POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Control of a PMSM-PMDC Back-to-Back setup

Automation and Control Laboratory
Automation and Control Engineering

Joshua Mohebban, Federico Guglielmo Mariani, Andrea Mora, Marco Nicolao, Alessandro Zanfrini

---

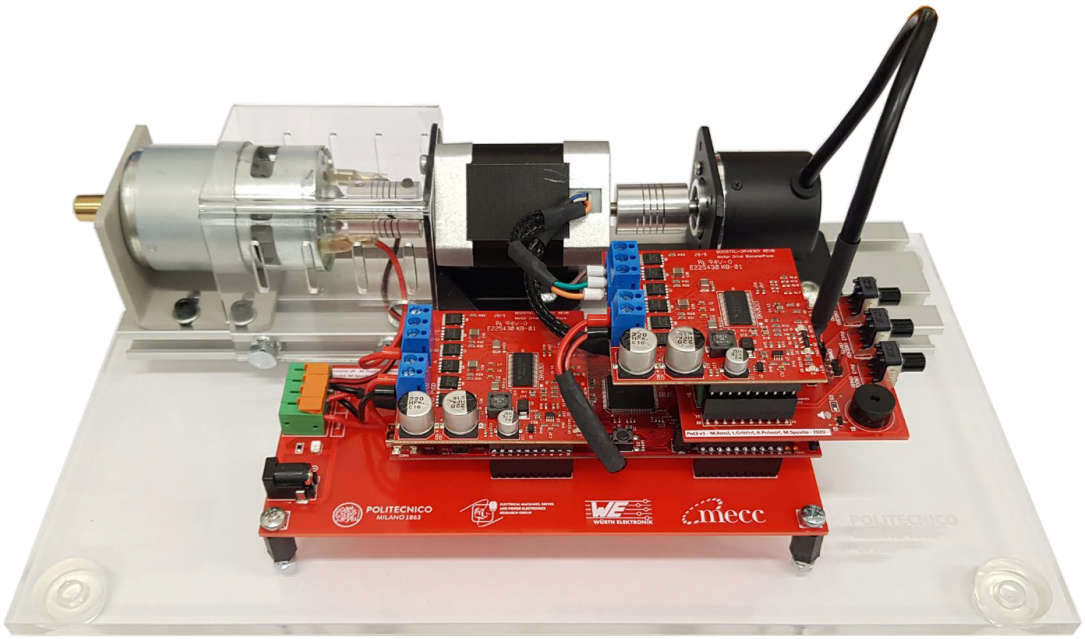
Professor:
Prof. Matteo Cazzulani

Assistants:
Dr. Matteo Pozzi
Dr. Sarper Ozturk

Academic year:
2024-2025

Key Words:
PMDC, PMSM,
Linear Quadratic Control,
Kalman Filter,
Field-Oriented Control,
Reduced Order Observer

Abstract: This report presents the development and implementation of a control system for a Permanent Magnet Synchronous Motor (PMSM) and a Permanent Magnet Direct Current (PMDC) Motor. The project, conducted in the Automation and Control Laboratory, encompasses both hardware and software aspects including motor modeling, sensor calibration, parameter identification, and advanced control strategies. The PMSM serves as the traction motor, while the PMDC operates as a braking unit, simulating load dynamics. Control strategies such as cascade PI, Linear Quadratic Regulator (LQR), and Field-Oriented Control (FOC) are implemented and evaluated. Furthermore, sensorless control techniques including Kalman filtering and reduced-order observers are explored to eliminate reliance on mechanical sensors. Experimental and simulation results validate the effectiveness of the proposed solutions.



Contents

1. Introduction

This report presents the development, modeling, and control implementation of a back-to-back Permanent Magnet Direct Current (PMDC) motor and Permanent Magnet Synchronous Motor (PMSM) system. The project was carried out aiming to provide hands-on experience with electric motor control systems, including both simulation and real-time hardware implementation.
The main objective was to design, simulate, and experimentally validate advanced control strategies for both PMDC and PMSM systems. These strategies include cascade PI control, Linear Quadratic Regulator (LQR) design, and Field-Oriented Control (FOC) techniques. Particular attention was given to sensorless control methods, such as Kalman filtering and reduced-order observers, to enhance system reliability and reduce dependency on mechanical sensors.
The report outlines the theoretical modeling of the motors, describes the experimental setup, presents the implemented control architectures, and discusses the achieved results. Challenges related to sensor calibration, parameter identification, and practical hardware limitations are also analyzed, providing a comprehensive understanding of modern electric drive systems.

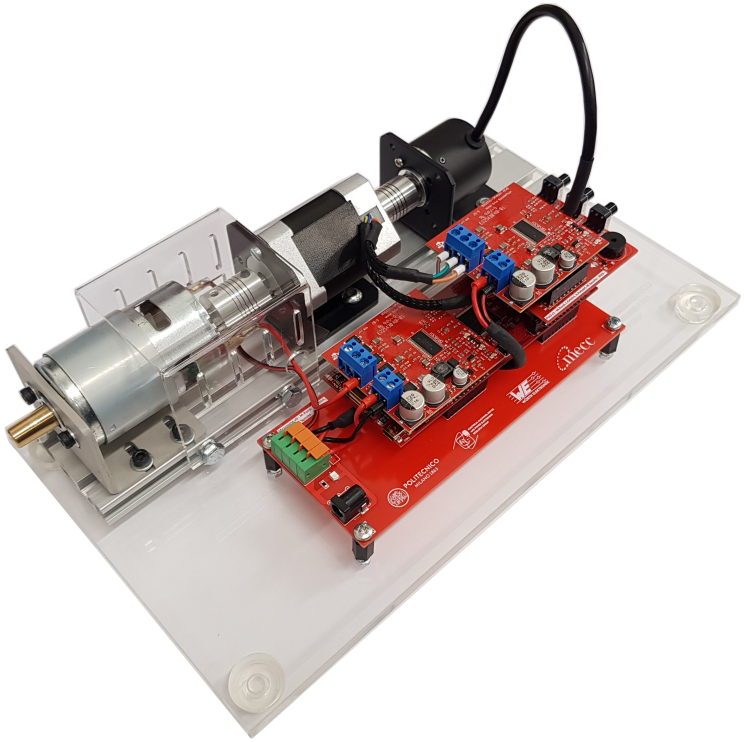


Figure 1: View of the B2B-BLDC kit and its connections [3]

2. Setup

2.1. Software Environment

The development board can be programmed directly in the MATLAB/Simulink environment. To set this up, it is required to install:

- Texas Instruments™ Code Composer™ Studio (CCS)
- Texas Instruments™ ControlSUITE™
- Embedded Coder Support Package for TI C2000™ processors

This toolchain allows the user to develop embedded software directly through Simulink's block diagram

interface. Once the design is complete, Simulink automatically converts it into optimized C/C++ code which can run standalone on the board.

2.1.1 Simulink Configuration

The configuration of Simulink was performed following the step by step procedure described at the chapter 9 of [3].

2.1.2 Firmware, Testing and External Mode

The Simulink environment generates and deploys the code to the microcontroller board enabling autonomous execution. The Simulink file designed with this purpose is referred as *Firmware*. On top of that, the PC can establish a constant communication with the board, sending and receiving data through a serial port. This is achieved by either utilizing a different Simulink file, called *Testing* or by using the *External Mode*.

Firmware

In this programming environment, the logic controlling the input/output peripherals is defined through configurable firmware. Once the firmware design is complete, the corresponding C code can be compiled and uploaded to the hardware by selecting *Build, Deploy & Start*.
To verify the correct operation of all components involved in the firmware process, an initial test was performed using a simple program to toggle the onboard blue LED.
For running the firmware, the following configuration settings were used:

- **Switching frequency** (*fsw)*: 30 kHz
- **Sampling time** (*Ts)* : [1e-4 ; 1e-5] s

Testing

After setting up the firmware environment, the Processor-in-the-Loop (PiL) technique can be used as a debugging tool. To do this, a dedicated file, *testing.slx*, must be created. In this file, a Serial Configuration block is added to establish data exchange with the host PC. The required parameters include the baud rate, communication type, and the appropriate USB port. During testing, transmitted data can be visualized using a Scope block, and specific speed reference values can be sent using the Serial Send block. However, due to the relatively low baud rate and signal noise associated with serial communication, it is preferable to use External Mode when implementing closed-loop control systems.

External Mode

For the entire project, this configuration has been adopted as a more effective alternative to PiL (Processor-in-the-Loop) simulation. The firmware file has been slightly modified by removing the SCI Transmit block and adding Scope blocks for signal monitoring. Although, using the External Mode, variables are still acquired from the hardware via serial communication, in this setup the real-time code first stores the data in memory. Data transfer to the host then occurs with lower priority, which is less critical than maintaining the strict timing requirements needed for real-time updates in a typical PIL configuration. On the other hand, code size and memory consumption are increased. The setup for External Mode is carried out as follows:

- Open the Model Configuration Parameters window and navigate to the Hardware Implementation pane. Ensure that the TI Piccolo LaunchPad™ F28069M is selected as the hardware board.
- Under Target Hardware Resources, select the External Mode menu to configure the necessary settings.
- In the Simulink editor, set the desired simulation time, then click the Monitor & Tune icon to start the real-time simulation and data exchange with the hardware.

2.2. Hardware Environment

This hardware kit consists of the following devices:

- One LaunchPad™ F28069M board;
- One Texas Instrument BOOSTXL DRV8323RS Booster Pack;
- One Texas Instrument BOOSTXL DRV8301RS Booster Pack;
- A mezzanine board to hold the MCU and manage the external power supply;
- One PMDC motor;
- One brushless DC (BLDC) motor;
- Encoder LPD3806-600BM-G5-24C;
- Aluminum base plate, motor supports and a joint.

The kit is designed to operate with a 24 V and 10 A power supply at rated conditions.

2.2.1 Half Bridge vs Full Bridge Connection

In order to be driven, the PMSM motor requires the usage of all three phases of the converter to which it is connected, due to its tri-phase nature. The PMDC's armature windings equivalent circuit can instead be seen as a simple RL circuit with only two terminals. For this reason, two possible electrical connections can be employed.

Half Bridge Configuration

In the Half-Bridge configuration, only one of the three legs of the converter is employed. Specifically, the terminals of the PMDC motor are connected between the midpoint of leg A and ground. As a result, only the ePWM1 module is required for operation in this topology. A primary limitation of the Half-Bridge configuration is that, after establishing the electrical connection, with V_{DC} connected to the positive terminal and ground to the negative terminal, the motor can only be driven only in one direction. Consequently, it is not possible to control the motor with a bipolar speed reference. However, it is much easier to drive and to control the motor in this configuration, in fact the armature voltage is given by:

$$V_a(k) = v_{a,avg}(k) = d(k)\, V_{DC} \tag{1}$$

where $d(k)$ is the duty cycle.
Notice that, although the PWM modulation technique generates a discontinuos square wave signal, the motor's armature circuit experiences a constant continuous voltage. This is due to both the converter's inherent filtering action and the motor's armature circuit itself, which attenuates the higher order harmonics of the square wave.

Full Bridge Configuration

In the Full Bridge configuration, two of the three legs of the converter are employed. The terminals of the PMDC motor are connected between the midpoints of leg A and leg B. The main advantage of the Full Bridge topology, with respect to the Half Bridge connection, is the ability to rotate the motor both in clockwise and counter-clockwise directions, without needing to change the electrical connections. However, this approach requires to generate two duty cycles, $d_a(k)$ and $d_b(k)$, to drive the legs A and B of the converter, respectively; consequently, it is required to utilize two ePWM modules. For each leg, the voltages measured from the central point to ground are

$$v_A(k) = d(k)\, V_{DC} \tag{2}$$

$$v_B(k) = d(k)\, V_{DC} \tag{3}$$

the armature voltage $V_a(k)$ is given by

$$V_a(k) = v_A(k) - v_B(k) = (d_a(k) - d_b(k))\, V_{DC} \tag{4}$$

To generate the duty cycles, the `quickstart_motor_control.slx` utilizes the *unipolar* voltage switching technique.

Unipolar Voltage Switching

Using unipolar voltage switching strategy, the armature voltage varies between $+V_{\mathrm{DC}}$ and $-V_{\mathrm{DC}}$. This is possible by forcing the duty cycle as follows:

$$d_B(k) = 1 - d_A(k) \tag{5}$$

At this point, it is possible to define $v_a(k)$ as a function of the duty cycle:

$$v_a(k) = (2d_A(k) - 1)V_{\mathrm{DC}} \tag{6}$$

The main advantage of the unipolar voltage switching method, with respect to the bipolar one, is that the the magnitude of the harmonic components of $v_a(k)$ and $i_a(k)$ is reduced; which means lower ripple and harmonic distortion, while keeping the converter switching frequency constant.

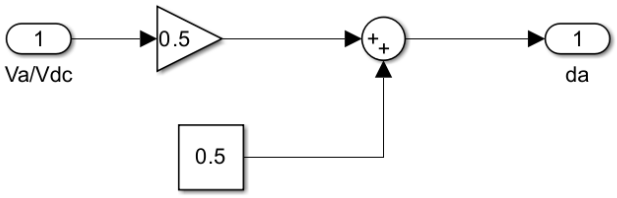


Figure 2: Unipolar voltage switching implementation

2.2.2 Pulse-Width Modulation (PWM)

Now, for what concerns the actuation of a two-level voltage source converter using a pulse-width modulation (PWM) the following clarifications are necessary:

1. The converter consists of two switches ($S1(t)$ and $S2(t)$) with anti-parallel diodes.
2. The duty-cycle is compared with triangular signal, both normalized in the [0,1] range, to generate a switching signal. $S1(t)$ is controlled directly by the PWM signal $S(t)$, $S2(t)$ is driven in a complementary way: $S2(t) = \mathrm{not}(S1(t))$.
3. This switching produces a pulsed output voltage given by:

   $$v_{\mathrm{out}}(t) = S(t) \cdot V_{\mathrm{DC}}$$

   which varies beetween 0 and V_{DC}. The fundamental component of the output matches the reference voltage, while high-frequency harmonics appear around multiples of the switching frequency f_{sw}.
4. Different types of carrier waveforms (e.g., triangular, sawtooth) can be used, but the carrier frequency must be higher than the modulation frequency. For a 2-level converter, the switching frequency equals the carrier frequency $f_{\mathrm{sw}} = f_{\mathrm{c}}$.
5. **PWM logic:**
   - If the modulation signal is less than the carrier:

     $$S(t) = 1$$

   - If the modulation signal is greater than the carrier:

     $$S(t) = 0$$

In short, Pulse Width Modulation (PWM) is particularly advantageous in power electronics for small motor applications due to its efficiency and simplicity. Specifically, discontinuous PWM reduces power losses in switching devices that operate only in two states (on/off). In addition, the inherent dynamics of the motor, modeled as an RL circuit with back-emf, acts as a low-pass filter. This characteristic helps to attenuate high-frequency harmonics generated by the PWM, reducing the need for external filtering components.

In the simulation of the PMDC control, considering a full bridge and the unipolar voltage switching configuration seen before, the PWM block was implemented as shown in Figure 3:

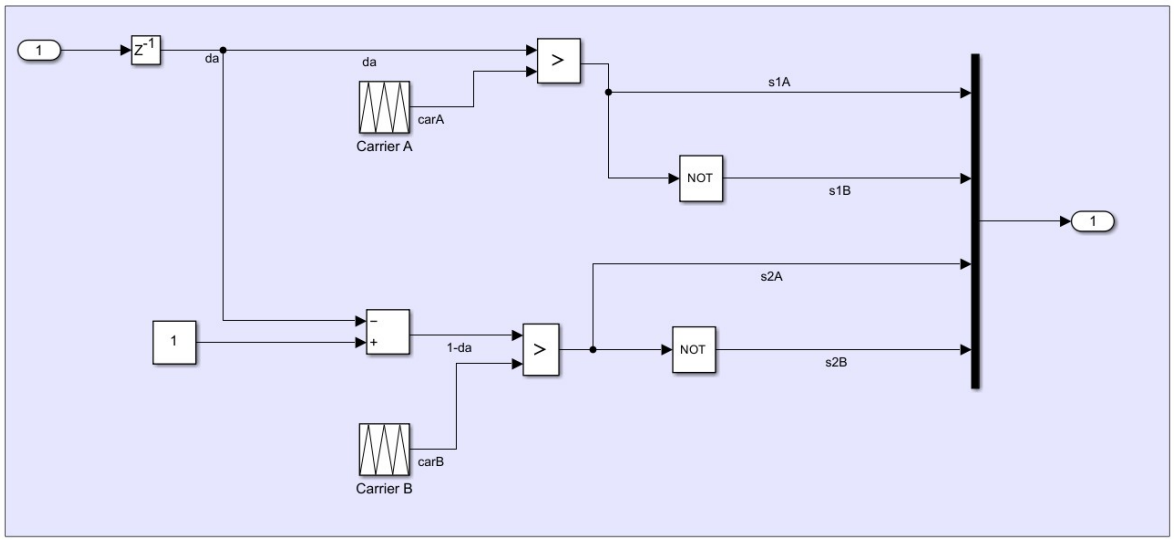


Figure 3: PWM implementation in the PMDC motor

On the other hand, for the PMSM motor, the implementation of PWM is shown in Figure 4.

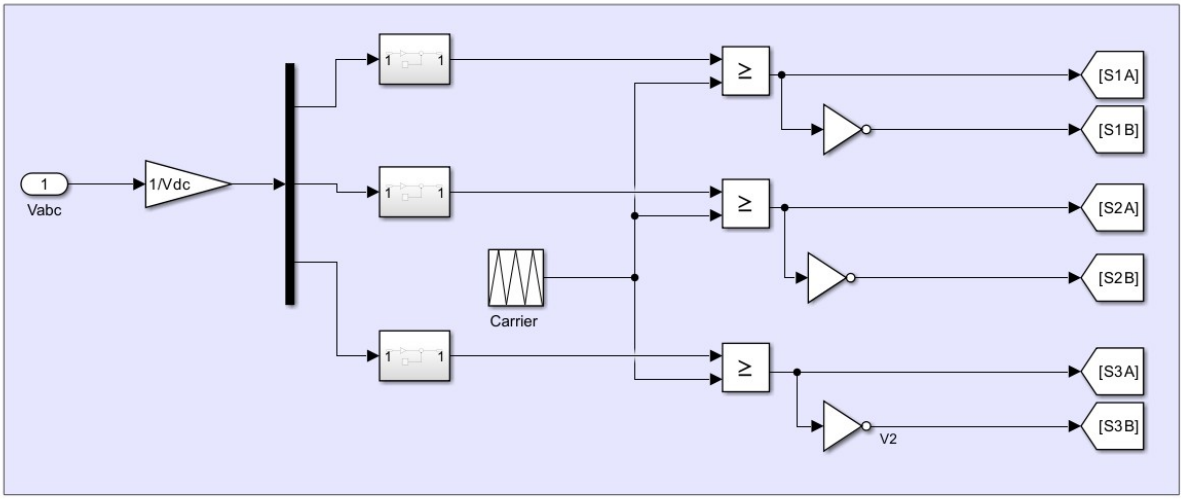


Figure 4: PWM implementation in the PMSM motor

For both simulations, a triangular carrier waveform was employed.

To conclude, the inverter models used in Simulink enviroment with unipolar PWM control for the PMDC and PMSM motors are shown in Figure 5 and 6:

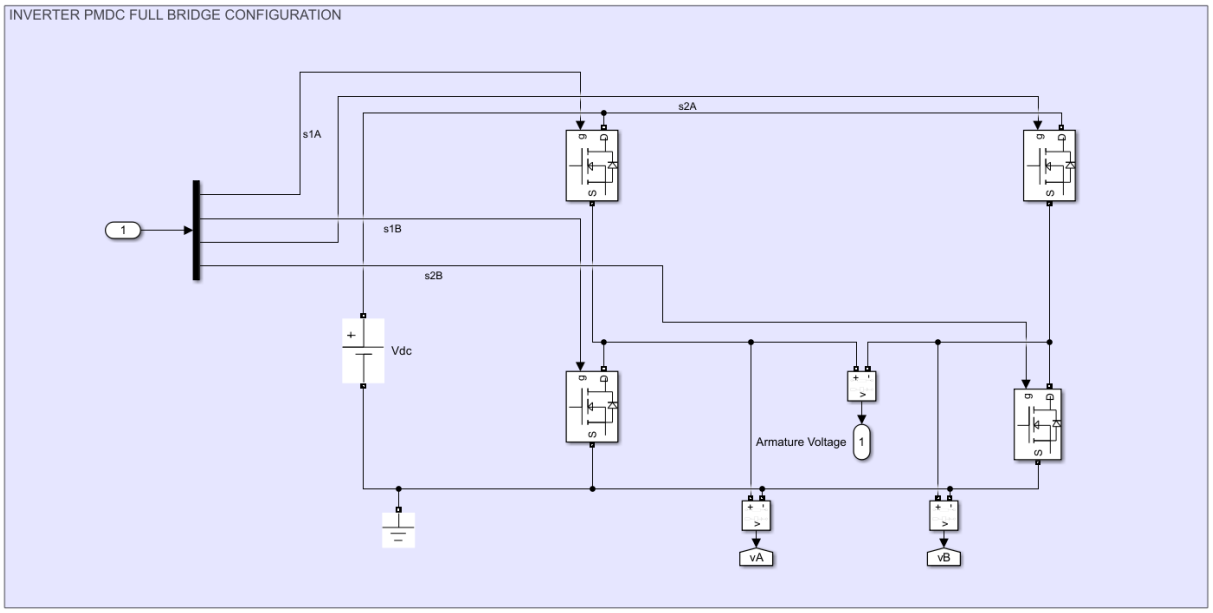


Figure 5: PMDC inverter in full-bridge configuration

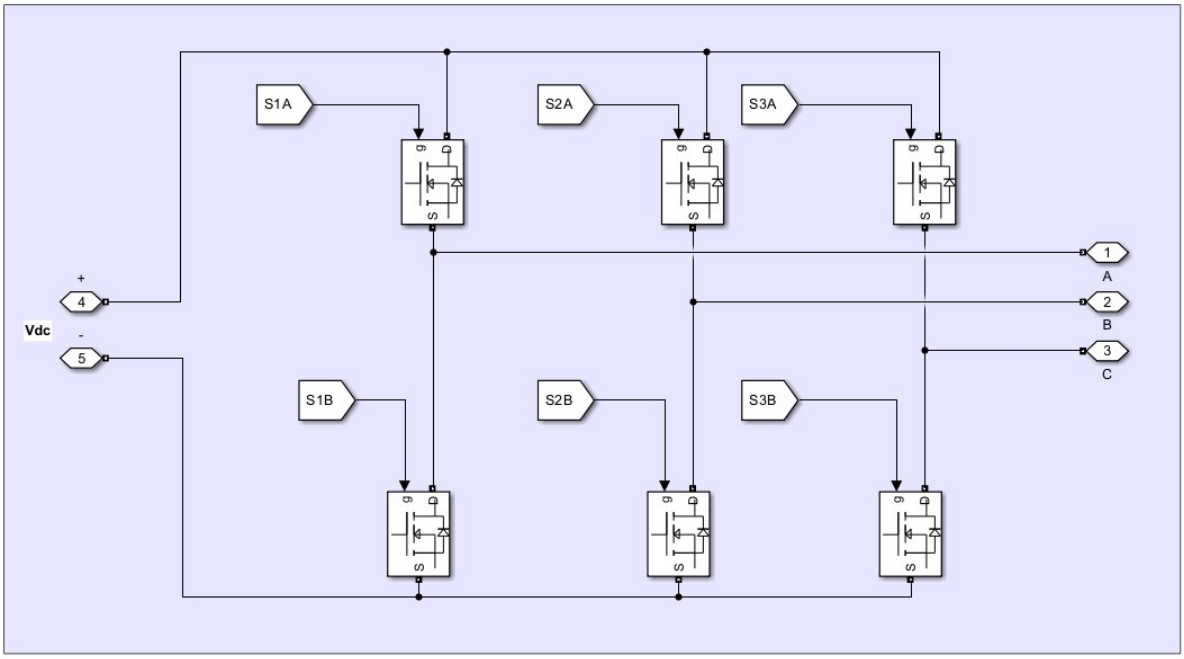


Figure 6: 3-leg inverter for PMSM

3. Sensor Characterization

In power electronics applications, the sensors embedded in standalone systems are often much cheaper and simpler than those used in test benches. Therefore, correctly selecting the parameters required to obtain accurate measurements from these sensors is a crucial step in the system setup. To support this process, more precise and specialized sensors can be used during an initial calibration phase to determine the appropriate parameters.
In our experimental setup, the BOOSTXL DRV8323RS converter includes both a voltage sensor and a low-side shunt current sensor for each phase of the inverter. In order to achieve accurate ADC conversions and consequently correct current measurements, an experimental characterization of each measurement channels is necessary.
In the following, the sensor characterization procedure that has been performed for each phase of the inverter is described, reporting the experimental offset and gain values of each measurement channel.

3.1. Setup of Current Sensing

The current sensor integrated in our converter is based on a shunt resistor positioned below the low-side MOSFET. Anytime a current flows through the shunt resistor, the voltage drop given by Ohm's first law ($V = R \cdot I$) is read by a differential operational amplifier, obtaining such an indirect measure of current. The differential amplifier is configured with a fixed gain, an internal reference voltage offset of 1.65V and a range of values for its output voltage in the interval [0V; 3.3V]. The output of the amplifier is then sent to the 12 bit ADC peripheral.
The ADC reads the voltage across the operational amplifier and outputs a number (*counts*) between 0 and 4095.
Theoretically speaking, since the current is represented by a 12-bit signed integer, the zero-current (1.65V) offset should be 2048. In order to obtain a current value, the counts number should then be multiplied by the board by a gain g_{adc}, which can be calculated from the datasheet values using the following formula:

$$g_{adc} = \frac{v_{\max} - v_{\text{ref}}}{i_{\max}} \cdot \frac{v_{\max}}{2^{12} - 1} \tag{7}$$

However, an empirical approach to calculate the offset and gain is often more reliable. The internal current sensors were calibrated by comparing the ADC readings with those obtained using an external current probe.

3.1.1 Offset Characterization

To determine the offset, the output value of the shunt resistor sensor must be read while no current flows through the motor. This condition is achieved by powering the board with a 10 V supply while applying zero voltage to the motor terminals; this can be accomplished by either imposing a 0 duty cycle in the ePWM block in Simulink or by physically disconnecting the motor from the board. In this state all current is absorbed by the board and thus none flows across the phase A's shunt resistor. The sensor's digital output represents the offset (1.65V voltage on the operational amplifier condition). It is convenient to momentarily set the gain g_{adc} in Simulink to 1, in order to be able to directly read the *counts* number. In the given board, the measured average offset was 2043 for phase A, 1430 for phase B and 1980 for phase C.

3.1.2 Gain Characterization

To characterize the gain of the ADC, the board must be connected to the power supply and a current clamp has to be set on one of the two cables linking the board to the motor, so that it could directly sense the current flowing out from the gate driver and visualize it through the oscilloscope. Five experiments were conducted by applying different voltage levels to the board while mechanically locking the rotor using a gripper. During each experiment, current signals measured via the current clamp were recorded, along with the corresponding count values displayed by the oscilloscope in Simulink. It should be noted were acquired the measurements only for positive current values, assuming a symmetric behaviour for the internal current sensor. These data points were then linearly interpolated in MATLAB®, using the

`polyval` function. The slope of the resulting characteristic curve represents the experimental ADC gain, g_{adc}, which was found to be approximately 0.0057 A/bit for phase A, 0.007 A/bit for phase B and C.

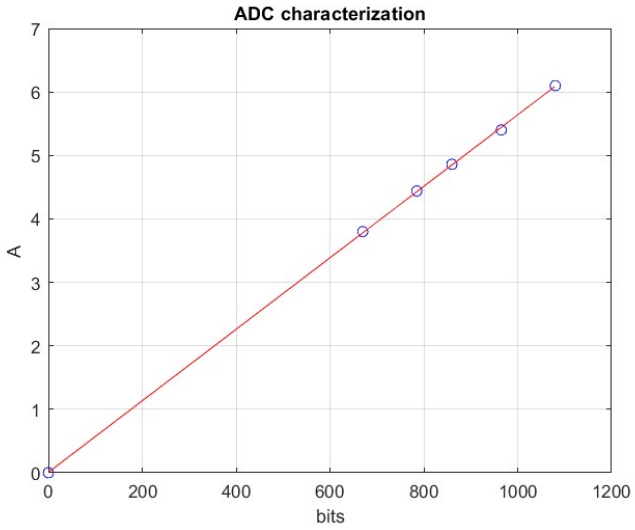


Figure 7: Plot of the fitting of the characteristic curve of the current sensor

Within the ADC block, the `int16` data type was adopted, and the sample time was set to T_s, presented in section 2.1.2. At the output of this block, an offset is subtracted from the signal. The resulting value is then directly transmitted via the serial interface, minimizing communication overhead, and simultaneously converted into a current signal to be used as feedback in the control loops.
To achieve this, the value is first cast to a `double` data type using a `Data Type Conversion` block. Subsequently, it is multiplied by the experimentally determined gain, yielding the corresponding feedback current.

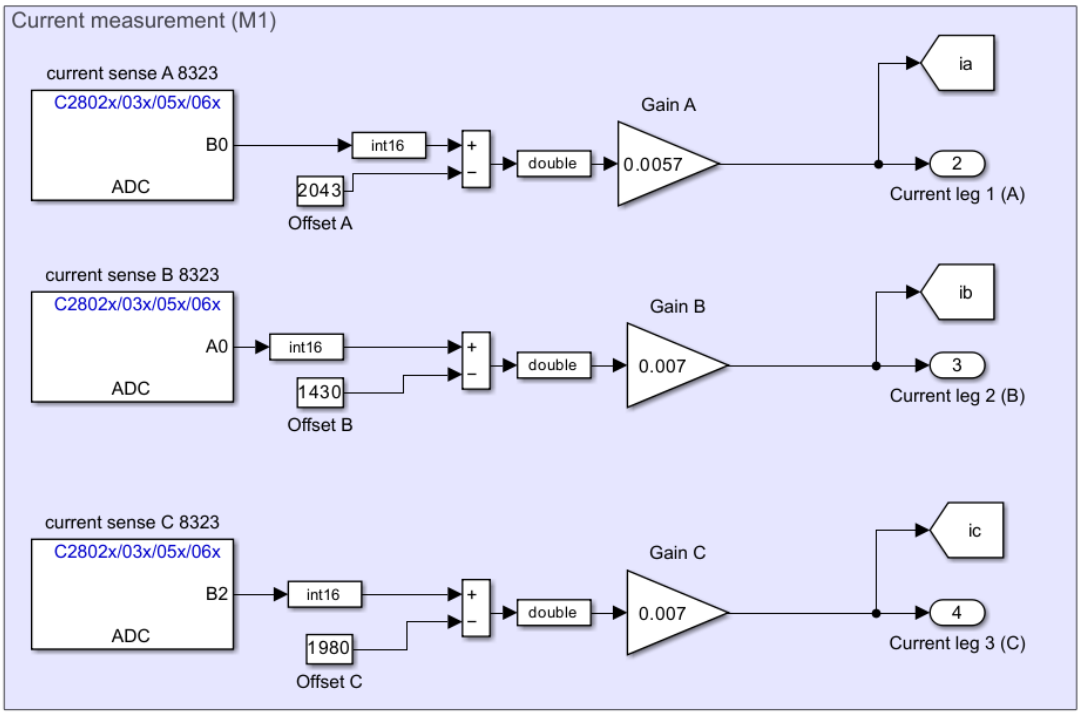


Figure 8: Overview of the offset and gain correction

3.2. Setup of Speed Sensing

To measure the angular speed of the motors, we employ the LPD3806-600BM-G5-24C **incremental encoder** (2400 cpr) and the microcontroller's eQEP2 peripheral.
Following section 14.5 of [3], we employed the `eQEP` Simulink block, then set:

- **General (Tab)** Module = `eQEP2`, Quadrature mode = X4, PositiveRotation = Clockwise, Sample time = T_w, Output position counter = ON.
- **Position counter (Tab)** Max position = $2^{32} - 1$, Enable software init = ON, Init value = 0, Reset on max = ON.
- **Signal data types (Tab)** Position counter data type = `int32`.
- **Digital filter (General Tab)** Filter length = 4 SYSCLK ticks (suppress jitter).

Then build the speed computation chain:

1. **Delay block**: Sample time = T_w, Delay length = 1.
2. **Add + Data Type Conversion**: Compute $\Delta x_p = x_p(k) - x_p(k-1)$, convert to `single`.
3. **Gain block 1**: $K_1 = \dfrac{60}{\text{cpr}\, T_w}$ converts counts/T_w into rpm.
4. **Gain block 2**: $K_2 = \dfrac{\pi}{30}$ converts rpm into rad/s.

With $T_w = 5T_s$ s = 2.5 ms and cpr = 2400, speed is updated at 400 Hz.
With this configuration, the minimum detectable speed increment is

$$\Delta\text{rpm} = \frac{60}{\text{cpr}\, T_w} = 10 \text{ rpm} \quad \left(\Delta\omega = \frac{2\pi}{\text{cpr}\, T_w} \approx 1.05 \text{ rad/s}\right),$$

defining the resolution of our speed measurement.

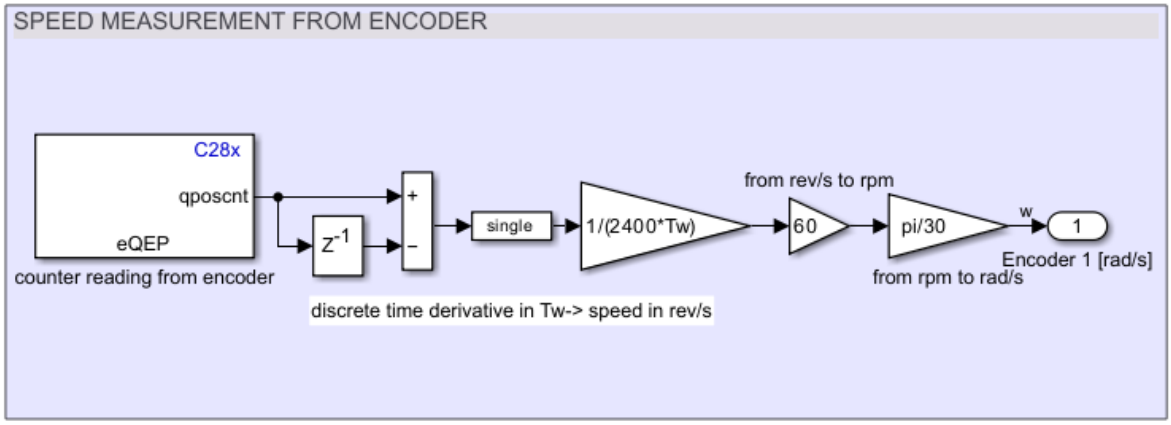


Figure 9: Simulink scheme of the speed computation

Choice of T_w

Considering a constant cpr, a large value of T_w implies a **higher speed measurement accuracy**, being able to measure down to small values of $\Delta x_p(k)$. Nevertheless, in order to keep the synchronized event sequence in the embedded target, the acquisition time must be a multiple of T_s. Hence, a large value of T_w may lead to a **quite slow speed computation task**, adding a relevant delay in the whole cascaded control action (e.g., the duty cycles tend to saturate for long periods). In the worst-case scenario this may lead to system instability.
Based on this fact, even if a sufficiently large value of T_w is necessary to well approximate the actual speed, it is quite often recommended to keep T_w in the $[2T_s, 10T_s]$ range for switching frequencies below 30 kHz [3]. Ultimately, we set $T_w = 5T_s$.

4. Model and Structure

4.1. Permanent Magnet DC Motor (PMDC)

4.1.1 Structure

The electro-mechanical structure of a **PMDC motor** consists of an external **stator** with permanent magnets that create the magnetic flux, and an internal **rotor**, as represented in Figure 10:

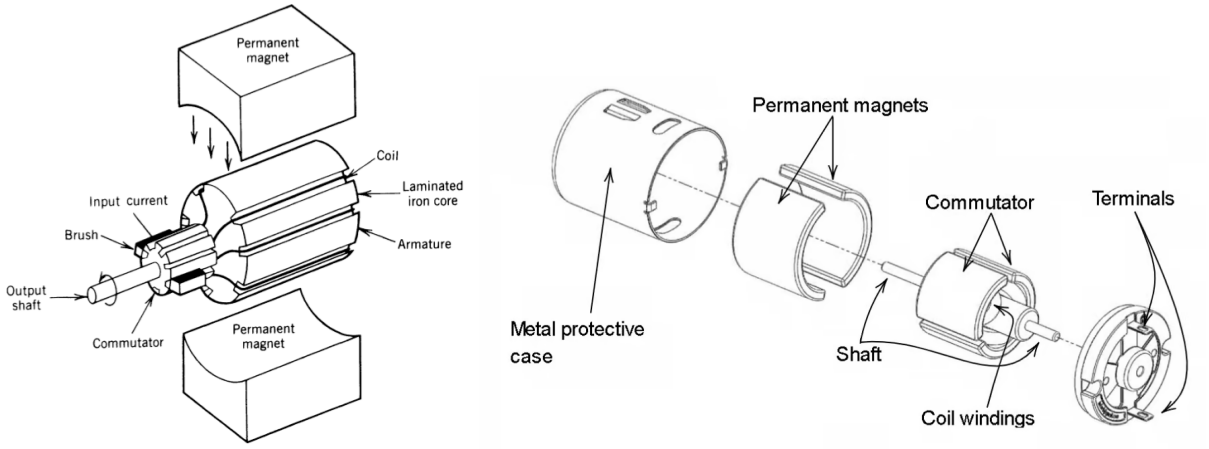


Figure 10: PMDC structure

The rotor is composed by a cylinder made of laminated ferromagnetic material with distributed slots (around which a coil is wound) and a **commutator**, connected to this coil. The role of the commutator is to keep the **armature current** flowing always in the same direction, while the continuity of the connection is guaranteed by the brushes [2].
The PMs create a constant **flux density** B and, accordingly, a constant excitation flux defined as ϕ_{PM}.

4.1.2 Electrical and Mechanical Model

The basic equations describing the behavior of the motor can be derived by resorting to an elementary machine composed by a single-turn armature winding rotating on itself under the action of a constant magnetic field.

$$v_a = R_a\, i_a + L_a \frac{di_a}{dt} + E \tag{8}$$

$$E = k_e\, \phi_{PM}\, \Omega \tag{9}$$

$$m_e - m_l = \beta\, \Omega + J \frac{d\Omega}{dt} \tag{10}$$

$$m_e = k_T\, \phi_{PM}\, i_a \tag{11}$$

where:

- i_a is the armature current (i.e., the current flowing through the winding of the motor).
- v_a is the armature voltage (i.e., the voltage applied to the terminals of the machine).
- E is the back electromotive force (back-emf) induced on the armature winding.
- R_a is the armature resistance.
- L_a is the armature inductance.
- k_e is the back-emf constant.
- k_T is the torque constant.
- ϕ_{PM} is the excitation flux generated by the permanent magnets.

- Ω is the rotational speed of the motor.
- m_e is the rotor torque.
- m_l is the resisting (or load) torque.
- β is the viscous damping constant.
- J is the inertia seen from the rotor.

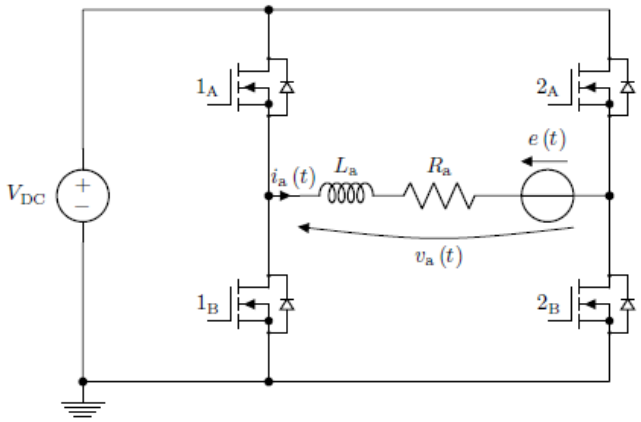


Figure 11: Equivalent circuit of a PMDC motor driven by a Full-Bridge topology converter

In DC machines, both electromagnetic torque m_e and back-emf E depend on the magnetic flux linked with the armature windings ϕ_{PM} and the related constants k_T and k_e. Moreover, k_T and k_e can be approximately considered equal, assuming that the energy conversion is almost perfect. Thus, a new constant **K** can be defined as

$$K = \phi_{PM} \cdot k_T = \phi_{PM} \cdot k_e.$$

Therefore, the electrical torque and the back-emf can be computed as:

$$E = K\,\Omega \tag{12}$$

$$m_e = K\,i_a \tag{13}$$

4.1.3 State-Space Form

In control theory a lot of techniques are designed especially for linear systems. In order to use them, it is necessary to elaborate the continuous time system into a state space form. The voltage v_a is a controllable variable, i.e. the input $u(t)$, which allows to increase/decrease the current flowing into the PMDC motor (according to the system parameters). The torque load m_l is considered as a disturbance, while i_a and Ω are the state variables. In this way, the model can be represented in state-space form as:

$$\frac{d}{dt}\begin{bmatrix}\Omega(t)\\ i_a(t)\end{bmatrix} = \begin{bmatrix}-\beta/J & K/J\\ -K/L_a & -R_a/L_a\end{bmatrix}\begin{bmatrix}\Omega(t)\\ i_a(t)\end{bmatrix} + \begin{bmatrix}0\\ 1/L_a\end{bmatrix} v_a(t) + \begin{bmatrix}-1/J\\ 0\end{bmatrix} m_l(t), \tag{14}$$

$$\begin{bmatrix}\Omega(t)\\ i_a(t)\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\Omega(t)\\ i_a(t)\end{bmatrix}. \tag{15}$$

4.2. Permanent Magnet Synchronous Machine (PMSM)

4.2.1 Structure

The Permanent Magnet Synchronous Machine (PMSM) is a three-phase synchronous motor in which the rotor magnetic field is generated by permanent magnets, eliminating the need for external excitation. Its structure includes:

- **Stator**: equipped with three-phase distributed windings, similar to those in an induction motor. When powered by a balanced three-phase voltage, these windings generate a rotating magnetic field.
- **Rotor**: contains permanent magnets —typically rare-earth types— either mounted on the surface or embedded inside the rotor structure. These magnets provide a constant magnetic field without requiring current supply.

PMSMs can be further classified based on rotor construction:

- **Surface-Mounted PMSM (SPMSM)**: Magnets are placed on the rotor surface. This results in a non-salient rotor, i.e., the inductances in the direct and quadrature axes are approximately equal ($L_d \approx L_q$).
- **Interior PMSM (IPMSM)**: Magnets are embedded inside the rotor, producing magnetic saliency ($L_d \neq L_q$). This configuration enables additional reluctance torque production.

4.2.2 Electrical Model in the *abc* Frame

The stator voltage equations in the stationary three-phase (abc) reference frame are:

$$\begin{aligned} v_a &= R_s i_a + \frac{d\psi_a}{dt} \\ v_b &= R_s i_b + \frac{d\psi_b}{dt} \\ v_c &= R_s i_c + \frac{d\psi_c}{dt} \end{aligned} \tag{16}$$

where:

- $v_{a,b,c}$: stator phase voltages,
- $i_{a,b,c}$: stator phase currents,
- R_s: stator winding resistance,
- $\psi_{a,b,c}$: stator flux linkages.

Assuming a linear magnetic system, the flux linkages can be modeled as:

$$\begin{aligned} \psi_a &= L_s i_a + M(i_b + i_c) + \psi_{m,a}(\theta) \\ \psi_b &= L_s i_b + M(i_a + i_c) + \psi_{m,b}(\theta) \\ \psi_c &= L_s i_c + M(i_a + i_b) + \psi_{m,c}(\theta) \end{aligned} \tag{17}$$

where:

- L_s: self-inductance,
- M: mutual inductance,
- $\psi_{m,x}(\theta)$: magnet-induced flux linkage in phase x, dependent on rotor position θ.

In a balanced system, the total magnet-induced flux linkage satisfies:

$$\psi_{m,a} + \psi_{m,b} + \psi_{m,c} = 0$$

The voltage equations can also be rewritten in terms of back-EMF $e_x(\theta)$:

$$v_x = R_s i_x + L_s \frac{di_x}{dt} + e_x(\theta), \quad x \in \{a, b, c\} \tag{18}$$

Figure 12 illustrates the three-phase representation.

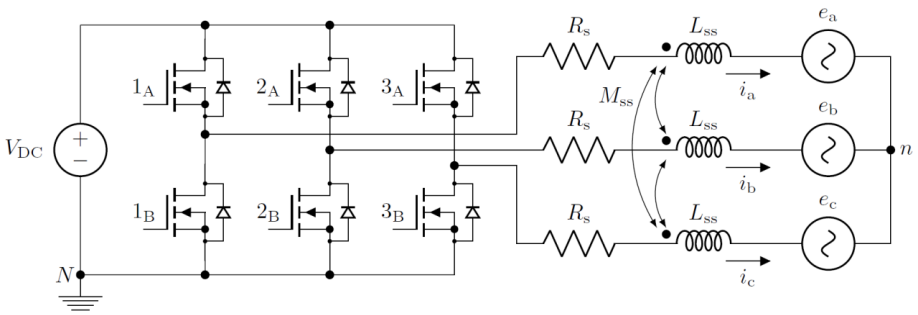


Figure 12: Equivalent circuit of a PMSM motor driven by the converter

4.2.3 Mechanical Model

The mechanical dynamics of the rotor are described by:

$$m_e - m_l = \beta\Omega + J\frac{d\Omega}{dt} \tag{19}$$

where:

- m_e: electromagnetic torque,
- m_l: external load torque,
- Ω: mechanical angular speed,
- J: moment of inertia,
- β: viscous friction coefficient.

The electromagnetic torque is given by:

$$m_e = \frac{3}{2}p(\psi_a i_a + \psi_b i_b + \psi_c i_c) \tag{20}$$

with p being the number of pole pairs.

4.2.4 Electrical Model in the dq Frame

To enable efficient control of the PMSM, particularly via Field Oriented Control (FOC), the three-phase (abc) system is typically transformed into a rotating two-axis (dq) reference frame using the Park Transformation.

- The **d-axis** aligns with the rotor flux.
- The **q-axis** leads the d-axis by 90° electrically.

In this frame:

- i_d controls the flux.
- i_q controls the torque.

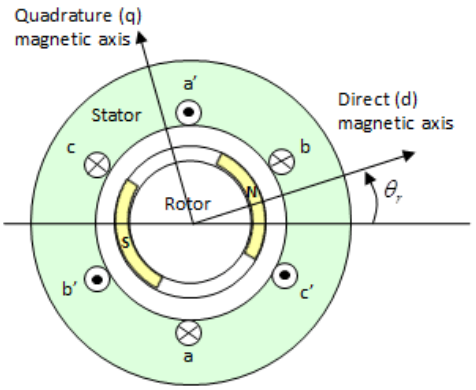


Figure 13: Rotating dq reference frame aligned with rotor flux

The stator voltage equations in the dq frame are:

$$v_d = R_s i_d + L_d \frac{di_d}{dt} - \omega_e L_q i_q \tag{21}$$

$$v_q = R_s i_q + L_q \frac{di_q}{dt} + \omega_e (L_d i_d + \phi_{PM}) \tag{22}$$

where:

- $v_{d,q}$: voltages in the dq frame,
- $i_{d,q}$: currents in the dq frame,
- $L_{d,q}$: inductances in the d and q axes,
- ϕ_{PM}: flux linkage from the magnets,
- $\omega_e = p\Omega$: electrical angular speed.

The electromagnetic torque becomes:

$$m_e = \frac{3}{2} p \left(\phi_{PM} i_q + (L_d - L_q) i_d i_q\right) \tag{23}$$

Note: For SPMSM, where $L_d \approx L_q$, the torque simplifies to:

$$m_e = \frac{3}{2} p \phi_{PM} i_q \tag{24}$$

4.3. B2B Configuration

The complete system comprehends one PMSM and one PMDC motor arranged in a B2B (Back-to-Back) configuration (see Figure 1). The two motors are mechanically coupled, with one of them behaving as a traction motor, while the other as a braking one and they are separately controlled and supplied by two separate converters.
Assuming that the PMSM is tracking a speed reference (full cascade structure) while the PMDC motor is aimed to regulate a friction/braking torque applied to the shaft (i.e., armature current control only), the controlled B2B system is equivalent to a cascade control of a PMSM (traction) motor subject to a load torque that can dynamically vary in time.
Therefore, the dynamic equation of the whole setup can be considered as follows:

$$m_{e,a} - m_{e,b} = \beta_{eq} \cdot \Omega + J_{eq} \cdot \frac{d\Omega}{dt} \tag{25}$$

with $m_{e,a}$ and $m_{e,b}$ the electrical torques of the traction motor and of the braking motor, respectively.

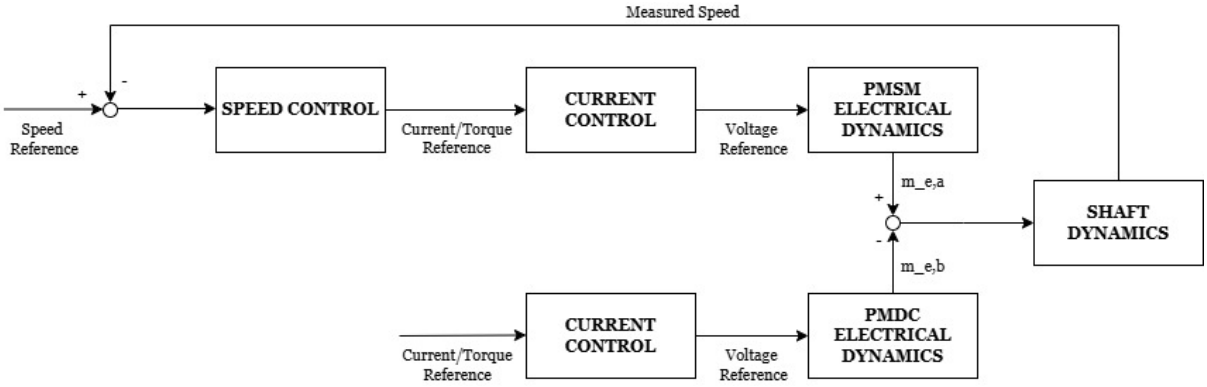


Figure 14: Sketch of B2B Control Scheme

5. Parameter Identification

The accuracy of the dynamic models for both the PMDC and PMSM machines critically depends on precise knowledge of their electrical and mechanical parameters. Therefore, a two-step identification procedure was carried out, comprising:

1. **Identification of electrical parameters**: determination of armature (stator) resistance and inductance, and electrical constant.
2. **Identification of mechanical parameters**: estimation of inertia and viscous friction.

5.1. Experimental Setup and Tools

To ensure the highest measurement accuracy, an **oscilloscope with current and voltage probes** was used instead of a standard multimeter. Specifically, a PicoScope 2204A-D2 oscilloscope was employed, paired with a custom 30 A current probe configured with a sensitivity of 100 mV/A. To visualize and analyze the signals, the PicoScope 7 T&M software was installed and used for data acquisition and signal assessment.
The measurement setup involved connecting both current and voltage probes to the PicoScope. The voltage probe alligator clips were attached directly to the motor cable terminals (for both PMDC and PMSM motors), while the current probe was clamped around one of the motor cables, ensuring that the correct direction of current flow was verified. A voltage input was then applied using a power supply by toggling it on and off, allowing observation of both the transient response and steady-state behavior of the signals.

Figure 15: Setup of the measurement process

5.2. Electrical Parameters

5.2.1 PMDC Motor

Armature resistance and inductance identification was performed using the **locked-rotor test**. With the rotor mechanically fixed as shown in Figure 16, a voltage step v_s was applied to the motor terminals by means of the power supply.

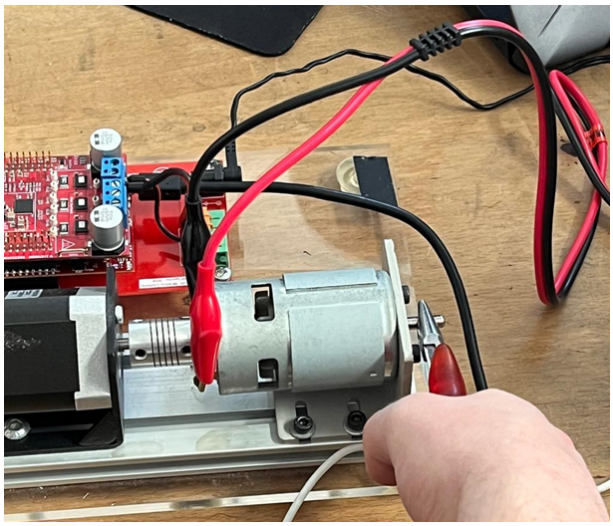


Figure 16: Locked Rotor Test - PMDC setup

In this way, since $\Omega = 0$, the equation 8 becomes $v_s = R_a \cdot i_a + L_a \cdot \frac{di_a}{dt}$
The resulting current transient $i_a(t)$ was recorded by means of an oscilloscope and filtered.
From the steady-state current $i_a(\infty)$, the resistance was calculated as

$$R_a = \frac{v_s}{i_a(\infty)}$$

The inductance was computed deriving the electrical time constant τ_{el} by identifying current value equal to 63.2 % value of steady state value.
Therefore the value of the inductance was obtained as:

$$L_a = R_a \cdot \tau_{el}.$$

In addition, the obtained value was validated fitting the step response value with *System Identification Toolbox* from Matlab.
These experiments were led using 5 values of voltages steps in between 1.5 V and 3 V.

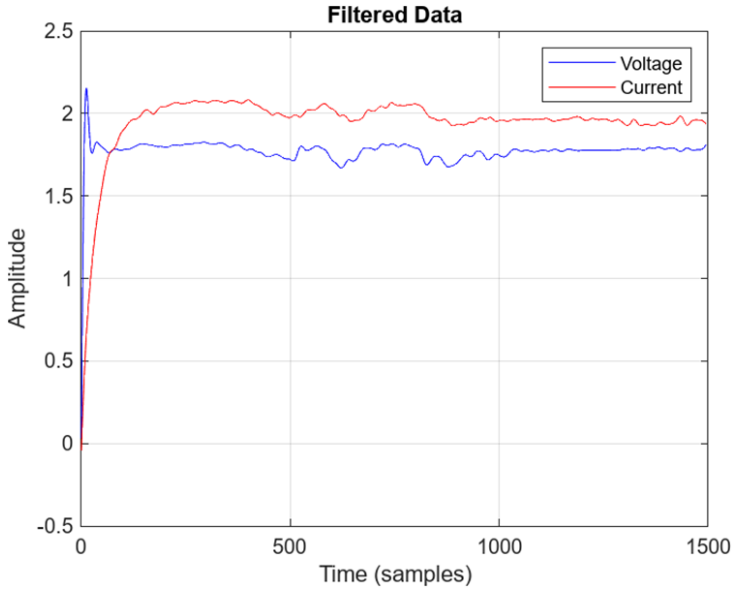


Figure 17: Example of Locked Rotor Filtered Step Response

Furthermore, **the torque/back-emf constant** K (Equations (12) and (13)) was identified through the **no-load test**.
The motor was driven by means of the power supply at four different voltage inputs between 4 V and 10 V. The steady speed Ω was measured by the encoder and the current through an oscilloscope. Since at steady state the derivative of the current is equal to zero, the torque/back-emf constant K is given by:

$$K = \frac{v_s - R_a \cdot i_a}{\Omega}.$$

The average results obtained for the PMDC motor electrical parameters were as follows:

R_a	0.75 Ω
L_a	1.2 mH
K	0.028 Vrad/s

5.2.2 PMSM

Phase resistance and phase inductance identification followed a similar approach to the one applied for the identification of armature resistance and inductance of the DC motor. With the rotor once again locked as shown in Figure **??**, a voltage step was applied to two phases of the three of the PMSM motor by means of a power supply. Since a constant DC voltage was applied to two legs the equivalent circuit is composed by: Vdc, 2 in series phase resistances R_s, 2 in series self-inductances L_s and two contributes given by two back-EMFs.
Due to the Locked-Rotor $\omega_e = 0 \rightarrow \theta = 0 \rightarrow e_x(\theta) = 0$, equation 18 between two legs (i.e. legs a and b) becomes:

$$v_{ab} = 2R_s i_{ab} + 2L_s \frac{di_{ab}}{dt} \tag{26}$$

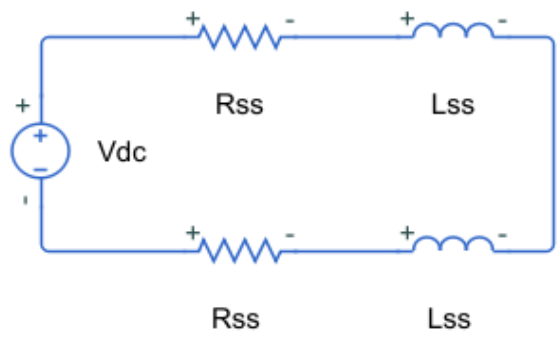


Figure 18: Equivalent circuit for parameter identification of Rs and Ls

Notice that Equation 26 is similar to the equation used in the identification of the DC motor parameters, thus a similar approach can be used to estimate R_s and L_s. Experiments were conducted between all possible combinations for legs A, B, C with voltages steps ranging from 2 to 3 volts in order to not excitate the circuits too much.
Being the motor a Surface-Mounted PMSM, it is assumed that the inductance along the d-axis is equivalent to the one along the q-axis, thus both having the same value equal to the self-inductance L_s.

The number of pole couples and the torque constant were identified by supplying in open loop the PMDC motor with different voltages, using the Simulink program provided. Utilizing the *Pico Oscilloscope*, the peak-peak voltage drops between the two ports of the Brushless Motor were measured. Lastly, the V_{rms} values were calculated from the V_{pp} values, and the frequency of the sinusoidal wave was directly measured from the *Pico*.

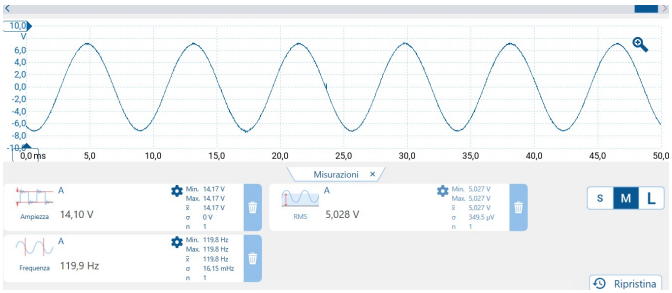


Figure 19: Back-EMF waveform between phases A and B measured at rotor speed $\omega = 188 rad/s$

Once the frequency and velocity of the shaft are measured, the amount of pole pairs can be estimated as $n_p = \frac{2f\pi}{\omega}$; the back-emf constant can be calculated with different unit measures, if we define it as $[\frac{V_{rms}s}{rad}]$ it can be obtained by the following formula $K_{e,rms} = \frac{V_{rms}}{\omega}$. If calculated in SI units K_t and K_e have the same absolute value so $K_{e,rms}$ can then be converted in $K_{t,pp}[\frac{Nm}{A}]$ by: $K_{t,pp}[\frac{Nm}{A}] = K_{e,rms}[\frac{V_{rms}s}{rad}]\sqrt{2}$.
The average results obtained for the PMSM motor electrical parameters were as it follows:

R_s	0.40 Ω
L_s	0.6 mH
n_p	4 Pole Pairs
K_t	0.037 Nm/A

5.3. Mechanical Parameters

Since the two motors are mechanically coupled, an equivalent inertia and friction coefficient can be associated to the whole system (i.e., the two motors and the mechanical joint), following the Equation 25. These parameteres were identified via a **no-load test**, supplying directly the PMDC motor (with the same set of voltage inputs used for the identification of the back-emf constant) and keeping the PMSM inactive so to not have any external load.
At steady state, the electromagnetic torque produced by the PMDC equals the viscous friction torque:

$$m_{e,b} = K\, i_a = \beta_{\text{eq}}\, \Omega,$$

hence

$$\beta_{\text{eq}} = \frac{K\, i_a}{\Omega}. \tag{27}$$

In practice, i_a was measured through an oscilloscope and Ω through the encoder.
The mechanical time constant τ_m is defined as the time required for the speed to reach 63.2% of its final value in response to a step in applied voltage (no-load). From the speed transient:

$$\Omega(t) = \Omega(\infty)\Big(1 - e^{-t/\tau_m}\Big),$$

the time constant τ_m is obtained by finding t such that $\Omega(t) = 0.632\,\Omega(\infty)$. Then, the equivalent inertia follows from:

$$J_{\text{eq}} = \tau_m\, \beta_{\text{eq}}. \tag{28}$$

It should be noted that the purely viscous friction model used in the motors' equations is an approximation. In reality, friction exhibits a residual torque at low speeds that the linear term alone cannot capture and consequently the measured drag tends to be higher than $\beta_{\text{eq}}\,\Omega$, when Ω is small. Indeed, the obtained measurements of β tended to oscillate, so we decided to consider an average value.
The average results obtained for the dynamic parameters were as follows:

β_{eq}	$1.2 \cdot 10^{-4}$ N m s
J_{eq}	$4.6 \cdot 10^{-6}$ Kg m^2

6. Control Implementation

This chapter presents an overview of the control strategies implemented for electric motors, focusing primarily on Permanent Magnet DC (PMDC) motors and Permanent Magnet Synchronous Motors (PMSM). In Section (6.1) a PI cascade control based on nested loops is proposed for the PMDC motor. As an alternative, the Linear Quadratic Regulator (LQR) is introduced, providing an optimal control solution for a comparison with the classical PI approach.
Section (6.2) shifts the focus to the PMSM motor, where the advanced Field Oriented Control (FOC) technique is employed. These implementations were made possible through the use of physical sensors (see chapter 3) which allowed a validation of the simulated models in a practical hardware setup.

6.1. PMDC Motor

In Figure 20 is shown the Simulink implementation of the simulation model of the motor. As already anticipated in Section 2.2.2, the PWM and the Inverter were built by scratch, while the model of the PMDC was implemented using the transfer function blocks.

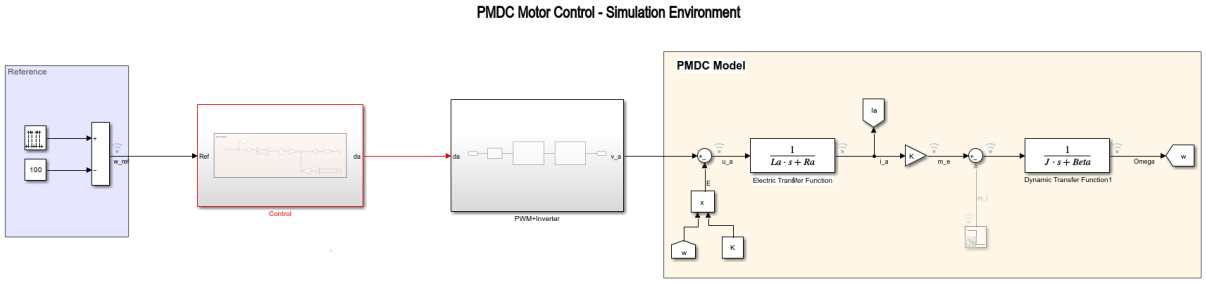


Figure 20: PMDC Control Simulink Implementation - Simulation Environment

In Figures 21 and 22 the Simulink implementation for the experimental environment is shown. The *Reference* and *Industrial Plant* subsystems stay the same for all the controls that have been implemented, meanwhile the *Control* subsystem changes based on what implementation has been used.

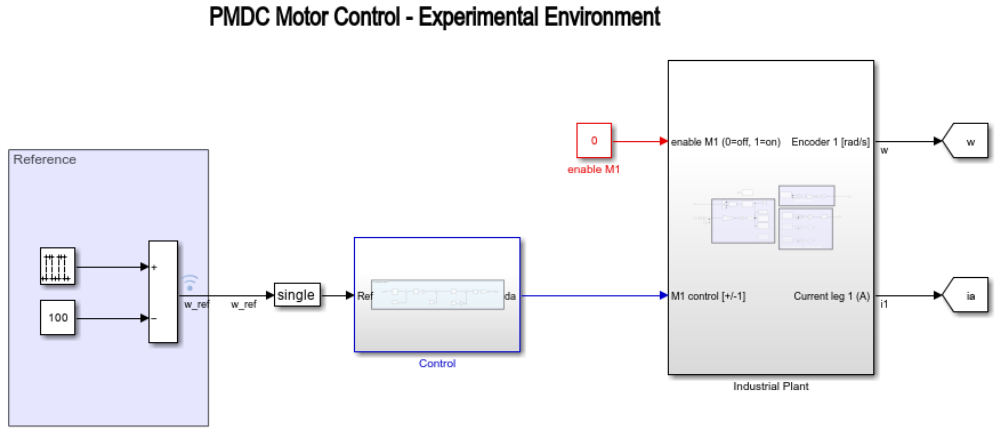


Figure 21: PMDC Control Simulink Implementation - Experimental Environment

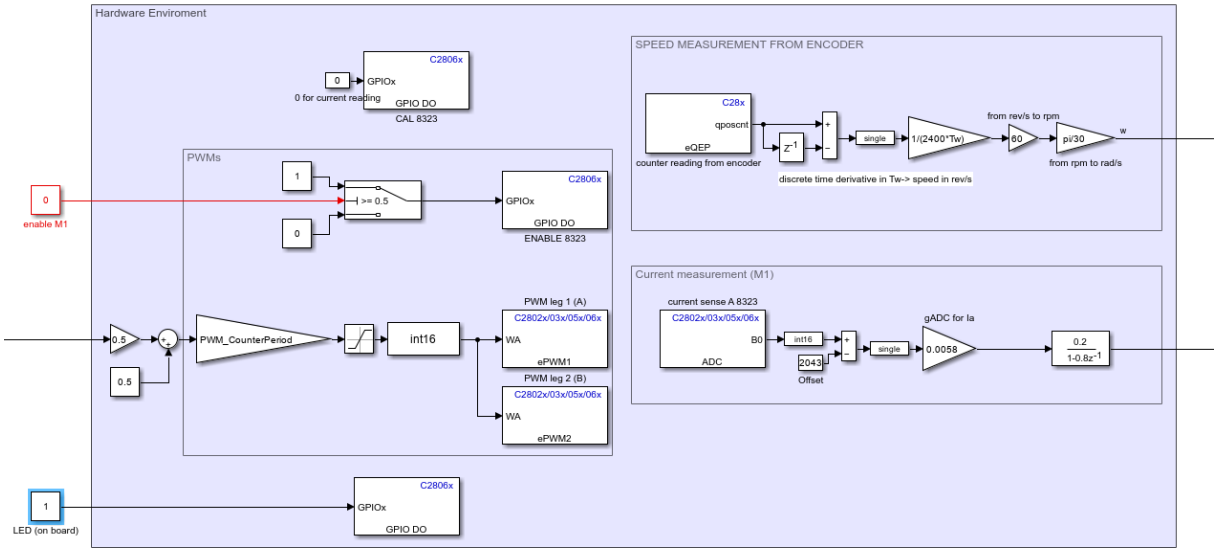


Figure 22: PMDC Control Interface to board

6.1.1 PI Cascade Control

Proportional-Integral (PI) control is one of the most widely used strategies in industrial control systems due to its simplicity, effectiveness, and robustness. In the context of Permanent Magnet DC (PMDC) motor control, a common and effective structure is the **cascade PI controller**, which employs two nested feedback loops: an inner loop regulating the motor current, and an outer loop controlling the motor speed. This structure exploits the different dynamics of the electrical and mechanical subsystems of the motor, ensuring both fast current regulation and accurate speed tracking.

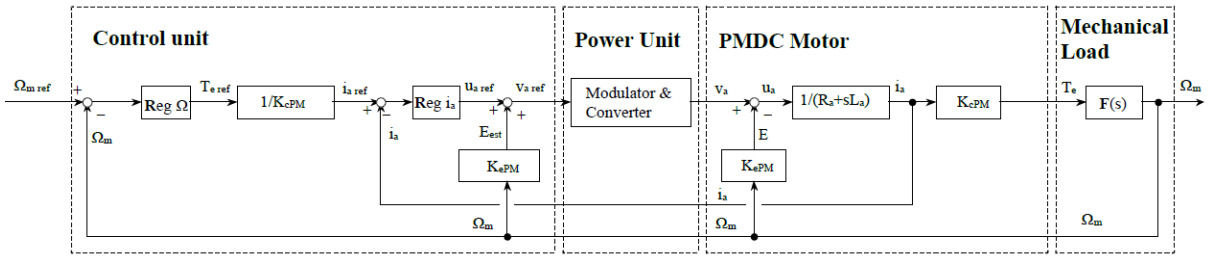


Figure 23: Speed control scheme of a PM DC machine

The general form of a PI controller in the Laplace domain is:

$$R(s) = k_p + \frac{k_i}{s}$$

where k_p and k_i are the proportional and integral gains, respectively. The PI controller generates a control action based on both the present error (via the proportional term) and the accumulation of past error (via the integral term), enabling effective disturbance rejection and zero steady-state error for step inputs.

The PMDC motor can be modeled by separating its dynamics into two parts:

- Electrical subsystem: $G_{\text{el}}(s) = \dfrac{1}{R_a + sL_a}$
- Mechanical subsystem: $G_{\text{mec}}(s) = \dfrac{1}{\beta + sJ}$

Here, R_a and L_a are the armature resistance and inductance, respectively, while β and J represent the viscous friction and moment of inertia of the rotor. Since electrical dynamics are faster, the inner loop is designed first.
Considering the electrical transfer function, can be rewritten in standard form as:

$$G_{\text{el}}(s) = \frac{1/R_a}{1 + s\tau}, \quad \tau = \frac{L_a}{R_a}$$

Designing a PI controller for this system, the open-loop transfer function becomes:

$$L(s) = R(s)G_{\text{el}}(s) = \left(k_p + \frac{k_i}{s}\right) \cdot \frac{1/R_a}{1 + s\tau}$$

To simplify the design, a **pole-zero cancellation** can be applied by setting $\frac{k_p}{k_i} = \tau = \frac{L_a}{R_a}$, resulting in:

$$L(s) = \frac{k_p}{R_a} \cdot \frac{1 + \frac{k_i}{k_p} s^{-1}}{1 + s\tau} \approx \frac{k_p}{R_a} \cdot \frac{1}{s}$$

This approximates an integrator with a phase margin close to 90°, ensuring good stability. The cutoff frequency of the current loop is then:

$$\omega_c = \frac{k_i}{R_a} = \frac{k_p}{L_a}$$

Given a desired settling time T, the time constant is $\tau = T/5$. Thus, the cutoff frequency is:

$$\omega_c = \frac{1}{\tau} = \frac{5}{T}$$

and the PI gains are computed as:

$$k_{p,c} = \omega_c L_a, \quad k_{i,c} = \omega_c R_a$$

The speed controller is designed around the mechanical transfer function:

$$G_{\text{mec}}(s) = \frac{1}{\beta + sJ}$$

Following a similar reasoning, the PI gains for the speed loop are:

$$k_{p,m} = \omega_m J, \quad k_{i,m} = \omega_m \beta$$

The mechanical loop should be at least one decade slower than the electrical loop to ensure a clear separation of time scales.
The chosen bandwidth for the external and internal loop were:

ω_c	1200 rad/s
ω_m	$100 rad/s$

which leads to the following controllers:

$k_{p,c}$	1.44
$k_{i,c}$	900
$k_{p,m}$	1.38e-4
$k_{i,m}$	3.6e-3

Furthermore, as dealing with microcontroller-based systems, several practical aspects influence the implementation:

- **Sampling frequency**: the electrical loop must be updated at a much higher frequency (typically tens of kHz) than the mechanical loop (hundreds of Hz).
- **Anti-windup**: integrators in both PI controllers must include anti-windup logic to prevent excessive accumulation of error during actuator saturation.
- **Fixed-point precision**: is required scaling of gains.
- **Noise filtering**: since the derivative term of a PID controller is often omitted due to sensitivity to high-frequency measurement noise, low-pass filters are recommended for current and speed sensing [3].
- **Discrete time domain**: the microcontroller works in the discrete time domain. Consequently, the PI controllers must be discretized; this was achieved by employing the *Forward-Euler* discretization technique.

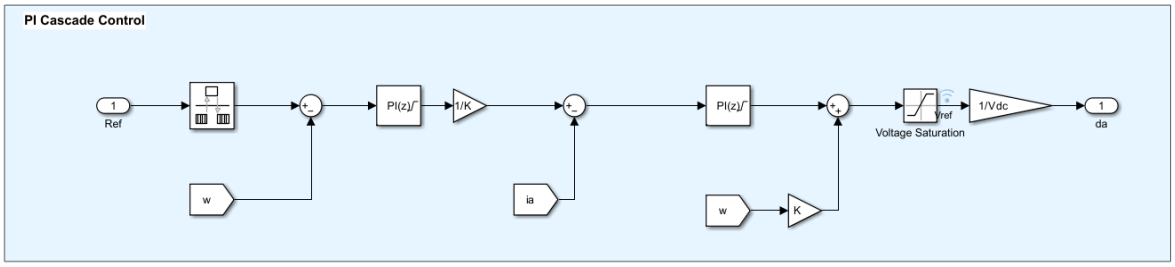


Figure 24: Cascade Control Scheme Simulink Implementation

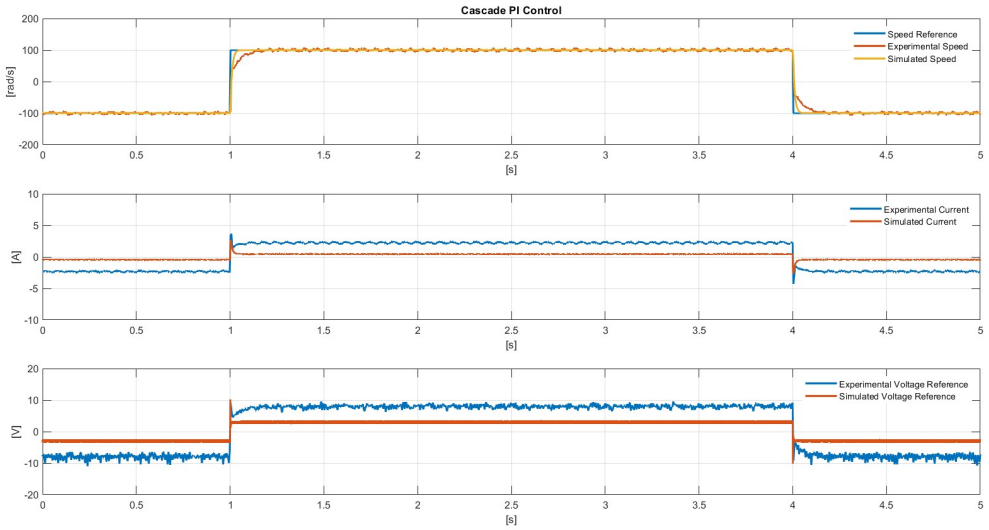


Figure 25: PI Cascade Control Results

As illustrated in Figure 25, both the experimental and simulated results successfully track the desired speed reference. As expected, the simulated response exhibits faster dynamics compared to the experimental one. In our opinion, the discrepancies observed in the voltage and current dynamics between the two cases may be attributed to uncertainties in motor parameter identification, inaccuracies in sensor characterization, and the influence of environmental noise on the experimental measurements.

6.1.2 LQ Control

Linear Quadratic Regulator (LQR) control is an **optimal control technique** that minimizes a quadratic cost function to achieve the best trade-off between control performance and energy consumption [4]. This control structure guarantees **asymptotic stability**, **fast transient response**, and **zero steady-state error** for constant references, while keeping the control effort within practical limits.
In this project, LQR was implemented as an alternative approach to regulate the speed of the PMDC motor.
The motor dynamics are modeled in state-space form (as already presented in equation 14):

$$\dot{x} = Ax + Bu, \quad y = Cx$$

where $x = [i, \Omega]^\top$ is the state vector composed of armature current i and angular speed Ω, u is the input voltage, and $y = \Omega$ is the measured output.
To ensure **zero steady-state error** for constant references, an integrator of the speed tracking error is added. This leads to an extended state vector:

$$x_{\text{ext}} = \begin{bmatrix} i \\ \Omega \\ z \end{bmatrix}, \quad \text{with} \quad \dot{z} = y_{\text{ref}} - y$$

The extended system is therefore described by:

$$A_{\text{ext}} = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix}, \quad B_{\text{ext}} = \begin{bmatrix} B \\ 0 \end{bmatrix}$$

The LQR controller is designed to minimize the infinite-horizon cost function:

$$J = \int_0^\infty \left(x_{\text{ext}}^\top Q_{\text{ext}} x_{\text{ext}} + u^\top R u \right) dt$$

The matrix $Q_{\text{ext}} \in R^{3\times3}$ penalizes deviations in the system states. Each diagonal term corresponds to a specific state:

- Q_{11}: penalty on **armature current** i
- Q_{22}: penalty on **angular speed** Ω
- Q_{33}: penalty on the **integrator state** z, which directly influences steady-state accuracy

The scalar $R \in R^{1\times1}$ penalizes the **control input** u, effectively limiting the control effort and reducing actuator stress.
Proper tuning of these weights is essential to achieve the desired balance between performance and robustness:

- Larger values in Q_{ext} lead to **tighter tracking** and reduced steady-state error, but may increase control activity.
- A higher R value softens the control action, reducing **energy consumption** and improving **smoothness**, at the cost of slower dynamics.

In this application, the most effective configuration was found to be:

$$Q_{\text{ext}} = \text{diag}(10,\ 10,\ 1000), \quad R = 100$$

This choice places strong emphasis on **reducing steady-state error** via the integrator, while still maintaining good regulation of speed and current without overly aggressive control inputs.
The resulting optimal state-feedback control law is:

$$u = -K_x x - K_z z$$

where $K_x \in R^{1\times2}$ and $K_z \in R$ are the feedback gains computed by solving the algebraic Riccati equation, using MATLAB's `lqr` function. The obtained results:

K_x	1.4123
K_x	0.2847
K_z	-3.1623

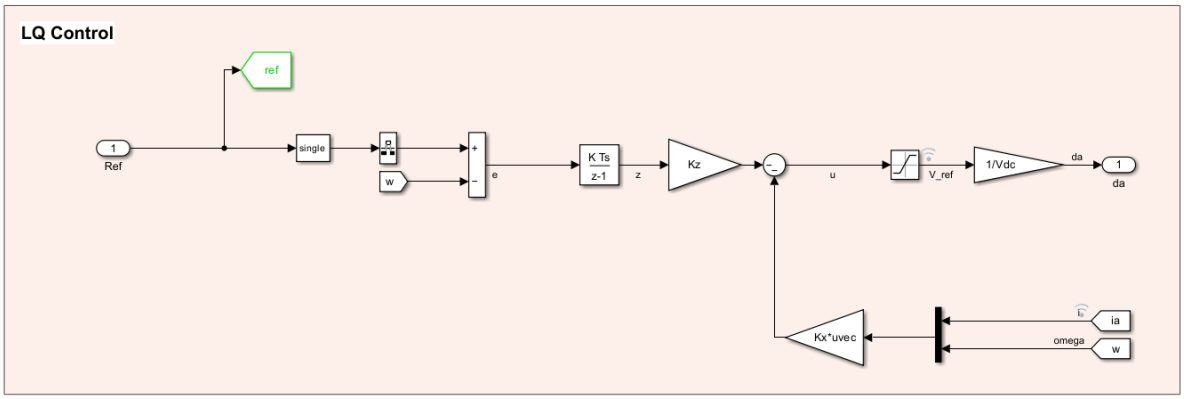


Figure 26: LQ Control Scheme Simulink Implementation

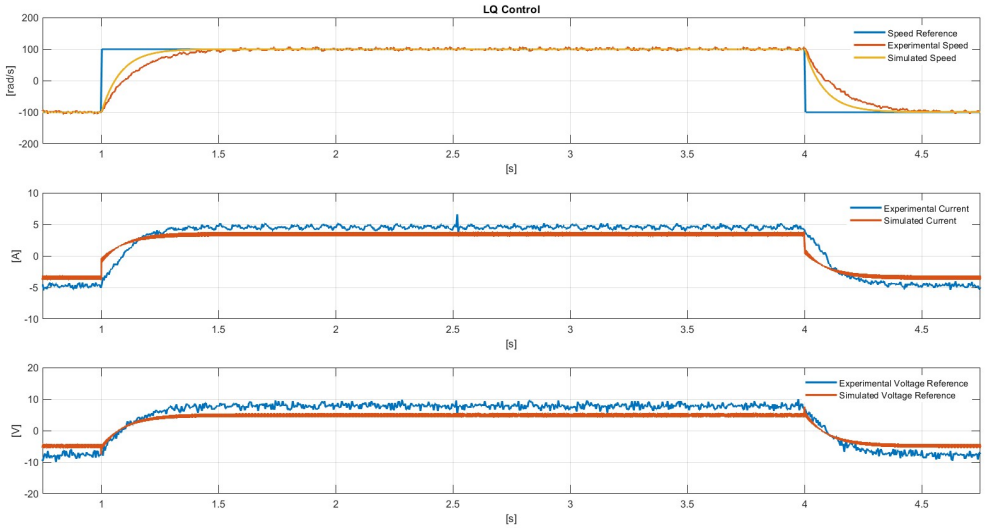


Figure 27: LQ Control Results

The speed reference is accurately achieved in both the simulated and real environments. However, noticeable differences remain in the current and voltage measurements, which, as previously hypothesized, are likely due to uncertainties in parameter identification and sensor characterization.

Comparison between Cascade PI and LQ Control

The goal of this analysis is to compare the results obtained with the two previously presented control strategies: the Cascade PI controller and the Linear Quadratic (LQ) controller.
To ensure a fair comparison, the weighting matrices of the LQ controller have been selected as follows:

$$Q_{\text{ext}} = \text{diag}(10,\ 10,\ 5000), \quad R = 10$$

This configuration has been chosen to achieve a settling time comparable to that of the Cascade PI controller. By aligning the dynamic response characteristics of the two control systems, particularly in terms of settling time, the comparison focuses more effectively on differences in performance metrics, rather than being biased by inherently different tuning aggressiveness.
Regarding the **angular velocity**, both controllers successfully track the reference signal. However, the Cascade PI controller exhibits a faster and more reactive response. In contrast, the LQ controller provides a more damped behavior, with a slightly slower rise time. This reflects the LQ controller's tendency to optimize performance by minimizing control effort and avoiding aggressive transients.
Analyzing the **current response**, the PI controller exhibits an overshoot while the LQ presents a more smooth tracking. This is consistent with the LQ strategy, which aims to minimize energy expenditure.
In terms of the **applied voltage**, the LQ controller results in a more dynamic and variable control action, while the PI controller generates a smoother voltage signal. The increased activity in the LQ voltage response highlights its continuous effort to optimize the system state trajectory, though it may lead to increased stress on the actuator or power stage.
Overall, the LQ controller offers more efficient and stable behavior at steady state, whereas the Cascade PI controller is simpler to tune and responds more promptly during initial transients.

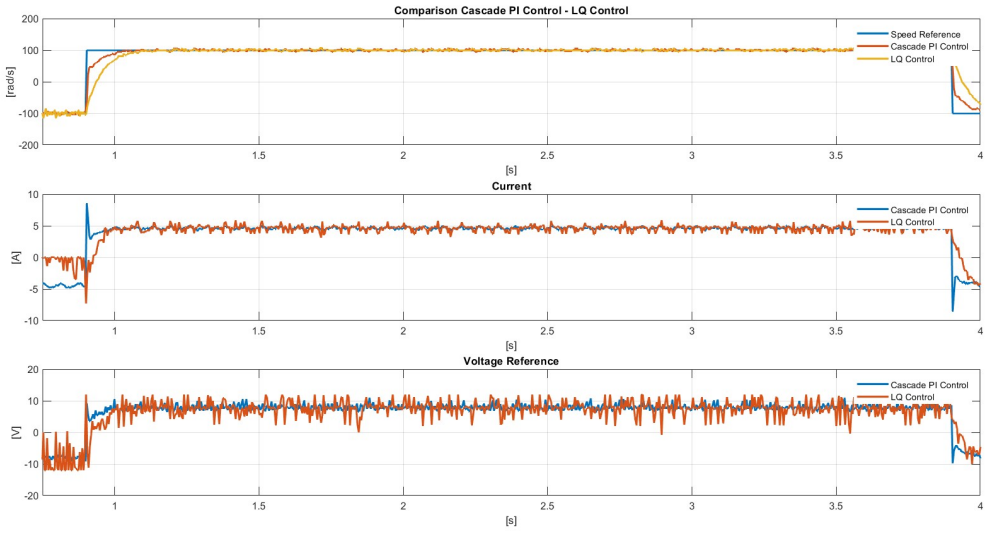


Figure 28: Comparison between PI Cascade and LQ Control

6.2. PMSM

6.2.1 Field Oriented Control

The control strategy employed to drive the PMSM is **Field Oriented Control (FOC)**, a technique that enables dynamic and decoupled control of torque and flux by leveraging a transformation of stator currents into a rotating reference frame aligned with the rotor magnetic field. [1]
The control architecture relies on the theory behind the PI Cascade Control presented in Chapter 6.1.1, so is composed of nested feedback loops:

- An **outer speed controller** computes a torque-producing current reference based on the speed error.
- **Inner current controllers** generate the voltage commands to regulate the current components.

Since in SPMSM the torque expression is dependent only on i_q (as already mentioned in Equation 24), for the torque birth, only the quadrature component of stator current is effective, while direct component has no effect on the torque. So in order to minimize the magnitude of the current space phasor (and consequently the losses), vector control should operate on the power converter in such a way that is, at anytime $i_d = 0$. The controller ensures this by continuously adjusting the voltage component v_d via the inner current loop, while i_q is dynamically regulated according to the torque demand imposed by the speed controller.
The voltage references v_d^* and v_q^* are transformed back to the stationary abc frame using the **inverse Park transformation** and subsequently passed to the pulse width modulation block, which drives the inverter supplying the PMSM.
Rotor position information, obtained from the incremental encoder, is fundamental to these transformations. It allows correct alignment between the reference frame and the rotor magnetic axis, enabling accurate decoupling of the d and q components.

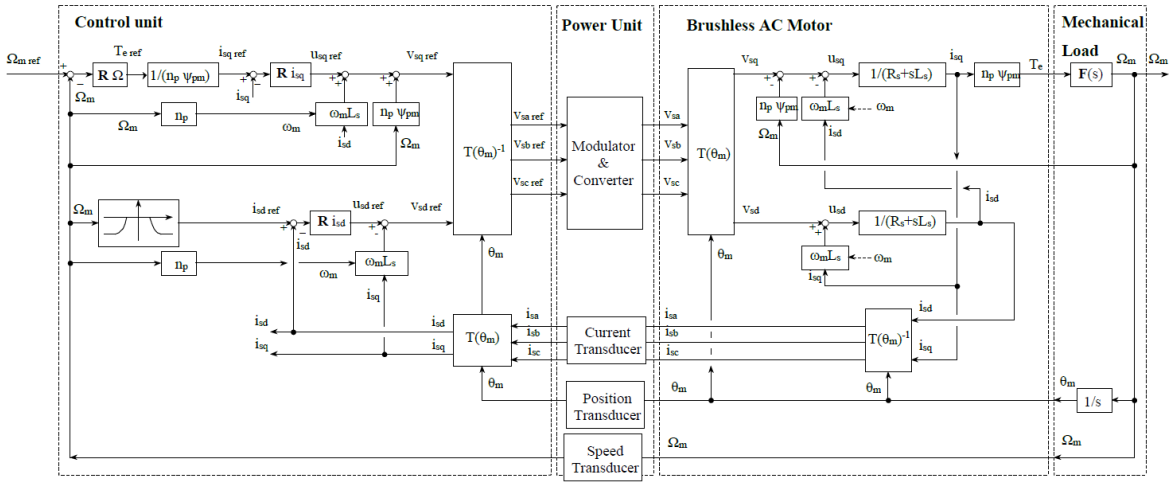


Figure 29: Field Oriented Control Scheme

The PI controllers used in both the current and speed loops are designed using the same methodology described in Section 6.1.1, based on the system's electrical and mechanical parameters. In particular, the controllers are implemented with anti-windup mechanisms and saturation handling to ensure robust operation under dynamic conditions and avoid integral windup during actuation limits.

Note on Modulation Techniques

In Field-Oriented Control (FOC) of PMSMs, it is essential to define the maximum allowable stator voltage magnitude v_s that can be applied by the inverter in order to determine the motor's operating boundaries. The inverter is supplied by a DC link voltage $V_{\text{DC}} = 24\ V$. However, this does not directly translate to the maximum usable stator voltage in the d-q reference frame. Under **Sinusoidal Pulse Width**

Modulation (SVPWM) (the modulation technique applied in this work), the maximum amplitude of the fundamental component of the phase voltage is limited to:

$$v_s^{\max} = \frac{V_{\mathrm{DC}}}{\sqrt{3}}$$

This value corresponds to the radius of the largest circular voltage vector that can be synthesized in the v_d-v_q plane using SVPWM. Exceeding this value leads to inverter saturation and distortion of the output voltage waveform.

Simulation Environment

In the Simulink environment initially a scheme similar to the one provided in the documentation was adopted, in which also the PWM and the Inverter scheme were modelled.

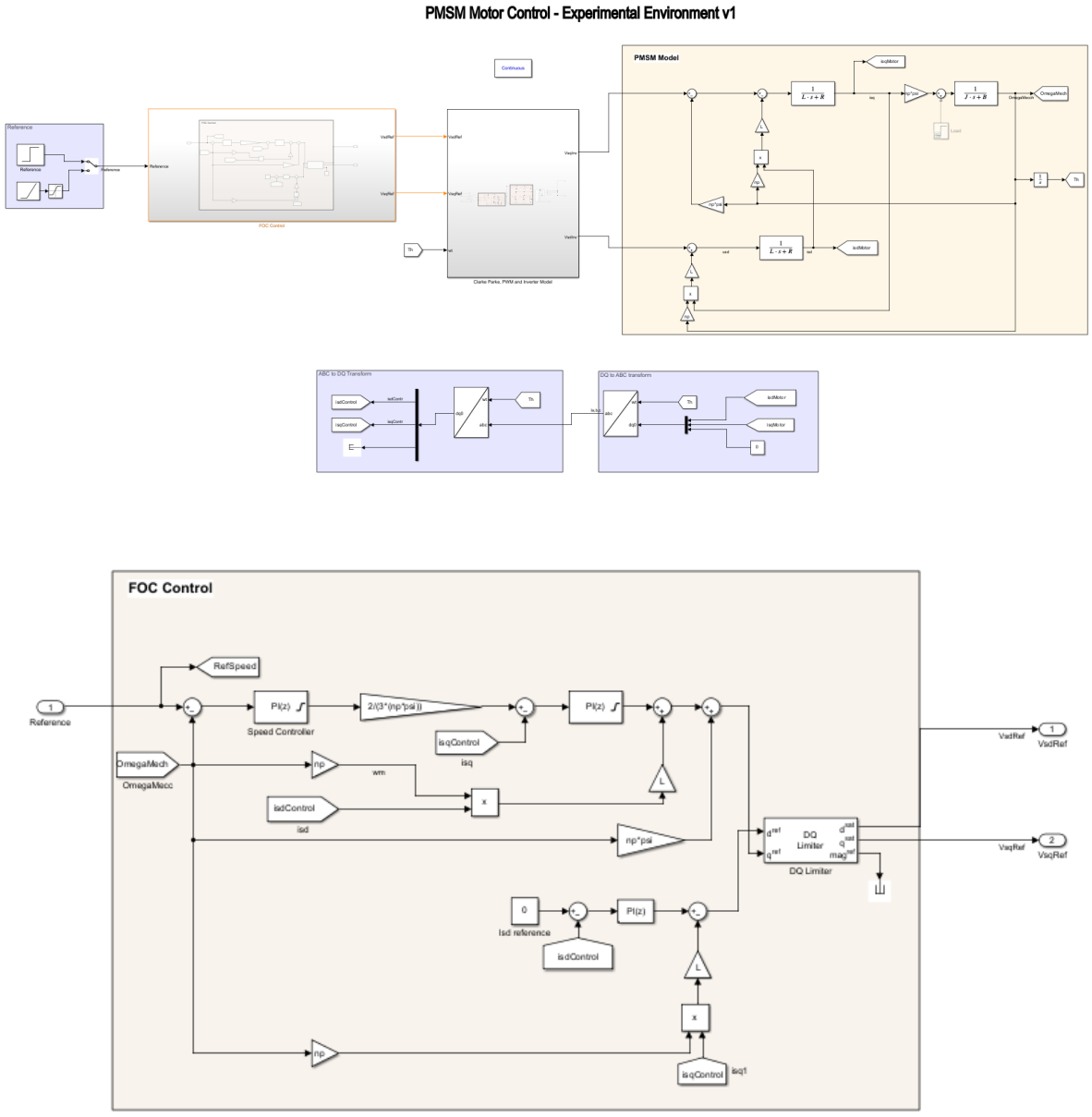


Figure 30: PMSM FOC Simulation Environment - version 1

However this simulation proved to not be sufficient to implement actual control for the hardware system at hand, it was noted that this implementation assumed that the PMSM motor should have been already

aligned at the start of the control, which were not true when the control on the actual motor and the board was activated. Hence a second simulation was developed, using a PMSM block integrated in the motor control toolbox, which allowed to simulate also an initial alignment phase, while keeping the scheme for PWM and Inverter. The result was much more complex than the former scheme, and it was divided into multiple steps, first an alignment phase is activated, then once it is complete the pression of a button allows to reset the values of the PI controllers in order to not have issues with the integral actions of the PIs. Then by pression of another button a switch between the alignment and the speed reference control schemes is actuated. Lastly another button can be pressed to activate the speed reference tracking, of which the desired value can be chosen using the rotary switch available on screen.

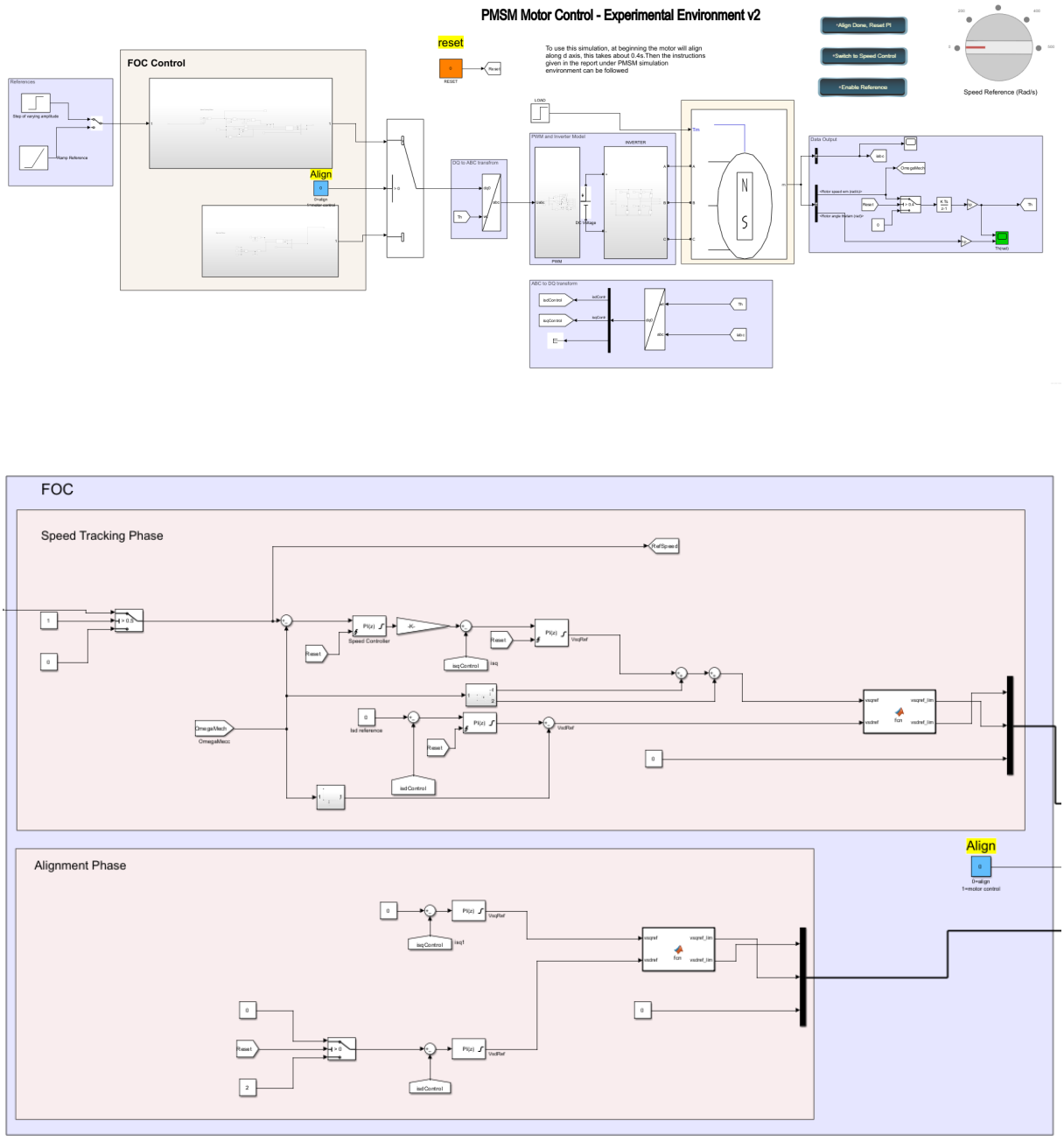


Figure 31: PMSM FOC Simulation Environment - version 2

For both the simulation environments the reference tracking performances were similar, but for the sake of readability only one of the two is shown. To choose the values of Kp and Ki, a similar approach to the one used in the DC motor was used and as such, after choosing the desired crossing frequency, the

values of the integral gain and proportional gain were computed using: $K_p = \omega_c \cdot L_s$ and $K_i = \omega_c \cdot R_s$ for Current PIs, and $K_p = \omega_m \cdot J$ and $K_i = \omega_m \cdot \beta$ for the Speed PI.

ω_c	600 rad/s
ω_m	6 rad/s

which leads to the following gain values:

$k_{p,c}$	0.360
$k_{i,c}$	240
$k_{p,m}$	$2.76e^{-5}$
$k_{i,m}$	$7.26e^{-4}$

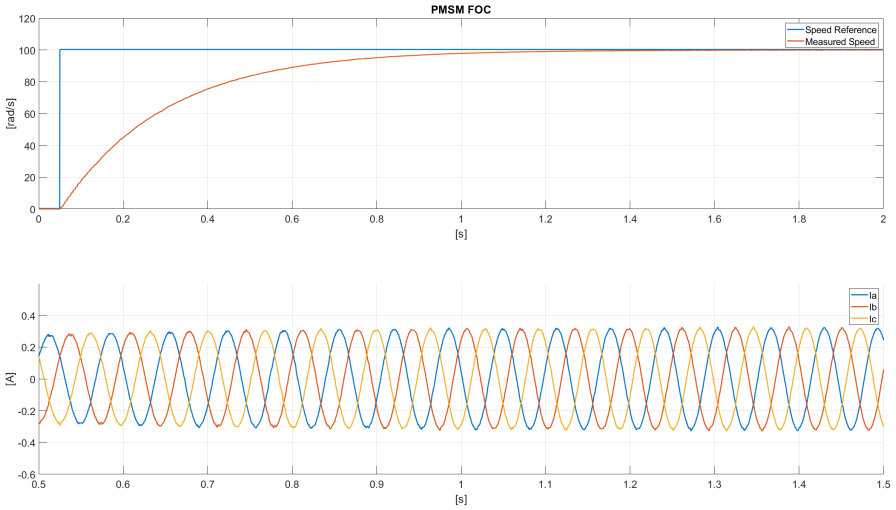


Figure 32: Response to a Step Reference

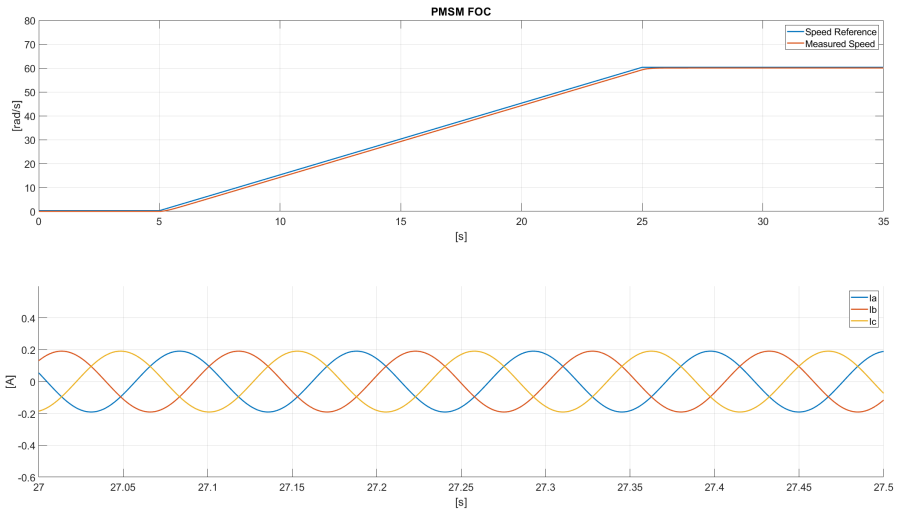


Figure 33: Response to a Ramp Reference

In both cases, the speed reference and current limits are satisfied. It should be noted that the transient response is relatively slow due to the low bandwidth selected.

Experimental Environment

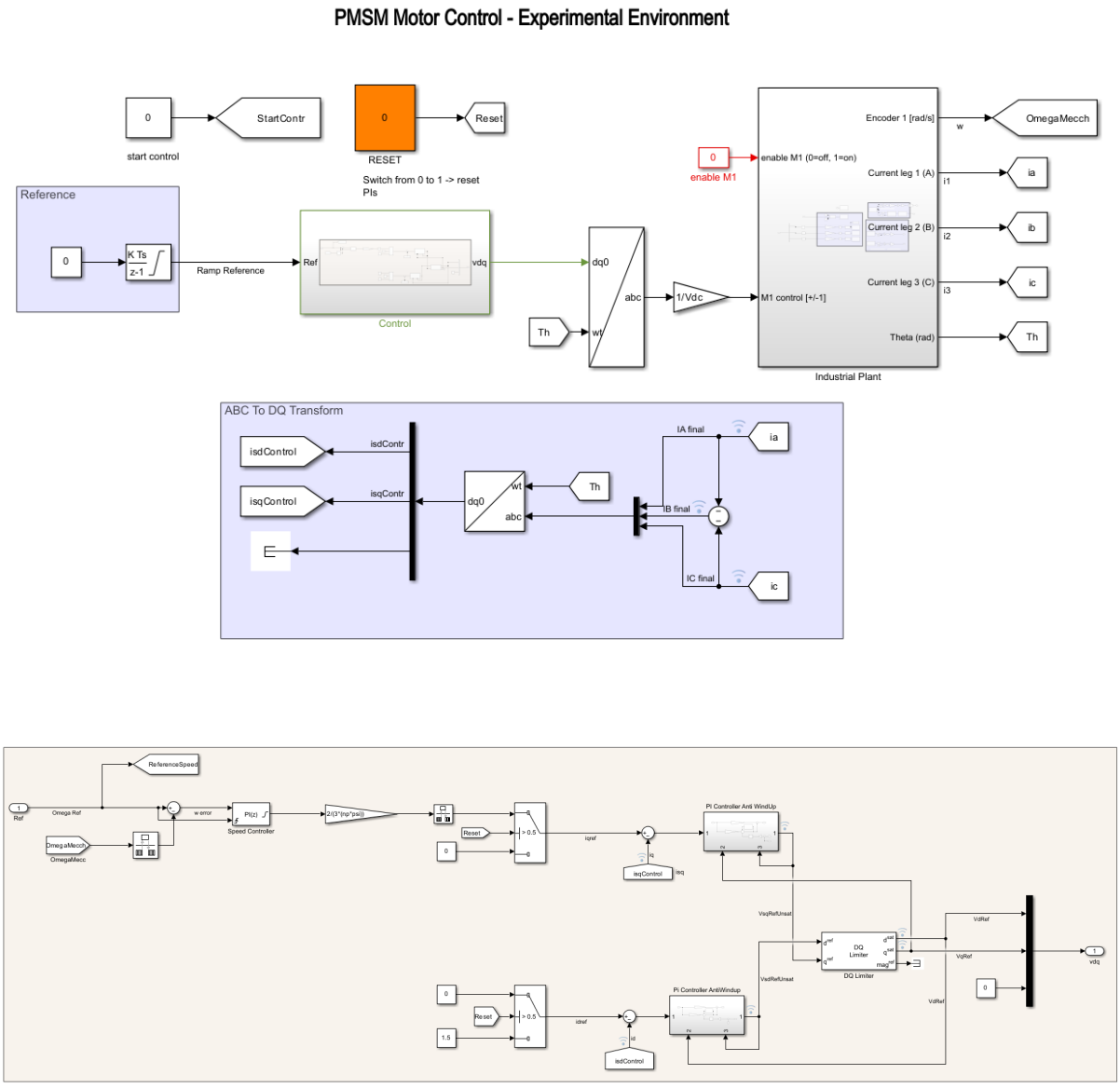


Figure 34: PMSM FOC Experimental Environment

In the real system, the implementation of the FOC algorithm encountered several challenges, particularly in tracking a step reference signal. During initial testing, it was observed that applying a step input caused the state variables to saturate almost immediately, triggering the board's protection mechanisms and interrupting operation. To address this issue, a ramp-shaped reference was employed instead. This allowed for a gradual increase in the reference speed, effectively preventing abrupt transients and ensuring a smoother response from both the electrical and mechanical subsystems.
Following the correct implementation of the rotor alignment procedure, both encoder-based and Hall sensor-based position measurements were tested in order to obtain the most accurate measurements of mechanical speed and rotor angle. However, despite these efforts, the tracking performance remained suboptimal. As illustrated in Figure 36 , although the motor eventually follows the ramp reference, significant vibrations are present during the initial transient phase.

To mitigate the impact of encoder noise, a very low bandwidth was employed in the speed and current control loops ($\omega_c = 600\ rad/s;\ \omega_m = 6\ rad/s$), aiming to filter out as much measurement noise as possible. Despite this, we believe that the tracking issues are primarily due to inaccuracies or noise in the encoder measurements. This effect could be related to the characteristic **stick-slip behavior** at low speeds, where static friction can cause oscillations or hesitations before the motor begins to accelerate smoothly. This phenomenon is particularly pronounced in PMSMs due to the interaction between the rotor's permanent magnets and the stator slots.

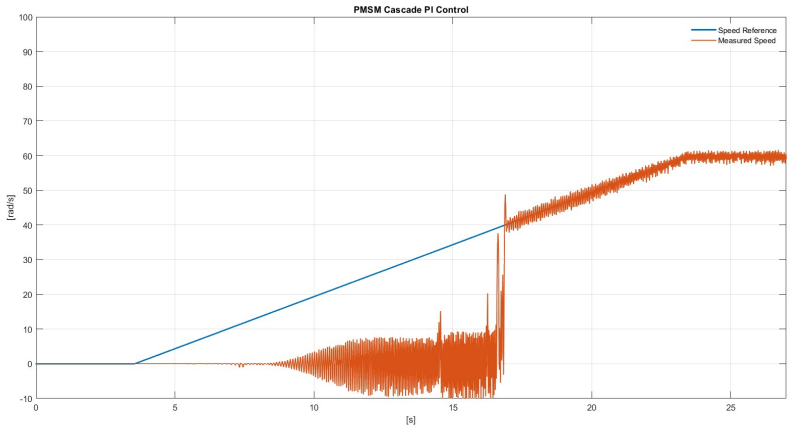


Figure 35: Speed Reference Tracking - PMSM FOC

Regarding the current waveforms shown in Figure 37, it is evident that the phase currents are not perfectly sinusoidal. To reduce the impact of sensor uncertainties, especially considering that the current sensor on phase B exhibits a significant offset deviation compared to the others (as seen in Figure **??**), a compensation strategy was implemented. Specifically, the mathematical relationship governing the sum of the three-phase currents in a balanced system (i.e. $i_a + i_b + i_c = 0$) was used to estimate one of the phase currents indirectly. Despite this corrective approach, the current waveforms still show noticeable distortion, which may be attributed to frequency-related effects, such as aliasing or phase delay in the measurement and filtering chain.

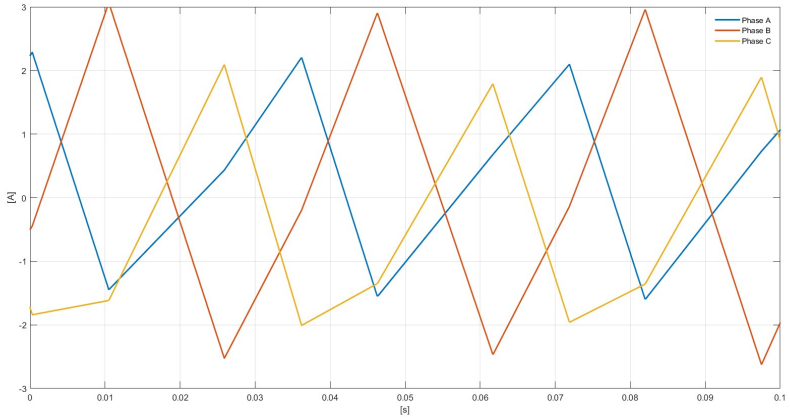


Figure 36: Three Phase Currents - PMSM FOC

6.2.2 Operating Regions

The behavior of a Surface Permanent Magnet Synchronous Motor (SPMSM) under Field-Oriented Control (FOC) can be divided into distinct operating regions, depending on the rotor speed and the voltage and current constraints imposed by the inverter. These regions are determined by the interaction between stator voltage, back-EMF, and stator current magnitude.

- **Constant Torque Region (Below Base Speed):**
  In this region, the mechanical speed is low enough that the back-EMF remains within the inverter's voltage capability. The control strategy sets the direct-axis current $i_d = 0$ to maximize efficiency, while the quadrature-axis current i_q is set to its maximum allowable value under the current constraint $i_{\max} = 9.4$ A.
  The stator voltage magnitude v_s is given by:

  $$v_s^2 = (\omega_e \psi)^2 + (\omega_e L i_q)^2$$

  where:
  - ω_e is the electrical angular speed (rad/s),
  - $\psi = 0.0108$ Wb is the rotor flux linkage,
  - $L = 0.6 \times 10^{-3}$ H is the stator inductance.

  The base speed is the speed at which v_s reaches its maximum value $V_s = \frac{V_{DC}}{\sqrt{3}} \approx 13.86$ V, assuming sinusoidal PWM. Solving the equation for ω_e yields:

  $$\omega_{\text{base}} = \frac{V_s}{\sqrt{\psi^2 + (L \cdot i_{\max})^2}} \approx \frac{13.86}{\sqrt{(0.0108)^2 + (0.0006 \cdot 9.4)^2}} \approx \frac{13.86}{0.01218} \approx 1138\,\text{rad/s}$$

  This corresponds to a mechanical base speed of:

  $$n_{\text{base}} = \frac{\omega_{\text{base}}}{p} \cdot \frac{60}{2\pi} \approx \frac{1138}{4} \cdot \frac{60}{2\pi} \approx 2719\,\text{rpm} \approx 284.5\,\text{rad/s}$$

  Below this speed, the motor can generate full torque with constant i_q and $i_d = 0$ **(a)**.
- **Field Weakening Region (Above Base Speed):**
  When the rotor speed exceeds n_{base} **(b)**, the back-EMF would cause the required v_q to exceed V_s **(c)**. To maintain operation, a negative i_d is applied, weakening the total flux linkage:

  $$\psi_{\text{eff}} = \psi + L i_d$$

  Reducing ψ_{eff} allows continued operation at higher speeds, but since:

  $$i_s^2 = i_q^2 + i_d^2 \leq i_{\max}^2$$

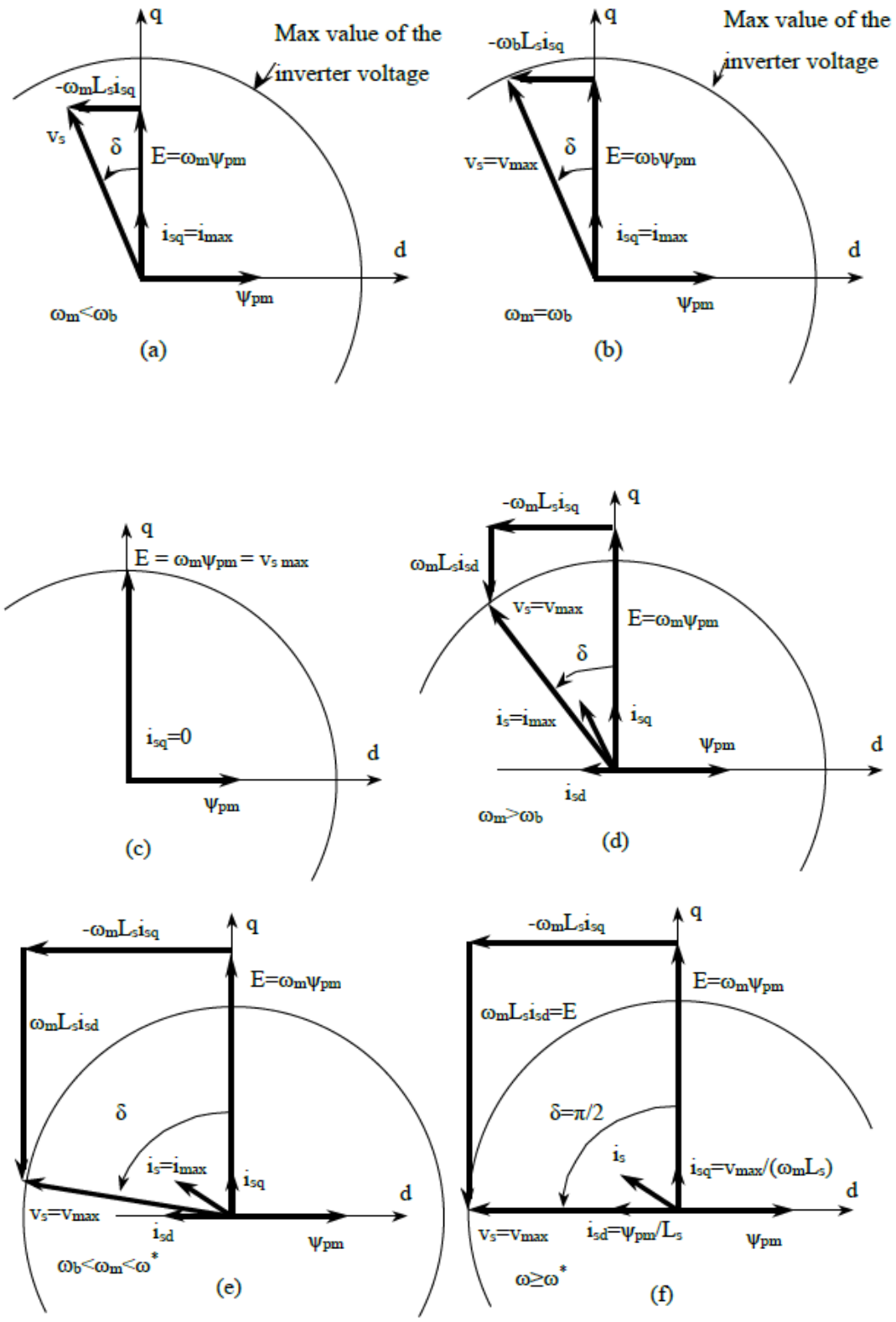
  increasing $|i_d|$ reduces the allowable i_q, and thus the available torque. The motor operates with decreasing torque as speed increases **(d), (e)**.
- **Voltage-Limited Region (Theoretical Maximum Speed $\omega^\star$):**
  This regime is defined by the theoretical condition in which the magnet flux is fully canceled **(f)**:

  $$i_d = -\frac{\psi}{L} \Rightarrow \omega^\star = \frac{V_s}{L \cdot i_{\max}}$$

  However, in this motor:

  $$i_{d,\max} = -\frac{\psi}{L} = -\frac{0.0108}{0.0006} = -18\,\text{A}$$

  This current greatly exceeds the available $i_{\max} = 9.4$ A. Therefore, **the motor cannot physically reach** $\omega^\star$, because it would require almost twice the allowed current to cancel the magnet flux entirely.



Simulation Environment

This strategy was only tested in the simulation environment. Figure 37 shows the implementation of the field-weakening strategy in Simulink using the MATLAB Function Block. The idea behind this implementation is to follow a switching logic that uses the base speed (computed in the previous section) as a threshold—specifically, 95% of the base speed is used to provide a conservative margin.


```
if omega>base_speed_rads
    isdref=-psi/L;
    isqref=sqrt(Imax^2-isdref^2);
else
    isdref=0;
    isqref=Tref/(np*psi);
end
```

Figure 37: Implementation of Operating Regions in Simulink

Figure 38 illustrates the difference between adopting and not adopting the strategy. To properly test the motor's actual limits, a speed reference of 500 rad/s was selected, which is significantly higher than the base speed. The plot clearly shows that with "classical" FOC, saturation occurs near the base speed, while applying the switching strategy allows the motor to reach much higher speeds.

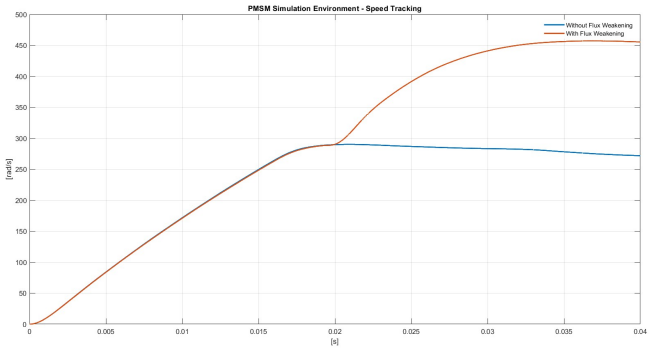


Figure 38: 500 rad/s Step Response — Flux Weakening Effect

Figure 39 is helpful to better visualize the evolution of the stator current components during a transition from normal operation to the field weakening region. Initially, the system operates in the constant torque region, with $i_{d,ref} = 0$ and i_q tracking the maximum allowable current $i_{\max} = 9.4$ A.
When the base speed threshold is exceeded, the controller activates the field weakening strategy by injecting a negative $i_{d,ref}$ to reduce the effective flux linkage and prevent voltage saturation. As expected, this leads to a reduction in i_q in order to respect the current constraint $i_s^2 = i_q^2 + i_d^2 \leq i_{\max}^2$.

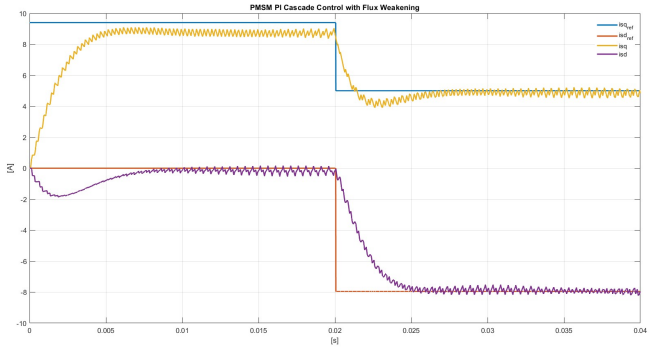


Figure 39: d-q Current Components - Flux Weakening

6.3. B2B Configuration

In the context of the back-to-back (B2B) configuration, a comprehensive designed control architecture was developed and implemented in a simulated environment. As detailed in Chapter 4.3, this configuration involves the PMSM mechanically coupled to the PMDC motor. The PMSM acts as the traction motor and is responsible for tracking a predefined speed reference. Its control strategy is based on a cascade structure with two nested feedback loops: an inner current control loop and an outer speed control loop, both governed by PI controllers. This hierarchical structure enables fast current regulation and accurate speed tracking, leveraging the electrical and mechanical time-scale separation inherent to the motor.
Conversely, the PMDC motor is utilized as a variable load torque generator to emulate resistive effects or braking conditions acting on the shared shaft. Its control strategy is simpler, involving a single current control loop implemented with a PI controller. By adjusting the armature current, the PMDC motor can dynamically apply a controlled braking torque to the system. This configuration allows the emulation of realistic load disturbances acting on the PMSM.

Simulation Environment

As illustrated in Figure 40, the simulation begins with a step input in the speed reference, which is effectively tracked by the PMSM motor thanks to its control scheme. At approximately 0.15 seconds, a load torque is applied by the PMDC motor, which is driven by a current step reference of 1.5 A. This abrupt change in load results in a temporary undershoot in the PMSM's speed response. Nevertheless, the control system successfully compensates for the disturbance, and the motor reaches steady-state conditions without further issues, demonstrating the robustness of the implemented control strategy.

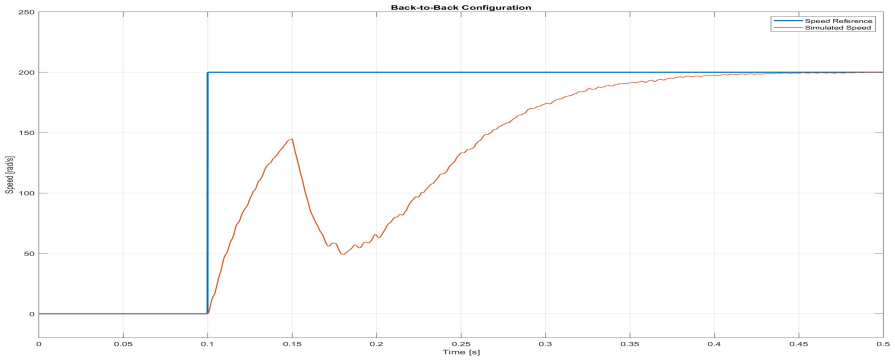


Figure 40: Back-to-Back Configuration: Speed Tracking

7. Sensorless Control

In the field of motor speed control, the estimation of the speed starting from measurements of the voltage applied to the motor and of the current flowing through represents a fundamental step.
Notably, eliminating position or speed sensors (such as Hall-effect sensors or encoders) reduces hardware costs, which is crucial in cost-sensitive applications such as consumer electronics or automotive components and there are fewer components to monitor or replace, simplifying long-term maintenance.
Sensors are prone to mechanical wear, electrical noise, and failure in harsh environments. Furthermore sensorless control improves system robustness by removing these potential failure points.
In this laboratory activity, three different speed estimation techniques focusing only on the DC motor were developed and tested.

7.1. Model-based Approach

The first estimator leverages the knowledge of the system model and the available measurements to estimate the speed. The model uses the identified parameters from Section 5, with armature voltage and current as inputs. Based on Equation 2, the speed is given by:

$$\omega_m = \frac{v_a - L_a \frac{di_a}{dt} - R_a i_a}{K}$$

This equation cannot be implemented directly because it includes a non-causal derivative term. To approximate it, a derivative filter $\frac{s}{0.01s+1}$ was applied. However, this filtered derivative amplifies measurement noise, requiring an additional low-pass filter $\frac{1}{0.01s+1}$ to limit its effect.

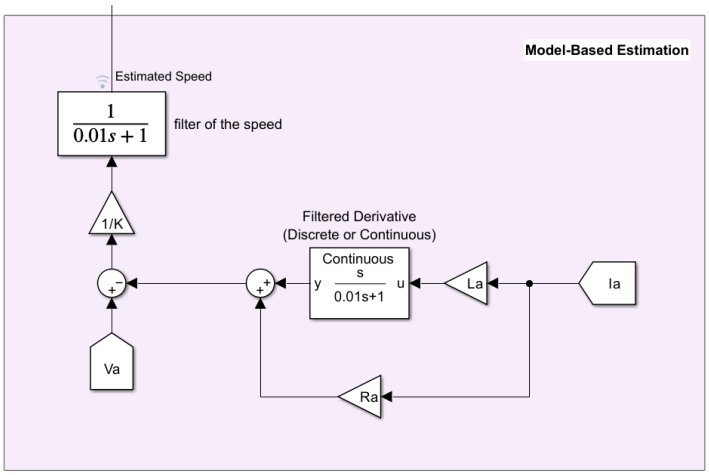


Figure 41: Model-based estimation - Simulink implementation

Simulation Environment

The undershoot in the estimated speed (Figure 42) is primarily caused by the phase lag introduced by the derivative filter. During a step response, the estimated derivative of the current lags behind the true rate of change, delaying the compensation effect in the estimator. This results in an initial drop below the steady-state speed before convergence.
This effect is typical when using real filters to approximate derivatives: the compromise between noise reduction and responsiveness leads to a slower, sometimes non-monotonic transient response. In steady state, the estimator aligns well with the simulated speed, but the transient performance is limited by the filtering dynamics.

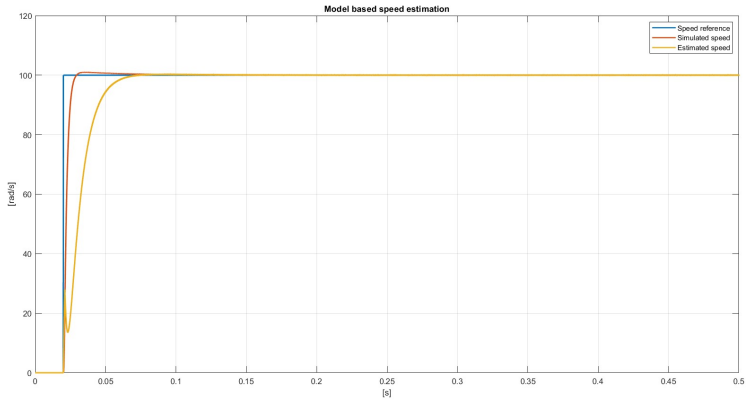


Figure 42: Model-based speed estimation - Simulation Environment

7.2. Kalman Filter Approach

The second method used to estimate the shaft rotational velocity of the DC motor was the Kalman Filter, based on the state space representation of the DC motor itself.
Kalman filtering could be applied in this situation due to the fact that available measurements are affected from a noise that can be modeled as a random Gaussian signal of which variance can be estimated.
Considering the current measurement to be available, the implementation of the Kalman Filter is based on the continuous-time state space form seen in Equations (14) and (15) discussed in chapter 4.1. First of all, the matrices need to be discretized due to the fact that the micro-controller works in discrete time domain, to do so the previously chose sample time T_s was used.
The resulting matrices are:

$$F = \begin{bmatrix} 0.9967 & 0.5685 \\ -0.0022 & 0.9388 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.0239 \\ 0.0808 \end{bmatrix} \qquad H = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

The matrix H is considering only the current as an output and is not using the velocity anymore, this notation is useful for the utilization of the dedicated blocks in the Simulink Environment.
A key aspect of Kalman Filter (KF) development involves defining and tuning the state and measurement covariance matrices. This has been carried out experimentally by comparing the filter's performance using various Q and R matrix configurations.

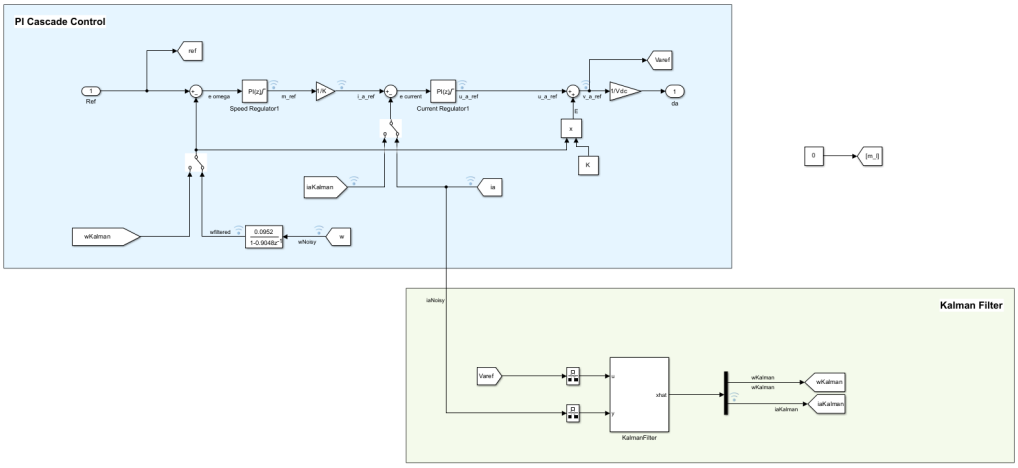


Figure 43: Sensorless Cascade PI Control PMDC Motor

The best Q and R matrices were:

$$Q = \begin{bmatrix} 50 & 0 \\ 0 & 0.1 \end{bmatrix} \qquad R = \begin{bmatrix} 20 \end{bmatrix}$$

Simulation Environment

Speed and Current Estimation The quality of the estimation can be observed in Figures 44 and 45, where it is difficult to distinguish any difference between the measured and estimated values of both speed and current.

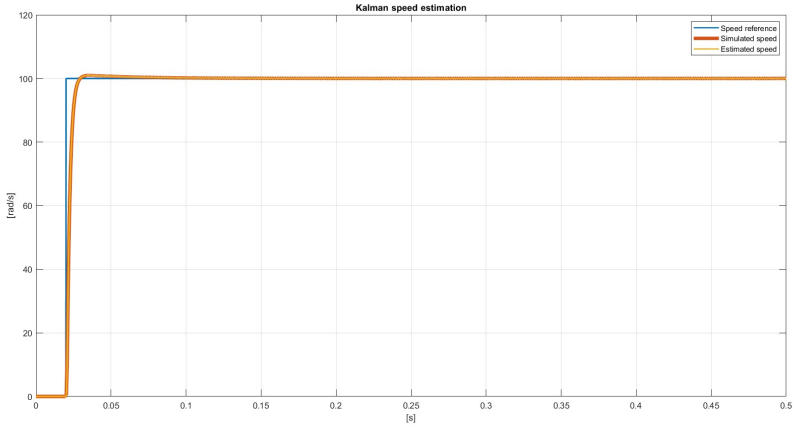


Figure 44: Kalman Filter Speed estimation - Simulation Environment

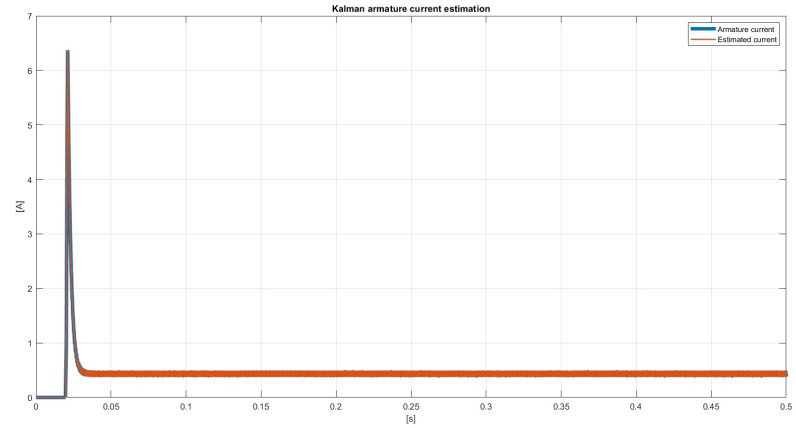


Figure 45: Kalman Filter Current estimation - Simulation Environment

Speed Control Considering the good quality of the estimation, it was decided to use the Kalman Filter estimates to close the loop. To better replicate the experimental environment in the simulation, noise was intentionally added to both the current sensors and the speed sensor. As expected, the velocity control results with a given reference did not show significant differences compared to the previously considered control approach, as shown in Figure 46.

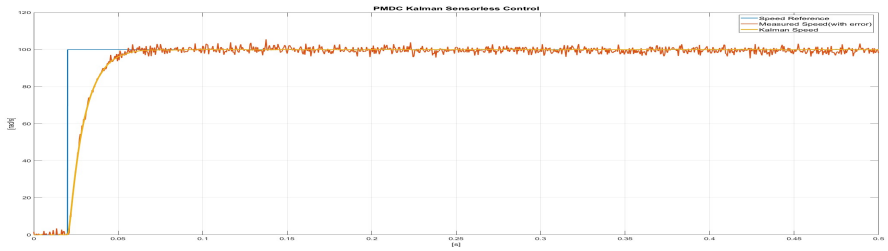


Figure 46: Speed Control using Kalman Filter Estimation - Simulation Environment

Experimental Environment

Speed Estimation The Kalman filter was then tested onto the real system, letting it be controlled by the cascade PI structure discussed in chapter 6.1.1, while still using the encoder to feedback the velocity and giving different steps of velocities as references:

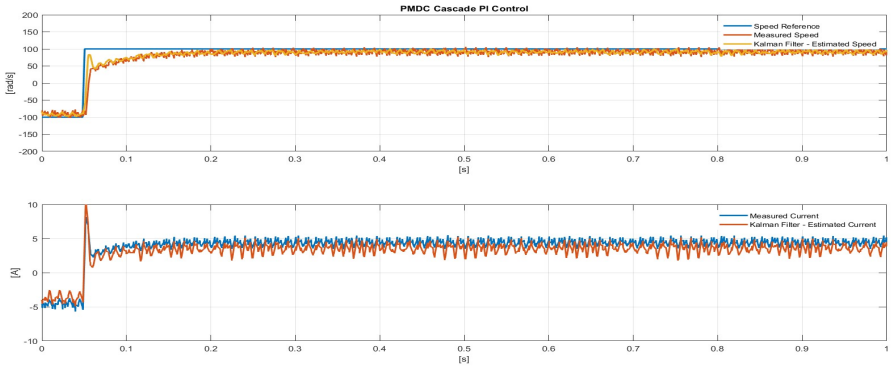


Figure 47: Kalman Filter Speed and Current Estimation - Experimental Environment

From Figure 47 is possible to appreciate the quality of the estimation also in the simulation with the real motor.

Sensorless Control

Lastly the KF was used to implement a control without the usage of the encoder:

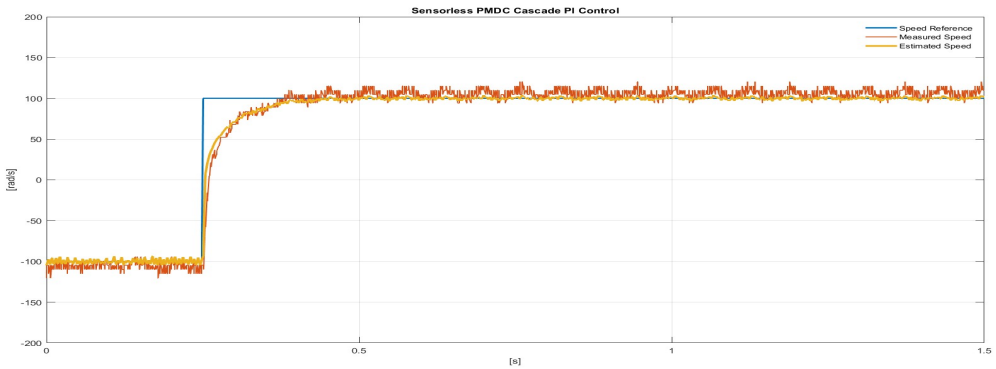


Figure 48: Sensorless Cascade PI Control PMDC Motor - Experimental Environment

Figures 47 and 48 show a minimal difference between the measured and the estimated values for both speed and current. However, the quality of the estimation remains excellent, demonstrating the effectiveness of the Kalman Filter. The filtering effect is particularly appreciable, as it smooths out the sensor noise while closely following the actual signal.

7.3. Reduced Order Observer Approach

A state observer is a system that, based on the input and output measurements, computes an estimate of the state of the system. The reason why we have implemented a reduced order observer is that is less computationally demanding.
Given a general dynamical system in state space representation:

$$\dot{x} = Ax + Bu, \quad y = Cx$$

where the output is a linear combination of the states, the idea is the following one:

1. Apply a state transformation to the system so that the p outputs coincide with p new states.
2. Estimate the remaining $n - p$ states.

External Torque and Speed Estimations

Our goal is to estimate the external torque, which acts as a disturbance. This estimation is valuable because it can be directly incorporated into the control scheme for disturbance compensation.
In order to do that first we need to extend the system. A reasonable choice is to define the external torque as a new state and assign to it a fictitious dynamics considered constant. The extend system becomes:

$$\begin{aligned}
\dot{i}_a &= -\frac{R_a}{L_a} i_a - \frac{K}{L_a}\omega + \frac{V}{L_a} \\
\dot{\omega} &= \frac{K}{J} i_a - \frac{B}{J}\omega - \frac{1}{J} T_e \\
\dot{T}_e &= 0
\end{aligned}$$

Consequently, the extended state vector and the extended system matrices are:

$$x_e = \begin{bmatrix} i_a \\ \omega \\ T_e \end{bmatrix} \qquad A_{m,e} = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{K}{L_a} & 0 \\ \frac{K}{J} & -\frac{B}{J} & -\frac{1}{J} \\ 0 & 0 & 0 \end{bmatrix}$$

$$B_{m,e} = \begin{bmatrix} \frac{1}{L_a} \\ 0 \\ 0 \end{bmatrix} \qquad C_{m,e} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

Notice that only the current was considered as output since the main scope of this observer is the estimation of the speed. On the other hand the voltage is the input. Of course both current and voltage are considered measurable.
The extended system is observable if and only if the original one is and there are no invariant zeros from disturbance to output. Both these two conditions were checked, and the estimation of the speed and of the external torque could be carried out.
The transformed system becomes:

$$\begin{aligned}
\dot{y(t)} &= \tilde{A}_{11} y(t) + \tilde{A}_{12} \tilde{x}_r(t) + \tilde{B}_1 u(t) \\
\tilde{x}_e(t) &= \tilde{A}_{21} y(t) + \tilde{A}_{22} \tilde{x}_r(t) + \tilde{B}_2 u(t)
\end{aligned}$$

where:

$$\tilde{A}_{11} = -\frac{R_a}{L_a}, \quad \tilde{A}_{12} = \begin{bmatrix} -\frac{K}{L_a} & 0 \end{bmatrix}, \quad \tilde{A}_{21} = \begin{bmatrix} \frac{K}{J} \\ 0 \end{bmatrix}, \quad \tilde{A}_{22} = \begin{bmatrix} -\frac{B}{J} & -\frac{1}{J} \\ 0 & 0 \end{bmatrix}, \quad \tilde{B}_1 = \frac{1}{L_a}, \quad \tilde{B}_2 = 0$$

We define now the following variables:

$$\eta(t) = \dot{y}(t) - \tilde{A}_{11}y(t) - \tilde{B}_1u(t)$$
$$\zeta(t) = \tilde{A}_{21}y(t) + \tilde{B}_2u(t)$$

and we are able to setup the observer for the following system:

$$\tilde{x}_e(t) = \tilde{A}_{22}\tilde{x}_r(t) + \zeta(t)$$
$$\eta(t) = \tilde{A}_{12}\tilde{x}_r(t)$$

The observer becomes:

$$\hat{x}_e(t) = \left(\tilde{A}_{22} - L\tilde{A}_{12}\right)\hat{x}_r(t) + \tilde{A}_{21}y(t) + \tilde{B}_2u(t) + L\eta(t)$$

Problem: $\eta(t)$ contains the derivative of y, which is not wise from a numerical point of view.

Explicit the expression of $\eta(t)$ in the observer, add and subtract $(\tilde{A}_{22} - L\tilde{A}_{12})Ly(t)$,
define $\xi(t) = \hat{x}_r(t) - Ly(t)$.
The final reduced order observer becomes:

$$\dot{\xi}(t) = (\tilde{A}_{22} - L\tilde{A}_{12})\,\xi(t) + (\tilde{A}_{21} - L\tilde{A}_{11} + \tilde{A}_{22}L - L\tilde{A}_{12}L)\,y(t) + (\tilde{B}_2 - L\tilde{B}_1)\,u(t)$$

Once the value of $\xi(t)$ has been computed, the state estimate is:

$$\hat{x}_e(t) = \xi(t) + Ly(t)$$

with:

$$L = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix} \qquad \hat{x}_e(t) = \begin{bmatrix} \hat{\omega}(t) \\ \hat{T}_e(t) \end{bmatrix}$$

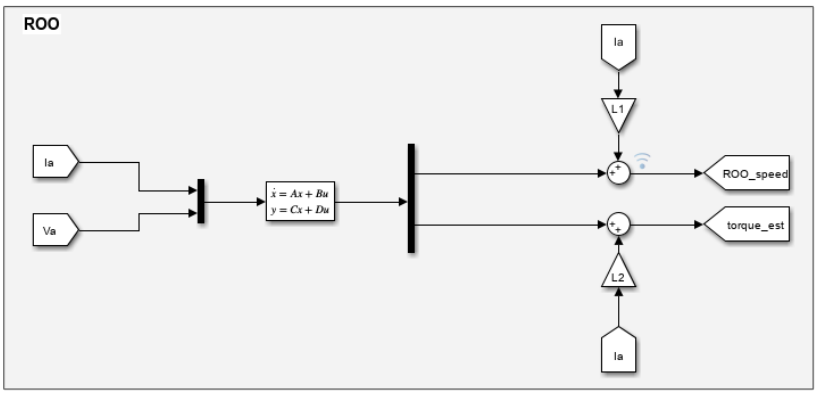


Figure 49: Reduced Order Observer Simulink implementation

Simulation Environment

Figure 50 illustrates the estimated speed compared to the actual speed. Despite the speed not being directly measured in this configuration, its estimation closely aligns with the real speed. This accuracy is attributed to the strong dynamic coupling between the speed and the measured current within the system's equations. As a result, the observer effectively reconstructs the speed without significant oscillations.

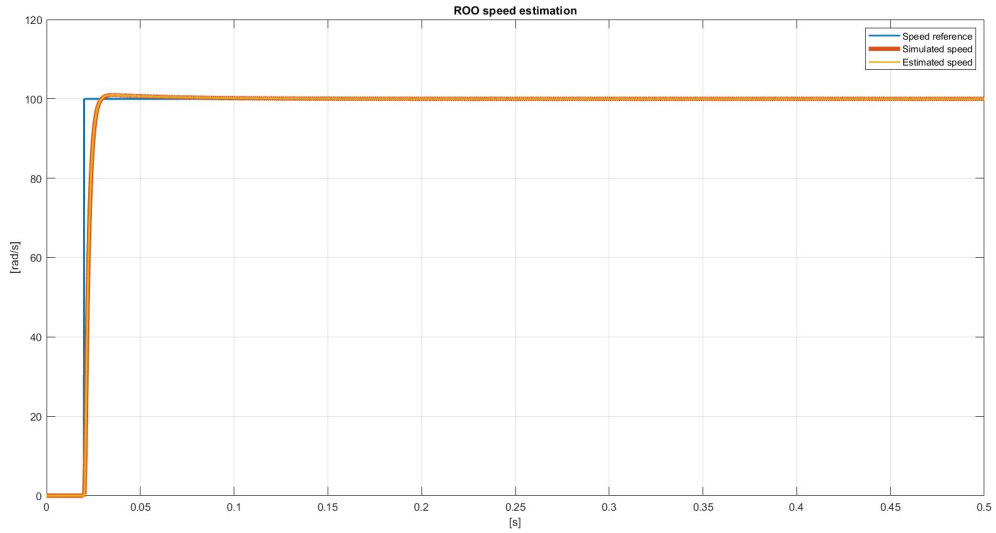


Figure 50: ROO Speed Estimation

In contrast, Figure 51 presents the estimated load torque, where an initial oscillatory transient is evident before converging to the actual load torque value. This behavior stems from the inherent dynamics of the extended system. Specifically, the system matrix $A_{m,e}$ possesses complex conjugate poles:

$$\lambda = -11.07 \pm 953.40i$$

These poles indicate that the system's natural response includes oscillatory components. Although the observer gain L was designed to place the observer's poles on the real axis to ensure numerical stability, the observer still captures the oscillatory nature of the plant's dynamics. Consequently, the estimated load torque exhibits transient oscillations reflecting the system's inherent behavior.

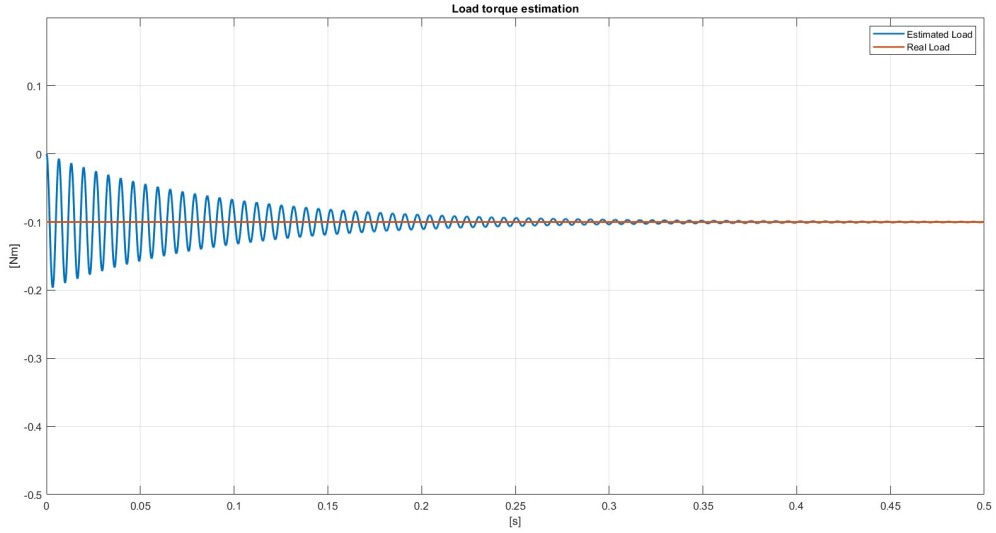


Figure 51: ROO Load Torque Estimation

8. Conclusions

This laboratory project successfully explored the modeling, control, and experimental validation of a back-to-back (B2B) setup comprising a Permanent Magnet Synchronous Motor (PMSM) and a Permanent Magnet DC (PMDC) motor. Through rigorous parameter identification, sensor calibration, and advanced control strategy implementation we were able to construct and analyze a robust electromechanical system.

For the PMDC motor, both cascade PI and LQ controllers were implemented and compared. The PI controller offered simplicity and ease of tuning, while the LQ controller provided more efficient steady-state performance and reduced control effort. Experimental results confirmed the effectiveness of both approaches, highlighting the strengths and trade-offs of classical and optimal control strategies.

The PMSM was controlled using Field Oriented Control (FOC), enabling decoupled torque and flux regulation. While simulations showed excellent performance, real-world implementation faced challenges, mainly due to noisy and inaccurate rotor position feedback. Future work could involve using higher-resolution sensors or developing a sensorless control strategy to replace the built-in Hall-effect sensors.

Additionally, although the motor's operating regions including field weakening were validated in simulation, their implementation on hardware was not completed. Realizing this feature experimentally would be a key step to fully assess high-speed performance and control robustness.

Further, the B2B configuration was tested to emulate realistic operating conditions by coupling the PMSM to a load generated by the PMDC motor. The simulations demonstrated strong disturbance rejection and system stability under load variations. A natural next step in this work would be to implement the complete B2B model on the hardware platform. This would allow for the experimental validation of the coordinated control strategies under real-world conditions and provide deeper insights into the dynamic interactions between the two motors.

Finally, sensorless estimation techniques including model-based, Kalman filter, and reduced-order observer methods were investigated for the PMDC motor. These approaches showed promising results in estimating motor speed and torque, reducing dependence on mechanical sensors and paving the way for more cost-effective and reliable motor control systems.

Overall, the project provided a comprehensive hands-on experience in motor modeling, control design, and embedded implementation, deepening our understanding of modern electric drive systems and the practical challenges involved in their real-world deployment.

A. Adaptive Control

Adaptive control techniques can be employed to successfully control plants with parametric uncertainties. Regular PID controllers can easily tackle additive uncertainties such as sensor noise, but struggle heavily against parametric uncertainties, requiring to diminish the performance targets in order to obtain a stable closed loop response for the system. Although various adaptive techniques are available, the Self-Tuning approach has been implemented in this context due to its suitability for the specific requirements of the application.

A.1. Self Tuning

The core idea behind Self Tuning is to run a recursive least squares algorithm in real time, in order to learn the parameters of the system and to consequently update the parameters of the chosen controller class, effectively tuning it online.

A.1.1 Taxonomy of the Self Tuning Approach

Self Tuning is an *indirect, online* control technique, defined as follows:

- **Online** means that the tuner updates the controller parameters at every time instant. This is particularly useful when high performance is required from the very beginning.
- **Indirect** means that the tuner relies on an identification algorithm which utilizes the available data to construct a model $\hat{S}$ of the open-loop plant S. The tuner then determines how to adjust the controller parameters by employing the **Certainty Equivalence Principle** (CEP), hypothesizing that $\hat{S} = S$, neglecting the modeling mismatch.

The Exploitation VS Exploration problem

The control action has a dual effect on the controlled plant S:

1. **Primary Effect** - The control system imposes the desired dynamics on the plant, effectively hiding its natural behaviour
2. **Secondary Effect** - By interacting with the plant, the controller influences the accuracy of the model estimate $\hat{S}$, and thus affecting the tuning of the controller C

It is intuitive to understand that the primary and secondary effects are inherently in conflict. On the one hand, suppressing the natural dynamics of the system is the objective of the controller itself. On the other hand, to build an accurate model of the plant, it is necessary to excite the system as much as possible, revealing its true behaviour.
One major drawback of Self Tuning is that it is *over-exploitative*. From the start, it attempts to force the output of the closed-loop system to follow the reference signal as closely as possible, obscuring the true dynamics of the system and imposing the designed poles and zeros onto the system. As a result, the quality of the model $\hat{S}$ is reduced, potentially leading to further problems later on.

A.1.2 Recursive Least Squares (RLS-III)

The Self-Tuning approach is applicable only to **ARX** (Auto Regressive with Exogenous input) systems of the following form:

$$y(t) = a_1^o(t)y(t-1) + a_2^o(t)y(t-2) + \cdots + a_n^o(t)y(t-n) + b_0^o(t)u(t-d) + b_1^o(t)u(t-d-1) + \cdots + \\ + b_m^o(t)u(t-d-m) + e(t) \tag{29}$$

where n, m, $d \geq 1$ need to be know and $b_0^o \neq 0 \ \forall t$
$e(t)$ represents an exogenous white noise signal, modeling the sensor measurement noise.

It is therefore necessary to derive the discretized equations of the PMDC, using, for instance, the Forward Euler method, obtaining two ARX models:

$$\begin{cases} ia(t+1) = (1 - \frac{Ra \cdot T_s}{La}) \cdot ia(t) + \frac{T_s}{La} \cdot u(t) \\ \omega m(t+1) = (1 - \frac{B \cdot T_s}{J}) \cdot \omega m(t) + \frac{T_s}{J} \cdot ia(t) \end{cases} \tag{30}$$

where $T_s = 2 \cdot 10^{-4}$ is the chosen sample time (about 7 times smaller than the armature current's time constant).
The identification algorithm employed is the **PEM** (Prediction Error Minimization) method. Given a data set $D^N = \{u(1) \dots u(N), y(1) \dots y(N)\}$, the predicted output $\hat{y}(t)$ at time $t-1$ is defined as $\hat{y}(t|t-1) = \phi(t-1)^T\theta$.
Where $\phi(t-1)$ is the regression vector, containing past input-output data up to time $t-1$, and ϕ is the unknown parameters vector.
We define an indicator of the prediction error magnitude as

$$J(\theta) = \frac{1}{N}\Sigma_{i=1}^{N}(y(i) - \hat{y}(i|i-1,\theta))^2 \tag{31}$$

$J(\theta)$ is a positive quadratic function of θ, which ensures the existence of a closed-form solution for its minimization:

$$\hat{\theta}_N = [\Sigma_{i=1}^{N}\phi(i-1)\phi(i-1)^T]^{-1}\Sigma_{i=1}^{N}\phi(i-1)y(i) \tag{32}$$

This solution is well defined only if the information matrix $S = [\Sigma_{i=1}^{N}\phi(i-1)\phi(i-1)^T]$ is non-singular. To ensure invertibility, the data points in the regression vector must be sufficiently informative (distant from zero and persistently exciting). This requirement can be particularly problematic in the Self-Tuning control paradigm, where the controller itself determines how to excite the system S. Consequently it is not guaranteed to obtain a sufficiently rich output y, especially at the very beginning of the control window. To address this, it is possible to introduce in the cost function $J(\theta)$ an initial strictly positive term, in order to force the positive definiteness of the information matrix.
We can therefore redefine the Least Squares identification problem as follows:

$$min_\theta(J(\theta)) = \frac{1}{N}\Sigma_{i=1}^{N}(y(i) - \hat{y}(i|i-1,\theta))^2 + (\theta - \bar{\theta})^T\bar{S}(\theta - \bar{\theta}) \tag{33}$$

with solution:

$$\hat{\theta}_N = [\bar{S} + \Sigma_{i=1}^{N}\phi(i-1)\phi(i-1)^T]^{-1}[\bar{S}\bar{\theta} + \Sigma_{i=1}^{N}\phi(i-1)y(i)] \tag{34}$$

where $\bar{\theta}$ corresponds to an initial guess of the parameters and $\bar{S}$ represents the weights given to the initial guess.
However, this solution still presents practical implementation issues:

- The identification algorithm must be executed at run-time, consequently the data set D^N is continuously growing, this may lead to memory limitations issues.
- The solution requires computing the inverse of a matrix, which can be computationally expensive, particularly for large values of n and m.

By addressing these limitations, we derive the **RLS-III** (Recursive Least Squares III) identification algorithm

$$\begin{cases} V(t+1) = V(t) - \frac{V(t)\phi(t)\phi(t)^T V(t)}{1+\phi(t)^T V(t)\phi(t)} \\ \hat{\theta}_{t+1} = \hat{\theta}_t + V(t+1)\phi(t)(y(t+1) - \phi(t)^T\hat{\theta}_t) \\ V(0) = \bar{S}^{-1} \\ \hat{\theta}_o = \bar{\theta} \end{cases} \tag{35}$$

It can be proven that $\hat{\theta}_t$ always converges to a finite value. However, convergence to the correct value of the system parameters is not guaranteed and is entirely dependent on the quality (informativeness) of the collected data points.

Simulink Implementation

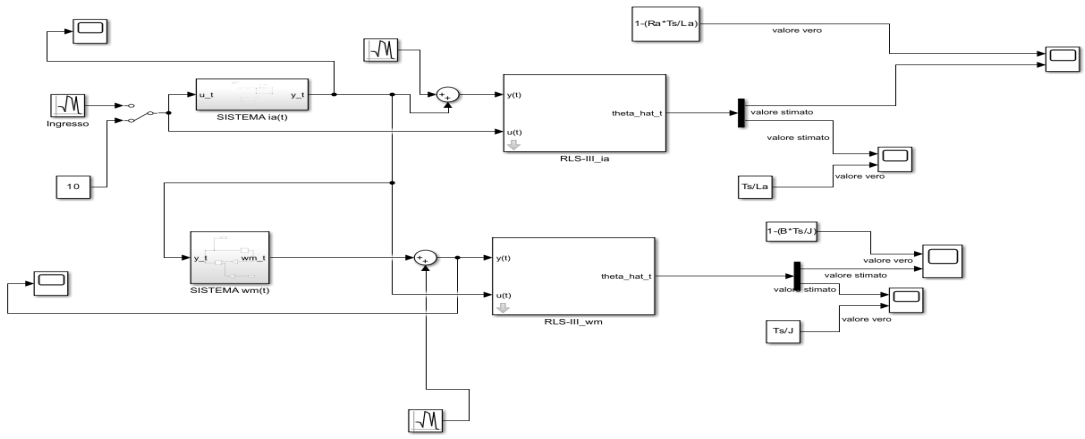


Figure 52: RLS block scheme

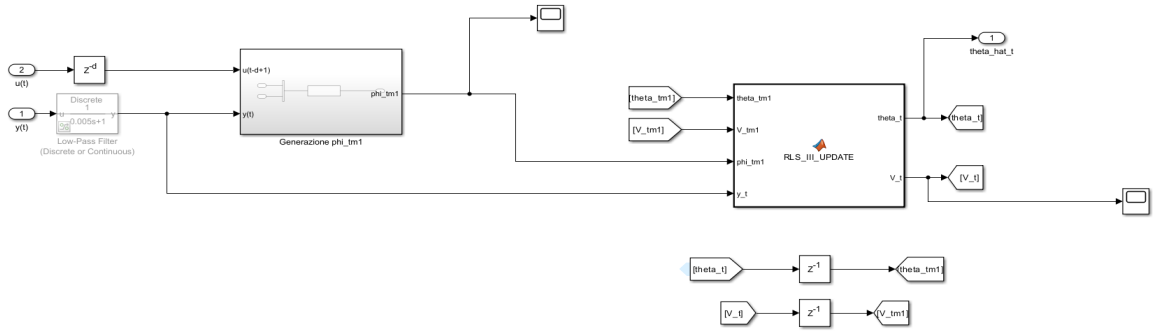


Figure 53: Inside the RLS-III-ia block

In the implementation pictured above, A hypothetical control input u(t) and measurement noise were introduced to assess the accuracy of the estimation algorithm. As expected, the accuracy of the parameters estimates strongly depends on both the characteristics of the input signal and the variance of the measurement noises. To improve estimation accuracy, a low pass filter may be applied on the measurements of the output signals, yielding varying degrees of effectiveness. Additionally, a pre-processing step is required on the input and output signals measurements, to construct the $\phi(t)$ vector.

Simulation Results

The results pictured below correspond to the estimations performed with $u(t) = 10$ and an initial choice of $\bar{\theta} = zeros(2, 1)$ and $\bar{S} = eye(2, 2)$. Some initial overshoot can be observed, which is due to the limited amount of data points available at the beginning of the simulation.

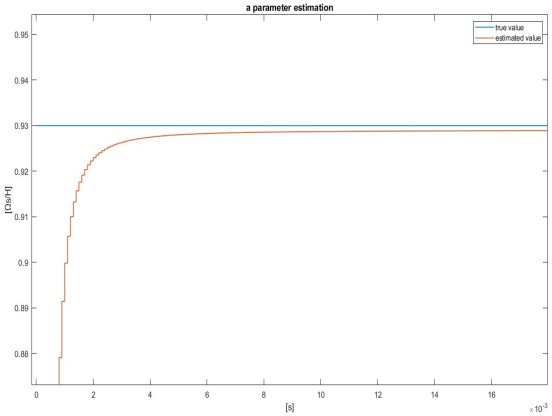


Figure 54: $1 - \frac{Ra \cdot T_s}{La}$ estimate

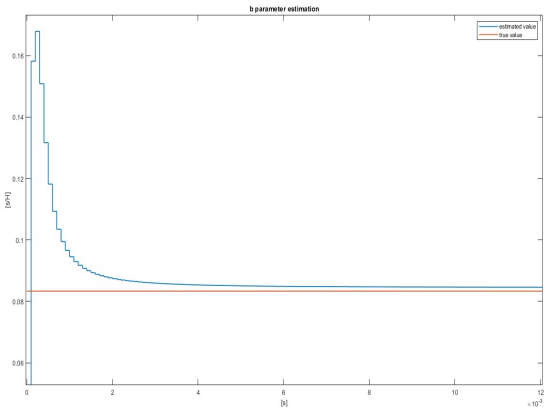


Figure 55: $\frac{T_s}{La}$ estimate

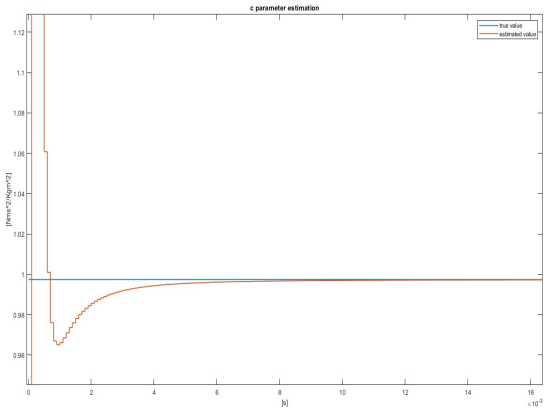


Figure 56: $1 - \frac{B \cdot T_s}{J}$ estimate

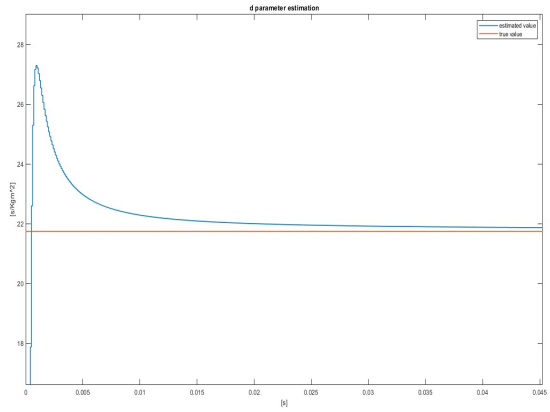


Figure 57: $\frac{T_s}{J}$ estimate

A.1.3 Adaptive Current Controller

A simple **PI** (Proportional Integrative) controller has been chosen to regulate the armature current $i_a(t)$, described previously by equation (30).
By applying the Z-transform to equation (30), the following transfer function, which relates $u(t)$ to $i_a(t)$, is obtained:
$G(z) = \frac{b}{z-a}$ where $a = 1 - \frac{Ra \cdot T_s}{La}$, $b = \frac{T_s}{La}$.
The PI regulator is expressed as:

$$R(z) = K_p + (\frac{K_i T_s}{z-1}) = \frac{K_p z + K_i T_s - K_p}{z-1} \tag{36}$$

Due to the uncertain nature of the available (estimated) system's parameters, it is best to not impose a zero-pole cancellation.
Now, consider the closed loop transfer function relating the reference signal to the output:

$$F(z) = \frac{R(z)G(z)}{1 + R(z)G(z)} = \frac{b(K_p z + K_i T_s - K_p)}{z^2 + z(bK_p - a - 1) + a + bK_i T_s - bK_p} \tag{37}$$

Defining $D_1 = (bK_p - a - 1)$ and $D_0 = (a + bK_i T_s - bK_p)$, it is possible to rewrite the denominator of $F(z)$ as $D(z) = z^2 + D_1 z + D_0$.
Coherently with the performance targets achieved in chapter 6.1.1, we impose a closed loop bandwidth

of 1200 rad/s. This is accomplished by setting the roots of $D(z)$ equal to the complex conjugate pair $z_{1,2} = e^{s_{1,2}T_s}$ where $s_{1,2} = -\xi\omega_n \pm i\omega_n\sqrt{1-\xi^2}$, with $\omega_n = 1200\ \frac{rad}{s}$ and $\xi = 0.707$.

$$\begin{cases} D_1 = -(z1 + z2) \\ D_0 = z1 \cdot z2 \end{cases} \tag{38}$$

Notice that, with $T_s = 2 \cdot 10^{-4}$ it wouldn't be possible to impose a particularly higher bandwidth without losing stability.
The equations system is easily solved, obtaining the following (asymptotic) values for K_p and K_i:

$$\begin{cases} K_p = \frac{1+a-(z1+z2)}{b} \simeq 1.9 \\ K_i = \frac{z1 \cdot z2 - a + bK_p}{bT_s} \simeq 1330 \end{cases} \tag{39}$$

The calculated integral gain K_i is considerably high, for this reason an anti-windup scheme should be incorporated into the controller.
Furthermore, the control action must be saturated in the range $[-0.98 \cdot V_{dc}; +0.98 \cdot V_{dc}]$.

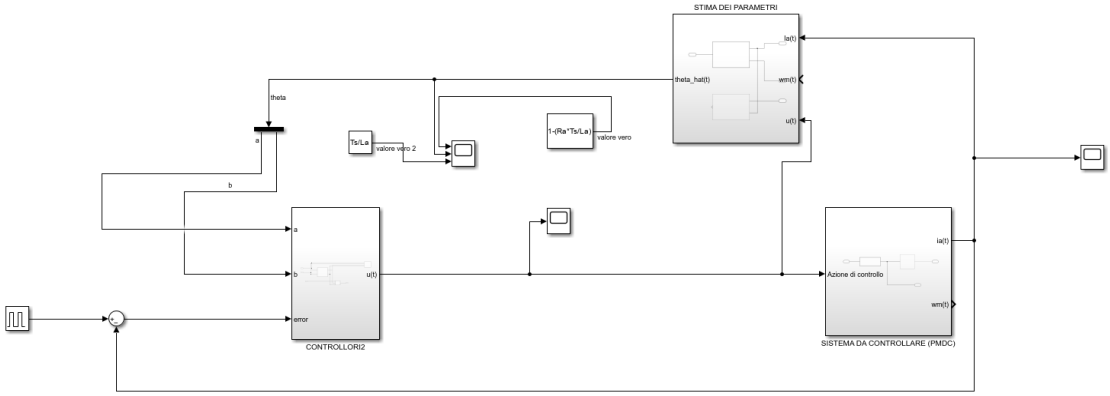


Figure 58: Adaptive current controller block scheme

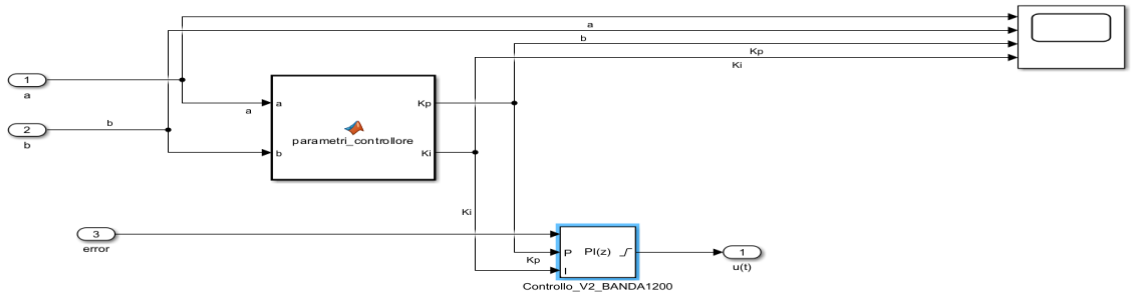


Figure 59: Inside the "CONTROLLORI2" subsystem

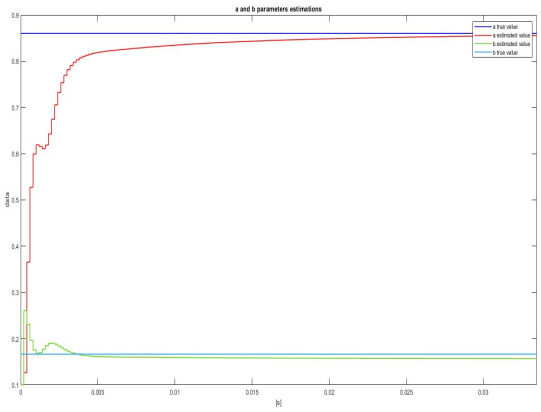


Figure 60: $1 - \frac{Ra \cdot T_s}{La}$ and $\frac{T_s}{La}$ estimations

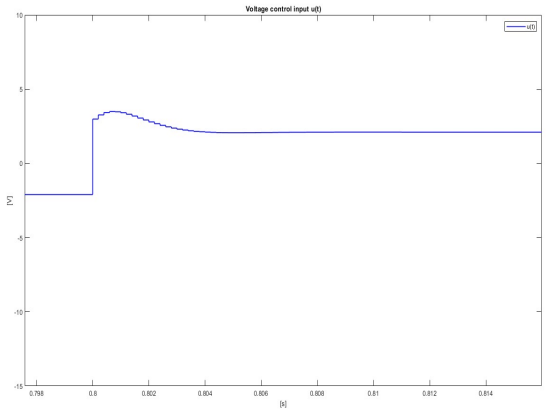


Figure 61: control action u(t)

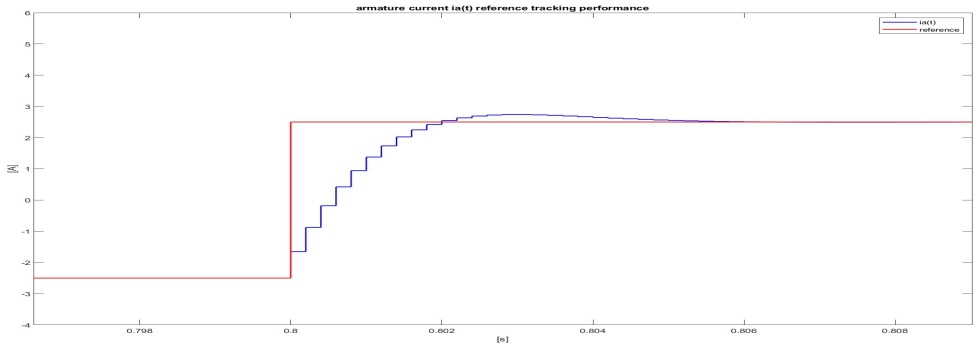


Figure 62: Armature current reference tracking performance, with reference amplitude in the range $[-2.5A; 2.5A]$

The plots above reveal a small overshoot in the system output and control input $u(t)$. These effects are likely due to the residual zero in the closed loop transfer function. A finer tuning of K_p and K_i, possibly with a small reduction of the closed-loop bandwidth; or the introduction of a derivative action in the controller (thereby providing an additional degree of freedom to enforce the zero-pole cancellation and/or to impose a desired zero) may yield better results, giving the controlled system more time to reach the steady state condition and eliminating the overshoots. Additionally, the ideal initial values for the estimations were found to be: $\bar{\theta} = [0.1 \quad 0.1]$ and $\bar{S} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

A.1.4 Adaptive Cascade Controller

A cascade PI controller architecture has been adopted to jointly regulate the armature current of the PMDC and its rotational velocity. To ensure proper functioning of the two loops, a substantially lower closed-loop bandwidth must be chosen for the outer velocity loop. Consistently with chapter 6.1.1, the speed loop bandwidth has been set to 40 rad/s.
Once again, we derived the transfer function $G_\omega(z)$ by applying the Z-transform to equation 30, obtaining $G_\omega(z) = \frac{d}{z-c}$ where $c = 1 - \frac{B \cdot T_s}{J}$ and $d = \frac{T_s}{J}$. Following the same procedure used previously, we assigned the desired closed-loop complex conjugate poles, obtaining:

$$\begin{cases} K_p = \frac{1+c-(z1+z2)}{d} \simeq 2 \cdot 10^{-4} \\ K_i = \frac{z1 \cdot z2 - c + dK_p}{dT_s} \simeq 0.01 \end{cases} \tag{40}$$

An anti-windup scheme has been incorporated in the controller. Furthermore, a saturation in the range $[10 \cdot K; -10 \cdot K]$ has been added to limit the control action $u(t)$.

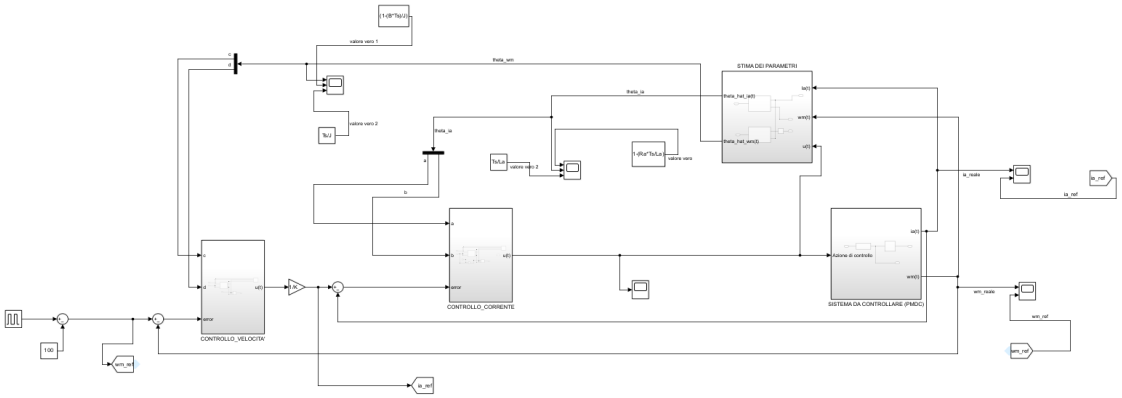


Figure 63: Adaptive cascade controller block scheme

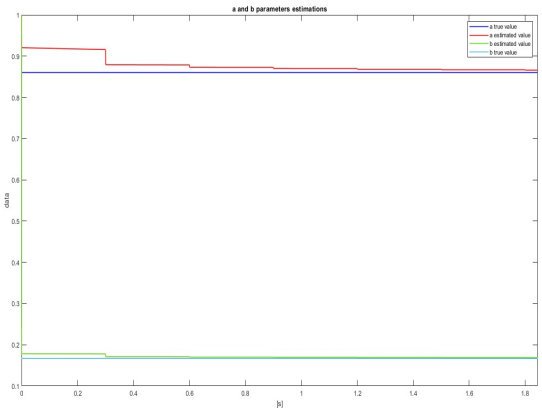


Figure 64: $1 - \frac{Ra \cdot T_s}{La}$ and $\frac{T_s}{La}$ estimations

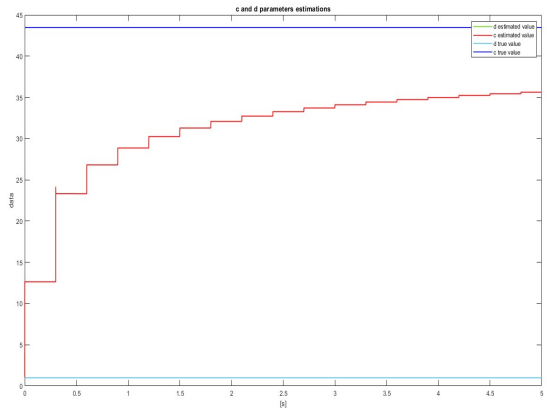


Figure 65: $1 - \frac{B \cdot T_s}{J}$ and $\frac{T_s}{J}$ estimations

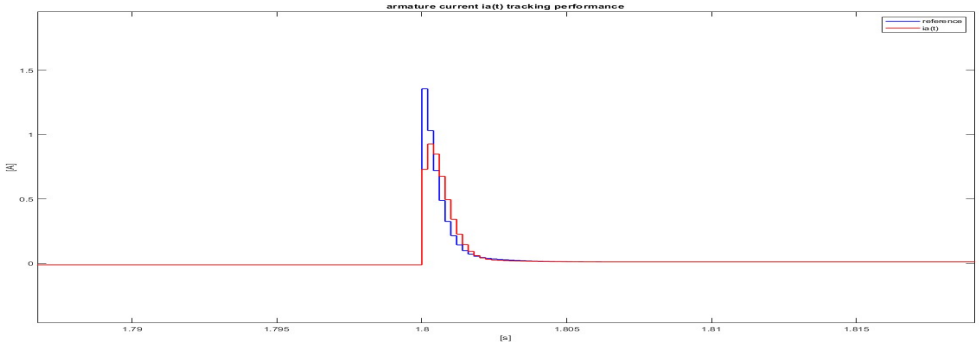


Figure 66: Armature current reference tracking performance

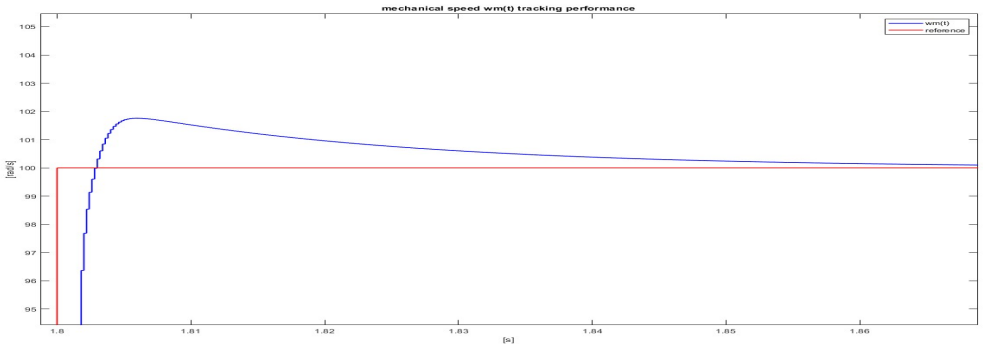


Figure 67: Mechanical speed reference tracking performance, with reference amplitude in the range [-100$\frac{rad}{s}$;100$\frac{rad}{s}$]

A small overshoot with a long drift is evident in the speed signal, and the speed controller generates a nearly impulsive reference for the armature current. These criticalities indicate that the tuning of the controllers may be refined. Even so, the speed response tracks the reference closely, exhibiting zero steady state error and a satisfactory rise time.
The quality of the parameter c estimation has also noticeably declined with respect to chapter A.1.2. This degradation is likely due to the poor excitation provided by the $ia(t)$ signal to the RLS estimation algorithm for the $\omega_m(t)$ subsystem.
Additionally, the best choices for the initial values of the parameters were found to be: $\bar{\theta}_{ia} = [1 \quad 1]$, $\bar{S}_{ia} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $\bar{\theta}_{\omega m} = [1 \quad 1]$, $\bar{S}_{\omega m} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

References

[1] MathWorks. Motor control part 6: Practical considerations for implementing foc with simulink, March 2022. Online video.

[2] Marco Mauri, Maria Stefania Carmeli, and Mattia Rossi. *Macchine Elettriche. Modelli a regime: teoria ed esercizi*. McGraw-Hill Education, 1st edition, 2018.

[3] Mattia Rossi, Nicola Toscani, Marco Mauri, and Francesco Castelli Dezza. *Introduction to Microcontroller Programming for Power Electronics Control Applications: Coding with MATLAB® and Simulink®*. CRC Press, 1st edition, 2021.

[4] Riccardo Scattolini and Lalo Magni. *Advanced And Multivariable Control*. Società Editrice Esculapio, corrected reprint edition, 2020.