POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Field Oriented Control of PMSM motor enhanced with Neural Networks

Tesi di Laurea Magistrale in
Automation and Control Engineering - Ingegneria dell'automazione

Juan Camilo Nustes, 10793367

Advisor:
Prof. Giambattista Gruosso

Co-advisors:
Danilo Pau

Academic year:
2022-2023

Abstract: This thesis presents a Field Oriented Control scheme implemented in Matlab/Simulink as well as the modelling for a Permanent Magnet Synchronous Motor which acts as the plant to be controlled. The simulated environment is used to generate data with great capabilities for training and validating multiple architectures of Neural Networks. The dataset is gathered after the design of specific speed profiles applied to the motor in order to verify how the motor and control scheme behave to the appearance of possible non-linearities that occur when high magnitude speed variations take place. Using the TensorFlow libraries in python, different Neural Networks are trained and validated from the gathered dataset, this trained networks are then brought back to the matlab environment to complement the PID regulator with the goal of improving the overall control capabilities in means of stabilization time and overshoot reduction.

An enhanced version of the FOC scheme is implemented, where a neural network is developed to compensate the torque generated with the PID regulator. This improved control scheme is able to totally eliminate the overshoot generated by the control action while still being able to stabilize the system. Furthermore, the control sees an improvement of 40% in the stabilization time.

Key-words: Motor control, Simulink, Neural Networks, Field Oriented Control, Machine Learning

1. Introduction

Motor control has been one of the engineering targets for the previous decades, as it is a field of engineering concerned with the regulation and management of electric motors, which are essential components in many industrial and consumer applications. Efficient motor control is crucial for achieving optimal performance and energy efficiency in a wide range of devices, from washing machines to electric cars. One of the most common approaches for controlling Permanent Magnet Synchronous Machines is the Field Oriented Control (FOC), which consists of two control loops connected in cascade, where the outer loop controls the motor speed, while the internal loop oversees controlling the current.

Field oriented control is a widely used technique in motor control that allows for precise regulation of the motor's speed and torque. FOC involves transforming the AC current and voltage signals of a motor into two orthogonal

components: the magnetic flux and torque producing components. This allows for independent control of the two components and enables efficient management of motor operation. Permanent Magnet Synchronous Motors (PMSM) are widely used in industrial and consumer applications due to their high efficiency, high power density, and low maintenance requirements. PMSMs are synchronous machines with permanent magnets embedded in the rotor, and their operation requires precise control of the stator current and voltage using techniques such as FOC.

Regression neural networks are a subset of artificial neural networks used to forecast numerical values; they are made up of several interconnected layers of nodes, or neurons, that process the input data to produce a set of output values. Each hidden layer after the input layer transforms the data in a nonlinear way, enabling the network to learn intricate relationships and patterns in the input data. The predicted value is then produced by the output layer using the recognized patterns. In order to reduce the discrepancy between the projected output and the actual output for a particular input, the weights of the connections between the neurons are changed during the training of a regression neural network. Using optimization techniques like stochastic gradient descent and a loss function like mean squared error or mean absolute error, this is accomplished.

The sophisticated control method known as enhanced field oriented control (FOC) is used in electric motor drives to increase performance and robustness. It combines classic FOC with extra compensating mechanisms like feedforward or feedback control. In order to increase the accuracy and robustness of the control system while reducing the reliance on physical sensors, enhanced FOC may also incorporate the use of cutting-edge sensing and estimating approaches, such as model-based observers or sensorless control.

2. System description and Control

A simulated environment of the motor and the control scheme is developed in Matlab / Simulink in order to test the control capabilities of the system without the risk of damaging any component when the overall system is brought to the limit conditions. Changes in the speed reference can and will produce non-linearities in the system, as an example, when the speed variations are of a high magnitude or when the reference is constantly fluctuating at a high frequency. In this section, the structure of the proposed speed controller is explained by defining each component of the FOC scheme as well as the overall implementation of the system in the simulink environment. The control capability is then verified by feeding the system with different input signals such as step and ramps to measure the stabilization time and overshoot percentage, as well as multiple speed profiles specifically designed as tests to bring to the limit the control capabilities of the PID regulators.

2.1. Mathematical model of PMSM

A synchronous motor's working principle is based on the interaction of the rotor's constant magnetic field and the stator's revolving magnetic field. Like three-phase induction motors, synchronous motors operate with a revolving magnetic field in their stator. According to Ampere's Law, the magnetic field of the rotor interacts with the synchronous alternating current of the stator windings to produce torque, which causes the rotor to revolve. A consistent magnetic field is produced by permanent magnets on the PMSM's rotor. The rotor poles lock with the revolving magnetic field of the stator at a synchronous speed of rotation of the rotor with the stator field. In this regard, when linked directly to the three-phase current network, the PMSM cannot start itself.

The mathematical model of a PMSM is described in the two-axis d–q synchronous rotating frame [7, 11]:

$$\begin{cases} \dfrac{\partial i_d}{\partial t} = -\dfrac{\partial R_s}{\partial L_d} i_d + \omega_e \dfrac{\partial L_q}{\partial L_d} i_q + \dfrac{1}{L_d} v_d, & \text{(1a)} \\ \dfrac{\partial i_q}{\partial t} = \omega_e \dfrac{\partial L_q}{\partial L_d} i_d - \dfrac{\partial R_s}{\partial L_q} i_q + w_e \dfrac{\partial \Lambda_f}{\partial L_q} + \dfrac{1}{L_q} v_q, & \text{(1b)} \end{cases}$$

where i_d, i_q and v_d, v_q are respectively the currents and voltages in rotating coordinates. R_s is the phase winding resistance, L_d and L_q are respectively the d and q axis inductance. ω_e is the rotating speed, and Λ_f is the permanent magnet flux linkage.

The torque T_e of the PMSM can be expressed as:

$$T_e = \frac{3}{2} * P * [\Lambda_f i_q + (L_d - L_q) i_d i_q], \tag{2}$$

The rotating speed ω_e is defined by:

$$w_e = P\omega_m, \tag{3}$$

where P is the number of poles in the motor and ω_m is the rotor angular velocity .

The rotor angular velocity is given by:

$$\frac{\partial \omega_m}{\partial t} = \frac{3}{2} * P * [\Lambda_f i_q + (L_d - L_q) i_d i_q], \tag{4}$$

The rotor angular position can be estimated from the angular velocity as shown in equation 5:

$$\frac{\partial \theta_m}{\partial t} = \omega_m, \tag{5}$$

A visual representation of the PMSM construction is presented in Figure 1

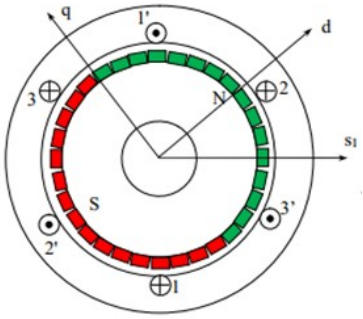

Figure 1: Motor construction of a PMSM.

2.2. Field Oriented Control Scheme

FOC is a variable-frequency drive control method. It has seen significant advancements since its launch, making it feasible to utilize induction motors in places where only DC motors were previously employed. FOC is one of the greatest motor control methods available due to its effectiveness and precision, particularly in terms of speed and torque ripple, making it suitable for both high and low power applications [1]. In Figure 2 is shown a graphical representation of the sensorless FOC focused on the motor speed control.

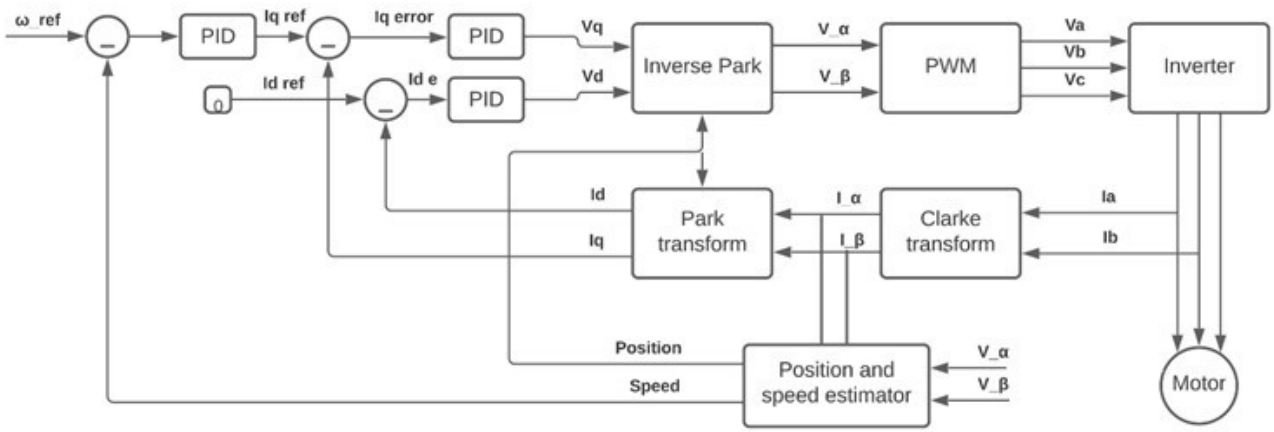

Figure 2: Block scheme of the field oriented control.

FOC depicts stator currents as vectors, with one vector corresponding to the motor's torque (subscript Q) and the other corresponding to the magnetic flux (subscript D). The algorithm figures out the references for the currents that PID controllers require to produce reference voltages. These voltages are sent through the inverse Park transform, which produces the two-phase orthogonal components in the stationary reference frame. At this point, these voltages are represented in a QD orthogonal reference frame [2, 10]. V_α and V_β represent the inputs of the Space Vector Modulator (SVM), The motor phase currents are first acquired and then used as input of the Clarke transform which provides the components in a stationary $\alpha\beta$ reference frame.

The internal control loop receives the reference of the I_q current and subtracts the measured value of the I_q current, this set of currents are then fed to two PID regulators which will the produce the voltages V_q, V_d, these voltages are then fed to the inverse park transform, which changes the reference frame of the voltage from a stationary to a rotary reference frame. Through the Park transform, the $\alpha\beta$ stationary reference frame is converted into a qd rotating reference frame, providing the signals that close the feedback loop. For the speed control, a second PID is introduced upstream the Q-D PID controllers. Through this cascade, it is possible to regulate the electric motor speed. The phase currents are measured from the motor and then fed to the Clarke transform, this transform provides the current in a $\alpha\beta$ reference frame. The position value is gathered using a Flux Observer block which computes the electrical motor position θ_e from the Voltages and currents in the $\alpha\beta$ reference frame.

Each of the components presented in the FOC scheme are explained in detail bellow:

- **Clarke transform** converts 3-phase currents (I_a, I_b, I_c) into a 2-phase currents (I_α, I_β). Since both arrangements produce a rotating magnetic field of the same strength and speed, they are equivalent [13]. Equations 6a and 6b present the computation of I_α, I_β currents respectively.

$$\begin{cases} I_\alpha = I_a = I * sin(\omega t), & \text{(6a)} \\ I_\beta = \dfrac{\sqrt{3}I_a}{3} = \dfrac{2\sqrt{3}I_b}{3} = I * sin(\omega t + \pi/2), & \text{(6b)} \end{cases}$$

  Figure 3 represents the 3-phase currents (I_a, I_b, I_c) and their transformation into a 2-phase currents (I_α, I_β).

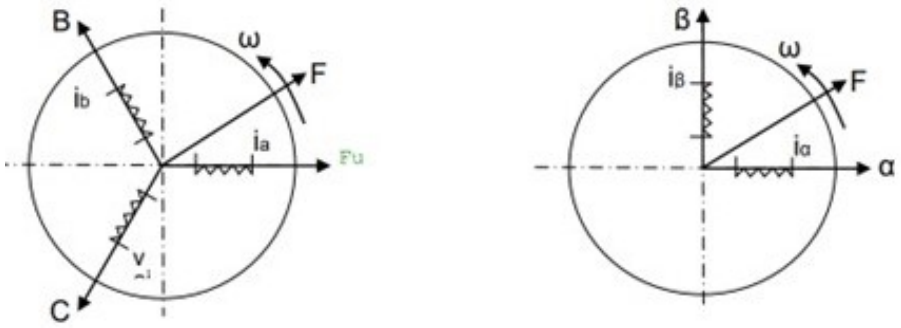

Figure 3: Clarke transformation from 3-phase to 2-phase AC currents.

- The **Park transformation** takes the 2-phase (I_α, I_β) currents represented in a stationary reference frame, and converts the system into a rotating frame (Q-D) by means of the C_p matrix shown in equation 7 where θ is the angle between I_α and I_q [13].

$$\begin{bmatrix} I_d \\ I_q \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} I_\alpha \\ I_\beta \end{bmatrix} = C_p \begin{bmatrix} I_\alpha \\ I_\beta \end{bmatrix} \quad (7)$$

  The transformation is represented in Figure 4 resulting in the 2-phase (I_d, I_q) in a rotating reference frame.

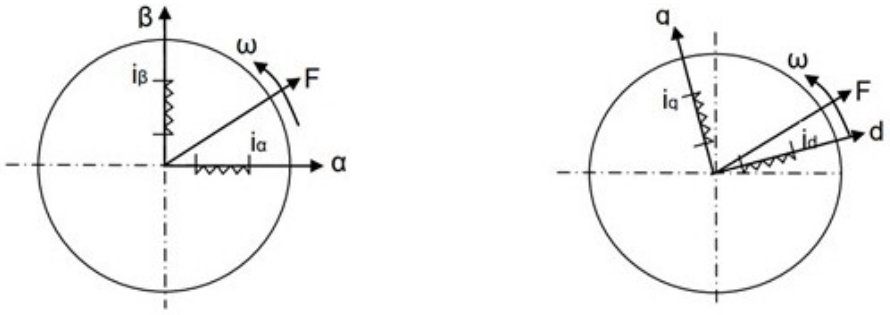

Figure 4: Park transformation from a stationary to a rotating reference frame.

- **Space vector modulation** creates pulse width modulated signals to regulate the switches of an inverter and create the necessary modulated voltage to drive the motor at the required speed or torque [8]. The viable space in the $\alpha\beta$ plane for SVM space vector synthesis is a hexagon, which is then split into six triangular sections by the active space vectors. The active vectors V1 and V2 are applied to the first sector, vectors V2 and V3 are applied to the second sector and so on, as showed in Figure 5.

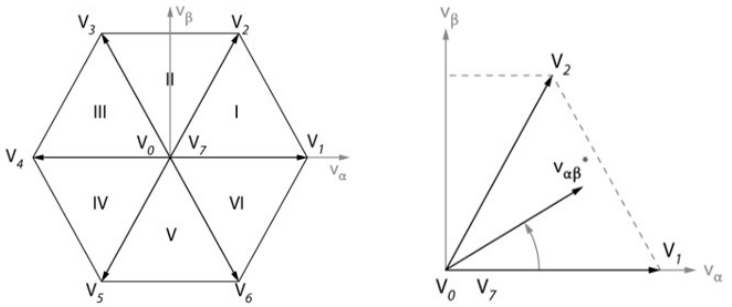

Figure 5: SVM hexagon space vectors.

- To implement the **inverter** in the simulation, the Average-Value Inverter block is used (Figure 6), which converts the DC voltage duty cycles to the 3-phase AC phase voltage that feeds the motor. The inverter is switched on when the DC supplied voltage is higher than the specified parameter V_{DC}, which is selected according to the motor specifications. When the inverter is turned off, the output AC current is null.

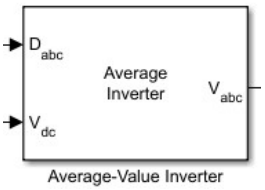

Figure 6: Average-value inverter block in Simulink.

- The **flux observer** block measures the voltage and current values of the rotating reference frame and computes the value for the electrical position of the motor. The electrical position angle is then fed back to the park and Clarke transforms, while also being used to calculate the motor speed in a sensorless approach by obtaining the derivative of the motor position carried by the **speed measurement block**. The flux observer block and the speed measurement block are presented in Figure 7 (a) and (b) respectively.

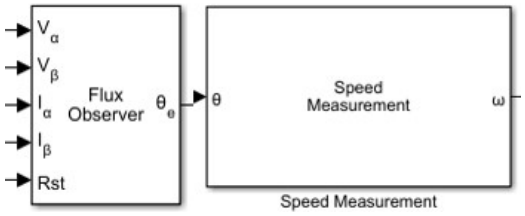

Figure 7: Flux Observer (a) and Speed Measurement (b) block in Simulink.

- To factor into the motor model inside the Simulink environment, a series of blocks inside the Motor control blockset [4] can be used. In this case, the "**Surface mount PMSM**" is the block used to simulate the motor behavior. This block internally considers the motor equations for the non-sinusoidal back-EMF, flux linkage and peak currents presented in and it can be fine-tuned with the desired motor parameters such as the pole pairs and maximum rated speed. The parameters are presented in Table 1:

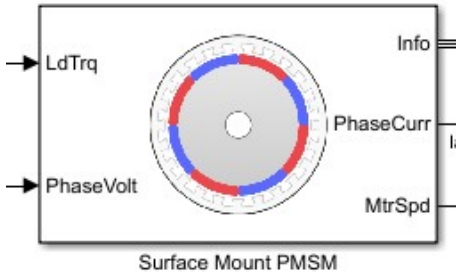

Figure 8: Surface mount PMSM block in Simulink

Parameters	Abbreviation	Value	Measure unit
Pole pairs	p	7	-
Max. speed	ω_{max}	15000	Rpm
Nominal current	I_n	1.20	Apk
Nominal DC voltage	V_{DC}	12.0	V
Stator resistance	R_s	0.11	Ohm
Stator inductance	L_s	0.018	mH
Back-Emf constant	B-Emf	0.4	$V_{ms}/krpm$

Table 1: Motor parameters used in the Simulink model.

After defining the functionality of each component that takes place in the Field Oriented Control scheme, an overall Simulink implementation is presented in Figure 9. This control scheme considers as an input the speed reference applied to the motor, which will also be used to test the control capabilities of the system as explained in section 2.3. The output of the system can be considered as the motor measured speed which is then fed back to achieve the closed-loop control.

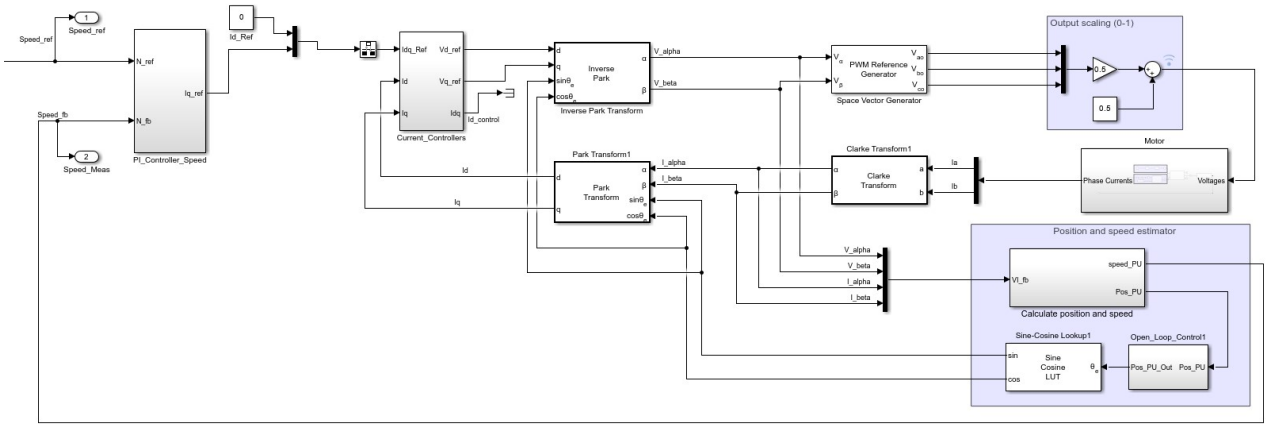

Figure 9: Block scheme of the field oriented control developed in Simulink.

2.3. Control tests

In the previous subsection, all the FOC scheme components where described except for the brain of the control system, the PID regulators. PID stands for Proportional-Integral-Derivative, which is a control method used to regulate or maintain a desired output or process variable in engineering and industrial applications. The PID controller continuously monitors the error between the desired setpoint and the actual output, and applies corrective actions to minimize that error [17]. The three components of the PID algorithm work together to adjust the control output in response to changes in the input.

Proportional (P): Produces an output proportional to the error between the reference and the output, where the larger the error, the larger the corrective action taken. **Integral (I)**: Integrates the error over time, and generates an output proportional to the accumulated error. Eliminates the steady-state errors and stabilize the system. **Derivative (D)**: Produces an output proportional to the rate of change of the error over time, improving the system's response to perturbations. By adjusting the gains of each component, the PID controller can be tuned to achieve optimal performance for a given system.

Setting the ideal PID parameters is a time consuming task due to the fact that there is no straightforward method to define the ideal gain values as the controller can be targeted at different control objectives. One could need a an extremely fast controller which stabilizes in the lowest possible time and not care about the overshoot produced by the controller, while other might be more concerned on not exceeding and overshoot value in exchange on longer stabilization times. The goal for this thesis is to have a fast stabilizing controller capable on accurately following the speed reference while keeping the overshoot value under 12%.

In order to set the starting values for the PID regulators, the "Motor Control Workbench" software developed by ST Microelectronics [16] is used, where the user is able to plug to the computer a control microprocessor connected to the physical motor and then run tests directly on the hardware. After setting up the hardware and the computer, the gains of the controllers for both the inner and outer loop are displayed in the sofware UI as shown in Figure 10.

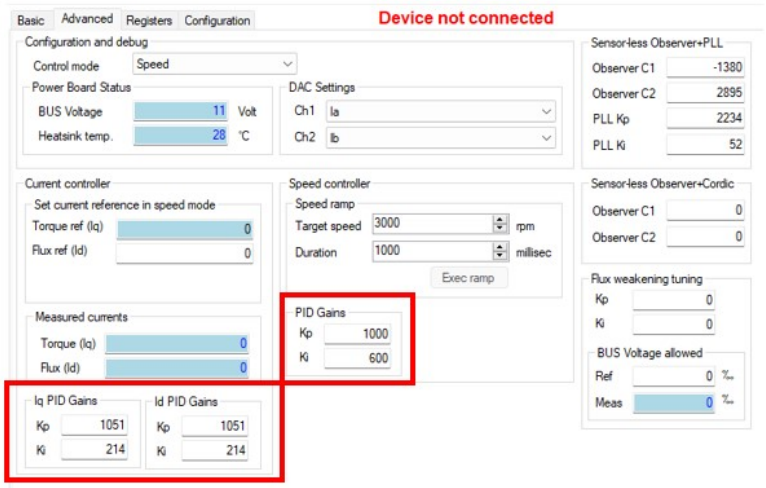

Figure 10: Motor control workbench UI displaying the PID gains.

To start with the control performance regarding the settling time, overshoot and bandwidth, a series of test have been designed, where a speed profile is applied to the motor model, and the control capabilities of the FOC are tested. This testing is approached by varying the input desired speed and comparing it to the measurement of the motor speed value through the scope in Simulink. In Figure 11 is shown an example of a speed profile (a) that will be applied to the motor as well as some other test cases (b) that will be explained in more detail after defining the PID controllers.

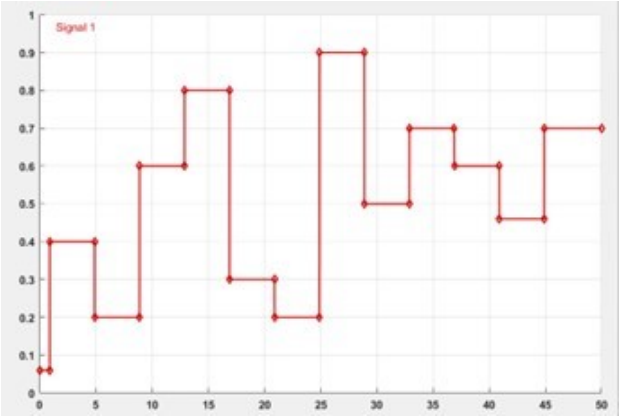

Figure 11: Speed profile to be applied to the motor.

Once the gain values for the PID regulators are set inside the simulated environment, the following tests are carried out to verify the control capabilities of the overall system:

- **Test case: Step Input**
  For the first test, a step block is used as an input to the FOC scheme considering the starting point as the open loop speed (10% of maximum motor speed). Once one simulation second has elapsed, the final value of the step is the desired motor speed. As an example, Figure 12 shows the system response to a specific input step which is set up with the following values:

  - Initial value: Open loop speed (1500 rpm).
  - Step time: 1 second.
  - Final value: 80% of the maximum speed value (12000rpm).

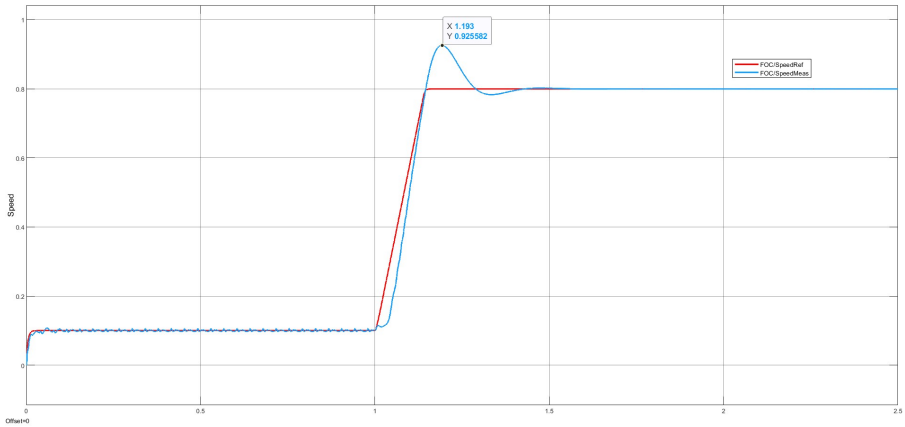

Figure 12: Motor response to a step input of 12000 rpms.

Using the measurement tool from the Simulink scope we can obtain the overshoot value (17.9%) and we are also able to measure the settling time which is around 0.44s, this means that the obtained PID regulator is very aggressive, causing a high overshoot but no oscillations, and therefore, a faster stabilization time.

- **Test case: Ramp Input**
  The ramp block will increase the motor speed from the open loop value (10% of max speed) with a constant slope. The test cases are defined by changing the slope value, which is expressed as a percentage of the maximum motor speed (15000 rpm).

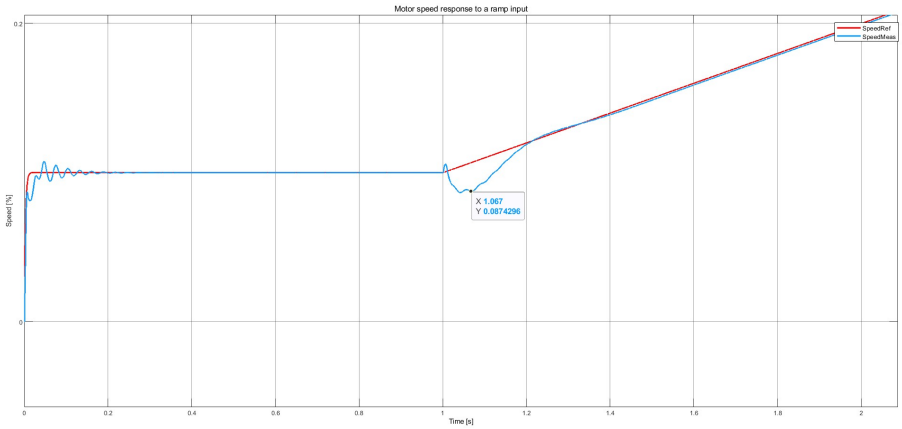

Figure 13: Motor response to a ramp input.

In Figure 13, the motor experiences a small undershoot of (6.56%) when the ramp starts to act on the speed reference due to the change from a open-loop to a closed-loop speed control. Furthermore, the FOC is able to match the speed reference in 220ms after the ramp starts affecting the motor and the system is able to follow the speed reference without problem. Figure 14 shows a different test case scenario, where a step takes place before the ramp in order to first bring the motor to a closed-loop control, and the result is that the undershoot is removed, but an overshoot value of (4.4%) appears due to the presence of the step input and a settling time of 370 ms.

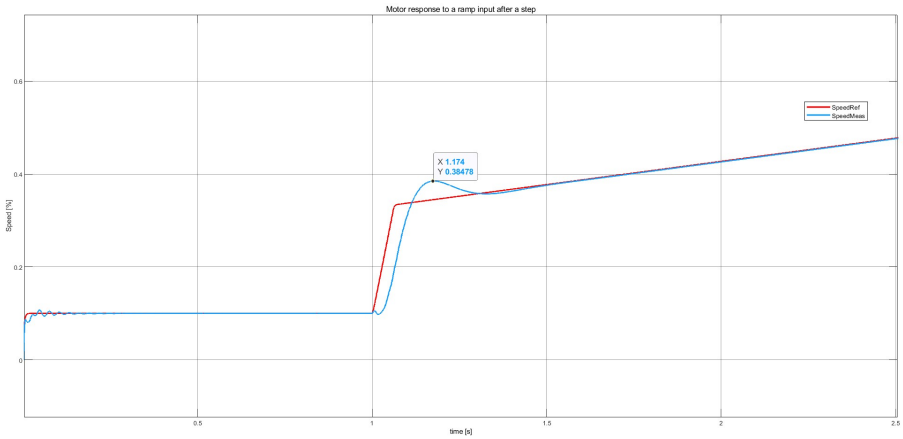

Figure 14: Motor response to a step of 3000 rpm followed by a ramp input.

- **Test case: Speed profile**
  This test case is a combination of various speed changes that will affect the motor during a specified simulation, the aim of this test is to verify how the control scheme will behave when multiple speed changes are affecting the motor reference, similar to a real-life scenario. Figure 15 shows a simulation where the motor is subject during 50 seconds to 12 speed changes, all of them with different magnitude. The objective of this test was verify how the overshoot percentage depends on the magnitude of the speed change, resulting in a maximum of a 10% overshot while being able to quickly stabilize the system (300-400 ms) and accurately following the reference.

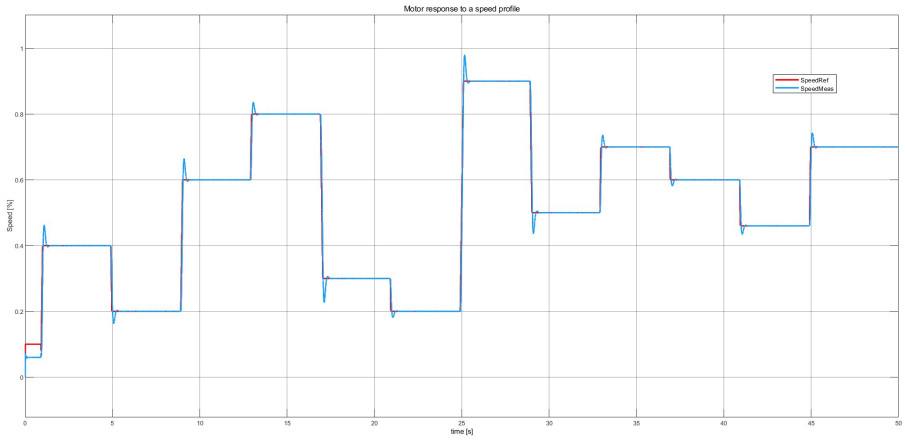

Figure 15: Motor response to a certain speed profile.

With the presented control tests it can be said that the control scheme is capable of stabilizing the plant and accurately following the reference. The obtained control is a robust control with very low stabilization time and overshoot values that fall under the acceptable range (around 10%). The overall control capabilities of the system are presented in table 2.

PID control parameters

Parameters	Value	Units
Settling time	0.41	[s]
Overshoot	10.2	[%]
Bandwidth	-	[rad/s]
Phase margin	-	[degrees]

Table 2: Control capabilities of the system.

3. Neural Network implementation

After the control of the system is verified, the next step is to develop a Neural Network which can replace the PID regulator inside the simulated environment, and then run a comparison between the behavior of the system controlled by regular PID controller vs the system controlled by a neural network. The type of network required for this task is a regression network, as it will be trained with a string of data gathered from the motor trying to replicate the job carried out by the PID. Neural networks are particularly effective in situations where the relationship between the input data and the output is complex and non-linear. The ability of neural network estimators to learn from and adjust to new data is one of its key features. Additionally, they can handle vast amounts of data and spot patterns that could be challenging or impossible for a human to notice.

However, neural network estimators can also be costly in terms of computation and training data [14, 15]. When gathering the data to train the network, the inclusion of non-linear boundary brings the possibility to design a non-linear control approach such as Machine learning or Neural Network based control, capable of matching and, if possible, outperforming the behavior of a PID regulator in certain situations [30]. [6]

3.1. Dataset generation

The first step for the neural network development is to gather relevant data to train the model so that the obtained network is able to learn the most from available data before being tested and therefore obtain better results when deployed. The dataset was gathered from the Simulink environment where the stored measurements are the values of the real and reference speed, as well as the reference and measured values of the regulator currents in the DQ reference frame. Finally, the voltages V_q and V_d, which are the internal loop outputs of the PID regulators are also stored. To read the simulink data from the matlab script the, the "to workspace" block

is used as shown in Figure 16.

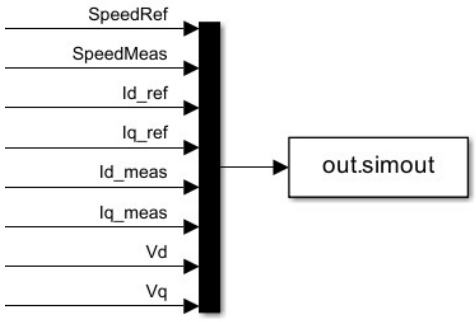

Figure 16: "To-workspace" block in simulink.

This dataset was created to mimic the response of a PMSM when subjected to a wide set of input speed shapes as well as test conditions specifically designed to highlight how the nonlinearities affect the PID controller present in the FOC scheme, in this way, a physical motor to gather data is no longer needed. To make sure that the dataset contains enough relevant information a set of input conditions where designed, where the motor will face a series of speed variations occurring at different magnitudes and frequencies. The first speed profile included in the dataset was the one presented in the control tests (section 2.3), based on the motor response when subject to this test, it was decided to generate a new speed profile where the reference step is increasing until the motor reaches the maximum speed.

This test mimics a Formula 1 car where the driver‘s acceleration pedal is acting as a switch, so the speed is no longer gradually increasing, but is pumped up from stop to maximum value. To make the test case richer in matter of data, this speed change is fixed to a change from 0 to 20%, 40%, 60%, 80% and full speed as it is shown in Figure 17. Then, the speed starts changing in an inverse pattern, where the base speed will be the maximum value and it will start decreasing by 20%, then 40% and so on.

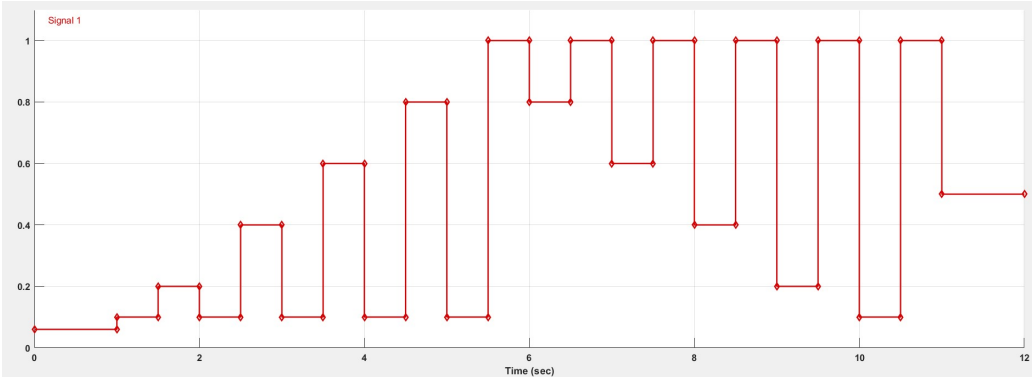

Figure 17: Test case for the data gathering.

Some variations of the previously explained speed profile are also considered, such as reducing the time between the speed changes in order to allow less time for the system to stabilize, and even to bring the system out of control. The next test case for data generation is the "random source generator" which is carried out by a simulink block with the same name which sets a mean value around which it will start generating a random signal. The goal of this test is to test the system against rapidly changing speed values and verify if the controller is able to follow the reference. An example of a generated random signal is presented in Figure 18.

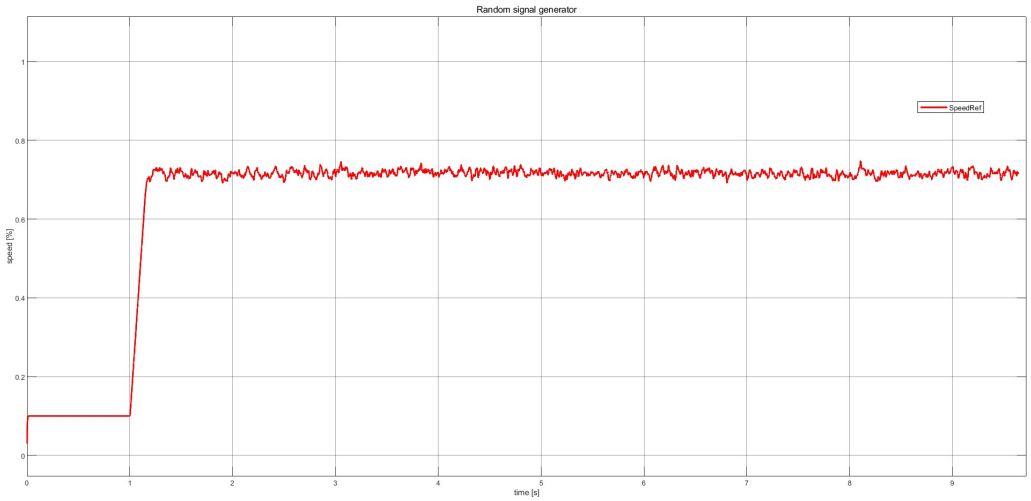

Figure 18: Test case of a random signal applied to the motor.

Once all the tests are applied to the control scheme, the resulting data array in the matlab script is brought to the the python environment. The data was saved as a *.mat array* which can be read in python so that the development of the neural network can continue. The obtained dataset is composed of 8 signals with 2.7 million data points each. A plot of the speed reference signal data is presented in Figure 19.

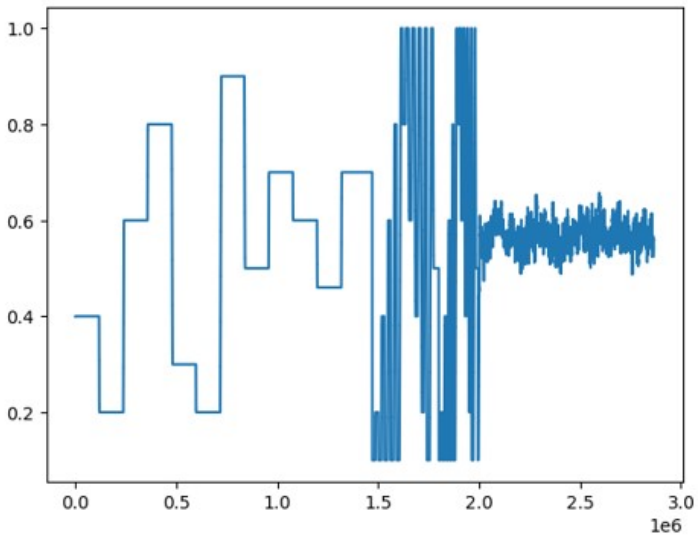

Figure 19: Dataset gathered from the motor model and simulation.

The full dataset was published with open access at Mendeley Data [3] and each test case is better explained in the related paper published while working on the thesis [9].

3.2. Neural Network development

Once the dataset has been obtained and brought to the python environment, the neural network architectures are developed and trained using Keras / TensorFlow. A Multi Layer Perceptron (MLP) architecture is developed, where the inputs of the network are the speed reference and speed error signal, and the output is the Iq current reference value that will be fed back to the internal control loop. The MLP is a type of neural network consisting of multiple layers of interconnected neurons based on feedforward propagation, meaning that data moves in a single path, without using any feedback loops. An input variable is represented by each node in the input layer, and a class label is represented by each node in the output layer. One or more layers of neurons in the hidden layers between the input and output layers alter the input data non-linearly before forwarding it to the following layer.

In an MLP, the neurons in each layer are fully connected to the neurons in the next layer. Each connection has a weight associated with it, which is adjusted during the training process to minimize the error between

the predicted output and the actual output. MLPs are commonly used in supervised learning tasks, such as classification and regression, where the goal is to learn a mapping between input features and output labels [5]. Figure 20 shows the mapping of the regression neural network that will replace the PID in the speed control loop of the FOC scheme.

Figure 20: MLP Neural Network for the speed loop.

The internal neural network structure is presented in Figure 21, where four layers are implemented: the input layer is composed of two neurons which receive the input signals (Speed reference, and error value), the second and third layers are the hidden layers of the neural network, this are dense layers composed of 8 and 4 neurons respectively. Finally, the output layer is defined by a single neuron, which has a linear activation function as is required for a regression approximation problem.

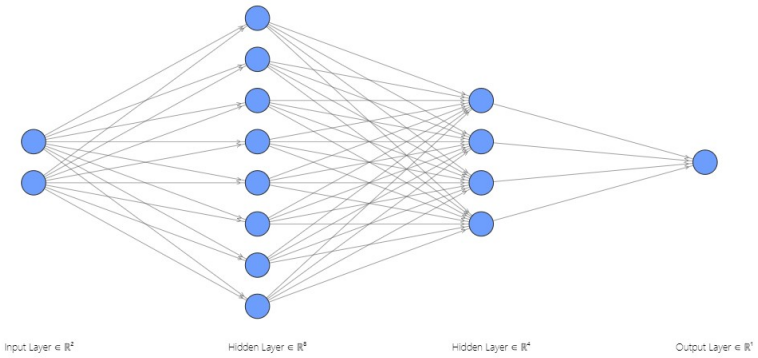

Figure 21: Internal Neural Network construction.

The Algorithm 1 shows how the network is defined inside the python environment by making use of the TensorFlow library. It is evident that the number of neurons in each hidden layer is larger than the ones presented in Figure 21, this is due to the fact that including the number of nodes in the figure example is reduced in order to ease the visual comprehension of the network architecture.

Algorithm 1 Generation of a MLP neural network

```
model = Sequential()
model.add(Dense(64, input_dim = 2, activation = 'relu'))
model.add(Dense(32, activation = 'relu'))
model.add(Dense(1, activation = 'linear'))
model.compile(optimizer='adam', loss='mse', metrics=['mse'])
```

Once the network model is defined and compiled, it is now able to be trained with the previously generated dataset, the data is split between training and validations sets by a 20% factor. The model is trained during 200 epochs while monitoring the mse_loss and accuracy in order to avoid overfitting the model. Figure 22 presents the learning curve of the network.

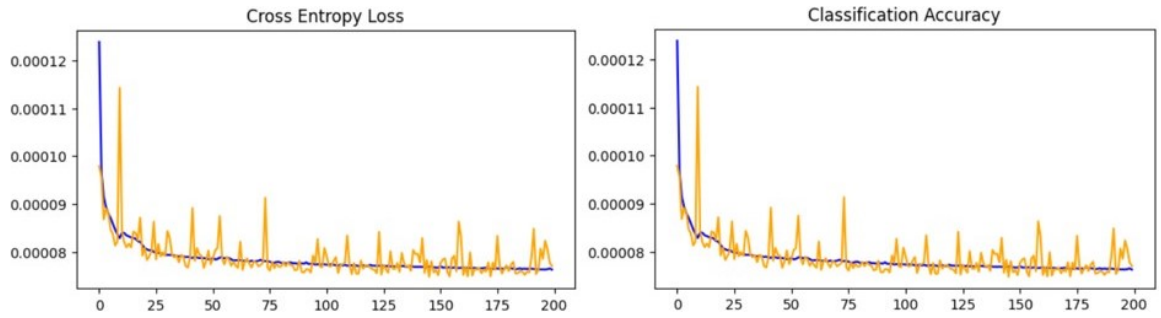

Figure 22: Learning curve of the neural network during training.

3.3. Network deployment on Simulink

The trained network is saved as an *.h5* file, storing its structure and weights, which can then be read by the matlab script in order to generate a network with the same architecture and weights as the one developed in python. The "predictor" block is used in simulink to read the imported neural network. The first task is to include the network inside the simulated control scheme just as a supervisor, while the control is still being carried out by the PID regulator as it is presented in Figure 23. The Network will produce an output that should be the same as the one generated by the PID as it was trained to replicate it.

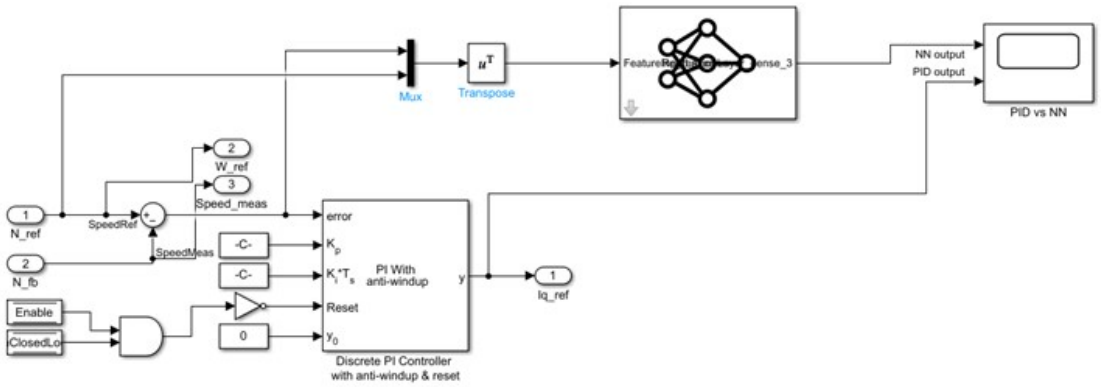

Figure 23: Neural network as a supervisor for the PID controller in the speed loop.

The output generated by the PID and by the neural network are comapred using the scope in Simulink in order to verify if the network is actually generating a control action when subject to the speed profile. Figure 24 shows that in fact, the NN output (blue), is similar to the output generated by the PID (red), however, it is not exactly the same and can lead to lack of control when the network is replacing the PID. The blue signal is replicating the red signal but is evident that some poles are skipped, wich could be corrected with a better training of the network.

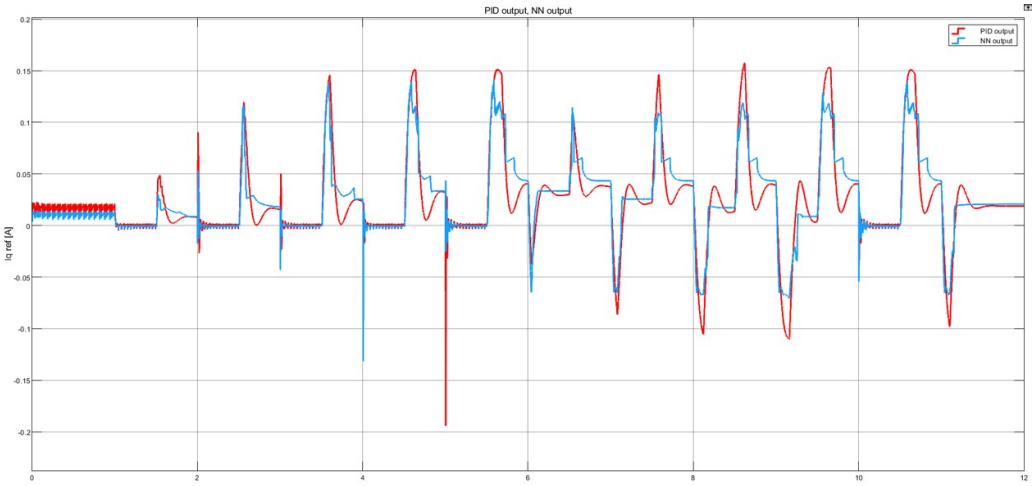

Figure 24: Neural network vs PID control output.

4. Control enhancement

To improve the control capabilities of the overall system, an adaptive neural network is proposed to compensate the overshoot generated by the PID regulator. The network is no longer replacing the PID but instead they both work to stabilize the system while the network is suppressing the overshoot. To achieve this control enhancement, the neural network is now defined as an optimization problem based on the results presented in [12] where the control of an induction motor was improved by the implementation of an "Adaline network". An ADALINE (Adaptive Linear Neuron) network is a type of artificial neural network that consists of a single layer of linear neurons. A linear activation function is applied to the weighted sum that results from multiplying the input values by a set of input weights in ADALINE neurons to produce the output. The weights are changed during training using the gradient descent-based supervised learning algorithm.

The weights for the ADALINE network are calculated using a recursive least square estimator fed by the motor speed and torque at 1 previous time instant, and the output signal is the motor speed in the current time instant as shown in Figure 25.

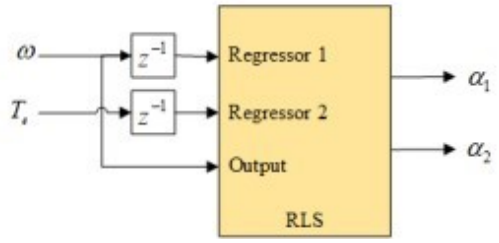

Figure 25: Recursive least square estimator.

The motor speed $\omega^*[k]$ can be obtained by the following weighted sum:

$$\omega^*[k] = \alpha_1 \omega[k-1] + \alpha_2 \Delta t[k-1], \tag{8}$$

Where α_1 and α_2 are the unknown parameters that will be estimated by the RLS. Equation 8 can be re-organized in terms of torque (which is the output generated by the PID inside the FOC control scheme) as follows:

$$\Delta\tau_{ff} = \frac{1}{\alpha_2}\omega[k] - \frac{\alpha_1}{\alpha_2}\omega[k-1], \tag{9}$$

The recursive least squares estimator is included in the simulink model simply by calling a pre-defined block, while the ADALINE network is implemented as a weighted sum following equation 9. In Figure 26 is shown the PID regulator generating the control output in the speed and the ADALINE network compenasting the produced torque value.

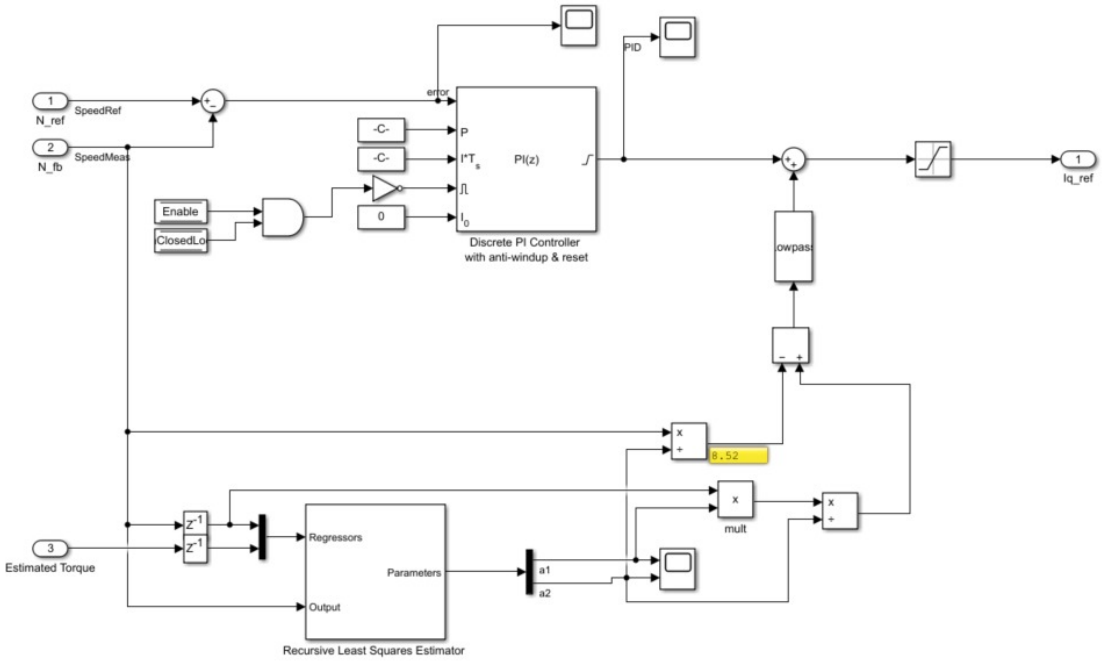

Figure 26: RLS and ADALINE network compensating the PID regulator.

Two control schemes are implemented in the simulation, one applying the base FOC scheme and another implementing the enhanced version of the control scheme in order to compare how both perform to the same input. An input signal is generated where two steps are applied to the motor to increase the speed.

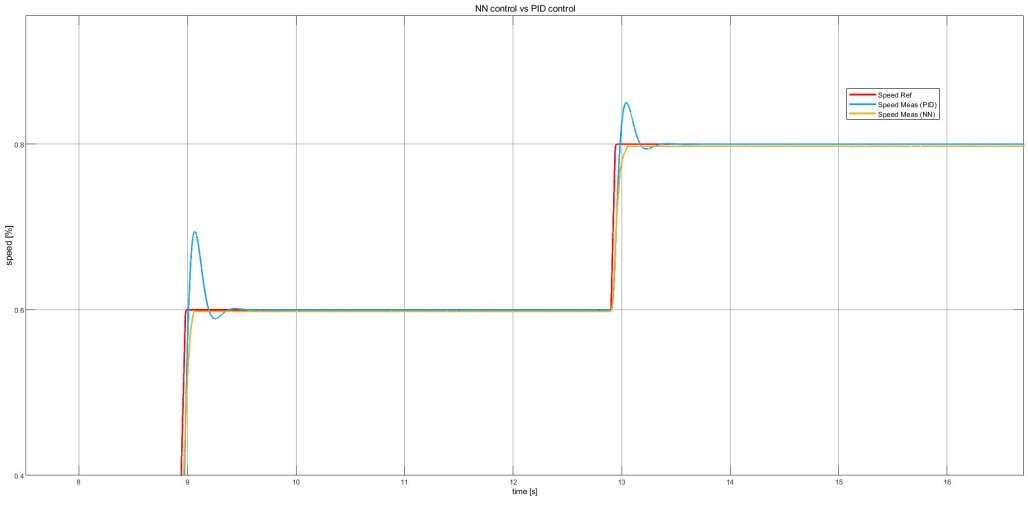

Figure 27: Neural network vs PID control output.

Figure 27 shows the speed comparison between the reference signal (red), the measured motor speed when controlled via PID (blue), and the motor speed when the FOC is enhanced with a NN (yellow). It is evident that both systems are able to stabilize the system, but the enhanced FOC scheme presents a better performance as the overshoot is no longer present.

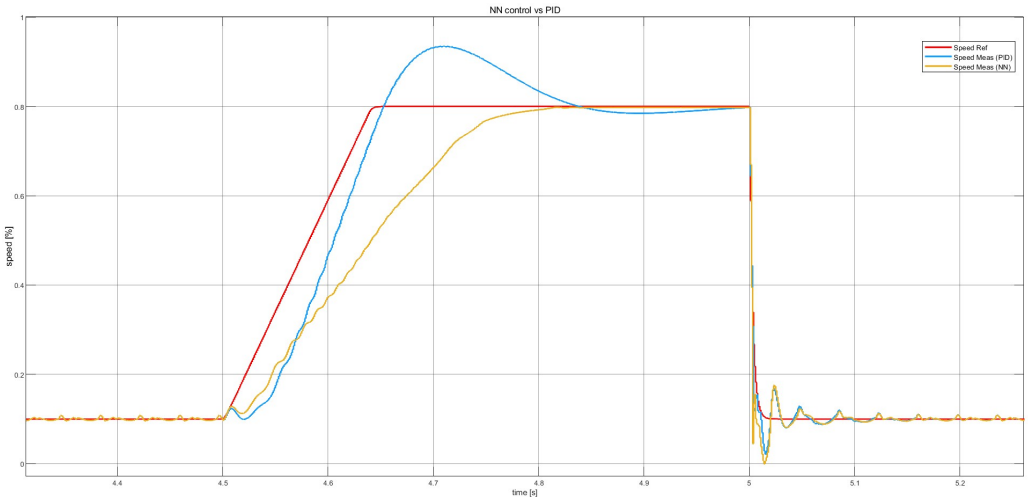

Figure 28: Neural network vs PID control output.

Figure 28 shows with higher detail the controller behavior when subject to a ramp input where it is evident that the overshoot is completely removed and the **settling time is reduced by 170ms which means a 40% improvement in the stabilization time**.

5. Conclusions

The enhanced version of the FOC scheme is able to stabilize the system at the target speed 40% faster than the classic FOC while also eliminating the overshoot generated by the PID. Further tests with different types of neural networks can be carried out in order compensate also non-linear operating conditions which are not supported by the ADALINE network. The simulated control scheme and motor model parameterized according to the actual motor parameters are of great help to test and verify different control schemes on various PMSM motors as it is just required to set the motor parameters to match the ones on the desired motor. The gathered and openly available dataset is of great use for the developing, training and validation of various types of neural networks as it eliminates the need on gathering data from a physical object while also being very vast as it includes both real-world and corner cases scenarios for the motor control.

References

[1] P. P. Acarnley and J. F. Watson. *Review of position-sensorless operation of brushless permanent-magnet machines.* in IEEE Transactions on Industrial Electronics, vol. 53, no. 2, pp. 352-362, April 2006, doi: 10.1109/TIE.2006.870868.

[2] Korkmaz F. et al. *Comparative performance evaluation of FOC and DTC controlled PMSM drives.* in: 4th POWERENG. May 2013, pp. 705–708.

[3] G. Gruosso J. Nustes, D. Pau. *PMSM_ FOC dataset.* in: Mendeley Data, V2, doi: 10.17632/6tjkgtfnky.2, (2023).

[4] Mathworks. Motor control blockset. url: https://www.mathworks.com/products/motor-control.html.

[5] F. Murtagh. *Multilayer perceptrons for classification and regression.* in: Neurocomputing, 2(5-6), 183-197, (1991).

[6] N. Adami S. Benini N. Federici, D. Pau. *Tiny Reservoir Computing for Extreme Learning of Motor Control.* in: 2021 International Joint Conference on Neural Networks (IJCNN), 2021, pp. 1–8.

[7] N. Kim Anh H. Than T. That Dong N. Khanh Quang, V. Q. B. Ngo and "Neural Network PID Controller for PMSM Drives N. Duc Tho. *Neural Network PID Controller for PMSM Drives.* 2022 7th National Scientific Conference on Applying New Technology in Green Buildings (ATiGB), Da Nang, Vietnam, 2022, pp. 146-149, doi: 10.1109/ATiGB56486.2022.9984109.

[8] D.O. Neacsu. *Space vector modulation - An introduction – Tutorial.* at IECON2001. In: (2001).

[9] Juan Camilo Nustes, Danilo Pietro Pau, and Giambattista Gruosso. Field oriented control dataset of a 3-phase permanent magnet synchronous motor. *Data in Brief*, 47:109002, 2023.

[10] TA Arafah Shawky Ramdan SG. Osman, Gomaa Dakrory. *Sensorless Field Oriented Control of Permanent Magnet Synchronous Motor.* 1.7. 2015.

[11] P. Pillay and R.Krishnan. *Modeling of permanent magnet motor drives*, volume 35, chapter 4, pages 537–541. in IEEE Transactions on Industrial Electronics, November 1988.

[12] R. Prasad. *Enhancing Speed Loop PI Controllers with Adaptive Feed-forward Neural Networks: Application to Induction Motor Drives.* at: 2022 25th International Conference on Electrical Machines and Systems (ICEMS), Chiang Mai, Thailand, 2022, pp. 1-6, doi: 10.1109/ICEMS56177.2022.9983335.

[13] S. Sengupta S. Chattopadhyay, M. Mitra. *Clarke and Park Transform.* In: Electric Power Quality. Power Systems. Springer, Dordrecht. (2011). https://doi.org/10.1007/978-94-007-0635-4_12.

[14] D. F. Specht. *A general regression neural network.* in IEEE transactions on neural networks 2.6 (1991): 568-576.

[15] D. F. Specht. *Probabilistic neural networks for classification, mapping, or associative memory.* in Proc. IEEE Int. Conf. Neural Nerworks, vol. 1, pp. 525-532, June 1988.

[16] STMicroelectronics. *ST Motor Control Workbench.* URL: https://www.st.com/en/embeddedsoftware/x-cube-mcsdk. html.

[17] Karl Johan Åström and Tore Hägglund. *Control PID avanzado.* Pearson, Madrid, 2009.

Abstract in lingua italiana

Questa tesi presenta uno schema di controllo orientato al campo implementato in Matlab/Simulink, nonché la modellizzazione di un motore sincrono a magneti permanenti che agisce come impianto da controllare. L'ambiente simulato viene utilizzato per generare dati con grandi capacità per addestrare e convalidare diverse architetture di reti neurali. Il dataset è raccolto dopo la progettazione di specifici profili di velocità applicati al motore al fine di verificare come il motore e lo schema di controllo si comportino di fronte alla comparsa di possibili non linearità che si verificano quando si verificano variazioni di velocità di grande magnitudine. Utilizzando le librerie TensorFlow in python, diverse reti neurali vengono addestrate e convalidate dal dataset raccolto, queste reti addestrate vengono quindi riportate nell'ambiente Matlab per completare il regolatore PID con l'obiettivo di migliorare le capacità di controllo complessive in termini di tempo di stabilizzazione e riduzione del sovraelongamento.

Viene implementata una versione migliorata dello schema FOC, in cui una rete neurale viene sviluppata per compensare la coppia generata con il regolatore PID. Questo migliorato schema di controllo è in grado di eliminare completamente il sovraelongamento generato dall'azione di controllo pur essendo in grado di stabilizzare il sistema. Inoltre, il controllo vede un miglioramento del 40% nel tempo di stabilizzazione.

Parole chiave: Controllo motori, Simulink, reti neurali, apprendimento automatico