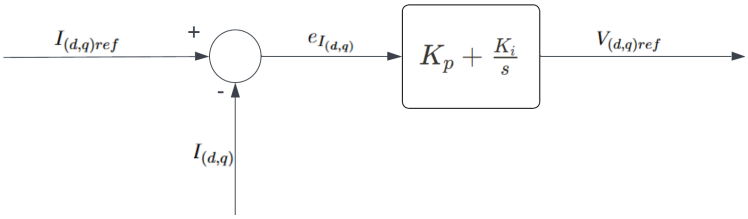
Figure 2.3: Sliding Mode Control subsystems

PI Control Proportional Integral control (PI) is the method implemented in the Whirlpool industrial productions. Within the simulator, as in the sliding mode case, we have two blocks dedicated to the implementation of this technique: one for speed control and one for current control Fig.(2.6).

The speed control loop (Fig 2.4a) ensures that the motor speed follows the desired reference. The PI controller for the speed loop is designed based on the error between the reference speed $\omega_{r,\text{ref}}$ and the actual speed ω_r. The controller adjusts the reference torque T_{ref}. The reference current $I_{q,ref}$ for the current control loop is generated via the torque/current transformation (Fig 2.5). The current control loop (Fig 2.4b) regulates the stator currents, ensuring that they track their reference values. Typically, two PI controllers are used for the d-axis and q-axis currents. Since $I_d = 0$, the main focus is on controlling I_q, which directly influences the torque production.



(a) Speed Control PI



(b) Current Control PI

Figure 2.4: PI Control scheme

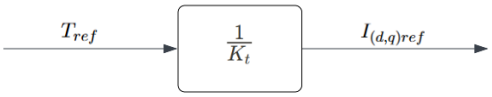
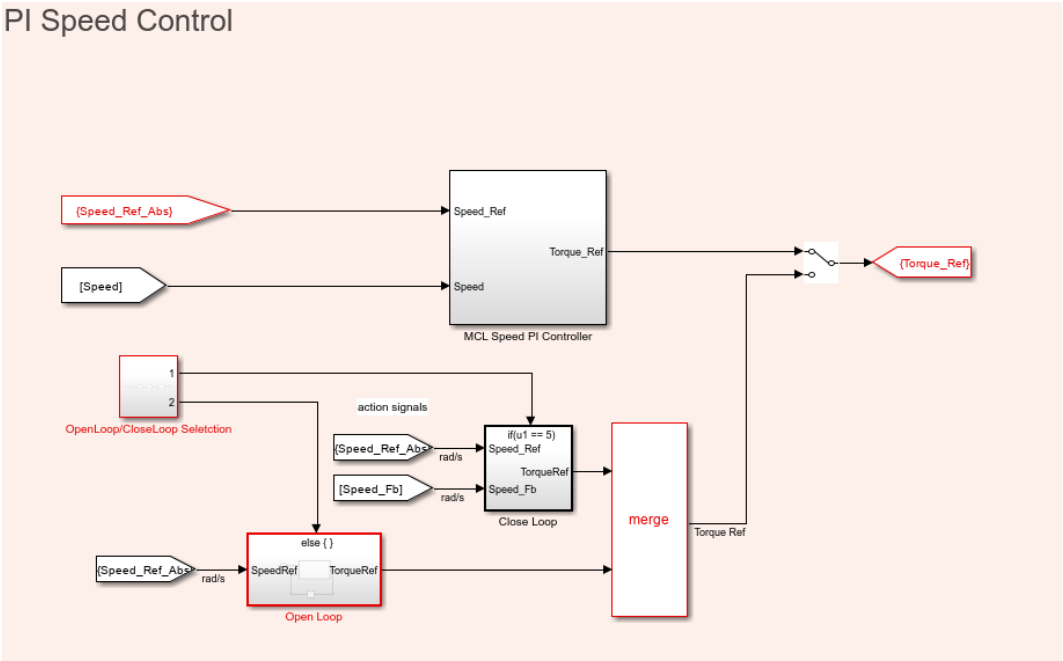
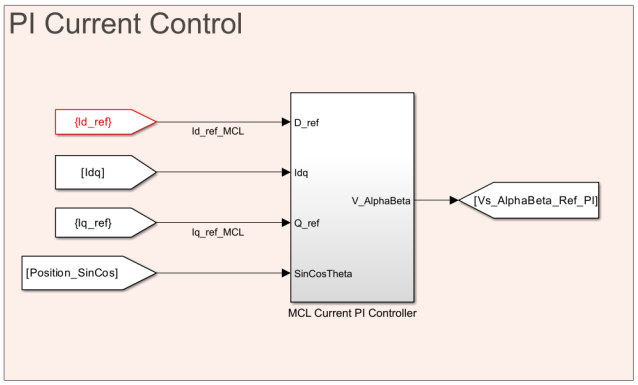


Figure 2.5: Torque/Current Transformation Scheme

Below is the modelling of the PI control in Simulink. There are additional blocks that deal with activating the close loop when the rotation speed reaches 1000 rpm.



(a) Speed PI



(b) Current PI

Figure 2.6: PI Control subsystems

2.1.2 Observers

MCL Observer The observer implemented by Whirlpool consists of a Luenberger observer (Fig. 2.7). It receives as input the current and voltage values in the $\alpha - \beta$ coordinates and returns as output the estimated rotational speed, sine and cosine of the estimated position and estimated current in the $d - q$ coordinates. The algorithm is implemented through an *S-function* contained in the block shown in the figure. The used parameters are specified within the subsystem. There are additional blocks useful for converting data into other units of measurement. In detail, the input and output signals are:

Input

- *I_AlphaBeta*: measured current in the $\alpha - \beta$ coordinates
- *Vs_AlphaBeta_Ref*: measured voltage in the $\alpha - \beta$ coordinates

Output

- *Speed_Est_Mech*: estimated speed
- *Position_Est_SinCos_MCL*: sine and cosine of the estimated position
- *Idq_Est_MCL*: current in the $d - q$ coordinates

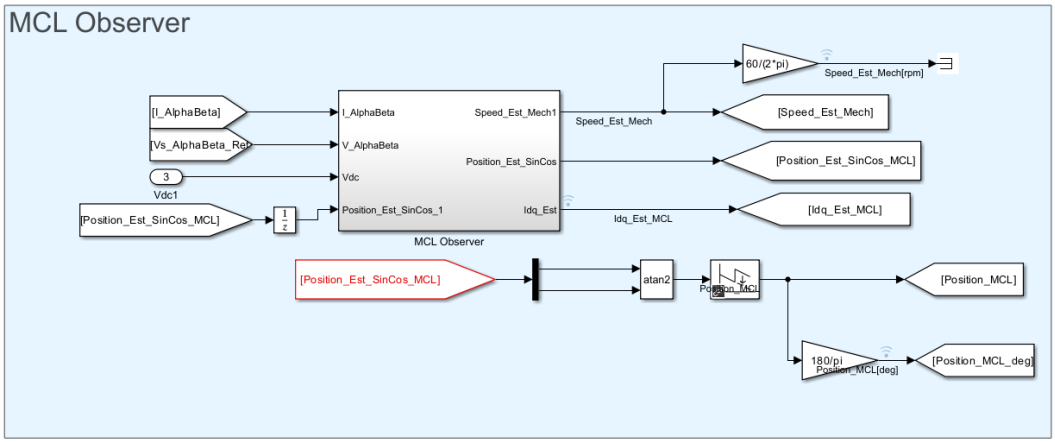


Figure 2.7: MCL Whirlpool Observer

Sliding Mode Observer (SMO) Again, for the sake of convenience, inputs and outputs pass through special buses (*obs_bus,obs_bus_updated*) dedicated to the data required for the processing carried out by the observer. Even in this case, there are switches to select the set of parameters according to the chosen mode. In detail, the input and output signals are:

Input

- *I_AlphaBeta*: measured current in the $\alpha - \beta$ coordinates
- *V_AlphaBeta*: measured voltage in the $\alpha - \beta$ coordinates

Output

- *Speed_Est_nSMO*: estimated speed
- *Posititon_Est_SinCos_nSMO*: sine and cosine of the estimated position
- *Idq_nSMO*: current in the $d - q$ coordinates

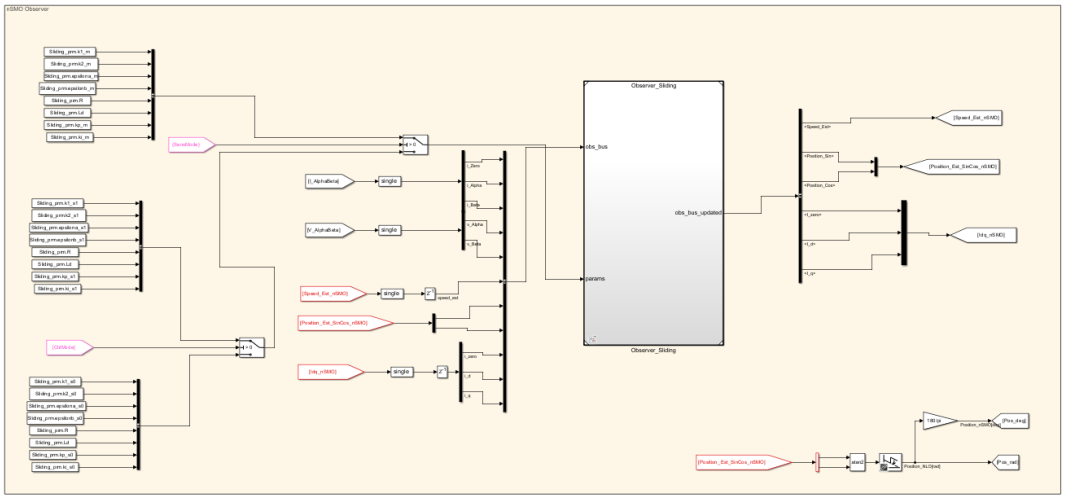


Figure 2.8: Sliding Mode Observer

The logic implemented in this block will be further explored in the following chapters.

2.1.3 Transformation

This block is dedicated to transforming the reference torque signal, coming from the speed control, into the corresponding current signal which will be the input reference for the current control block. The reference torque signal is chosen according to the control mode selected via a switch.

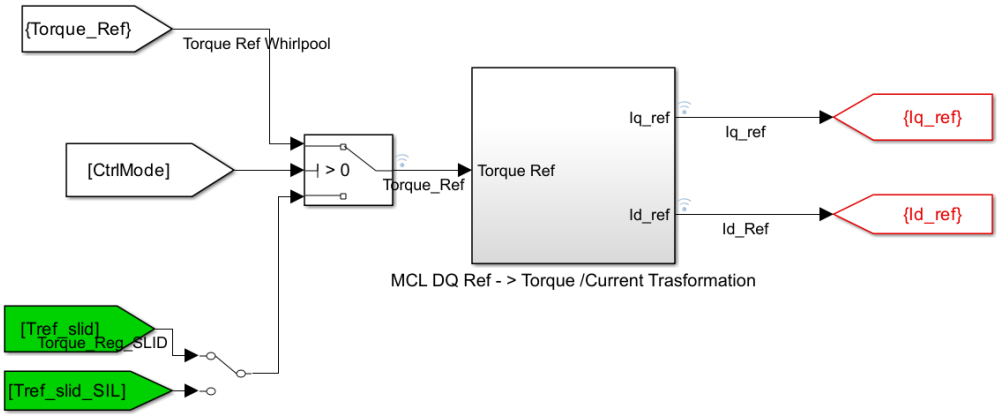
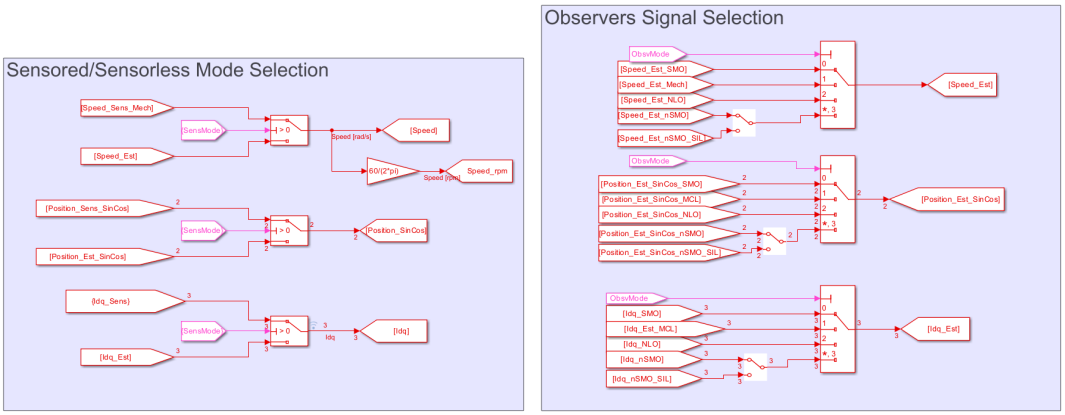


Figure 2.9: Torque/Current transformation

2.1.4 Mode Switches



(a) Sensored/Sensorless mode switches (b) Observer mode switches

Figure 2.10: Mode Switches

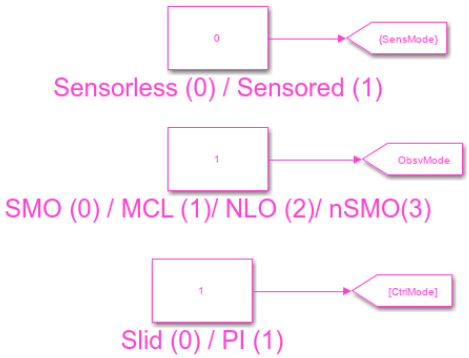


Figure 2.11: Assignment of the variables evaluated by the switches.

2.1.5 PWM Modulation

The control voltages in the *dq* reference frame from the current control block are transformed into the $\alpha\beta$ reference frame using the Inverse Park transform. Voltages with respect to the $\alpha\beta$ axes are exploited by Space Vector Modulation. This modulation scheme is used to apply a given voltage vector to the three-phased electric motor via a steady state DC-voltage.

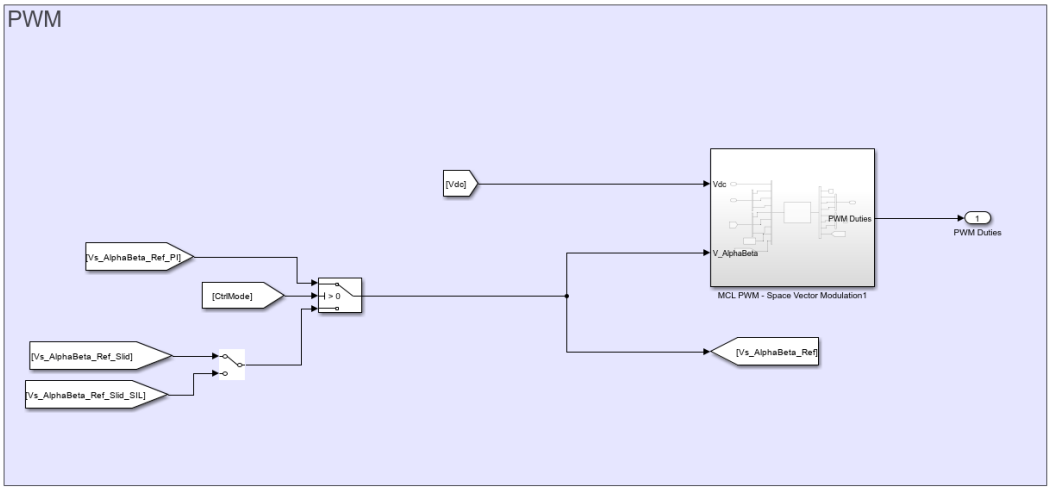
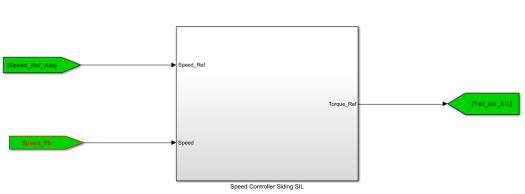


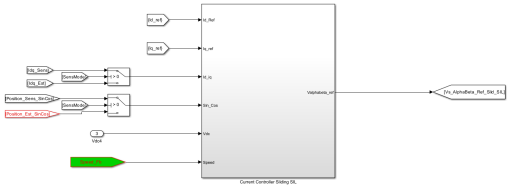
Figure 2.12: PWM Modulation

2.1.6 SIL Blocks

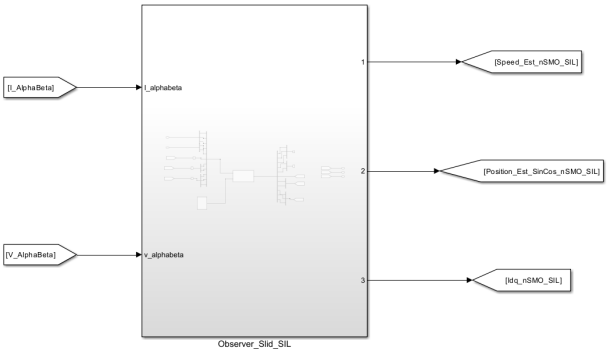
Below are the blocks dedicated to simulation in Software-in-the-loop mode (Fig. 2.13). The latter refers to the verification technique in which the control software to be verified is made to interact directly with the emulation of the system it is intended for in order to stress it as if it were under operating conditions. Within them is an *s-function* that implements the C-code resulting from the autogeneration.



(a) SIL SMC Speed



(b) SIL SMC Current



(c) SIL SMO

Figure 2.13: SIL blocks

2.2 Project Directory Structure

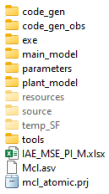


Figure 2.14: Project Folder

It is now useful to describe how the project folder is organised, and which files to execute to start a simulation. First, the working environment must be configured by opening the *"mcl_atomic.prj"* file. Double-clicking it loads the project structure, including all associated files, functions and settings.
Secondly, you need to access the *"main_model"* folder. The script files that manage the execution of the process and the simulink model of the simulator are all contained in this folder. To start a simulation, the following scripts must be executed in this order:

1. *"BusInit_CodeGen.m"*
2. *"parameters_sliding_init.m"*

All buses types are defined in the first file. They are used to simplify signal management, improve organisation and reduce the complexity of the model. They can be seen as the "skeletons" of the structures that will later be populated by the data. Two are assigned for each type of macro-component whose signals are to be grouped (one for inputs and outputs, one for parameters)

- MCL_CURRENT_SMC_IO_F_TYPE : bus type for current control I/O
- MCL_CURRENT_SMC_PARAMS_TYPE : bus type for current control parameters
- MCL_SPEED_SMC_IO_F_TYPE : bus type for speed control I/O
- MCL_SPEED_SMC_PARAMS_TYPE : bus type for speed control parameters
- MCL_SLID_OBS_IO_TYPE : bus type per observer I/O
- MCL_SLID_OBS_PARAMS_TYPE : bus type for observer parameters

The second script file is where all the tunable parameters of all algorithms are assigned to their respective variable. All variables are placed within a structure called "*Sliding_prm*".
Once these files have been executed, the simulation can be started, remembering to assign the values of *SensMode, CtrlMode, ObsMode* to set the desired mode.

Chapter 3

Theoretical background and Preliminary Notions

In the following chapter, a comprehensive review of all the fundamental concepts underlying this work is provided to ensure both clarity and completeness. Each concept is revisited in detail, offering explanations to understand the foundational ideas that support the subsequent discussions.

3.1 Park Transformation

The Park Transformation is a mathematical tool used in electrical engineering, particularly in the analysis of three-phase electrical machines such as induction motors and synchronous generators. Its main purpose is to simplify the analysis of AC circuits, especially in the context of machine modeling and control, by transforming time-varying three-phase quantities into a rotating reference frame where the quantities become time-invariant under steady-state conditions. As stated in [13], one of the difficulties inherent in describing the behavior of most rotating electric machinery is that the machine inductances are a function of both the electrical and the mechanical angles of the machine. To simplify this process, R.H. Park [14] developed a transformation that made the analysis of electric machines more straightforward by transforming the motor equations into a reference frame that is rotating synchronously with the fields in the machine.

Coordinate Systems Involved

- **abc (Stationary Frame)**: The natural three-phase coordinate system, representing voltages or currents in three phases (a, b, c).
- **dq0 (Rotating Frame)**: The transformed coordinate system, rotating with respect to the reference angle (typically rotor or stator flux angle), consisting of:
  - *d-axis (direct axis)*: Aligned with the rotating reference, often associated with flux.

- *q-axis (quadrature axis)*: Perpendicular to the d-axis, representing torque-producing components.
- *0-axis (zero-sequence component)*: Captures the zero-sequence element, typically zero in balanced systems.

Mathematical Basis of Park Transformation

The Park transformation consists of two steps: first, the three-phase variables are transformed into a two-phase stationary frame using the **Clarke transformation**, and then they are further converted into a rotating reference frame using the Park transformation itself.

Step 1: Clarke Transformation (abc to $\alpha\beta$ Frame)

The three-phase variables v_a, v_b, v_c are transformed into two components in a stationary, orthogonal reference frame v_α, v_β using the Clarke transformation:

$$\begin{pmatrix} v_\alpha \\ v_\beta \end{pmatrix} = \frac{2}{3} \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{pmatrix} \begin{pmatrix} v_a \\ v_b \\ v_c \end{pmatrix} \tag{3.1}$$

Where:

- v_a, v_b, v_c are the three-phase voltages or currents.
- v_α, v_β are the orthogonal components in the stationary frame.

Step 2: Park Transformation ($\alpha\beta$ to dq Frame)

The stationary frame v_α, v_β is then transformed into the rotating dq frame. This transformation is based on the electrical angle θ, which defines the reference frame's rotation, typically synchronized with the machine's rotor or stator flux:

$$\begin{pmatrix} v_d \\ v_q \end{pmatrix} = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} v_\alpha \\ v_\beta \end{pmatrix} \tag{3.2}$$

Where:

- v_d and v_q are the components in the rotating reference frame.
- θ is the angular position of the rotating reference frame, typically the rotor angle or the stator flux angle.

Step 3: Zero-Sequence Component (0-Axis)

In three-phase balanced systems, the zero-sequence component is zero. However, in the presence of unbalanced conditions or asymmetries, the zero-sequence component v_0 can be calculated as:

$$v_0 = \frac{1}{3}(v_a + v_b + v_c) \tag{3.3}$$

Thus, the full Park transformation that includes the zero-sequence component is:

$$\begin{pmatrix} v_d \\ v_q \\ v_0 \end{pmatrix} = \frac{2}{3} \begin{pmatrix} \cos(\theta) & \cos(\theta - \frac{2\pi}{3}) & \cos(\theta + \frac{2\pi}{3}) \\ \sin(\theta) & \sin(\theta - \frac{2\pi}{3}) & \sin(\theta + \frac{2\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} v_a \\ v_b \\ v_c \end{pmatrix} \tag{3.4}$$

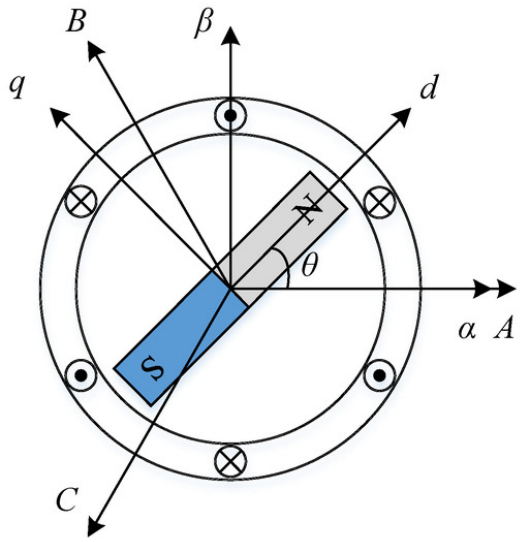


Figure 3.1: $ABC, dq, \alpha\beta$ reference frame

3.2 Space Vector Modulation

Overview of Space Vector Modulation

Space Vector Modulation (SVM) is a sophisticated technique used in the control of pulse-width modulated (PWM) converters and motor drives. It effectively generates the required output voltages to control three-phase systems by transforming the desired voltage vector into a series of switching states, allowing for improved performance compared to traditional PWM methods. Here's an overview of SVM, including its principles and formulas ([15]).

Basics of Space Vectors

In a three-phase system, the voltages can be represented as vectors in a two-dimensional plane ($\alpha - \beta$ plane) by using the concept of space vectors. Each of the three phases (A, B, C) can be represented in a complex plane, leading to the formation of a hexagon of possible voltage vectors.

The phase voltages can be represented as:

$$\begin{pmatrix} V_a \\ V_b \\ V_c \end{pmatrix} = V_{max} \cdot \begin{pmatrix} \cos(\theta) \\ \cos\left(\theta - \frac{2\pi}{3}\right) \\ \cos\left(\theta + \frac{2\pi}{3}\right) \end{pmatrix} \tag{3.5}$$

Here, V_{max} is the maximum phase voltage, and θ is the angle representing the position of the voltage vector.

Representation of Space Vectors
Space vectors in SVM can be categorized into:

- **Active Space Vectors**: These are the voltage vectors generated by switching combinations of the inverter (e.g., V_0 to V_7 in a two-level inverter).
- **Zero Space Vectors**: These represent the neutral points, typically V_0 and V_7, which do not contribute to active power.

The active space vectors correspond to the switching states of the inverter. For a three-phase inverter, the active vectors can be defined as follows:

$$V_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad V_1 = \begin{pmatrix} V_{max} \\ 0 \\ 0 \end{pmatrix}, \quad V_2 = \begin{pmatrix} V_{max}/2 \\ V_{max} \cdot \frac{\sqrt{3}}{2} \\ 0 \end{pmatrix}, \quad V_3 = \begin{pmatrix} -V_{max}/2 \\ V_{max} \cdot \frac{\sqrt{3}}{2} \\ -V_{max} \cdot \frac{\sqrt{3}}{2} \end{pmatrix}$$

$$V_4 = \begin{pmatrix} -V_{max} \\ 0 \\ 0 \end{pmatrix}, \quad V_5 = \begin{pmatrix} -V_{max}/2 \\ -V_{max} \cdot \frac{\sqrt{3}}{2} \\ 0 \end{pmatrix}, \quad V_6 = \begin{pmatrix} 0 \\ -V_{max} \\ 0 \end{pmatrix}, \quad V_7 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Modulation Process
For a three-phase system, the modulation time for each vector can be calculated using the formula:

$$t_a = \frac{T_s}{V_{max}} \cdot V_a \cdot \text{duty cycle} \tag{3.6}$$

$$t_b = \frac{T_s}{V_{max}} \cdot V_b \cdot (1 - \text{duty cycle}) \tag{3.7}$$

Where:

- T_s is the switching period.
- t_a and t_b are the time durations for the active vectors V_a and V_b.

Calculation of Duty Cycles
To find the duty cycles for the selected active vectors, the following equations are employed:

$$T_a = T_s \cdot D_a \tag{3.8}$$

$$T_b = T_s \cdot D_b \tag{3.9}$$

Where:

- D_a and D_b are the duty cycles of the active vectors V_a and V_b respectively.

The duty cycles are calculated based on the geometry of the vectors and the position of the desired voltage vector in the $\alpha - \beta$ plane.

Generation of PWM Signals
The PWM signals can be generated by comparing the desired voltage vector with a triangular carrier waveform. The output of the comparator determines when each switch in the inverter is turned on or off.

3.3 Sliding Mode Control

Sliding Mode Control (SMC) is a robust, nonlinear control strategy designed to drive and maintain a system on a predefined sliding surface, where system dynamics are insensitive to certain disturbances and parameter uncertainties. The control law consists of an equivalent control and a discontinuous control to ensure robustness and finite-time convergence ([16]).
Problem Formulation
Given the system described by this vector differential equation:

$$\dot{x} = f(x, u, t) \tag{3.10}$$

where $x, f \in \mathbb{R}^n, u \in \mathbb{R}^m, t \in \mathbb{R}^+$. It is also given the vector equation:

$$s(x) = 0 \tag{3.11}$$

with $s(x) : \mathbb{R}^n \to \mathbb{R}^m$, that is $s(x) = [s_1(x)...s_m(x)]^T$; it is assumed that the m scalar equations $s_i(x) = 0$ are linearly independent.
The problem consists of searching for a control law $u(x,t) = [u_1(x,t)...u_m(x,t)]^T$ of this type:

$$u_i(x,t) = \begin{cases} u_i^+(x,t) & \text{if } s_i(x) > 0 \\ u_i^-(x,t) & \text{if } s_i(x) < 0 \end{cases} \tag{3.12}$$

($i = 1, \ldots, m$, $u_i^+(x;t)$, $u_i^-(x;t)$ continuous functions, $u_j^+(x;t) \neq u_j^-(x;t)$ on $s_i(x) = 0$) so that, from a certain instant t_s onwards, the condition $s(x) = 0$ is verified, namely the plant's state is moving along the intersection of the m surfaces. The control law's synthesis occurs in two distinct phases: first, the equation $s(x) = 0$ 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 $u_i(x,t)$ with $i = 1, \ldots, m$, must be selected to satisfy the equation (3.11) from t_s onwards.

Sliding Mode conditions for existence and equations

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

$$\lim_{s\to 0^-} \dot{s} > 0, \quad \lim_{s\to 0^+} \dot{s} < 0 \quad \Leftrightarrow \quad \lim_{s\to 0}(s\dot{s}) < 0 \tag{3.13}$$

They ensure the sliding mode occurs 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 (3.13) is the definite positive function $V(s(x)) = \frac{1}{2}s^2(x)$ and its definite negative derivative. In general, when $u \in \mathbb{R}^m$, the following definite positive Lyapunov function is used:

$$V(s(x)) = \frac{1}{2}s^T(x)s(x) \tag{3.14}$$

with its definite negative derivative

$$\dot{V}(s(x)) = s^T(x)\dot{s}(x) < 0 \tag{3.15}$$

and a control function in the same form as the equation (3.12). The control function must fulfill the inequality (3.15) for each x . To define the sliding mode equation, Filippov's theory of differential equations with discontinuous right-hand sides is used:

$$\dot{x} = f(x,t) + B(x,t)u(x,t) \tag{3.16}$$

but this form is only proper for systems with linear control. Then the equivalent control method is embraced:

$$\frac{\partial s}{\partial x}\dot{x} = 0 \tag{3.17}$$

where $\frac{\partial s}{\partial x}$ is the Jacobian of $s(x)$. Replacing equation (3.16) into (3.17):

$$\frac{\partial s}{\partial x}f(x,t) + \frac{\partial s}{\partial x}B(x,t)u(x,t) = 0 \tag{3.18}$$

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

$$u_{eq} = -\left[\frac{\partial s}{\partial x}B(x,t)\right]^{-1}\frac{\partial s}{\partial x}f(x,t) \tag{3.19}$$

The u_{eq} function is the equivalent continuous control and replacing it in (3.16), it returns the ideal sliding mode equation:

$$\dot{x} = \left\{I - B(x,t)\left[\frac{\partial s}{\partial x}B(x,t)\right]^{-1}\frac{\partial s}{\partial x}\right\}f(x,t) \tag{3.20}$$

where I is the identity matrix of order n.

If $\frac{\partial s}{\partial x}$ B(x,t) is singular:

- The equivalent control is not unique, but the sliding mode equation continues to be 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:

$$s^*(x) = H_s(x,t)s(x) \tag{3.21}$$

where $H_s(x,t)$ is a $m \times m$ matrix with $\det[H_s(x,t)] \neq 0$, the sliding mode equation (3.20) is still valid if the m control vector components are discontinuous on the surfaces $s_i^*(x) = 0$, with $i = 1, \ldots, m$. Also defining

$$u^*(x) = H_u(x,t)u(x) \tag{3.22}$$

where $H_u(x,t)$ is a $m \times m$ matrix with $\det[H_u(x,t)] \neq 0$, the sliding equation does not change if the m components of the new control vector $u_i^*(x)$ are discontinuous on the surfaces $s_i^*(x) = 0$. These properties simplify control function synthesis when conditions like (3.6) are set. In fact, the last two equations (3.21 and 3.22) allow choosing control vector m components independently. Thus, the problem can be solved as monodimensional problems.

Issue and Characteristics

The sliding motion represents an ideal condition; however, in practice, trajectory oscillations occur in the vicinity of $s(x) = 0$ due to the non-ideal switching mechanism, which is not physically feasible due to the infinite switching frequency it would require. The delay introduced by the switching frequency also results in the state not aligning promptly with the sliding surface, leading to chattering: the generation of high-frequency oscillations caused by switching. The switching mechanism can be characterized by a hysteresis cycle of amplitude Δ, where $s(x) = 0$ is considered scalar. Due to the behavior of $u(x,t)$, trajectories starting from a point where $s(x) < 0$ do not remain at the points where $s(x) = 0$; instead, they proceed to $s(x) = +\Delta$ and then return to $s(x) = -\Delta$, and so forth, generating oscillations. It is evident that smaller Δ values result in higher frequency and lower chattering amplitude. When $\Delta = 0$ (ideal switching), the frequency would be infinite and the chattering amplitude would be zero; this implies that the state would perfectly "slide" on the surface $s(x) = 0$.

Requiring the state to be close to the sliding surface, rather than achieving perfect sliding motion, can help reduce chattering phenomena. Considering the scalar case:

$$|s(x)| \leq \varepsilon \tag{3.23}$$

An area called the "boundary layer" is defined, with $\varepsilon > 0$. Outside the boundary layer, the control function is defined as:

$$u(x,t) = \begin{cases} u^+(x;t) & \text{if } s(x) > \varepsilon \\ u^-(x;t) & \text{if } s(x) < -\varepsilon \end{cases} \tag{3.24}$$

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

$$u(x,t) = \begin{cases} k \cdot \text{sign}[s(x)] & \text{if } |s(x)| > \varepsilon \\ k \cdot \frac{s(x)}{\varepsilon} & \text{if } |s(x)| \leq \varepsilon \end{cases} \tag{3.25}$$

By employing this method, the control law is influenced by the value ε, which must be chosen to optimize the balance between minimizing chattering and meeting control requirements.

Choosing a sliding surface

The sliding surface is time-dependent. The selection of the sliding surface geometrically defines the control requirements. In the context of a tracking problem, we assume that the system output is $y = h(x)$, where $y, h \in \mathbb{R}^m$. Let the reference output be denoted as y_d and the tracking error as $e = y - y_d$. The sliding surface is defined as follows:

$$s(x,t) = \dot{e} + \Lambda e = 0 \quad \text{with } \Lambda = \text{diag}(\lambda_i), \lambda_i > 0,\ i = 1, \ldots, m \tag{3.26}$$

When the above equation is satisfied, each component of the tracking error e_i exponentially approaches zero with a time constant of $\frac{1}{\lambda_i}$.

Scalar case

Considering the following bidimensional system:

$$\begin{cases} \dot{x}_1(t) = x_2 \\ \dot{x}_2(t) = f(x) + g(x)u \end{cases} \tag{3.27}$$

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

$$f(x) = \widehat{f}(x) + \Delta f(x) \quad \text{(nominal value + uncertainty)} \tag{3.28}$$

$$g(x) = \widehat{g}(x) + \Delta g(x) \quad \text{(nominal value + uncertainty)} \tag{3.29}$$

The objective is to design a control law that can stabilize the state and guide it to the straight line $s = x_2(t) + ax_1(t) = 0 \; \forall \; t \geq t_s$; this line is the chosen sliding surface, and t_s is the instant at which the state reaches it. To ensure that $s = 0$, we utilize a Lyapunov function $V(s)$ that always decreases over time ($\dot{V}(s) < 0$):

$$V(s) = \frac{1}{2}s^2 \tag{3.30}$$

Thus, we have:

$$\dot{V}(s) = \frac{1}{2}s \cdot \dot{s} \cdot 2 = s \cdot \dot{s} \tag{3.31}$$

where $\dot{s}$ is the time derivative of $s = x_2 + ax_1$. Consequently, we have:

$$\dot{V}(s) = s \cdot \dot{s} = s \cdot \frac{d}{dt}[x_2 + ax_1] = s[\dot{x}_2 + a\dot{x}_1] = s[f(x) + g(x)u + ax_2] \tag{3.32}$$

$$= sg(x)\left[u + f(x) + ax_2\right] \tag{3.33}$$

Since $f(x)$ and $g(x)$ are uncertain quantities, we assume that:

$$\left|\frac{f(x) + ax_2}{g(x)}\right| \leq \rho \tag{3.34}$$

where ρ is a constant (but could be a known function $\rho(x)$). Therefore, from equation (3.32), we obtain the following expression:

$$\dot{V}(s) = sg(x)\left[u + \frac{f(x) + ax_2}{g(x)}\right] \leq sg(x)u + |s|g(x)\rho \tag{3.35}$$

We choose this control law:

$$u = -\beta \text{sign}(s) \quad \text{with } \beta = \rho + \beta_0 \text{ where } \beta_0 > 0,\, \rho > 0 \tag{3.36}$$

Substituting into equation (3.35), we get:

$$\dot{V}(s) \leq sg(x)\left\{-[\rho(x) + \beta_0]\text{sign}(s)\right\} + |s|g(x)\rho = -|s|g(x)\beta_0 \tag{3.37}$$

Consequently, we have:

$$s \cdot \dot{s} \leq -|s|g(x)\beta_0 \quad \text{with } 0 < g_0 \leq g(x) \tag{3.38}$$

$$s \cdot \dot{s} \leq -g_0\beta_0|s| = -\alpha_0|s| \tag{3.39}$$

This guarantees that $|s|$ always decreases, thus leading to $s = 0$. Equivalent Control

Law
It is assumed that $f(x)$ is subject to changes $f(x) = \widehat{f}(x) + \Delta f(x)$, while the function $g(x)$ is known, $g(x) = \widehat{g}(x)$. The equation (3.32) takes the following form:

$$\dot{V}(s) = sg(x) \left[u + \frac{\widehat{f}(x) + \Delta f(x) + ax_2}{g(x)} \right] \tag{3.40}$$

The input has two components:

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

Replacing u in equation (3.40):

$$\dot{V}(s) = sg(x) \left[-\frac{\widehat{f}(x) + ax_2}{g(x)} + v + \frac{\widehat{f}(x) + \Delta f(x) + ax_2}{g(x)} \right] \tag{3.41}$$

It is assumed that:

$$\left| \frac{\Delta f(x)}{g(x)} \right| \leq \rho\Delta \tag{3.42}$$

Thus, we have:

$$\dot{V}(s) = sg(x) \left[v + \frac{\Delta f(x)}{g(x)} \right] \tag{3.43}$$

Given that typically $|\Delta f(x)| < |f(x)|$, it is generally true that $\rho\Delta(x) < \rho(x)$. This sufficiently reduces the amplitude of oscillations. The discontinuous component is implemented as follows:

$$v = -\rho_\Delta \cdot sat(s) \tag{3.44}$$

Chapter 4

Observer design

In this thesis project, a Super-Twisting Sliding Mode Observer (STSMO) was utilized, combined with a Phase-Locked Loop (PLL), for the estimation of the PMSM's rotor speed and position. The super-twisting algorithm is an advanced variant of classical sliding mode control, addressing key limitations of first-order sliding mode observers [17]. In classical sliding mode observers, high-frequency switching (*chattering*) occurs due to discontinuous control signals, which can lead to system instability and noise amplification. The super-twisting algorithm eliminates these issues by producing continuous control inputs while retaining the inherent robustness of sliding mode control [18]. In the context of this thesis, the super-twisting sliding mode observer (STSMO) was used to estimate the back-EMF of the PMSM, which is crucial for determining the rotor position. The estimated back-EMF was then fed into the PLL to extract the rotor angle. The super-twisting algorithm significantly impacted the performance of the observer, which exhibits reduced chattering and excellent robustness to disturbances and uncertainties. This, paired with the PLL's precision in rotor angle estimation, enabled the system to achieve reliable control. The validity of the results obtained will be shown in Chapters 5-6.

4.1 Super-twisting algorithm

The PMSM model in the stationary reference frame $(\alpha - \beta)$ is shown as follows:

$$\begin{cases} u_\alpha = Ri_\alpha + L\dot{i}_\alpha + \omega_e(L_d - L_q)i_\beta + e_\alpha \\ u_\beta = Ri_\beta + L\dot{i}_\beta + \omega_e(L_d - L_q)i_\alpha + e_\beta \end{cases} \tag{4.1}$$

Where:

- u_α, u_β : stator voltages
- i_α, i_β : stator currents
- R : stator resistance
- ω_e : electrical rotor speed

- ψ_f : PM flux linkage
- L_d, L_q : stator inductance
- e_α, e_β : $\alpha\beta$-axis back-EMF with $e_\alpha = -E\sin\theta, e_\beta = E\cos\theta$

θ is the rotor position and E is the amplitude of the back-EMF satisfying:

$$E = (L_d - L_q)\left(\omega_e i_d - \dot{i}_q\right) + \omega_e \psi_f \tag{4.2}$$

As reported in [10], the formula underlying the algorithm can be written in the following form:

$$\begin{cases} \dot{\widehat{x}}_1 = -k_1 \left|\widehat{x}_1 - x_1\right| \mathrm{sign}(\widehat{x}_1 - x_1) + \widehat{x}_2 + \rho_1 \\ \dot{\widehat{x}}_2 = -k_2 \mathrm{sign}(\widehat{x}_1 - x_1) + \rho_2 \end{cases} \tag{4.3}$$

Where:

- x_t : state variables
- $\widehat{x}_t$: estimation of the state variables
- k_t : sliding-mode gains
- $sign(t)$: signum function
- ρ_t : perturbation terms

In [19] the conditions of stability of the super-twisting algorithm have been educed. If ρ_1 and ρ_2 satisfy the following conditions:

$$\rho_1 \leq \delta_1 |x_1|^{\frac{1}{2}} \quad \rho_2 = 0 \tag{4.4}$$

where δ_1 is a positive constant and the sliding-mode gains k_1, k_2 meet the condition:

$$k_1 > 2\delta_1, \quad k_2 > \frac{k_1 5\delta_1}{k_1 + 4\delta_1^2\left(\frac{1}{2}(k_1 - 2\delta_1)\right)} \tag{4.5}$$

then the stability of the system can be guaranteed.

4.2 Back-EMF estimation

Let consider the PMSM model in the stationary reference frame $(\alpha - \beta)$ reorganised as follows:

$$\begin{cases} L_s \dot{i}_\alpha = -R_s i_\alpha - e_\alpha + u_\alpha \\ L_s \dot{i}_\beta = -R_s i_\beta - e_\beta + u_\beta \\ e_\alpha = -\psi\omega_r sin(\theta) \\ e_\beta = \psi\omega_r cos(\theta) \end{cases} \tag{4.6}$$

Where:

- i_α, i_β are the phase currents
- u_α, u_β are the phase voltages
- e_α, e_β are the back EMF

Considering $\widehat{i}_\alpha, \widehat{i}_\beta$ as the currents estimated by the STSMO we have:

$$\begin{cases} L_s\dot{\widehat{i}}_\alpha = -R_s\widehat{i}_\alpha - e_\alpha + u_\alpha \\ L_s\dot{\widehat{i}}_\beta = -R_s\widehat{i}_\beta - e_\beta + u_\beta \\ e_\alpha = -\psi\omega_r sin(\theta) \\ e_\beta = \psi\omega_r cos(\theta) \end{cases} \tag{4.7}$$

Let consider

- $\tilde{i}_\alpha = \widehat{i}_\alpha - i_\alpha$ and $\tilde{i}_\beta = \widehat{i}_\beta - i_\beta$: the current observation errors
- $sgn()$: the sign function
- $k_1 > 0, k_2 > 0$: the two gains of the STSMO

According to the super-twisting sliding-mode theory, when $\tilde{i}_\alpha$ and $\tilde{i}_\beta$ reach sliding-mode surface, the estimated back-EMF $\widehat{e}_\alpha, \widehat{e}_\beta$ have the form:

$$\begin{bmatrix} \widehat{e}_\alpha \\ \widehat{e}_\beta \end{bmatrix} = \begin{bmatrix} k_1 \left|\tilde{i}_\alpha\right| sgn(\tilde{i}_\alpha) + \int_0^t k_2 sgn(\tilde{i}_\alpha)dt \\ k_1 \left|\tilde{i}_\beta\right| sgn(\tilde{i}_\beta) + \int_0^t k_2 sgn(\tilde{i}_\beta)dt \end{bmatrix} \tag{4.8}$$

Once the back-EMF estimates are obtained a phase-locked loop (PLL) is adopted to track the rotor position.
It is important to highlight that, in this project, the *saturation* function was used instead of the *sign* function. Indeed, by introducing a smoother transition near the sliding surface, the saturation function mitigates high-frequency oscillations.

4.3 Phase-locked loop (PLL)

The PLL is a second-order system that contains a phase detector (PD), a loop filter (LF) and a voltage-controlled oscillator (VCO) (Fig.4.1)[20]. The back-EMF error $\Delta E'$ is defined as:

$$\Delta E' = -\widehat{e}_\alpha cos(\widehat{\theta}_e) - \widehat{e}_\beta sin(\widehat{\theta}_e) = -k_L sin(\Delta\theta_e) \tag{4.9}$$

Where $k_L = \psi_f \omega_e$ and $\Delta\theta_e = \widehat{\theta}_e - \theta_e$. When the PLL has tracked the rotor position, $sin(\Delta\theta_e)$ is so small that is approximately equal to θ_e. From this, (5.4) can be rewritten as

$$\Delta E' \approx -k_L \Delta\theta_e \tag{4.10}$$

Therefore, the open-loop transfer function of the PLL $G_0(s)$ is:

$$G_0(s) = \frac{\widehat{\theta}_e}{\theta_e} = \frac{k_p s + k_i}{s^2} \tag{4.11}$$

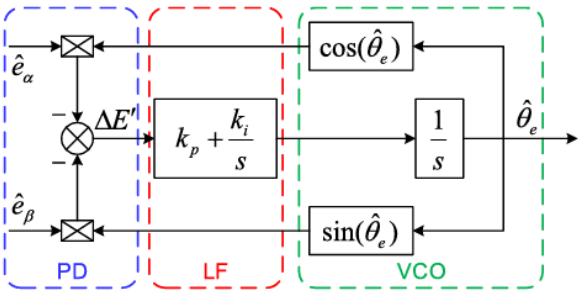


Figure 4.1: Structure of PLL in $\alpha - \beta$ reference frame

4.4 Simulink Model

In this section the structure of the code block designed to implement the observer will be presented. The entire subsystem can be seen as divided into the following sub-components:

- Bus-selector and input variables assignment
- Saturation calculation
- Estimates calculation
- Phase-locked loop (PLL)
- Direct Park
- Bus Updating

In table 4.1 all the variables involved in the process.

Mathematical Symbol	**Code Variable Name**	**Description**
i_α	`i_alpha`	Current signal with respect α axis
i_β	`i_beta`	Current signal with respect β axis
$\widehat{i}_\alpha$	`i_alpha_est`	Estimated current with respect α axis
$\widehat{i}_\beta$	`i_beta_est`	Estimated current with respect β axis
$\tilde{i}_\alpha$	`ia_error`	Estimation error of i_α
$\tilde{i}_\beta$	`ib_error`	Estimation error of i_β
$sat(\tilde{i}_\alpha)$	`sat_ia`	Saturation function of $\tilde{i}_\alpha$
$sat(\tilde{i}_\beta)$	`sat_ib`	Saturation function of $\tilde{i}_\beta$
$\int_0^t sat(\tilde{i}_\alpha)dt$	`int_sat_ia`	Integral of the saturation function of $\tilde{i}_\alpha$
$\int_0^t sat(\tilde{i}_\beta)dt$	`int_sat_ib`	Integral of the saturation function of $\tilde{i}_\beta$
$\widehat{e}_\alpha$	`e_alpha`	Estimated back-EMF e_α
$\widehat{e}_\beta$	`e_beta`	Estimated back-EMF e_β
$\widehat{\theta}$	`theta_est`	Estimated Rotor Position
$sin(\widehat{\theta})$	`sin_theta`	Sine function of $\widehat{\theta}$
$cos(\widehat{\theta})$	`cos_theta`	Cosine function of $\widehat{\theta}$
$\widehat{\omega}$	`omega_est`	Estimated Rotor Speed
i_d	`i_d`	Current with respect d-axis
i_q	`i_q`	Current with respect q-axis

Table 4.1: Sliding Observer Variables

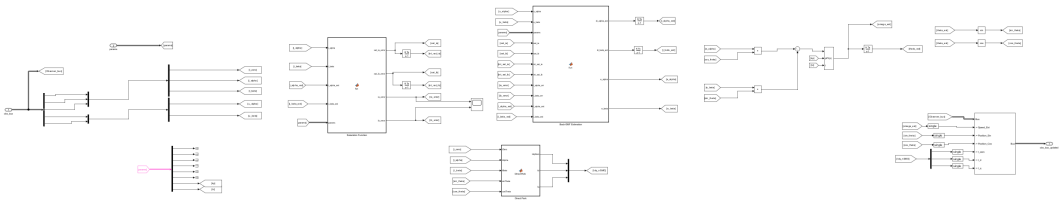


Figure 4.2: Sliding Observer Simulink Model

Each sub-component of the model is dedicated to a stage in the process, which consists of the following steps.

Bus Selector and Input Variables Assignment The values of the variables required for the process are extracted from the "*obs_bus*" bus via a bus selector and assigned to the respective variables.

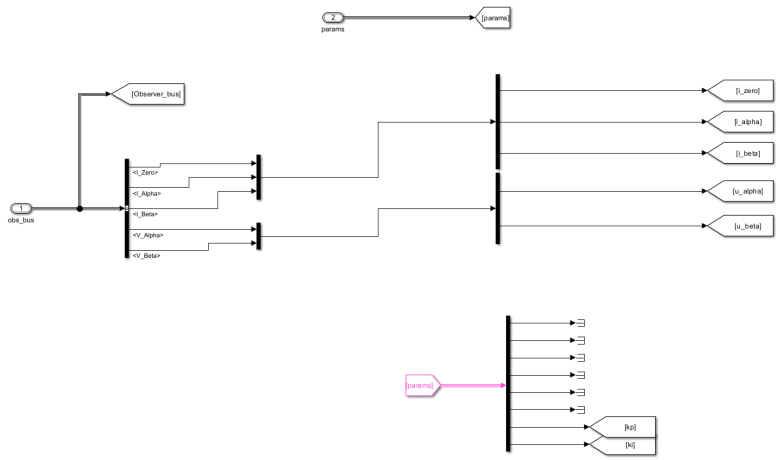


Figure 4.3: Observer input bus

Saturation Calculation In the *MATLAB function* shown in Fig. **??**, the saturation function of the current estimation error is calculated. The measured value of the current with respect to the $\alpha\beta$ axes and the corresponding estimated value are taken as inputs, and the error is then calculated. The function outputs the errors of the estimated currents (`ia_error,ib_error`) and calculated saturation function at those values (`sat_ia_error,sat_ib_error`). Outside the function, the integral of the saturation function is also calculated (`int_sat_ia,int_sat_ib`).

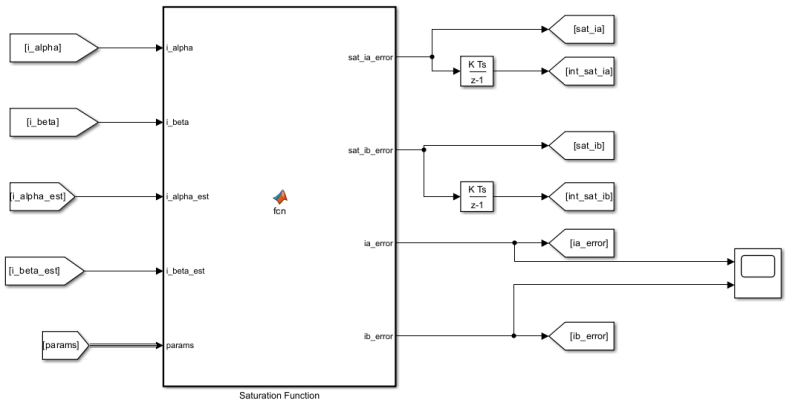


Figure 4.4: Saturation Matlab Function

Estimates Calculation The *MATLAB function* dedicated to this step deals with the estimation of the back-EMF. Receives as input the values of the measured voltages and currents, the values of the saturation function calculated at the previous step, the estimated current errors and the current estimates at the previous sampling time. Subsequently returns as output the values of the estimated currents with respect to $\alpha\beta$ reference frame (`i_alpha_est,i_beta_est`) and the values of the estimated back-EMFs with respect to $\alpha\beta$ reference frame (`e_alpha,e_beta`).

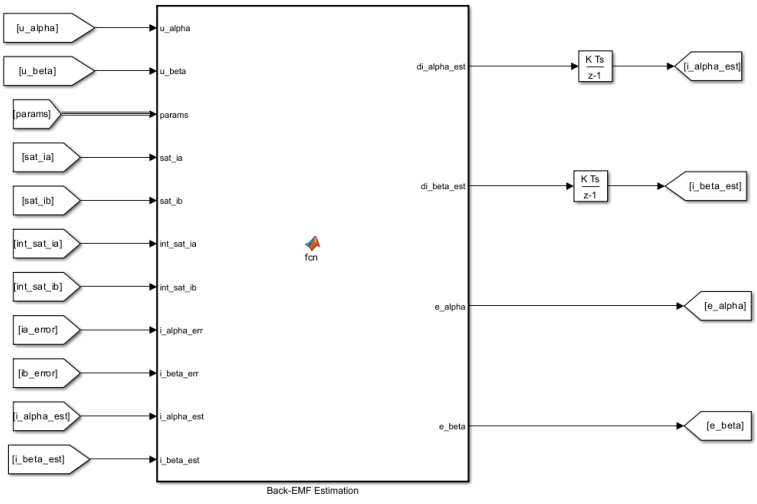


Figure 4.5: Back-EMF Estimation Matlab Function

Phase-locked Loop (PLL) Once the back-EMF estimates have been obtained, their values are transmitted to the PLL. The latter, via a PI controller, is to calculate the estimated rotor speed and position corresponding to that value of the back-EMFs.

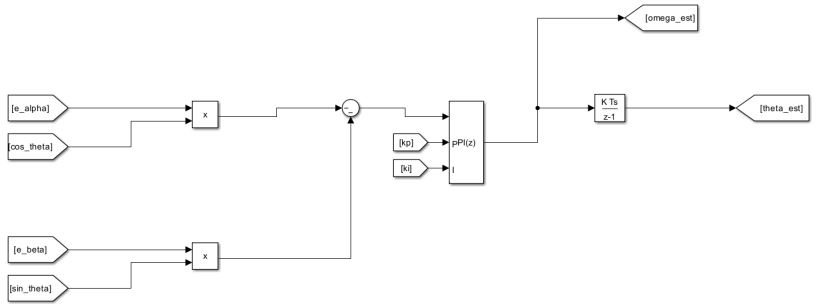


Figure 4.6: Phase-locked Loop (PLL) simulink scheme

Direct Park The values of the estimated currents and the sine and cosine of the estimated angular position are transmitted to the *MATLAB function* dedicated to the Direct Park Transformation. The function returns as output the values of the currents with respect to the rotating axes *dq*.

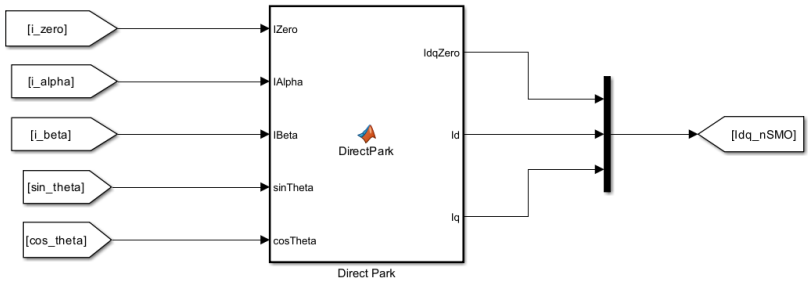


Figure 4.7: Direct Park Transformation Matlab Function

Bus Updating Once the values of all necessary estimates have been obtained, as a final step, the bus dedicated to the observer process is updated.

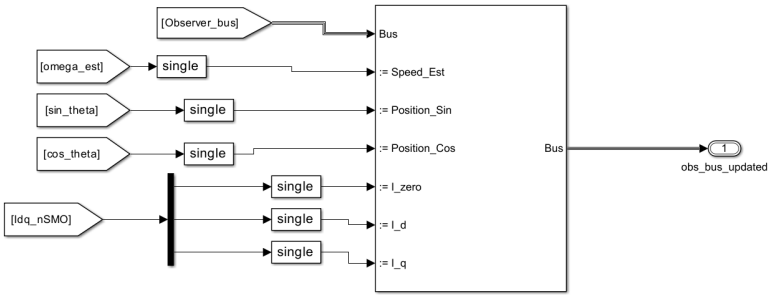


Figure 4.8: Observer Bus updating

Chapter 5

Discussion and Comparison of simulation results

This chapter is dedicated to the description of the results obtained in the simulation environment. Firstly, the open-loop performance of the designed observer was analysed (*sensored* mode). Subsequently, once the *sensorless* mode was activated and the observer connected to the control loop, the trend of the controlled signals was analysed. Finally, the trends of the variables controlled via the sliding mode approach were compared with those controlled via PI controllers.

5.1 STSMO estimations (sensored mode)

As mentioned above, in the first instance, the performance of the observer was analysed in the open-loop, i.e. without its influence in the control loop. In the open-loop configuration, the primary goal is to verify the observer's ability to accurately estimate the motor's speed and position based on the motor's electrical signals, without actively influencing the control inputs. The simulations that provided excellent results in this scenario indicate that the STSMO is able to reliably track the actual speed and position of the PMSM. The control mode selected for this type of simulation was PI control in *sensored* mode (*SensMode* = 1, *ObsMode* = 3, *CtrlMode* = 1). A good tuning of the observer's parameters was certainly crucial. Tuning concerned in particular the parameters `k1_m, k2_m, epsilona_m, epsilonb_m, kp_m, ki_m`. After several attempts, the resulting tuning is as follows:

- `k1` = 0.005
- `k2` = 6000
- `epsilona` = 0.7
- `epsilonb` = 0.7
- `kp` = 50
- `ki` = 10000

In the following figures, the high accuracy of the speed, position and current estimates can easily be seen. For the sake of clarity, it should be specified that the following color associations have been chosen in the graphical results that follow.

- **Reference** signals: blue
- **Sliding Mode** control signals/observed signals: red
- **PI** control signals: green

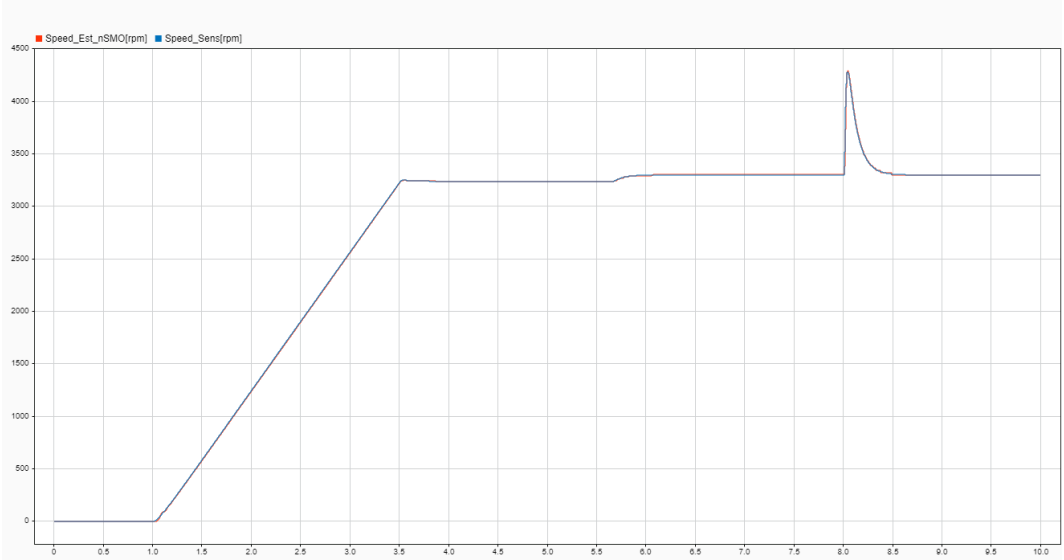


Figure 5.1: Estimated Speed [rpm]

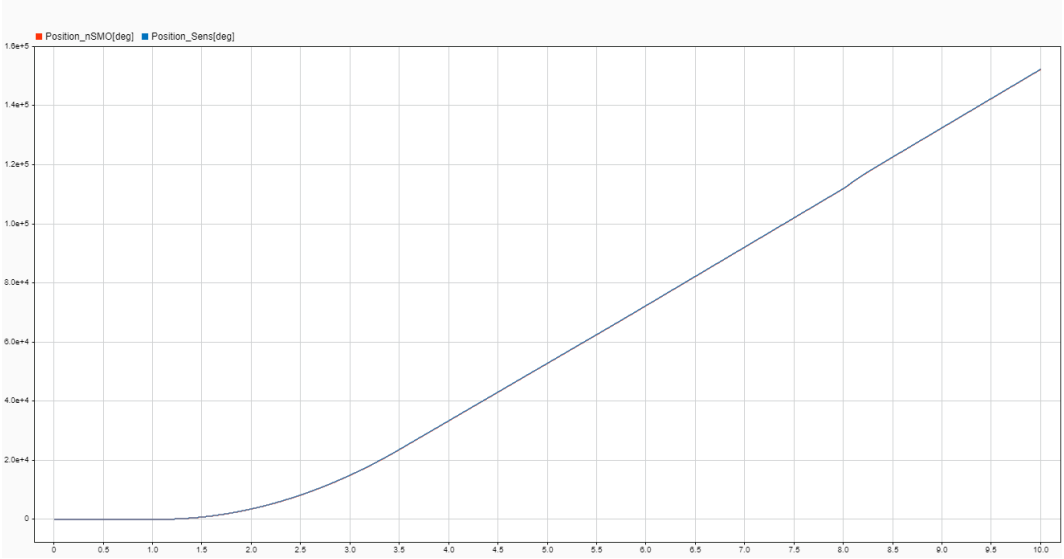


Figure 5.2: Estimated Position [deg]

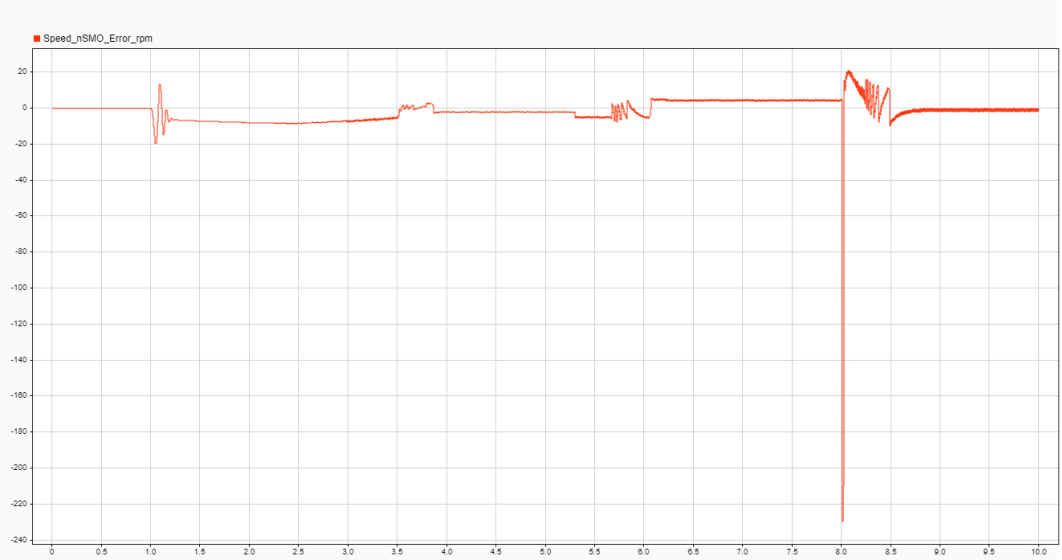


Figure 5.3: Speed Estimation Error

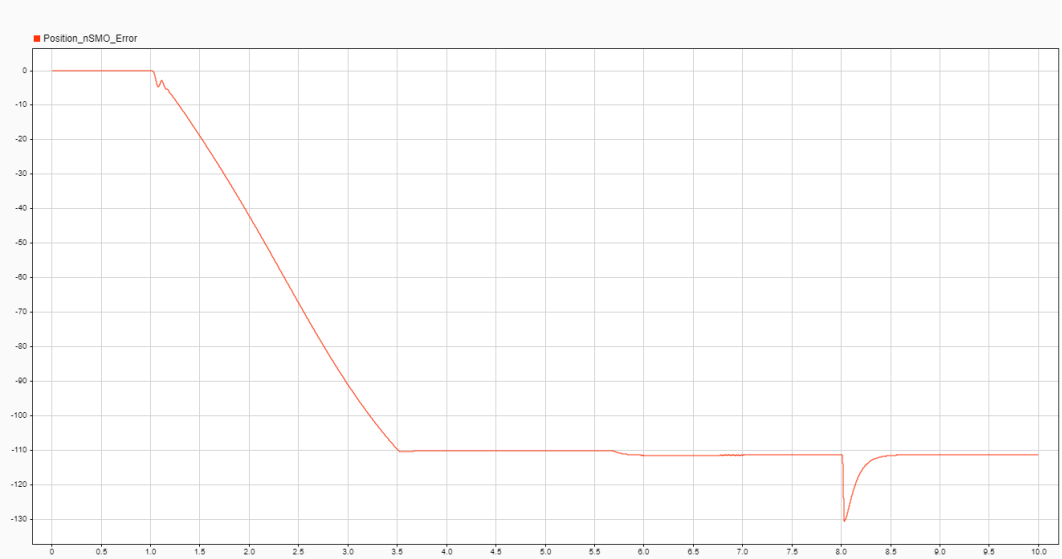


Figure 5.4: Position Estimation Error

It is worth specifying that the peak observed at the 8th second in the sensors signal (Fig.5.1) is due to the end of the load torque action. As shown in Fig.5.3, the speed estimate shows oscillations near zero, which are, however, sufficiently contained to consider the calculated estimate as good. Regarding the position as well (Fig.5.4), it can be observed that the estimation error increases proportionally with the speed but still remains limited. Therefore, based on what is observed in the graphical results, the observer has an excellent estimation capability. Hence, it is decided to proceed with the next simulations.

5.2 Sliding Mode Control + STSMO (sensorless mode)

In subsequent simulations, the STSMO was connected to the control loop, working in conjunction with the sliding mode controller (SMC) for speed and current regulation of the PMSM. This closed-loop configuration allows for a comprehensive evaluation of the system's overall performance, with the observer providing real-time estimates of motor speed and position, while the controller adjusts the inputs based on those estimates. To perform the simulation in this mode, the values to be used for the switch verification are as follows: $SensMode = 0$, $ObsMode = 3$, $CtrlMode = 0$.
During the initial integration of the observer into the closed-loop system, it was found that the previous tuning from the open-loop tests was insufficient for maintaining accurate speed regulation under all conditions. In the closed-loop configuration, the feedback from the observer directly influenced the controller's decisions, and any minor estimation errors from the observer could lead to degraded speed performance, especially during dynamic transitions. To address this, the controller gains were adjusted to improve the system's responsiveness to the observer's estimates. In addition, even observer gains had to be carefully retuned to enhance the convergence speed and stability of the estimates. Therefore, what results is a re-tuning of both control and observer parameters:

Speed Regulator

- `a1` = 18
- `ro` = 0.02
- `epsilon` = 80

Current Regulator

- `a2` = 1000
- `roq` = 7
- `epsilonq` = 10
- `a3` = 500
- `rod` = 15
- `epsilond` = 2

Observer

- `k1` = 450
- `k2` = 450
- `epsilona` = 0.05
- `epsilonb` = 0.05
- `kp` = 4
- `ki` = 80

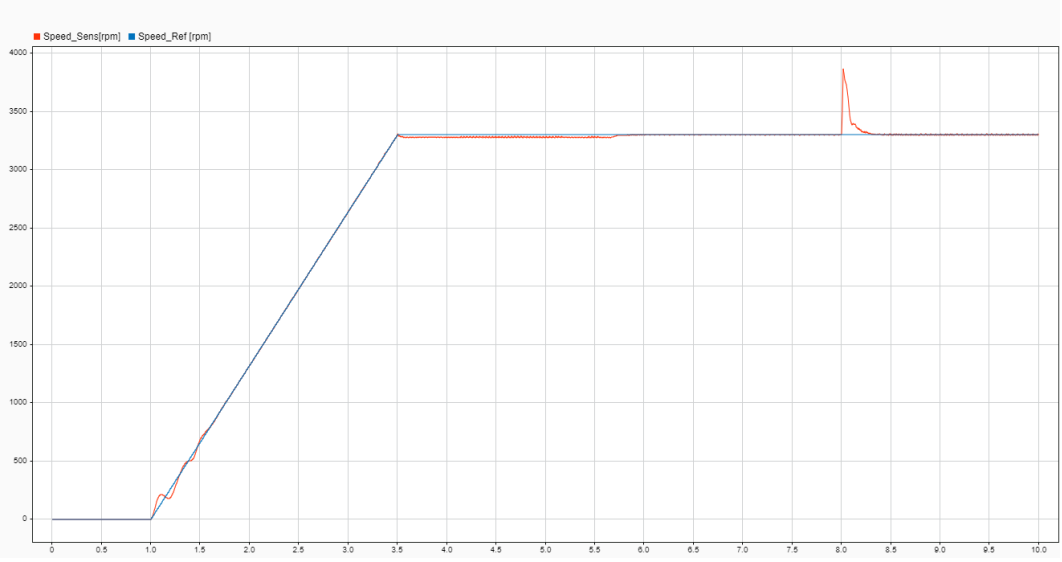


Figure 5.5: SMC + STSMO Speed

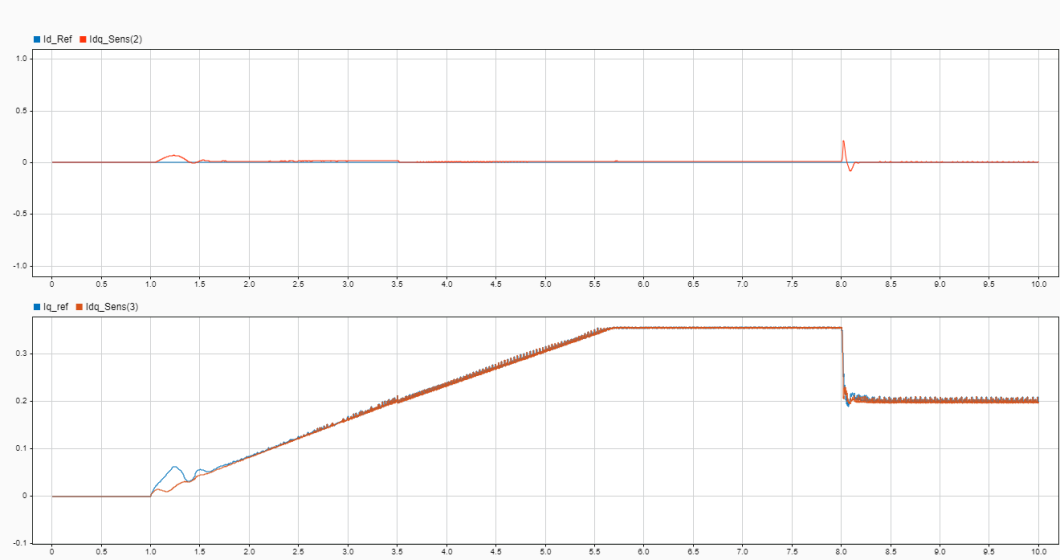


Figure 5.6: SMC + STSMO Current

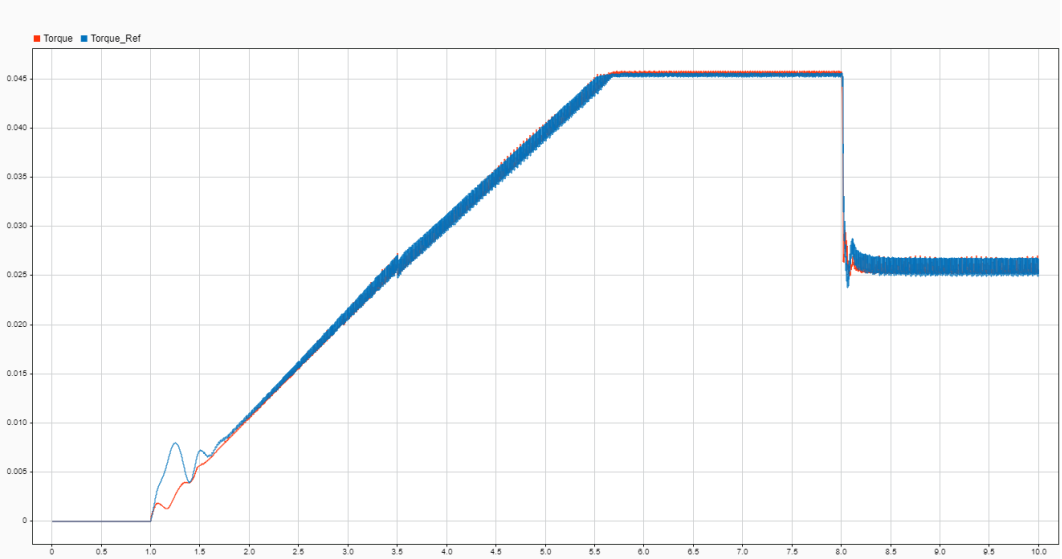


Figure 5.7: SMC + STSMO Torque

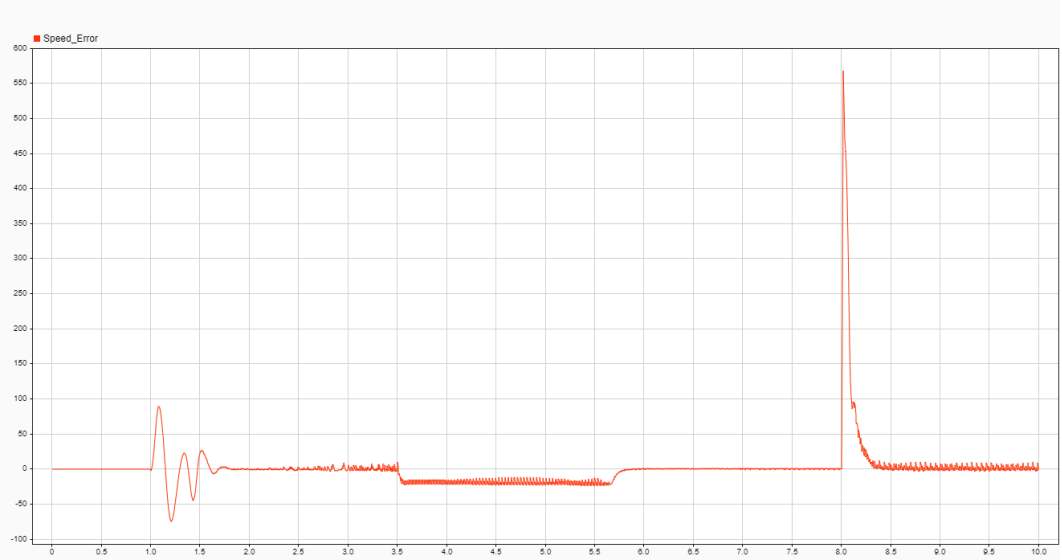


Figure 5.8: SMC + STSMO Speed Error

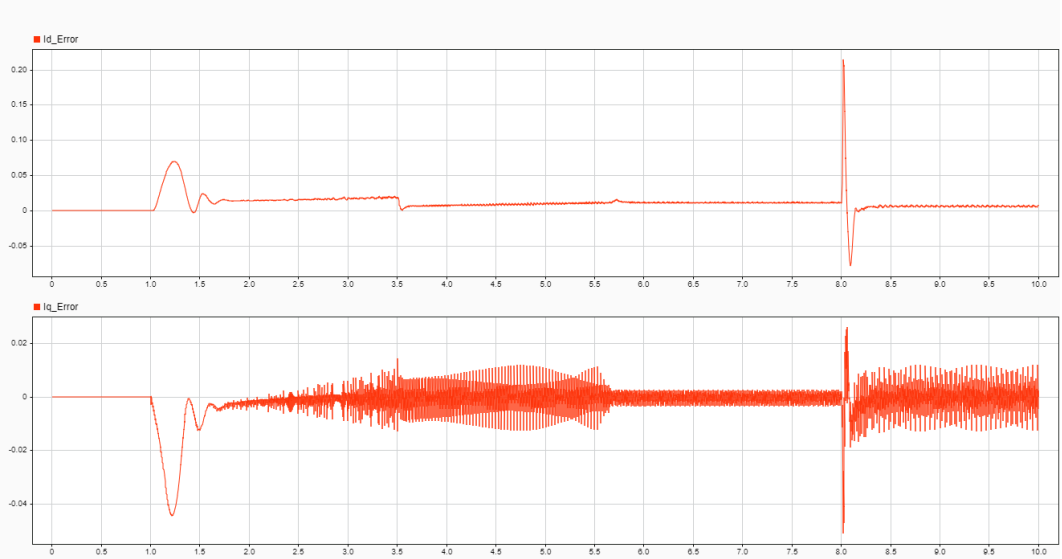


Figure 5.9: SMC + STSMO Current Error

In the figures above, we notice the excellent control achieved by the combination of control and sliding mode observer. Although the controlled signals show slight oscillations, which, as previously mentioned, are inherent to the nature of this approach, the simulations should be considered successful.

5.3 Comparison: Sliding Mode/PI

Thereafter, in order to prove the validity of the proposed approach, the results of the Sliding Mode control together with Sliding Mode observer were compared with those of the PI control and Whirlpool Luenberger observer.
Analysing the graphical results it's clear that one of the most significant trade-offs involves the inherent nature of each control strategy in terms of response speed and oscillations. In the simulations, the Sliding Mode controller consistently showed better performance in tracking the reference signals. However, this fast response comes with a downside: more oscillations or chattering in the control signals. The only measure that can be considered to reduce the oscillations is to find the optimal parameter tuning to minimize them, while being careful not to compromise the robustness of the system. In contrast, the PI controller, provides smoother control signals with fewer oscillations. While effective under steady-state conditions, the PI controller exhibited slower dynamic response, especially during rapid speed changes.

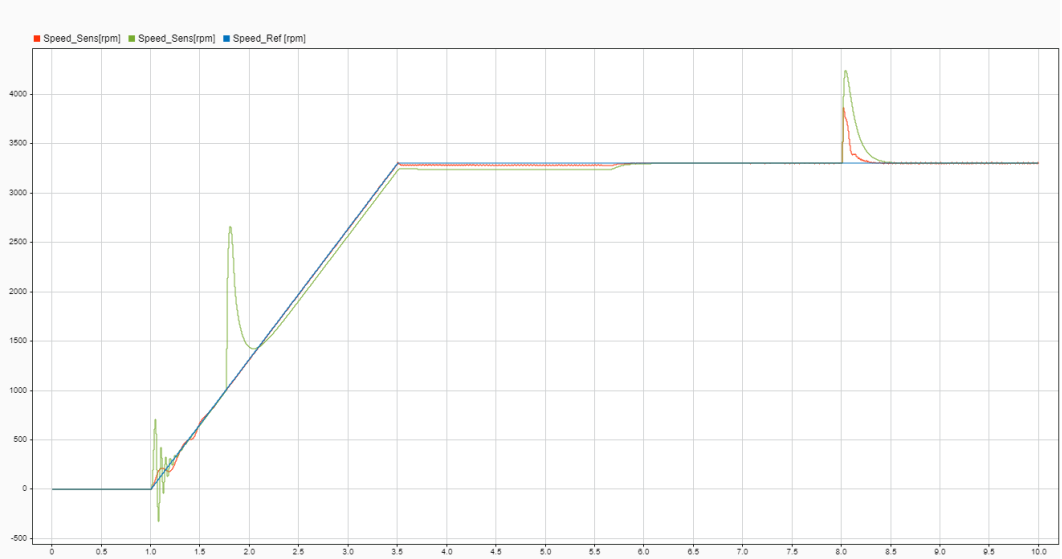
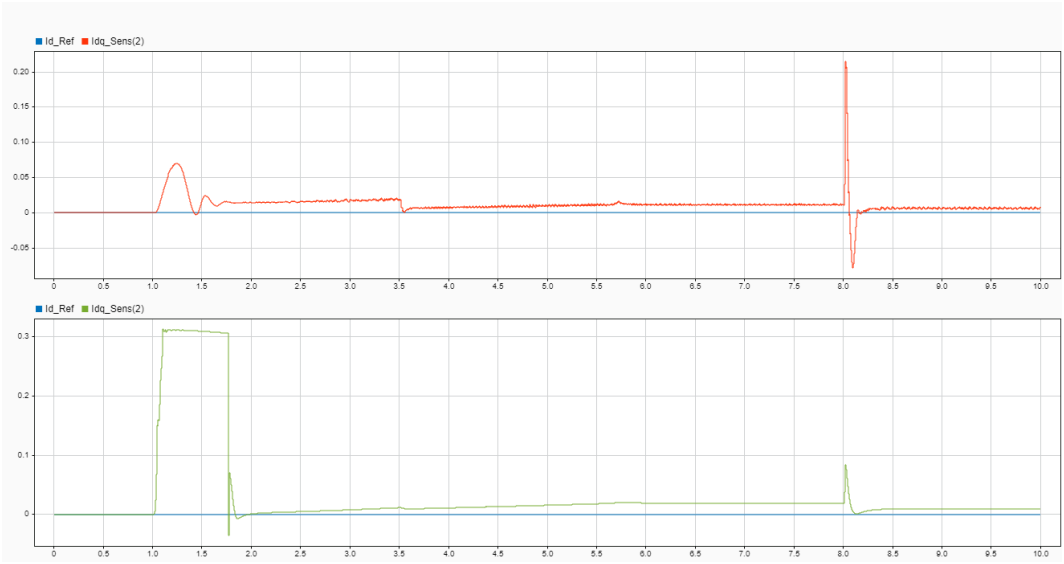
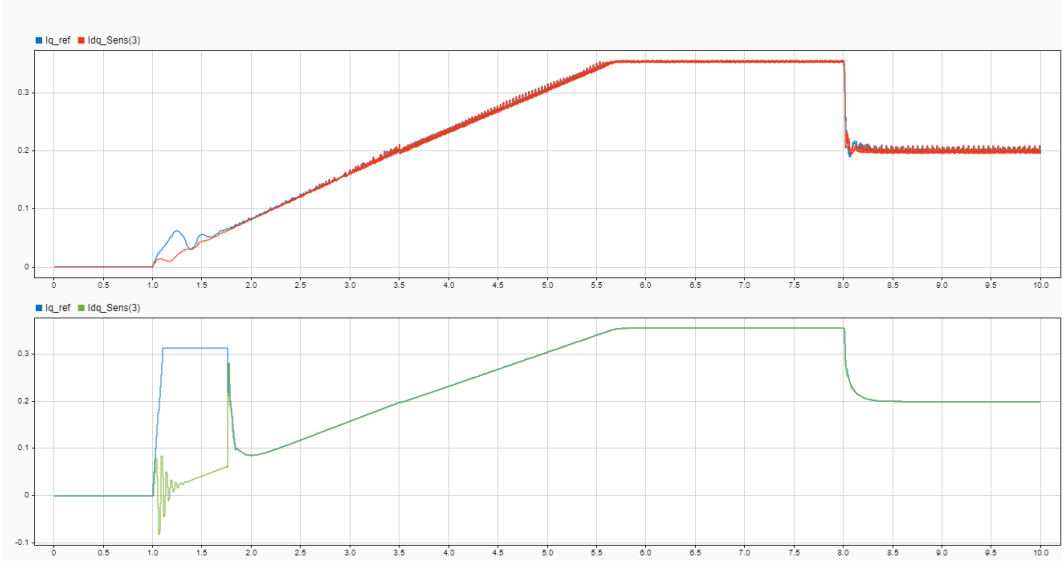


Figure 5.10: PI/SMC Speed



(a) PI/SMC Current I_d



(b) PI/SMC Current I_q

Figure 5.11: PI/SMC Current

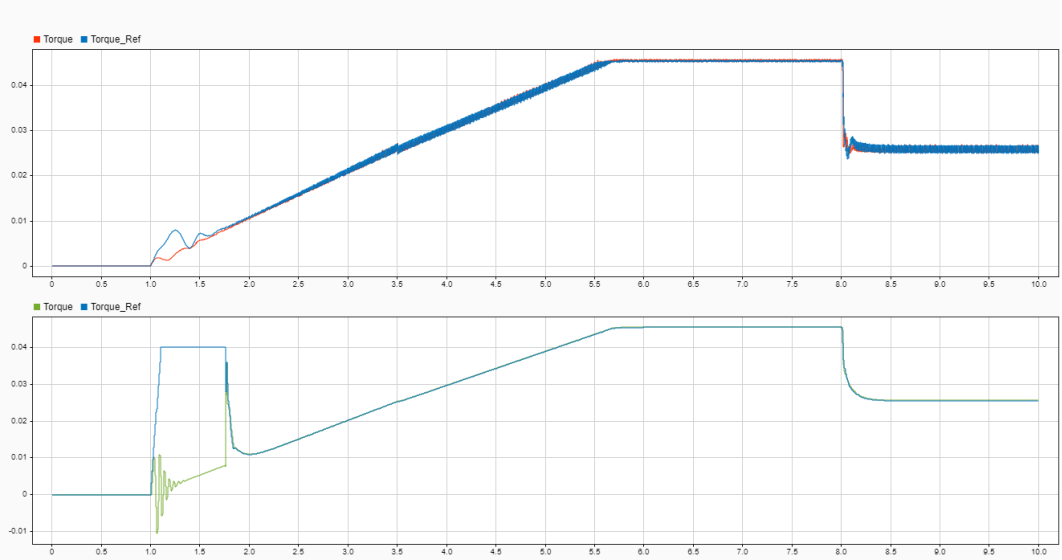


Figure 5.12: PI/SMC Torque

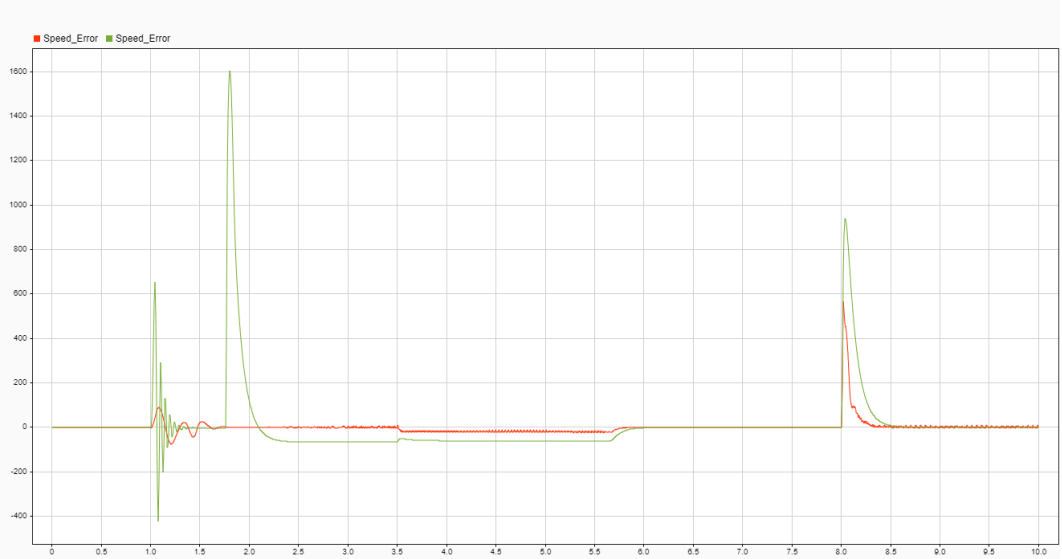


Figure 5.13: PI/SMC Speed Error

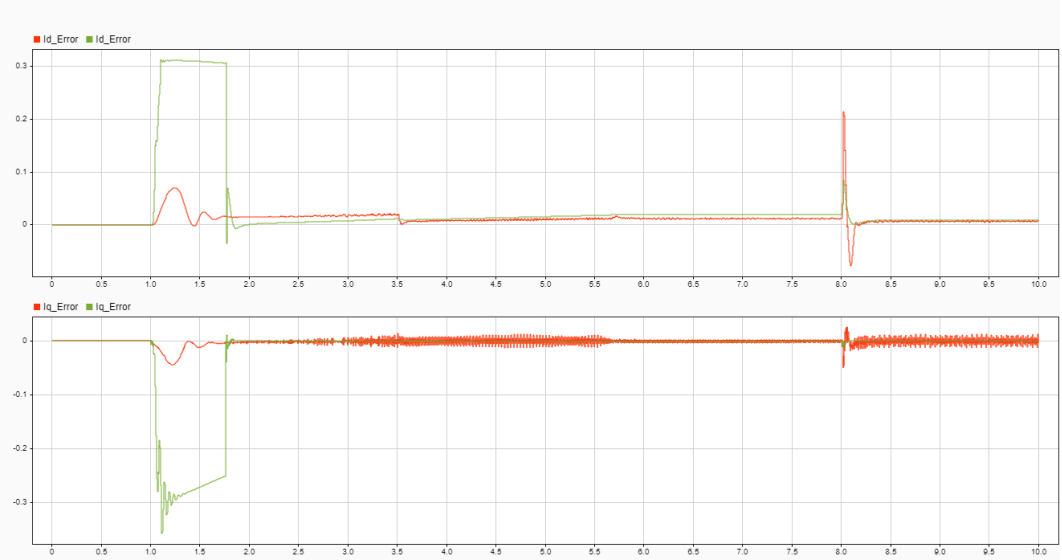


Figure 5.14: PI/SMC Current Error

Chapter 6

Robustness Tests

Robustness testing was carried out to evaluate the control system's ability to maintain performance under uncertain conditions, such as parameter variations and external disturbances. In practical applications, motor parameters like resistance, inductance, and the load torque often vary due to changes in temperature, mechanical wear, or external forces. Therefore, the robustness tests aimed to verify how well the proposed control strategy could handle these real-world variations while maintaining stable and accurate control of the motor's speed and current.
In order to perform such a test, the motor electro-mechanical parameters were varied. All the parameters were tested at their boundary values using a method called *Design of Experiments* (**DOE**). DOE is a statistical approach used to design and execute experiments to identify relationships between input and output variables. By testing the regulator's robustness in extreme scenarios, the system's performance was evaluated under unlikely real-world conditions where all parameters simultaneously reach their limit 6.1.

6.1 IAE and MSE

Two indices were chosen as indices to be evaluated for robustness: Integral Absolute Error (**IAE**), Mean Absolute Error (**MSE**). The IAE measures the sum of the absolute error between the desired value and the value obtained over time.

$$IAE = \int |e(t)|dt \tag{6.1}$$

The IAE provides an evaluation of the accumulated total error, so it is useful for assessing the accuracy of the system in the long run. It is particularly sensitive to errors that persist over time, so it can indicate how consistently the system is capable of reducing error, which is a desired characteristic in a robust system. MSE is the average of the squared errors between the desired value and the measured value. It

is calculated as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(e_i)^2 \tag{6.2}$$

Its sensitivity to large errors makes it effective in measuring how well the system can withstand significant disturbances or extreme operating conditions, which is essential for evaluating robustness.

6.2 Tests Configuration and Evaluation

The parameters involved in tests execution are:

- **Rs** : stator resistance
- **Ls** : inductance
- **Phim** : electromagnetic flux
- **J** : inertia coefficient
- **B** : friction coefficient

Parameter	**Nominal Value**	**Min** (10°C)	**Max** (100°C)	**Unit Measure**
Rs	45.5	40.2686	61.5965	Ohm
Ls	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	$Kg * m^2$
B	7.40E-05	7.03E-05	7.77E-05	$Nm/rad/s$

Table 6.1: Parameters

Tab.6.2 shows the configurations of the various tests. Each of the 5 parameters involved are varied to their maximum (1) or their minimum (-1). Thus resulting a total of 32 tests. Thanks to the support of a special MATLAB script, simulations were carried out automatically in each of the 32 cases.
First, the values of IAE and MSE were collected under nominal conditions, so that they could be compared with the values of the same indices under anomalous conditions.

N. test	**Rs**	**Ls**	**Phim**	**J**	**B**
1	1	1	-1	1	1
2	1	1	1	-1	1
3	-1	-1	-1	-1	1
4	1	-1	1	1	-1
5	-1	-1	-1	1	-1
6	1	1	-1	-1	1
7	-1	1	-1	1	1
8	-1	1	1	1	-1
9	1	-1	1	1	1
10	-1	1	-1	1	-1
11	-1	1	-1	-1	1
12	-1	-1	-1	1	1
13	-1	1	-1	-1	-1
14	-1	1	1	1	1
15	1	-1	-1	1	1
16	1	-1	1	-1	1
17	1	-1	1	-1	-1
18	1	1	1	1	1
19	1	-1	-1	-1	-1
20	1	1	1	1	-1
21	-1	-1	1	1	-1
22	-1	1	1	-1	-1
23	1	1	1	-1	-1
24	-1	-1	1	-1	1
25	-1	-1	-1	-1	-1
26	-1	-1	1	1	1
27	1	-1	-1	-1	1
28	-1	1	1	-1	1
29	-1	-1	1	-1	-1
30	1	1	-1	1	-1
31	1	-1	-1	1	-1
32	1	1	-1	-1	-1

Table 6.2: Tabella dei Test

6.2.1 1^{st} Attempt Results

N° test	IAE Nominal	MSE Nominal	IAE test	MSE test
1	16.9221	13.3985	1.8276e+03	2.2758e+05
2	16.9221	13.3985	17.2745	0.3002
3	16.9221	13.3985	1.8437e+03	2.3506e+05
4	16.9221	13.3985	17.4216	1.1117
5	16.9221	13.3985	1.6442e+03	2.3913e+05
6	16.9221	13.3985	1.8698e+03	2.2759e+05
7	16.9221	13.3985	1.8336e+03	2.3485e+05
8	16.9221	13.3985	15.6737	1.0532
9	16.9221	13.3985	17.3195	0.2922
10	16.9221	13.3985	1.6442e+03	2.3913e+05
11	16.9221	13.3985	1.8437e+03	2.3506e+05
12	16.9221	13.3985	1.8336e+03	2.3485e+05
13	16.9221	13.3985	1.6636e+03	2.3942e+05
14	16.9221	13.3985	15.4828	0.2626
15	16.9221	13.3985	1.8276e+03	2.2758e+05
16	16.9221	13.3985	17.2745	0.3002
17	16.9221	13.3985	17.3726	1.1089
18	16.9221	13.3985	17.3195	0.2922
19	16.9221	13.3985	1.7513e+03	2.3138e+05
20	16.9221	13.3985	17.4216	1.1117
21	16.9221	13.3985	15.6737	1.0532
22	16.9221	13.3985	15.6555	1.0557
23	16.9221	13.3985	17.3726	1.1089
24	16.9221	13.3985	15.4644	0.2651
25	16.9221	13.3985	1.6636e+03	2.3942e+05
26	16.9221	13.3985	15.4828	0.2626
27	16.9221	13.3985	1.8698e+03	2.2759e+05
28	16.9221	13.3985	15.4644	0.2651
29	16.9221	13.3985	15.6555	1.0557
30	16.9221	13.3985	1.7517e+03	2.3160e+05
31	16.9221	13.3985	1.7517e+03	2.3160e+05

Table 6.3: SMC + STSMO Robustness tests failed

A first attempt of the 32 tests was performed with the same tuning chosen for simulations under nominal conditions. As shown in table 6.3, the results suggest that with the chosen tuning, the system loses stability when perturbations of the chosen parameters occur. It has been noted that large variations in the metrics considered occur when the electromagnetic flux value *Phim* is set at its minimum

(0.0673). Therefore, once the particular sensitivity of the system to this parameter was understood, *Phim* was lowered below its minimum value and the tuning of the controller and observer parameters was performed again.
Retuning concerned the following parameters:

Controller

- `epsilon` (Speed regulator) = 200
- `lambda0` = 0.06
- `K_Torque` = 0.09

Observer

- `k1` = 600
- `k2` = 1000
- `kp` = 10
- `ki` = 200

Note that changing the value of *Phim*, represented in the code by `lambda0`(λ_0), also requires changing the value of the constant K_Torque as it is directly proportional to it ($K_{Torque} = (3/2)\lambda_0 N_r$, with N_r the number of pole pairs).

6.2.2 2^{nd} Attempt Results

N° test	IAE Nominal	MSE Nominal	IAE test	MSE test
1	36.04	1.47e+02	35.2885	3.0107
2	36.04	1.47e+02	426.9645	327.2921
3	36.04	1.47e+02	30.8898	3.6964
4	36.04	1.47e+02	407.8074	215.9621
5	36.04	1.47e+02	31.7898	2.4367
6	36.04	1.47e+02	35.2494	3.0067
7	36.04	1.47e+02	30.9262	3.7297
8	36.04	1.47e+02	32.3549	4.9990
9	36.04	1.47e+02	426.9100	317.6038
10	36.04	1.47e+02	31.7898	2.4367
11	36.04	1.47e+02	30.8898	3.6964
12	36.04	1.47e+02	30.9262	3.7297
13	36.04	1.47e+02	31.7471	2.4336
14	36.04	1.47e+02	31.5458	4.2950
15	36.04	1.47e+02	35.2885	3.0107
16	36.04	1.47e+02	426.9645	327.2921
17	36.04	1.47e+02	407.8756	219.4572
18	36.04	1.47e+02	426.9100	317.6038
19	36.04	1.47e+02	35.7350	5.0914
20	36.04	1.47e+02	407.8074	215.9621
21	36.04	1.47e+02	32.3549	4.9990
22	36.04	1.47e+02	32.3726	4.8868
23	36.04	1.47e+02	407.8756	219.4572
24	36.04	1.47e+02	31.6908	4.3920
25	36.04	1.47e+02	31.7471	2.4336
26	36.04	1.47e+02	31.5458	4.2950
27	36.04	1.47e+02	35.2494	3.0067
28	36.04	1.47e+02	31.6908	4.3920
29	36.04	1.47e+02	32.3726	4.8868
30	36.04	1.47e+02	35.7924	5.0696
31	36.04	1.47e+02	35.7924	5.0696
32	36.04	1.47e+02	35.7350	5.0914

Table 6.4: SMC + STSMO after parameters retuning

The results of applying the new tuning are shown in Tab.6.4. In this case, the variations in the indices have decreased a lot, but the common pattern correlating failed tests has been lost. So the choice was to restore the previous tuning and further lower `lambda0` and `K_Torque`.

The chosen values are:

- `lambda0` = 0.03
- `K_Torque` = 0.045

6.2.3 3^{rd} Attempt Results

N°test	IAE Nominal	MSE Nominal	IAE test	MSE test	diffIAE	diffMSE
1	12.6410	1.5600	20.4743	0.3337	7.8333	-1.2263
2	12.6410	1.5600	13.6277	0.6658	0.9867	-0.8942
3	12.6410	1.5600	14.3076	1.3688	1.6666	-0.1912
4	12.6410	1.5600	13.6716	1.5416	1.0306	-0.0184
5	12.6410	1.5600	14.3728	0.0449	1.7318	-1.5151
6	12.6410	1.5600	20.2677	0.3362	7.6267	-1.2238
7	12.6410	1.5600	14.3116	1.3200	1.6706	-0.2400
8	12.6410	1.5600	12.7928	1.7398	0.1518	0.1798
9	12.6410	1.5600	13.6026	0.7773	0.9616	-0.7827
10	12.6410	1.5600	14.3728	0.0449	1.7318	-1.5151
11	12.6410	1.5600	14.3076	1.3688	1.6666	-0.1912
12	12.6410	1.5600	14.3116	1.3200	1.6706	-0.2400
13	12.6410	1.5600	14.3697	0.0525	1.7287	-1.5075
14	12.6410	1.5600	12.6258	2.4840	-0.0152	0.9240
15	12.6410	1.5600	20.4743	0.3337	7.8333	-1.2263
16	12.6410	1.5600	13.6277	0.6658	0.9867	-0.8942
17	12.6410	1.5600	13.6947	1.3789	1.0537	-0.1811
18	12.6410	1.5600	13.6026	0.7773	0.9616	-0.7827
19	12.6410	1.5600	20.3965	1.3352	7.7555	-0.2248
20	12.6410	1.5600	13.6716	1.5416	1.0306	-0.0184
21	12.6410	1.5600	12.7928	1.7398	0.1518	0.1798
22	12.6410	1.5600	12.8644	1.9997	0.2234	0.4397
23	12.6410	1.5600	13.6947	1.3789	1.0537	-0.1811
24	12.6410	1.5600	12.6913	2.4153	0.0503	0.8553
25	12.6410	1.5600	14.3697	0.0525	1.7287	-1.5075
26	12.6410	1.5600	12.6258	2.4840	-0.0152	0.9240
27	12.6410	1.5600	20.2677	0.3362	7.6267	-1.2238
28	12.6410	1.5600	12.6913	2.4153	0.0503	0.8553
29	12.6410	1.5600	12.8644	1.9997	0.2234	0.4397
30	12.6410	1.5600	20.7150	1.2686	8.0740	-0.2914
31	12.6410	1.5600	20.7150	1.2686	8.0740	-0.2914
32	12.6410	1.5600	20.3965	1.3352	7.7555	-0.2248

Table 6.5: SMC + STSMO Robustness tests OK

As can be seen in Tab.6.5, this latest choice of parameters allows for good results in terms of robustness. Based on the obtained results, it can therefore be concluded that the system is particularly sensitive in terms of robustness to the parameter

`lambda0`.

By comparing these results with those obtained from the robustness tests conducted on the PI controller, it can be observed that they are fully comparable. In the last two columns of Tables 6.5and 6.6, the variations observed in the analyzed indices across the different cases are reported. When comparing those related to the SMC with those related to the PI, we notice that in some cases, particularly with regard to the IAE, the SMC control exhibits smaller variations compared to the PI. This consistent performance across multiple tests highlights the robustness and reinforces its reliability for real-world applications.

N°test	**IAE Nominal**	**MSE Nominal**	**IAE test**	**MSE test**	**diffIAE**	**diffMSE**
1	56.4062	2.1e-07	64.4127	7.54e-08	8.0065	-1.346e-07
2	56.4062	2.1e-07	53.8872	2.33e-08	-2.5190	-1.867e-07
3	56.4062	2.1e-07	64.4751	2.69e-07	8.0689	5.89e-08
4	56.4062	2.1e-07	53.7957	1.13e-07	-2.6105	-9.7e-08
5	56.4062	2.1e-07	64.2792	1.34e-07	7.8730	-7.59e-08
6	56.4062	2.1e-07	64.3975	4.56e-08	7.9913	-1.64e-07
7	56.4062	2.1e-07	64.5113	2.33e-08	8.1051	-1.867e-07
8	56.4062	2.1e-07	53.8351	1.83e-07	-2.5711	-2.69e-08
9	56.4062	2.1e-07	53.9064	9.31e-10	-2.4998	-2.09069e-07
10	56.4062	2.1e-07	64.2792	1.34e-07	7.8730	-7.59e-08
11	56.4062	2.1e-07	64.4751	2.69e-07	8.0689	5.89e-08
12	56.4062	2.1e-07	64.5113	2.33e-08	8.1051	-1.867e-07
13	56.4062	2.1e-07	64.2402	1.13e-07	7.8340	-9.7e-08
14	56.4062	2.1e-07	53.9458	8.38e-09	-2.4604	-2.0162e-07
15	56.4062	2.1e-07	64.4127	7.54e-08	8.0065	-1.346e-07
16	56.4062	2.1e-07	53.8877	2.33e-08	-2.5185	-1.867e-07
17	56.4062	2.1e-07	53.7716	1.13e-07	-2.6346	-9.7e-08
18	56.4062	2.1e-07	53.9064	9.31e-10	-2.4998	-2.09069e-07
19	56.4062	2.1e-07	64.1643	1.13e-07	7.7581	-9.7e-08
20	56.4062	2.1e-07	53.7957	1.13e-07	-2.6105	-9.7e-08
21	56.4062	2.1e-07	53.8351	1.83e-07	-2.5711	-2.69e-08
22	56.4062	2.1e-07	53.8111	1.83e-07	-2.5951	-2.69e-08
23	56.4062	2.1e-07	53.7716	1.13e-07	-2.6346	-9.7e-08
24	56.4062	2.1e-07	53.9276	9.31e-10	-2.4786	-2.09069e-07
25	56.4062	2.1e-07	64.2402	1.13e-07	7.8340	-9.7e-08
26	56.4062	2.1e-07	53.9458	8.38e-09	-2.4604	-2.0162e-07
27	56.4062	2.1e-07	64.3975	4.56e-08	7.9913	-1.64e-07
28	56.4062	2.1e-07	53.9276	9.31e-10	-2.4786	-2.09069e-07
29	56.4062	2.1e-07	53.8111	1.83e-07	-2.5951	-2.69e-08
30	56.4062	2.1e-07	64.1839	1.13e-07	7.7777	-9.7e-08
31	56.4062	2.1e-07	64.1839	1.13e-07	7.7777	-9.7e-08
32	56.4062	2.1e-07	64.1643	1.13e-07	7.7581	-9.7e-08

Table 6.6: PI + Luenberger Observer Robustness tests

Chapter 7

Code Generation and Experiment Setup

This chapter aims to outline the steps to be followed for the practical implementation of the designed project. First, the procedure for generating the C code will be illustrated, and then the setup of the laboratory experiments will be described.

7.1 Code Generation

Generating C code allows the model to be compiled and run on hardware platforms that support real-time processing. It's a crucial step in the process of transitioning from simulation to real-world implementation.

The Observer block mentioned in Chapter 2 is a model reference block: a reference to an external standalone Simulink model that implements the observer logic. The reference takes place through a call to file `Observer_Sliding.slx`.
As a first step, it is necessary to run script `Slidingparam_and_codegen_init.m`. Next, after opening file `Observer_Sliding.slx`, press Ctrl + B to generate the file ".C" containing the code. The C code must then be compiled by running the script `compile_CodeGen_obs.m`. All the files required for the generation procedure are contained in the folder `main_model`. The result of the compilation will be a file ".mex" that will be called by the *S-function* that implements the code in the Simulink model. The *S-function* block is placed within the SIL block (Fig.2.13c) responsible for simulating the generated code.
To switch to SIL simulation mode, it is necessary to double-click the *manual switches* (Fig.7.2), specifically placed in the model to connect the signals from the SIL labels to the control loop.

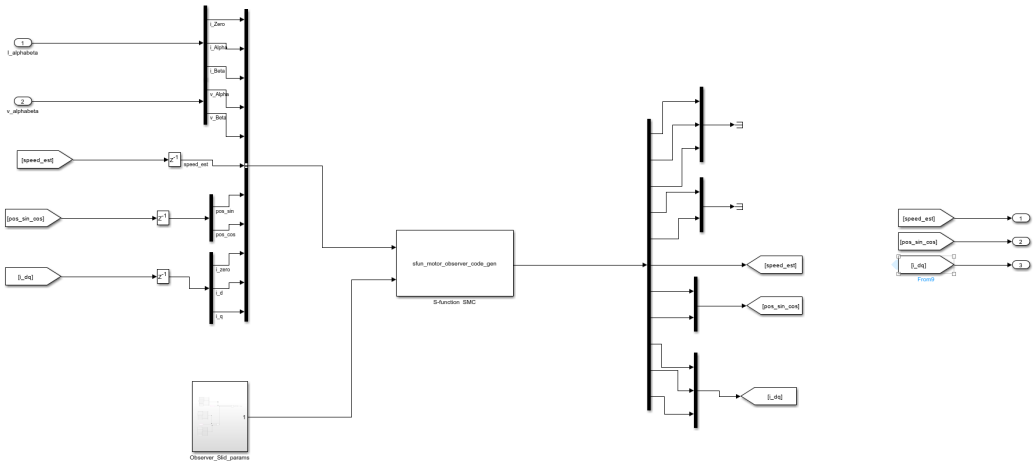


Figure 7.1: Observer *S-function*

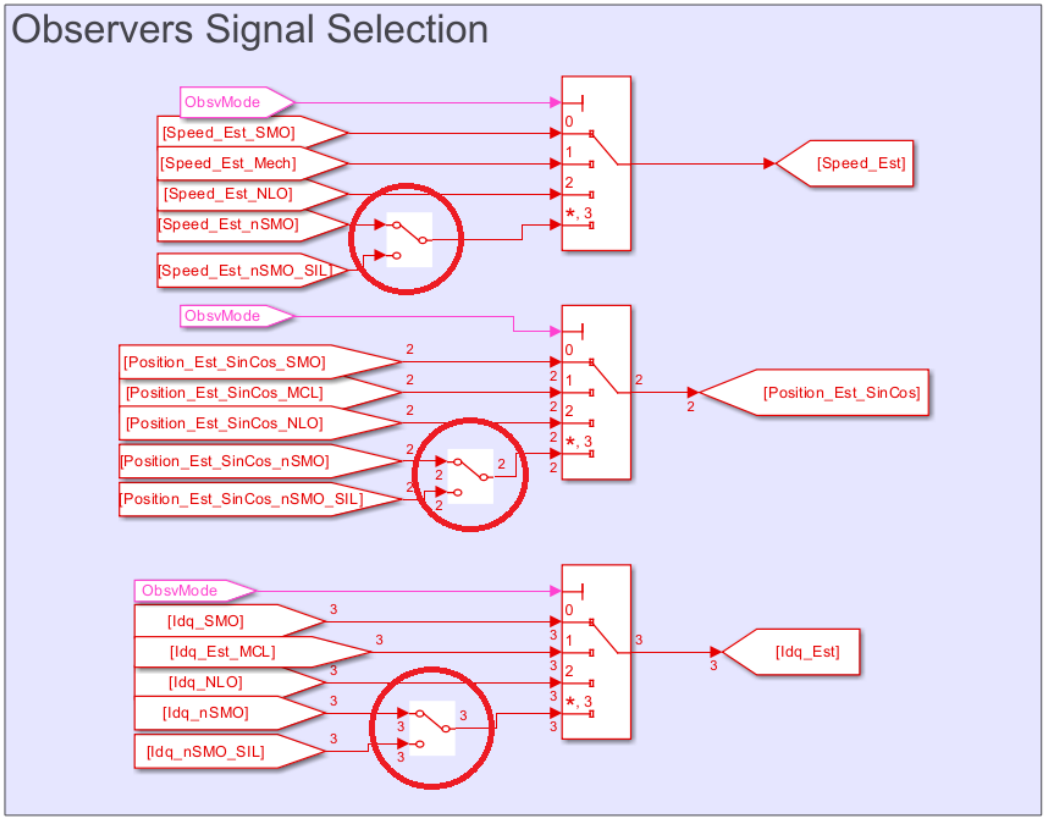


Figure 7.2: Manual switches for the estimated signals

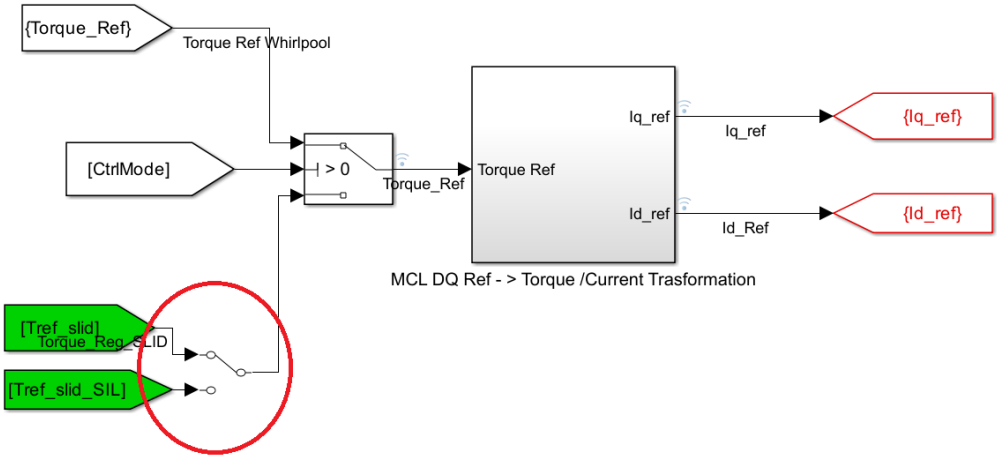


Figure 7.3: Manual switches for torque signal

7.2 Experiment Setup

The present chapter intends to describe the procedure adopted for the experiments setup , the devices employed, and the results achieved. As initially mentioned, the control algorithm was tested on the drain pump of a dishwasher(Fig.7.4,7.5). Various hardware and software tools were used to support the connection between the PC and the dishwasher board:

- **Debugger J-Link Arm** : for the firmware download onto the microcontroller.
- **U-Link device**: to enable communication between the PC and the microcontroller.
- **UART serial interface**: for reading and writing variables within firmware.
- **MAC Dish** : software used to operate the various dishwasher components through switches.
- **Free Master** : to display and modify control parameters.

Thanks to *Mac Dish* was possible to modify the variable of interest for the test, namely variable `fill_valve`. Through this variable, it was possible to control the solenoid valve that regulates the water filling of the dishwasher.In the conducted tests, the valve was kept active for 60 seconds.
Free Master allowed for the control of the start and end of the tests, as well as the configurations in which they were conducted. In particular, to start a test, it was necessary to follow the steps below:

1. Set `Dbg_Mode_Selector`: To choose which of the two controllers to use for the test (PI/Sliding Mode).
2. Set `Dbg_Obs_Selector`: To choose which of the two observers to use for the test (Whirl/Sliding Mode).
3. Set `BD_Target_Speed` = 3300
4. Set `BD_Target_Accel` = 2000
5. Set the `BD_Update_Cmd` = 1 to start the engine.

Figure 7.4: Model of the Whirlpool dishwasher used for the tests.

Figure 7.5: PM drain pump

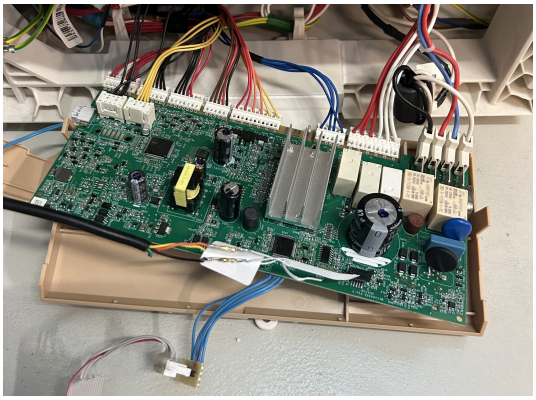


Figure 7.6: Board

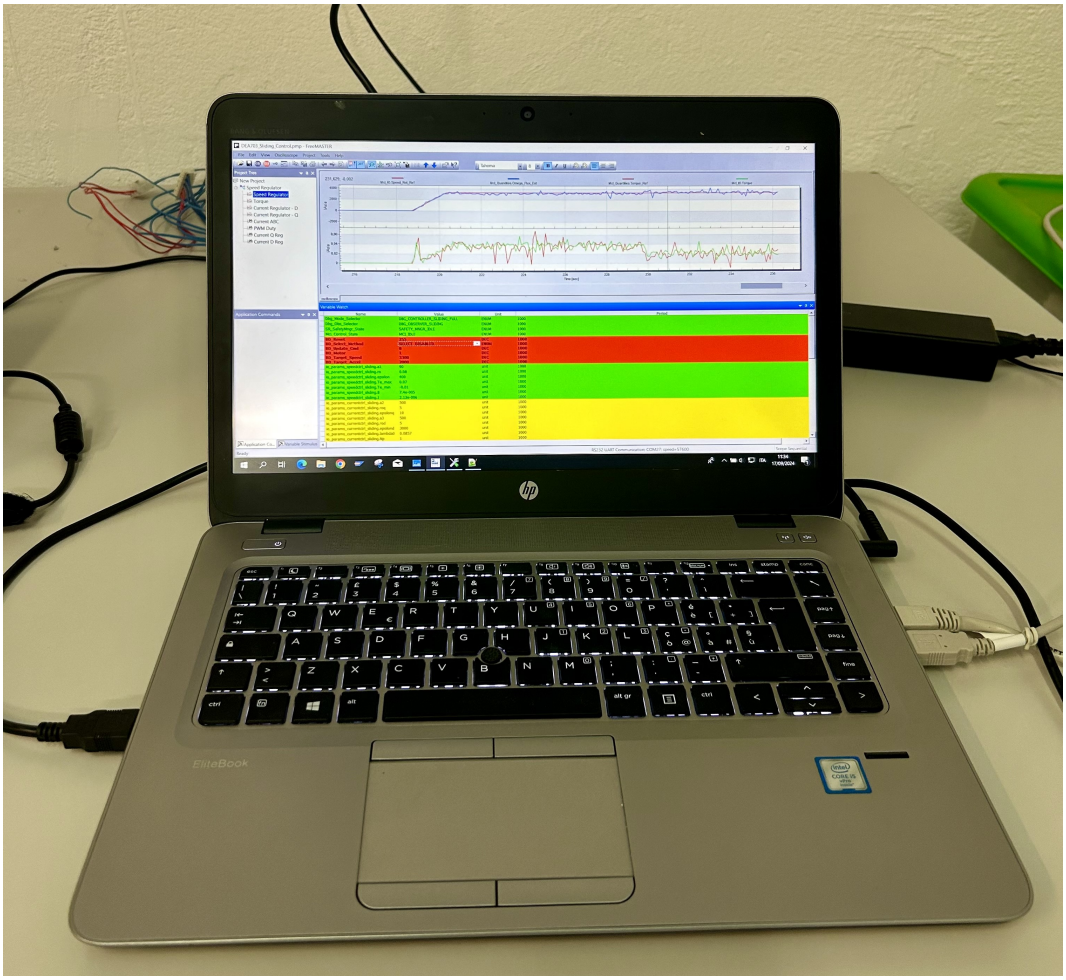


Figure 7.7: PC using *Free Master*

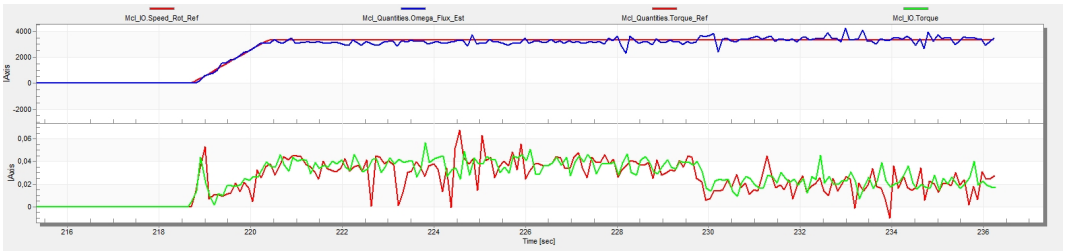


Figure 7.8: Speed Sliding - Free Master

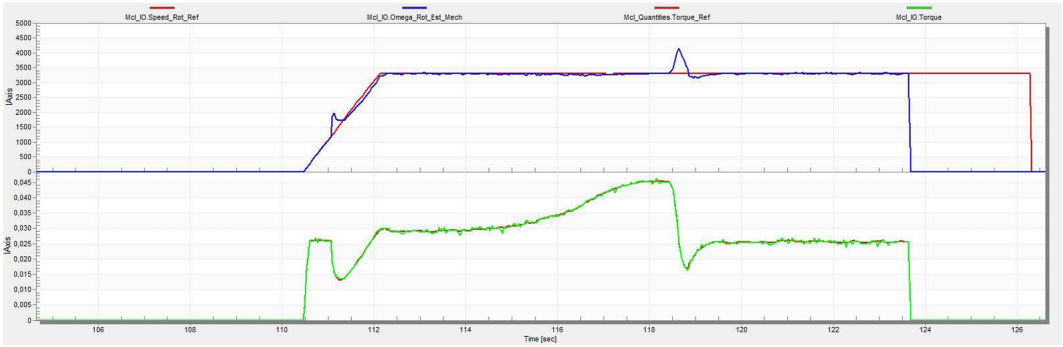


Figure 7.9: Speed PI - Free Master

7.3 Discussion of the results

As shown in the graphical results available from *Free Master* (Fig. 7.8,7.9), both controllers behave as expected: the PI controller provides a controlled signal with fewer oscillations, although it is sensitive to the torque drop after the eighth second of the simulation, resulting in a overelongation in the speed signal. The sliding mode controller, on the other hand, exhibits much more pronounced oscillations in the controlled signal, but when the torque drop occurs, it does not react and remains in its state.

As highlighted by the experimental results, the Sliding Mode Controller combined with the Sliding Mode Observer presents itself as a strong candidate to replace the PI controller and, in general, a more reliable choice in environments where fluctuations and external changes are expected.

Chapter 8

Conclusions and future developments

In this thesis project, the sensorless control problem of the Permanent Magnet Synchronous Motor (PMSM) of a Whirlpool dishwasher was addressed. A fully Sliding Mode-based approach was proposed. Specifically, a STSM Observer was introduced, capable of reliably estimating the rotor's speed and position without the use of physical sensors. After reviewing all the theory underlying the approach and the logic employed in the design, the system was first tested through simulations and subsequently, following robustness tests, applied to the real-world case of a dishwasher. The laboratory tests certainly met expectations: the fully sliding mode control system demonstrated its ability to track the desired signal and exhibited remarkable robustness. The results obtained, when compared with the performance of the PI controller, suggest that a control system based on the proposed approach could be a valid alternative to the type of control currently employed in Whirlpool productions.

As future development, an improvement of the sliding mode technique is proposed, for example by using the sigmoid function instead of the saturation function for both the control law and the observer. In particular, for position estimation in the observer, it may be worth attempting to use a sliding mode controller replacing the PI controller. Recent studies in the literature are moving towards Model Predictive Control (MPC). MPC approches are known for its ability to handle multivariable systems with constraints and for its flexibility in optimizing the system's behavior at each control instant. In particular, the Finite Control Set (FCS) technique is popular for its ease of implementation and its ability to reduce computation times, although it has been shown that performance is greatly influenced by the accurate knowledge of the model. For this reasons, this approach could represent an interesting direction for future research.

Appendice A

.1 Saturation MATLAB Function

```
function [sat_ia_error,sat_ib_error, ia_error, ib_error]  = fcn(i_alpha,i_beta,i_alpha_est,i_beta_est,params)

epsilon_a = params.epsilona;
epsilon_b = params.epsilonb;

ia_error = i_alpha_est - i_alpha;
ib_error = i_beta_est - i_beta;


if abs(ia_error) > epsilon_a
    sat_ia_error = sign(ia_error);
else
    sat_ia_error = ia_error/epsilon_a;
end

if abs(ib_error) > epsilon_b
    sat_ib_error = sign(ib_error);
else
    sat_ib_error = ib_error/epsilon_b;
end
```

.2 Back EMF Estimation MATLAB Function

```
function [di_alpha_est,di_beta_est, e_alpha, e_beta] = fcn(u_alpha,u_beta,params,sat_ia,sat_ib,int_sat_ia,int_sat_ib, i_alpha_err,i_beta_err,i_alpha_est,i_beta_est)


k1 = params.k1;
k2 = params.k2;
R  = params.R;
L = params.Ld;

z_alpha = k1*sqrt(abs(i_alpha_err))*sat_ia + k2*int_sat_ia;
z_beta = k1*sqrt(abs(i_beta_err))*sat_ib + k2*int_sat_ib;

di_alpha_est = -(R/L)*i_alpha_est + (1/L)*u_alpha - (1/L)*z_alpha;
di_beta_est = -(R/L)*i_beta_est + (1/L)*u_beta - (1/L)*z_beta;

e_alpha=z_alpha;
e_beta = z_beta;
```

.3 Direct Park MATLAB Function

```
function [IdqZero,Id,Iq] = DirectPark(IZero, IAlpha,IBeta,sinTheta, cosTheta)

T=[cosTheta sinTheta;
    -sinTheta cosTheta];

Idq=T*[IAlpha;IBeta];
IdqZero = IZero;
Id = Idq(1);
Iq = Idq(2);
```

Appendice B

Code Variable Name	Bus Type	Description	Type
Speed_Rot_Ref	MCL_SPEED_SMC_IO_F_TYPE	Speed Reference (rad/s)	Input
Speed_Rot	MCL_SPEED_SMC_IO_F_TYPE	Actual Speed (rad/s)	Input
Torque_Ref	MCL_SPEED_SMC_IO_F_TYPE	Torque Reference	Output
a1	MCL_SPEED_SMC_PARAMS_TYPE	Speed Control law tuning parameter	Parameter
ro	MCL_SPEED_SMC_PARAMS_TYPE	Speed Control law tuning parameter	Parameter
Te_max	MCL_SPEED_SMC_PARAMS_TYPE	Torque upper bound	Parameter
Te_min	MCL_SPEED_SMC_PARAMS_TYPE	Torque lower bound	Parameter
B	MCL_SPEED_SMC_PARAMS_TYPE	Coefficient of viscous friction	Parameter
J	MCL_SPEED_SMC_PARAMS_TYPE	Mechanical Inertia	Parameter
Id_ref	MCL_CURRENT_SMC_IO_F_TYPE	Current I_d Reference	Input
Iq_ref	MCL_CURRENT_SMC_IO_F_TYPE	Current I_q Reference	Input
Id_ref	MCL_CURRENT_SMC_IO_F_TYPE	Current I_d Reference	Input
Id	MCL_CURRENT_SMC_IO_F_TYPE	Measured/Estimated Current I_d	Input
Iq	MCL_CURRENT_SMC_IO_F_TYPE	Measured/Estimated Current I_d	Input
Sin	MCL_CURRENT_SMC_IO_F_TYPE	Estimated Position Sine function	Input
Cos	MCL_CURRENT_SMC_IO_F_TYPE	Estimated Position Cosine function	Input
Speed	MCL_CURRENT_SMC_IO_F_TYPE	Measured/Estimated Speed (rad/s)	Input
Vs_Alpha_Beta_Ref_Zero	MCL_CURRENT_SMC_IO_F_TYPE	Voltage V_0 Reference	Output
Vs_Alpha_Ref	MCL_CURRENT_SMC_IO_F_TYPE	Voltage V_α Reference	Output
Vs_Beta_Ref	MCL_CURRENT_SMC_IO_F_TYPE	Voltage V_β Reference	Output
Vd_Ref	MCL_CURRENT_SMC_IO_F_TYPE	Voltage V_d Reference	Output
Vq_Ref	MCL_CURRENT_SMC_IO_F_TYPE	Voltage V_q Reference	Output
a2	MCL_CURRENT_SMC_PARAMS_TYPE	Current I_q Control law tuning parameter	Parameter
roq	MCL_CURRENT_SMC_PARAMS_TYPE	Current I_q Control law tuning parameter	Parameter
epsilonq	MCL_CURRENT_SMC_PARAMS_TYPE	Current I_q Control law tuning parameter	Parameter
a2	MCL_CURRENT_SMC_PARAMS_TYPE	Current I_d Control law tuning parameter	Parameter
rod	MCL_CURRENT_SMC_PARAMS_TYPE	Current I_d Control law tuning parameter	Parameter
epsilond	MCL_CURRENT_SMC_PARAMS_TYPE	Current I_d Control law tuning parameter	Parameter
lambda0	MCL_CURRENT_SMC_PARAMS_TYPE	Flux linkage of the permanent magnet	Parameter
R	MCL_CURRENT_SMC_PARAMS_TYPE	Winding Resistance	Parameter
L	MCL_CURRENT_SMC_PARAMS_TYPE	Winding Inductance	Parameter
I_Zero	MCL_SLID_OBS_IO_TYPE	Measured Current I_0	Input
I_Alpha	MCL_SLID_OBS_IO_TYPE	Measured Current I_α	Input
I_Beta	MCL_SLID_OBS_IO_TYPE	Measured Current I_β	Input
V_Alpha	MCL_SLID_OBS_IO_TYPE	Measured Voltage V_α	Input
V_Beta	MCL_SLID_OBS_IO_TYPE	Measured Voltage V_β	Input
Speed_Est	MCL_SLID_OBS_IO_TYPE	Estimated Speed (rad/s)	Output
Position_Sin	MCL_SLID_OBS_IO_TYPE	Estimated Position Sine function	Output
Position_Cos	MCL_SLID_OBS_IO_TYPE	Estimated Position Cosine function	Output
I_zero	MCL_SLID_OBS_IO_TYPE	Estimated I_0 current	Output
I_d	MCL_SLID_OBS_IO_TYPE	Estimated I_d current	Output
I_q	MCL_SLID_OBS_IO_TYPE	Estimated I_q current	Output
k1	MCL_SLID_OBS_PARAMS_TYPE	Observer tuning parameter	Parameter
k2	MCL_SLID_OBS_PARAMS_TYPE	Observer tuning parameter	Parameter
epsilona	MCL_SLID_OBS_PARAMS_TYPE	Observer tuning parameter	Parameter
epsilonb	MCL_SLID_OBS_PARAMS_TYPE	Observer tuning parameter	Parameter
R	MCL_SLID_OBS_PARAMS_TYPE	Winding Resistance	Parameter
L	MCL_SLID_OBS_PARAMS_TYPE	Winding Inductance	Parameter
kp	MCL_SLID_OBS_PARAMS_TYPE	Observer tuning parameter	Parameter
ki	MCL_SLID_OBS_PARAMS_TYPE	Observer tuning parameter	Parameter

Table 1: Motor Control Block Variables

Bibliography

[1] Yuefei Zuo, Xiaoyong Zhu, Li Quan, Chao Zhang, Yi Du, and Zixuan Xiang. Active disturbance rejection controller for speed control of electrical drives using phase-locking loop observer. *IEEE Transactions on Industrial Electronics*, 66(3):1748–1759, 2019.

[2] Fortino Mendoza-Mondragón, Víctor Manuel Hernández-Guzmán, and Juvenal Rodríguez-Reséndiz. Robust speed control of permanent magnet synchronous motors using two-degrees-of-freedom control. *IEEE Transactions on Industrial Electronics*, 65(8):6099–6108, 2018.

[3] M. S. Khajueezadeh, Sajjad FeizHoseini, Z. Nasiri-Gheidari, and Mehdi Behzad. Analysis of torsional vibrations on the resolver under eccentricity in pmsm drive system. *IEEE Sensors Journal*, 22(22):21592–21599, 2022.

[4] Tanja Zwerger and Paolo Mercorelli. Backward extended kalman filter to estimate and adaptively control a pmsm in saturation conditions. *IEEE Journal of Emerging and Selected Topics in Industrial Electronics*, 5(2):462–474, 2024.

[5] Bahaa Hafez, Ayman S. Abdel-Khalik, Ahmed M. Massoud, Shehab Ahmed, and Robert D. Lorenz. Single-sensor-based three-phase permanent-magnet synchronous motor drive system with luenberger observers for motor line current reconstruction. *IEEE Transactions on Industry Applications*, 50(4):2602–2613, 2014.

[6] P. L. Xu and Z. Q. Zhu. Novel carrier signal injection method using zero-sequence voltage for sensorless control of pmsm drives. *IEEE Transactions on Industrial Electronics*, 63(4):2053–2061, 2016.

[7] Junjie Hong, Xijian Lin, Jianbo Zhang, Wei Huang, Baiping Yan, and Xiyu Li. A composite sliding mode control with the first-order differentiator and sliding mode observer for permanent magnet synchronous machine. *ISA transactions*, 147:489–500, 2024.

[8] Zhaowei Qiao, Tingna Shi, Yindong Wang, Yan Yan, Changliang Xia, and Xiangning He. New sliding-mode observer for position sensorless control of permanent-magnet synchronous motor. *IEEE Transactions on Industrial electronics*, 60(2):710–719, 2012.

[9] Asif Chalanga, Shyam Kamal, Leonid M Fridman, Bijnan Bandyopadhyay, and Jaime A Moreno. Implementation of super-twisting control: Super-twisting and higher order sliding-mode observer-based approaches. *IEEE Transactions on Industrial Electronics*, 63(6):3677–3685, 2016.

[10] Shuo Chen, Xiao Zhang, Xiang Wu, Guojun Tan, and Xianchao Chen. Sensorless control for ipmsm based on adaptive super-twisting sliding-mode observer and improved phase-locked loop. *Energies*, 12(7):1225, 2019.

[11] LAURA MORETTI. Controllo sensorless di un motore sincrono a magneti permanenti: validazione sperimentale su un elettrodomestico. 2022.

[12] RUMEYSA NUR GULESIN. Progetto e sviluppo di un sistema di controllo model based e sensorless per un motore sincrono a magneti permanenti. 2022.

[13] Christopher Laughman, Steven B Leeb, Leslie K Norford, Steven R Shaw, and Peter R Armstrong. A park transform-based method for condition monitoring of three-phase electromechanical systems. 2010.

[14] R. H. Park. Two-reaction theory of synchronous machines generalized method of analysis-part i. *Transactions of the American Institute of Electrical Engineers*, 48(3):716–727, 1929.

[15] Dorin O Neacsu. Space vector modulation-an introduction. In *IECON*, volume 1, pages 1583–1592, 2001.

[16] Giuseppe Orlando. *Tecniche di controllo a struttura variabile per un veicolo sottomarino comandato a distanza.* PhD thesis, 1996.

[17] Qiwei Xu, Donghao Jiang, Yiming Wang, Xuefeng Zhang, Jincheng Liu, and Yangming Chen. Variable-step close-loop angle compensation method of pmsm rotor position estimation based on super-twisting sliding-mode observer using tangent reaching law. *Energy Reports*, 9:356–361, 2023.

[18] Muhammad Rafiq, Saeed ur Rehman, Fazal ur Rehman, Qarab Raza Butt, and Irfan Awan. A second order sliding mode control design of a switched reluctance motor using super twisting algorithm. *Simulation Modelling Practice and Theory*, 25:106–117, 2012.

[19] Jaime A. Moreno and Marisol Osorio. A lyapunov approach to second-order sliding mode controllers and observers. In *2008 47th IEEE Conference on Decision and Control*, pages 2856–2861, 2008.

[20] Gang Liu, Haifeng Zhang, and Xinda Song. Position-estimation deviation-suppression technology of pmsm combining phase self-compensation smo and feed-forward pll. *IEEE Journal of Emerging and Selected Topics in Power Electronics*, 9(1):335–344, 2021.