Politecnico di Torino

ISTITUTO ITALIANO
DI TECNOLOGIA

Department of Control and Computer Engineering
Politecnico di Torino

Dynamic Interaction and Control Lab
Istituto Italiano di Tecnologia

Modeling, Identification and Control of Brushless Motors on Humanoid Robots.

Candidate **Giovanni Pizzolante**

Master's Degree thesis in Mechatronic Engineering

Academic Year 2020/2021, July 2021

Prof. Marcello Chiaberge	Politecnico di Torino	Supervisor
Dott. Daniele Pucci	Istituto Italiano di Tecnologia	Supervisor
Eng. Giulio Romualdi	Istituto Italiano di Tecnologia	Supervisor

Alla mia famiglia e
a MariaTeresa.

Abstract

A common approach for controlling humanoid robots is based on a hierarchical architecture composed by three different layers called: *trajectory optimization, simplified model control,* and *whole-body quadratic programming (QP) control*. When the *whole-body QP control* layer generates joint torques references, a reliable low-level joint torque control layer becomes pivotal.
The humanoid robot iCub is actuated by *permanent magnet synchronous motors (PMSM)*, thus the joint torque control inner layer should be composed by two nested control loop: a current and a torque control loop. This thesis aims at improving the overall torque control architecture of the humanoid robot iCub, by designing and implementing a low-level current vector controller as part of the fourth layer of the aforementioned architecture, that guarantees good current tracking performances. Firstly, an identification algorithm has been developed and tested for motor parameters estimation. Then, a current controller has been designed and implemented by exploiting the feedback linearization and the field oriented control approaches. Experiments in the Simulink environment and on the knee motor group of the iCub humanoid robot validate the proposed controller.

Acknowledgements

Sin dal primo giorno in cui ho iniziato a scrivere la Tesi, non mi sbagliavo sul fatto che questa sarebbe stata la sezione più difficile da completare.

Ai ragazzi del Lab.
Ovviamente ci tengo a ringraziare tutti i ragazzi del Lab che mi hanno supportato durante questa esperienza.
In particolare, ringrazio Giulio per la sua costante supervisione e soprattutto per non essersela mai presa (almeno per quello che so io) per il mio spirito un po' frizzantino. Lo ringrazio perchè mi ha aiutato molto durante questo percorso e mi ha insegnato davvero tanto... più di quanto immagina.
Ringrazio Dani perchè è sempre riuscito a ricavarsi del tempo da dedicarmi. Lo ringrazio per avermi dato fiducia, ma sopratutto per avermi trasmesso molta positività nei momenti in cui sembrava non esserci via d'uscita. Inoltre lo ringrazio per l' incredibile esperienza a Milano che davvero mi ha lasciato tanto.
Ringrazio Ines e Nuno per essere state delle figure di riferimento importanti quando mi sono trovato in difficoltà.
Un enorme grazie a Prajant, Giuseppe, Paolo, Antonello ed Italo per le magiche rovesciate che mi hanno concesso sia a San Giorgio che a Ronco Scrivia... ah no, scusate, era Serra Riccò.
Ringrazio il mitico e unico Valentino per i suoi regali culinari sempre apprezzati e soprattutto per avermi sopportato per gli ultimi 3 anni ... Vale mi dispiace che tu debba ancora continuare a sopportarmi.

Alla mia Famiglia.
Ed ora voglio ringraziare la mia famiglia per avermi supportato durante questi anni di formazione: senza di loro tutto questo non sarebbe stato possibile. Li ringrazio per avermi spinto a dare il meglio di me e per esserci stati nei momenti di gioia, e in quelli di difficoltà. Li ringrazio perchè i loro

sacrifici non ce li hanno mai fatti pesare, e li ringrazio per non averci fatto mai mancare nulla. Siete tutto.
Ringrazio Papà, per avermi dato quella dose di leggerezza quando serviva. Ringrazio Mamma, per avermi fatto sempre sfogare quando necessario. Ringrazio Francesco per tutti i "Giovà, nu mborta! È successo lo stesso anche a me!". Ringrazio Ale, per tutte le lezioni tecniche sulle galline, conigli, capre, e tutti gli animali dell'arca.
Un GRAZIE infinito va a mia Zia Lalla e Zio Carlo, Marilinda, Stefano, Andrea, e Antonietta, per avermi fatto sempre sentire a casa durante tutti questi anni, e per molto, molto altro. Non ci sono parole che possano esprimere il bene che vi voglio perchè Torino. Sappiate che Torino è casa soprattutto grazie a voi.
Ed un grazie va a Torino, per avermi fatto incontrare una persona speciale, con cui spero ancora di condividere tanto. Meri, ti ringrazio per tutto: per i sorrisi, per i pianti, per i momenti di paura, e per quelli di gioia, per i baci, per gli abbracci, per gli sguardi di complicità e anche per "gli sciarri"... ma soprattutto ti ringrazio per quello che verrà, che già so sarà incredibile.

E adesso ringraziatemi voi, perchè la mia famiglia è grandissima e andrebbero ringraziati tutti uno per uno, ma ve lo risparmio.

Contents

List of Tables

List of Figures

E per sognare poi qualcosa arriverà...
Pino Daniele

Chapter 1

Introduction

Human beings have always been enchanted by the opportunity to construct intelligent robots. One of the first man who starts imaging robots was the Greek author Apollonius of Rhodes. In "The Argonautica", one of his most famous epic poem written in the III century B.C., the author describes Talos that is a giant automaton made of bronze whose aim is to defend Crete from the invaders.

The dream of Apollonious become true in the 1495, when Leonardo Da Vinci creates the Leonardo's mechanical knight shown in Figure 1.1.

It is curios noticing that both Talos and Leonardo's Knigth are humanized

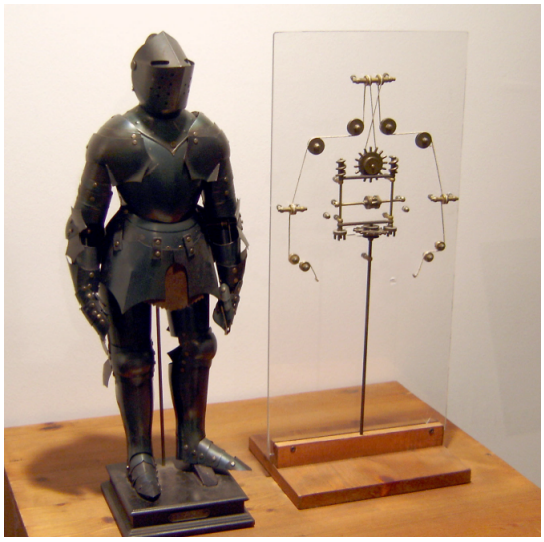


Figure 1.1: Leonardo's mechanical knight

automatons. So, the idea of autonomous systems cooperating with humans goes back to Apollonius times, the Czech author Karel Capek was the pioneer of the term "robot" in 1921. The first modern autonomous systems were not humanoids because their very first applications were inside factories, where the complexity of designing humanoids were meaningless. In 1961 *Unimate* was born: the first industrial robot operating at General Motors created for working with heated die-casting machines. *Unimate* gained popularity in a such way that from that moment robots started becoming a fundamental resource in the assembly line of the major companies. The most striking features of this robots are very high accuracy in movements and the capability of moving heavy objects. For this reasons industrial robots are a good substitute of human workers that do not possess such skills. Mobile robotics is gaining more and more popularity in companies as well. This because they are useful for intralogistics purpose. In fact, wheeled robots can easily move in an environment free of obstacles, such as a warehouse, but not only. Their usage could be extended for householding as vacuum cleaners, or for performing more complicated tasks such as washing dishes or serve dinner. Examples of wheeled robots are shown in Figure 1.2a, and 1.2b that shows respectively a vacuum cleaners produce by iRobot® and the humanoid robot $R1$ produced by the Italian Institute of Technology.

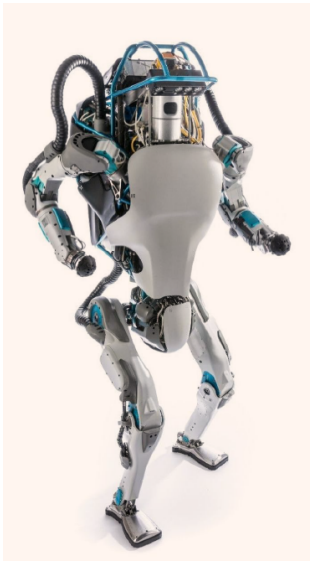
Even if wheeled robotics offers an intrinsic stability and simplify the control problems, it is strongly limited in a high structured environment. Thus the necessity of a more reliable and versatile locomotion becomes fundamental. Legged robots are more versatile than wheeled ones because they are equipped

(a) Vacuum Cleaner - produced by iRobot®

(b) R1 - produced by IIT

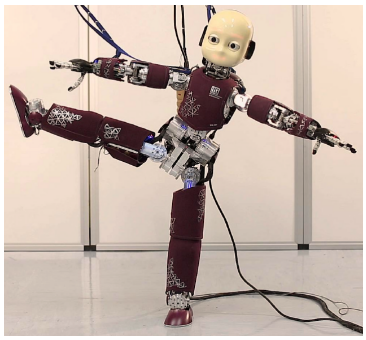
Figure 1.2: Wheeled Robots

with articular limbs to provide locomotion. The big advantage of such a mechanism is that they are able to traverse many different terrains, but the price to pay is that very high complexity and power consumption are required. Different legged humanoid robots have been designed over time, and among those in Figure 1.3, are shown Atlas produced by Boston Dynamics®, Asimo produced by Honda, and iCub2.5 produced by IIT.



(a) Atlas - produced by Boston Dynamics®

(b) Asimo - produced by Honda®



(c) iCub2.5 - produced by IIT

Figure 1.3: Bipedal Humanoid Robots

1.1 Thesis Contribution

Robust bipedal locomotion of humanoid robots is still an active research domain among the scientific community. Several factors make this a challenging topic: the terrain unpredictability, the complexity of the robot dynamics and the low efficiency of the actuation systems. During the DARPA Robotics Challenge, a layered control architecture for generating walking patterns was commonly adopted – Fig. 1.4. Each layer aims at generating references for the layer below both by processing inputs from the robot and the surrounding environment, and from the outputs of the layer before. The main layers are known as: *trajectory optimization, simplified model control, and whole-body quadratic programming (QP) control* [Romualdi et al. [2020]]. The *trajectory optimization* layer aims at generating the desired feet trajectory by means of optimization techniques. The *simplified model control* layer is in charge of finding feasible center-of-mass (CoM) trajectories considering simplified dynamicals models. The *whole-body QP control* layer uses a complete robot model to produce either desired positions, velocities, or torques inputs at the joint-level. In between the *whole-body QP control* and *the robot*, there is a low level inner layer aiming at driving the actuation system of the robot – Fig. 1.4. Recently, the scientific community has been interested in the possibility of using torque control based algorithms to perform locomotion tasks. Indeed torque-controlled robots have several advantages over position or velocity controlled ones. A torque-controlled humanoid robot is, in fact, intrinsically compliant in case of external unexpected interactions, and it can be also used to perform cooperative tasks alongside humans.

In this work, we focus our interest on the iCub humanoid robot [Metta et al. [2010]]. The actual torque-controlled architecture implemented on the humanoid robot iCub allows the robot to walk in a rigid and plane terrain. However, the weaknesses of this architecture have their roots in the fact that iCub is not equipped with joint torque sensors since it has originally been designed to be position controlled. As a consequence, the joint torques are estimated by exploiting the robot dynamics model and the readouts of the force/torque (F/T) sensors, so that the performances of this architecture are strictly related to the uncertainties of the model and the reliability of the F/T sensor's measurements. Therefore, this makes difficult to use a torque-control architecture in a real scenario. So when the *whole-body QP* control layer generates joint torques, a reliable low-level joint torque controller becomes pivotal.

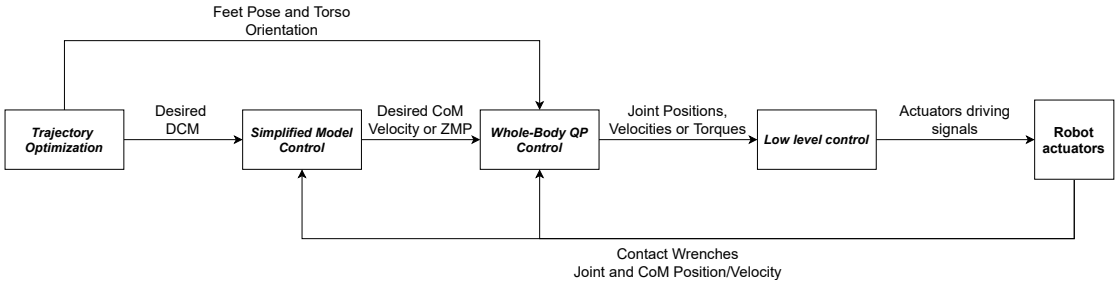


Figure 1.4: Layer-Based Architecture.

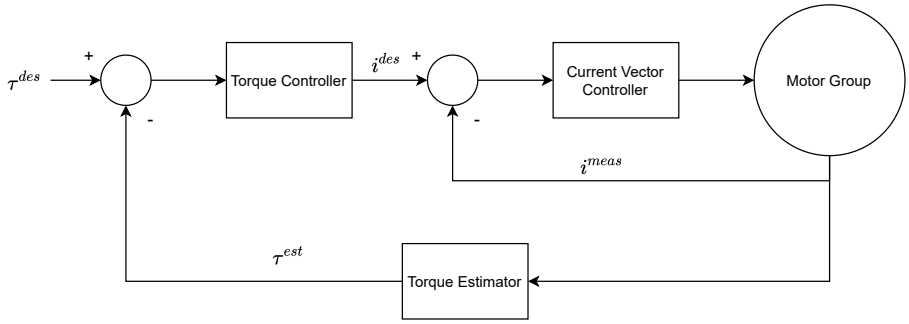


Figure 1.5: The fourth layer

This thesis aims at improving the overall torque control by designing and implementing a low-level current control that enhance the joint torques tracking performances. The proposed low-level controller actually is the fourth layer of the aforementioned architecture – Fig. 1.4. Considering that the main joints of iCub are actuated by motor groups composed by *permanent magnet synchronous motor (PMSM)*, this inner layer is composed by two controllers: namely a *torque controller* and a *current controller* – Fig. 1.5. The *torque controller* aims to generate a desired motor current considering the mechanical model of the motor. Finally the *current controller* is in charge of adjusting the motor voltages to stabilize the electrical dynamics of the motor along the desired trajectory.

The contribution is directed toward the modeling, identification, and control of a permanent magnet synchronous motor (PMSM) mounted on the iCub humanoid robot leg. More precisely, the contributions of this work follow. i) The system identification of the parameters related to the the electrical dynamics of the PMSM. ii) A low-level controller that guarantees good current tracking performances. The low level controller has been

designed by exploiting the field oriented control approach and the gains has been chosen by considering the identified model. iii) A validation of the approach both in simulation and on a test-bench.

1.2 Thesis Structure

This thesis is structured as follows:

- in Chapter 2, a full description of the humanoid platform iCub on which the contribution is integrated is provided. Particular focus is on the motor group and the control mode already implemented
- in Chapter 3, an overview on the electrical machines with particular emphasis on the PMSM machine is given. Then the dynamics of the SPM motor is illustrated
- in Chapter 4, the main concept of the field oriented control is briefly discussed together with the practical implementation in both the (a,b,c) and (d,q) frame
- in Chapter 5 the experimental setup used for the experiments and simulation enviroments built for validation and control purposes are described
- in Chapter 6, the identification algorithm for estimating the motor parameters and the technique used for solving conditioning problems are discussed. Then the estimation results and simulation that validate the approach are shown.
- in Chapter 7, the low level controller is designed. Firstly, an introduction on the control approach used, then the control law implemented and the additional tool needed for guaranteeing a good current tracking performances are shown. Finally, the experimental results obtained are commented.

Chapter 2

iCub: the Humanoid Robot

This chapter aims to briefly present iCub: the humanoid robot platform used to test the control algorithms illustrated in the Chapter 7. Particular attention is given to the motors and to the already implemented control modes.

iCub was built by the Italian Institute of Technology that is a scientific research centre based in Genova. The robot was involved into the RobotCub european project which main goal was to develop an embodied robotic child (iCub) with the physical (height 104 cm and mass n 33 kg) and ultimately cognitive abilities of a 2.5-year-old human child [Tsagarakis et al. [2007]]. The iCub humanoid robot is an enterely open-source platform created for research in cognitive development[Metta et al. [2010]]: the platform is open both in software but more importantly in all aspects of the hardware and mechanical design. It is fascinating to know that the robot is used by more than 20 laboratories in Europe, US, Korea and Japan. In this work, the version "iCub 2.5" has been adopted for the experiments made.

2.1 Hardware

The iCub humanoid robot has 53 degrees of freedom (DoF) that are allocated as follow:

- **Upper body** 38 DoF (hands, arms and head):

- 9 DoF fo each hand
- 7 DoF for each arm
- 6 DoF for the head (3 for the neck and 3 for eyes)

- **Lower Body** 12 DoF (legs):
  - 3 DoF for each hip
  - 2 DoF for each ankle
  - 1 DoF for each knee

- **Torso** 3 DoF

The kinematic structure of the robot is shown in Figure 2.1, where the joints of both hands and eyes are not represented for the sake of having a not confusing representation.

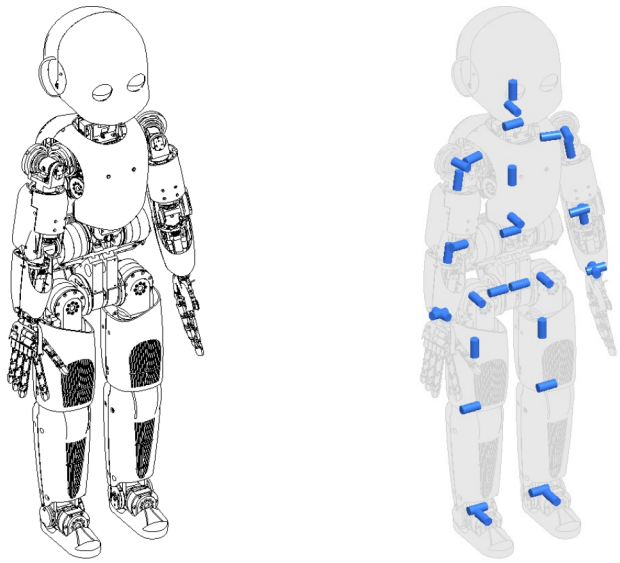


Figure 2.1: iCub kinematic structure

2.1.1 Motor group

In the following dissertation, we consider only the DoFs actuated by motor groups composed by a Permanent Magnet Synchronous motors (PMSM) and an harmonic drive. These motors exhibit robustness, higher power density,

and higher torque and speed bandwidths when compared with conventional DC brushed motor [Tsagarakis et al. [2007]]. The harmonic drive allows to have no backlash and high reduction ratio. The motor group is equipped with an high resolution encoder, suitable for implementing the field oriented control for controlling the current, thus the torque. The joint motor group used for testing the control algorithms implemented in this work is based on the MOOG C2900584 motor and a CSD-17-100-2A Harmonic Drive. It is capable of delivering $40Nm$ of torque. The motor group is represented in Figure 2.2. For the sake of this work it is important to state that the robot is not equipped with torque sensors at the joint level, thus estimation techniques for evaluating the joint torques are necessary. The presence of the current sensors on each phase, allows to close the loop in current, as discussed in Chapter 4.

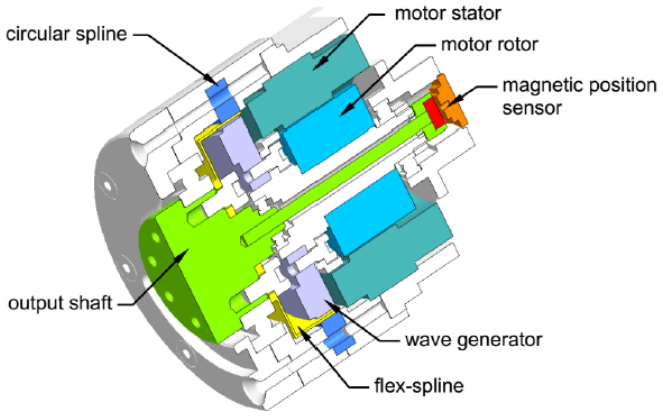


Figure 2.2: Motor Group cross section

2.1.2 Boards

iCub2.5 system architecture is shown in Figure 2.3. In this subsection the main boards of such architecture are briefly discussed. An on-board COMEXPRESS by CONGATEC CPU aims to run the high level controller for controlling the robot. The central unit sends the desired state to the Ethernet Motor Supervisor (EMS) exploiting an Ethernet bus. The EMS controller runs at 1kHz and it converts the desired reference into a PWM set point that the motor driver applies to the motor windings. The motor

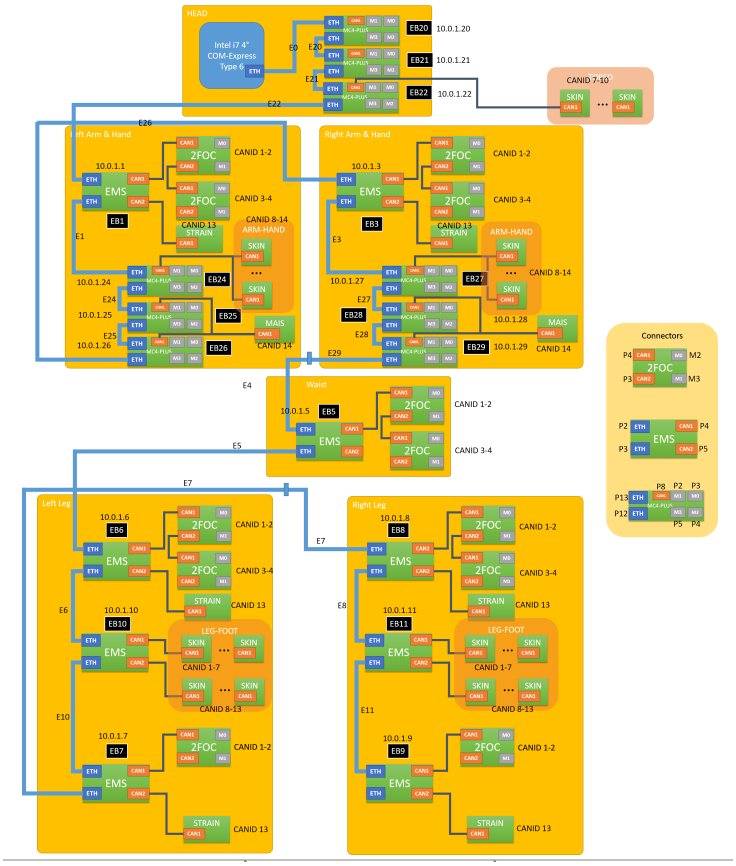


Figure 2.3: System Architecture - iCub2.5

driver is the 2FOC board, that is able to drive two motors at time. The 2FOC driver control algorithm, generats PWM signals at 20kHz that feed the motor. These two boards communicate between each other by means of a CAN communication protocol.

2.1.3 Sensors

The robot also integrates a huge number of sensors. As already said, each actuation group integrates position sensors: both an incremental rotary position sensor (Hall effect sensors) to measure the angular variation, and an absolute 12 bit angular encoder (AS5045 from Austria Microsystems) for measuring the joint absolute angular position are integrated. iCub 2.5 is also equipped with a distributed tactile skin for perceving external forces applied on its body (Figure 2.4). Another important integrated sensor is the

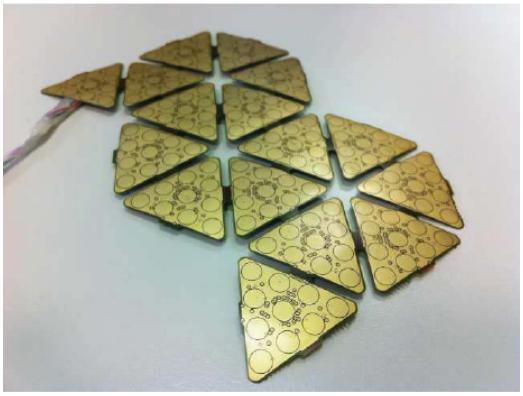


Figure 2.4: Artificial skin

force/torque sensor (F/T) used for estimating the internal torques and the external wrenches [Fumagalli et al. [2012]], that are fundamental information for the walking algorithm. Furthermore, iCub uses IMUs, two cameras, and microphones.

2.2 Software Architecture

The communication with the robot is provided by means of YARP (Yet Another Robot Platform): an open-source, multiplatform software middleware. YARP is written by and for researchers in humanoid robotics, who has to deal with the control of a large quantity of complex hardware and an equally complicated pile of software [Metta et al. [2006]]. More specifically, YARP is a set of C++ libraries, protocols and tools, needed to develop a number of programs that interact in a peer-to-peer way. YARP provides an abstraction layer to communicate with physical devices through the YARP Device Drivers: C++ classes for abstracting the functionality of robot devices. Those drivers implements one or more YARP Interfaces that are C++ libraries containing the attributes and methods definition. The interfaces (such as IPositionControl, IPWMControl, etc) can be interfaced directly with C++ applications or using the bindings in Simulink and MATLAB. Then, YARP drivers and interfaces are wrapped by means of the YARP Wrappers that use the network resources to satisfy their interfaces. YARP framework main purpose is to increase modularity and re-usability, fundamentals for maximizing the research progress.

2.3 Control Architecture

The Control Architecture on iCub can be differentiated by an high level control architecture and a low level one. The high level controller runs at 100Hz. It could send trajectories in position, velocity or torque depending on which kind of low level control mode is set. The EMS receives the high level reference, and then it sends to the 2FOC board the respective PWM set point. The motors could also be directly feed with a PWM set point, resulting in an open loop configuration.

2.3.1 Position Control

The control architecture of the position control mode is shown in Figure 2.5. From the high level controller, a trajectory in position is sent and the architecture provides only a feedback term closed on the measured joint position. The peculiarity of this control mode is that the trajectory is tracked by implementing a null jerk behaviour.

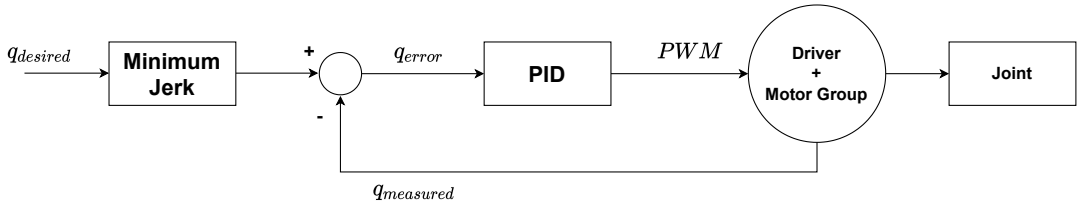


Figure 2.5: Position Control Architecture

2.3.2 Position Direct Control

The Position Direct Control mode (Figure 2.6) differs from the previous one only in the minimum jerk block: the desired trajectory is tracked without imposing any kinematic constraint on the acceleration.

2.3.3 Velocity Control

The Velocity control architecture is similar to the position one, in fact, also in this case any feed-forward action is present. As it can be seen in Figure 2.7, the loop is closed on the joint velocity.

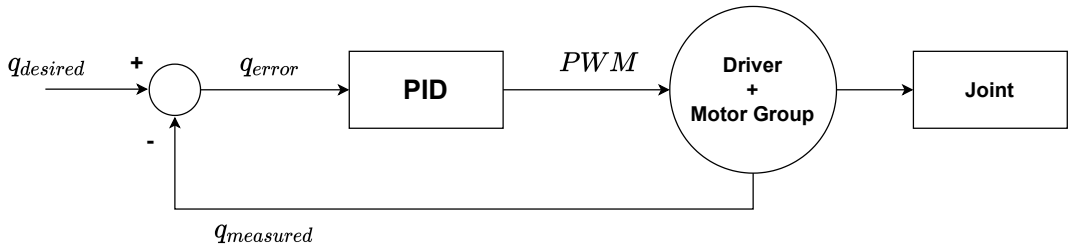


Figure 2.6: Position Direct Control Architecture

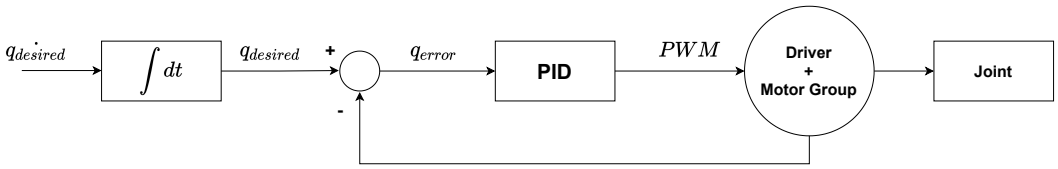


Figure 2.7: Velocity Control Architecture

2.3.4 Actual Torque Control

The Torque control architecture is shown in Figure 2.8. It is evident that, among the control modes described, this is the most complex one. This control mode allows to achieve torque based control exploiting the low-level PWM control. More specifically, there is a desired joint torque coming from the high level controller compared with the joint torque estimated from the whole-body model. A PID controller combined with a feed-forward action aims to generate a PWM set point for obtaining that desired torque. As already stated in Section 1.1, the estimation of the joint torques is crucial, and in this case is not reliable.

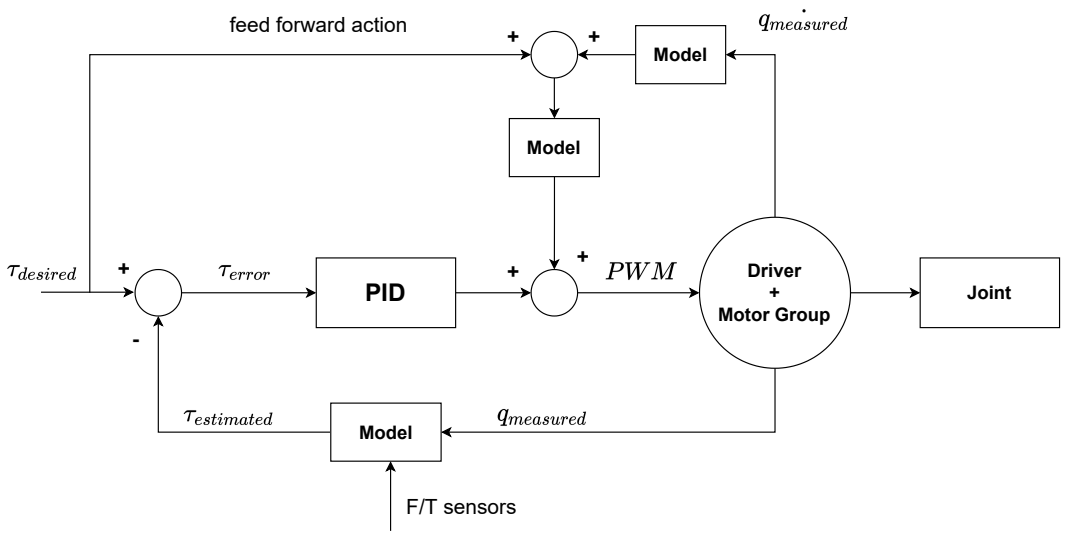


Figure 2.8: Torque Control Architecture

Chapter 3

Modeling iCub Motors

The motors mounted on iCub2.5 are all three phase surface mounted permanent magnet synchronous motors (SPM-PMSM) with distributed windings, star connected. In this Chapter, firstly an overview of how the electrical machines are classified is illustrated. Then the dynamic model of the SPM motor in the direct and quadrature frame is derived.

3.1 Classification of electrical machines

Electrical machines are largely used and their applications are destined to grow over time. At the same time, the required performance in terms of power, readiness, and precision in control are also increasing. Here, it is underlined the main differences between brushed and brushless machine and synchronous and asynchronous machines.

3.1.1 Brushed/Brushless and Synchronous/Asynchronous

The electrical machines can be generally categorized into two major groups, namely the brushed machines, and the brushless machines. The latter can enjoy the absolute advantage of maintenance-free operation so that these types of machines have become the major trend since the last few decades (Lee et al. [2017]). Furthermore, these kinds of machines show excellent controllability, high precision positioning, smooth running at low speeds, and very rapid accelerations and decelerations. Brushless machines also have good overload capacity, high torque, and can withstand strong impulsive stress despite the fact that are compact in size. All these advantages are

at the expense of fairly high costs and the indispensable presence of the drive. By the classification of machine materials, the brushless machines can be further divided into two subgroups: the Permanent-Magnet (PM) brushless machines, and the magnetless brushless machines. Moreover, we can distinguish between synchronous and asynchronous brushless machines considering the velocity of the rotor as rigidly related to the AC frequency imposed by the inverter in the first case. The asynchronous machines, instead, are characterized by the fact that the rotor speed slips with respect to the AC frequency imposed by the inverter.

$$\omega_m = \frac{2\pi f}{p}, \text{ Synchronous machines} \tag{3.1}$$

$$\omega_m \leq \frac{2\pi f}{p}, \text{ Asynchronous machines} \tag{3.2}$$

where in (3.1) and (3.2), ω_m, f, and p are respectively the rotor velocity, the AC frequency of the phase currents and the number of poles pair of the motor.

3.2 Permanent Magnet Synchronous Motors (PMSM)

As the name suggests, permanent magnet synchronous motors (PMSM) belong to the synchronous AC motors. A PMSM, like any rotating electric motor, consists of a rotor and a stator where the former is the fixed part and the latter the rotating part. Typically, the rotor is located inside the stator, but there are also structures that implements the rotor as the more external part.

The PMSM motors can be classified depending on the type of rotor and stator windings. In particular, based on the rotor type, they can be classified as:

- Surface Mounted Permanent Magnet (SPM)
- Interior Permanent Magnet (IPM)

while based on the stator windings they can be classified as:

- Distributed windings

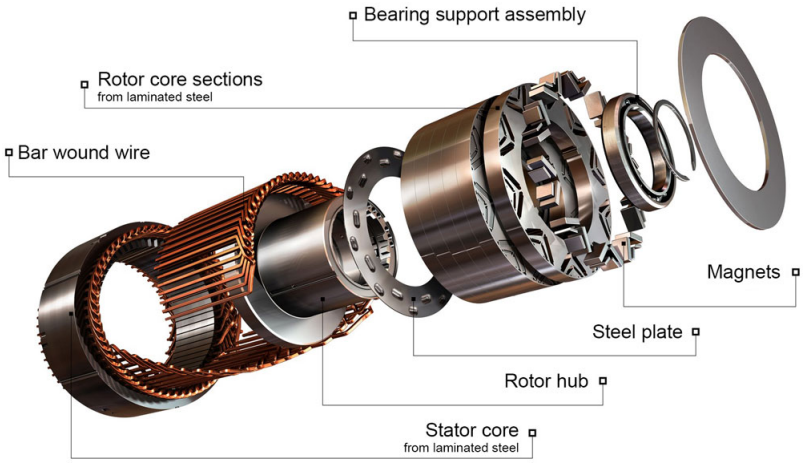


Figure 3.1: Structure of an Interior Permanent Magnet Synchronous Motor

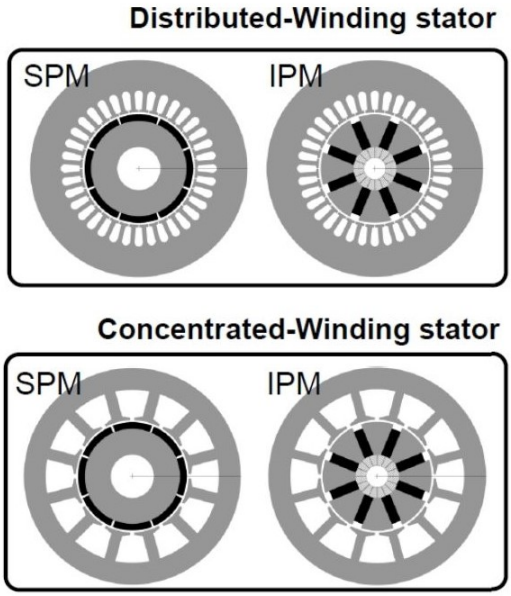


Figure 3.2: PMSM classification

- Concentrated windings

Independently of the stator windings, the rotor design imposes the motor name, as shown in the figure 3.2.

The stator laminations have slots for housing the stator windings, while the rotor can have holes for housing the permanent magnets in various

configurations and this is the case of the IPM motors. The magnets of the SPM motors are located on the surface of the rotor, thus the rotor does not present any slots. The two winding configurations produces magnetic fields of "different quality". In the case of distributed windings, the stator produces a nearly sinusoidal field distribution at the airgap; however, it has higher manufacturing cost and longer end-windings that implies more material and Joule losses and an increased axial lenght. The concentrated windings types are cheaper but they produce harmonics fields and they have another kind of application such as direct drive and safety-critical operations.

3.3 Modeling PMSM - Fundamentals

Vector control theory allows transforming the model of the three-phase machine into a model of a “fictitious” bi-phase machine using a proper transformation of variables (Clarke and Park transforms). The model of the AC machines gets the simplest expression in the rotating (d, q) frame, where for balanced operation with sinusoidal phase quantities, the (d, q) quantities become DC. For the synchronous machines, the (d, q) frame is the rotor frame. In this section, firstly the Clarke and Park transforms are illustrated and then the electric and magnetic equations both in the (a, b, c) and (d, q) frames are derived.

3.3.1 Clarke and Park transforms

The behavior of three-phase machines is usually described by their voltage and current equations. The coefficients of the differential equations that describe their behavior are time varying (except when the rotor is stationary). The mathematical modeling of such a system tends to be complex since the flux linkages, induced voltages, and currents change continuously as the electric circuit is in relative motion. For such a complex electrical machine analysis, mathematical transformations are often used to decouple variables and to solve equations involving time varying quantities by referring all variables to a common frame of reference. Among the various transformation methods available, the well known are:

- Clarke Transformation
- Park Transformation

Basically, the three reference frames considered in this implementation are:

1. Three-phase reference frame, in which x_a, x_b, and x_c are co-planar three-phase quantities at an angle of 120 degrees to each other.

2. Orthogonal stationary reference frame, in which x_α (along α axis) and x_β (along β axis) are perpendicular to each other, but in the same plane as the three-phase reference frame.

3. Orthogonal rotating reference frame, in which x_d is at an angle θ (rotation angle) to the α axis and x_q is perpendicular to x_d along the q axis.

The three reference frames are shown in figure 3.3.

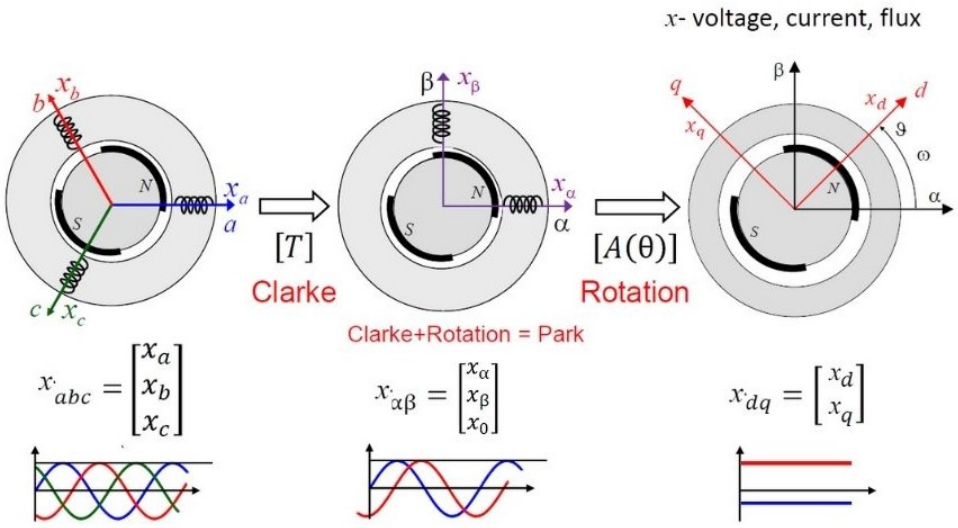


Figure 3.3: Clarke and Park transformations

Clarke: (a, b, c) to (α, β, 0)

The Clarke transformation converts balanced three-phase quantities into balanced two-phase quadrature quantities. This transformation is done by using a 3x3 matrix whose power non invariant form is:

$$T = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{3.3}$$

The zero-sequence components x_0, are not involved in the electromechanical energy conversion, thus it is possible to model the machine behavior with just the (α, β) components.

Park: (α, β) to (d, q)

The Park transformation converts vectors in balanced two-phase orthogonal stationary system into orthogonal rotating reference frame. This transformation is done by usign the following 2x2 matrix:

$$A = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \tag{3.4}$$

The (d, q) reference frame rotates with the synchronous speed w_m defined in 3.1.

From (a, b, c) to (d, q): transformation

By considering the transformations 3.3, 3.4, the complete transformation from the three-phase reference frame (a, b, c) to the rotating bi-phase one, is expressed by the following equations:

$$\begin{aligned} x_\alpha &= \frac{1}{3}[2(x_a - x_b) + (x_b - x_c)] \\ x_\beta &= \frac{\sqrt{3}}{3}(x_b - x_c) \end{aligned} \tag{3.5}$$

$$\begin{aligned} x_d &= x_\alpha cos\theta + x_\beta sin\theta \\ x_q &= x_\beta cos\theta - x_\alpha sin\theta \end{aligned} \tag{3.6}$$

where θ is the electric angle, that in the case of one pole pair coincides with the rotor angle θ_m, but if the rotor has p pole pairs than the electric angle θ_e becomes:

$$\theta_e = \theta_m p \tag{3.7}$$

3.3.2 Electrical and Magnetic equations - (a,b,c) frame

The three-phase machine presents three windings that cause the creation of the back electromotive force and the linkage of the fluxes. It is possible to define for each winding an electric equation and a magnetic one. Since the machine has three windings, it is possible to write six equations. Considering the line to neutral (start center) voltage of the x-th winding as V_x, it is possible to write the electric equation, also known as voltage equation as the sum of two components:

- contribution due to the stator phase resistance R_s
- contribution due to the variation over time of the total flux linkage λ_x with the x-th winding

$$\begin{aligned} v_a &= R_s i_a + \frac{d\lambda_a}{dt} \\ v_b &= R_s i_b + \frac{d\lambda_b}{dt} \\ v_c &= R_s i_c + \frac{d\lambda_c}{dt} \end{aligned} \tag{3.8}$$

The stator windings model is represented in figure in 3.4.

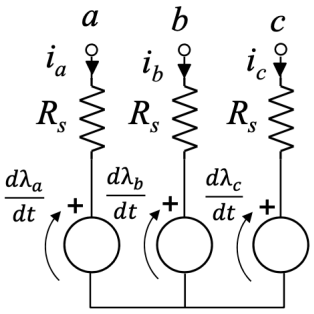


Figure 3.4: Stator Windings model

As regard the magnetic equations, on each winding, the total flux linkage is the sum of the contribution of the self and mutual inductances plus the contribution of the flux linkage due to the permanent magnets. The magnetic equations representing the total flux linkage on each phase are:

$$\underbrace{\begin{bmatrix}\lambda_a\\ \lambda_b\\ \lambda_c\end{bmatrix}}_{\lambda} = \underbrace{\begin{bmatrix}L_{aa} & L_{ab} & L_{ac}\\ L_{ba} & L_{bb} & L_{bc}\\ L_{ca} & L_{cb} & L_{cc}\end{bmatrix}}_{\mathbf{L}} \underbrace{\begin{bmatrix}i_a\\ i_b\\ i_c\end{bmatrix}}_{\mathbf{i}} + \underbrace{\begin{bmatrix}\lambda_m cos(\theta)\\ \lambda_m cos(\theta - \frac{2\pi}{3})\\ \lambda_m cos(\theta + \frac{2\pi}{3})\end{bmatrix}}_{\lambda_{\mathbf{PM}}} \tag{3.9}$$

The model described by the equation 3.9 is known to be the magnetic model of the synchronous machine and it states that the total phase flux linkage depends on all stator currents and on the rotor position. It is important to underline that λ_m that is the magnet flux linkage, is temperature dependent, but in this work it is approximated to be constant. To exploit the general magnetic model in the phase coordinates, the rotor anisotropy is assumed such that the magnetizing inductances expressed in L are not constant by depend on the rotor position. By assuming sinusoidal winding distributions, the self magnetizing inductances are depending on 2θ:

$$\begin{aligned}
L_{aa} &= L_{avg} + L_{\Delta} cos(2\theta)\\
L_{bb} &= L_{avg} + L_{\Delta} cos(2\theta + \frac{2\pi}{3})\\
L_{cc} &= L_{avg} + L_{\Delta} cos(2\theta - \frac{2\pi}{3})\\
L_{ab} = L_{ba} &= -\frac{1}{2} L_{avg} + L_{\Delta} cos(2\theta - \frac{2\pi}{3})\\
L_{ac} = L_{ca} &= -\frac{1}{2} L_{avg} + L_{\Delta} cos(2\theta + \frac{2\pi}{3})\\
L_{bc} = L_{cb} &= -\frac{1}{2} L_{avg} + L_{\Delta} cos(2\theta)
\end{aligned} \tag{3.10}$$

where L_{avg} is the average inductance and L_{Δ} is the differential inductance due to the rotor anisotropy.

The magnetic model in the phase coordinates it is too complex: a transformation into a bi-phase reference frame is needed in order to reduce the complexity of the model.

3.3.3 Electrical and Magnetic equations - (d, q) frame

To derive the machine model in the direct and quadrature frame, first it is needed to apply the Clarke transformation to the voltage equation 3.8, obtaining the equations in the (α, β) frame:

$$\underbrace{T\begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix}}_{\mathbf{v}_{\alpha,\beta}} = R_s \underbrace{T\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}}_{\mathbf{i}_{\alpha,\beta}} + \frac{d}{dt}\underbrace{T\begin{bmatrix} \lambda_a \\ \lambda_b \\ \lambda_c \end{bmatrix}}_{\lambda_{\alpha,\beta}} \tag{3.11}$$

where the homopolar components x_0 are not considered since, as stated before, they are not involved in the energy conversion. Then, the Park transform is applied to the (α, β) quantities, deriving the voltage equations expressed in the (d, q) frame:

$$\begin{aligned} v_d &= R_s i_d + \frac{d\lambda_d}{dt} - \omega_e \lambda_q \\ v_q &= R_s i_q + \frac{d\lambda_q}{dt} + \omega_e \lambda_d \end{aligned} \tag{3.12}$$

where ω_e is the electric pulsation. The equations 3.12 are valid for all the synchronous machines. The next step, is to derive the current-to-flux relationship, also know as the magnetic model, in the (d, q) frame by substituting 3.10 in 3.9 and by applying the Clarke and Park transformations. The obtained model is the following:

$$\begin{bmatrix} \lambda_d \\ \lambda_q \end{bmatrix} = \begin{bmatrix} L_d & L_{dq} \\ L_{qd} & L_q \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} \lambda_m \\ 0 \end{bmatrix} \tag{3.13}$$

where:

- L_d is the d-axis inductance
- L_q is the q-axis inductance
- L_{dq} and L_{qd} are cross-saturation inductances
- λ_m is the magnet flux

This magnetic model can be used to derive the magnetic models of all the synchronous machine.

3.4 Dynamics of an SPM motor

In this section, the equations governing the dynamics of the SPM motors are derived. Moreover, since the aim of this work is to design a current control loop, it is fundamental to develop a simulation environment which guarantees the testing of the algorithms designed. In this section only the simulink model of the motor it is briefly discussed.

3.4.1 Electrical dynamics in the (d, q) frame

The ordinary differential equations in the (d, q) frame of the SPM motor can be derived by considering that the machine is theoretically isotropic and this means that the direct and quadrature inductances have the same value:

$$L_d = L_q = L_s \tag{3.14}$$

and L_s is called synchronous inductance. In this work, the machine has been considered isotropic, but since the parameters have been also identified, it has been preferred to consider the direct and quadrature inductances as separate quantities. Furthermore, since in SPM motors no cross-saturation effects happen, then it can be said that

$$\begin{aligned} L_{dq} &= 0 \\ L_{qd} &= 0 \end{aligned} \tag{3.15}$$

Now, by substituting 3.13 into 3.12 the electric dynamics of an SPM motor is derived, as shown in the equation below:

$$\begin{aligned} v_d &= R_s i_d + L_d \frac{di_d}{dt} - \omega_e L_q i_q \\ v_q &= R_s i_q + L_q \frac{di_q}{dt} + \omega_e L_d i_d + \omega_e \lambda_m \end{aligned} \tag{3.16}$$

The (d, q) equivalent circuits are depicted in figure 3.5.

The equation of the electromagnetic torque T_e can be obtained by performing a power analysis. The output power of the motor is:

$$P_{out} = T_e \omega_m \tag{3.17}$$

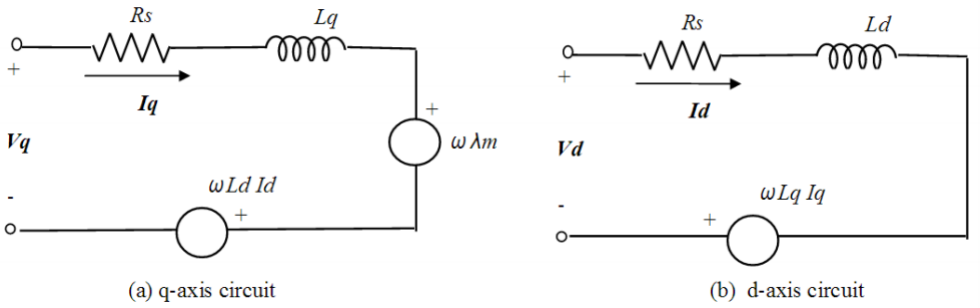


Figure 3.5: Equivalent circuit of an SPM

where ω_m is the rotor velocity. The output power can also be seen as the generation of an electromotive force, thus the output power can be expressed as:

$$P_{out} = \frac{3}{2}(-\omega_e \lambda_q i_d + \omega_e \lambda_d i_q) \tag{3.18}$$

By equalizing 3.17 and 3.18, and by substituting 3.13, the equation of the electromagnetic torque is derived :

$$T_e = \frac{3}{2} p[\lambda_m i_q + (L_d - L_q) i_d i_q] \tag{3.19}$$

The first term of 3.19 represents the synchronous torque (PM exitacion torque), while the second term the reluctance torque. By considering the isotropy of the SPM motor, the second term can be neglected.

3.4.2 Mechanical dynamics

The equation that models the mechanical part of the system is:

$$J\frac{d\omega_m}{dt} = T_e - T_{load} - T_f \tag{3.20}$$

where

- J is the total system mechanical inertia (rotor and load)
- ω_m is the rotor velocity
- T_e is the electromagnetic torque

- T_{load} is the torque imposed by the mechanical load
- T_f is the friction torque

In this work, the friction torque is modeled as a viscous plus Coulomb contribution. In particular, the model of the friction acting on the system is:

$$T_f = F_v \omega_m + K_c sign(\omega_m) \tag{3.21}$$

where F_v is the viscous coefficient and K_c the Coulomb one.

3.4.3 Simscape model

The simscape library contains a PMSM motor block that implements the equations that have been derived for the SPM motor. This model has been used for building the simulation environment (described in 5.2) for validation and control purposes. The Simscape motor block allows to choose among different configurations (i.e. rotor type, back EMF type, etc..). For the scope of this work the model configuration is chosen to be 3-phase with sinusoidal back EMF. The model is representend in the figure 3.6.

The A, B, C ports are the input stator voltages, the port S is the mechanical output port.

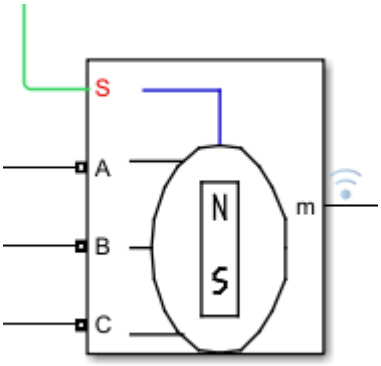


Figure 3.6: Library: Simscape / Electrical / Specialized Power Systems / Electrical Machines

Chapter 4

Current Control

The PMSM motors fall under the scope of variable-frequency vector control. The most important vector control techniques are the Field Oriented Control (FOC) and the Direct Flux Vector Control (DFVC). In this section, firstly the main objective of field oriented control is discussed, then the implementation in the (a,b,c) and (d,q) frame have been illustrated.

4.1 FOC - main concepts

Field Oriented Control is a current (torque) control algorithm that allows to design a current controller synchronized to the motor flux λ_m, in such a way that optimum control is obtained. It is an example where the cascade control topology it is used: the outer speed and position loop are cascaded around the inner current loop as shown in the figure 4.1.

Considering an AC motor, the three phases are distributed equally throughout the circumference of the machine in the stator slots. The three-phase currents flowing in the windings, creates a rotating magnetic field. Considering now the case of a permanent magnet machine, an SPM machine for example, the rotor of such a machine starts rotating trying to follow the rotating field generated by the three-phase currents. When the rotor tracks perfectly the rotating field, the machine does not generate any torque since the rotor flux angle and the stator magnetic field are aligned. When the motor starts to be loaded, the angle θ_λ between the rotor flux angle and the stator magnetic field starts increasing and so the torque, as shown in figure 4.2.

At $\theta_\lambda \pm 90\,\text{deg}$, the machine is generating the most torque possible for

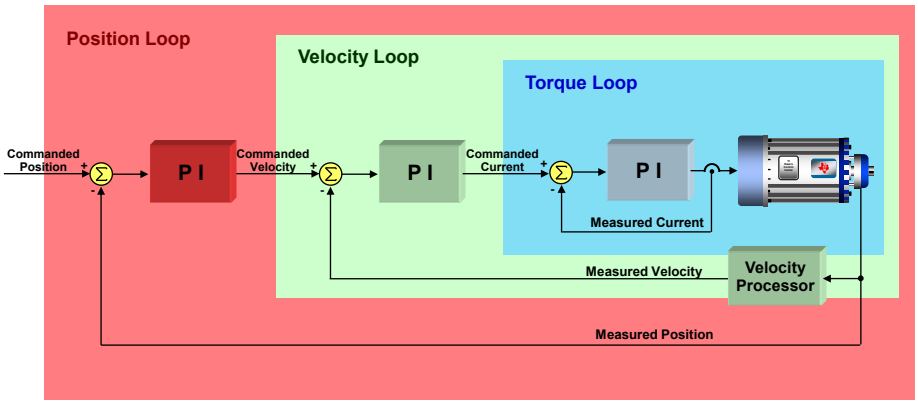


Figure 4.1: Cascade Control Topology

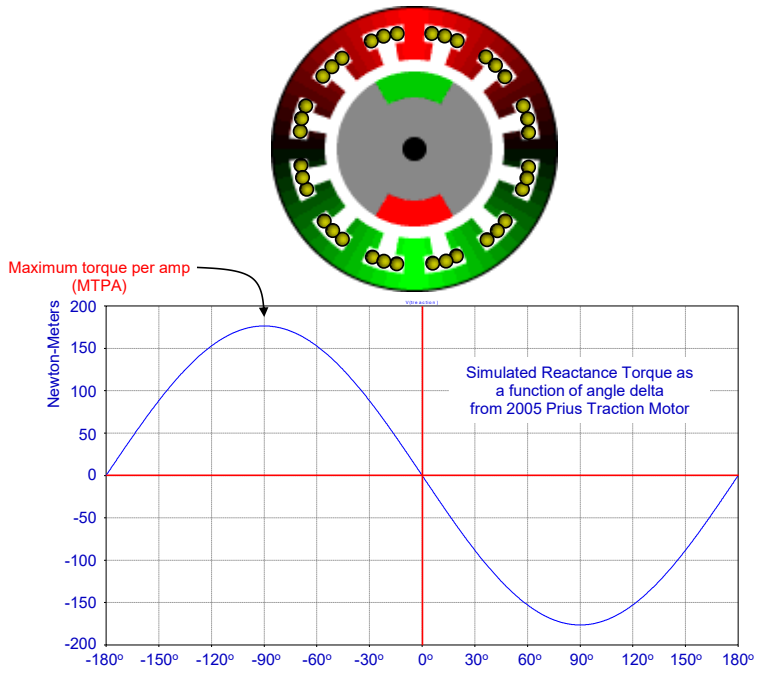


Figure 4.2: Maximum Torque per AMP (MTPA)

the given currents on that motor. Going over that torque by varying more the angle, means losing synchronization and reaching instability, in fact, the stable region belongs to the $\theta_\lambda \in [-90\,\text{deg}\,+90\,\text{deg}]$ (figure 4.2). The goal of the field oriented control algorithm is to get the maximum torque per

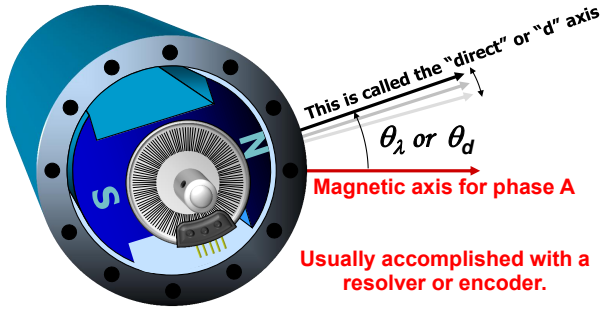


Figure 4.3: Magnetic and direct axis

ampere in a stable way, by feeding the motor windings in a such way that the vectorial sum of the three-phase currents gives a stator magnetic vector perpendicular to the rotor flux. A must for reaching this goal is knowing exactly the orientation of the rotor flux, both via sensored and sensoreless techniques. Once having this information, then it is known where to position the stator magnetic vector, thus the phase currents that have to be generated. In Figure 4.3, the rotor flux (or direct) axis and the phase a axis are shown.

The equation 4.1 models the generated torque depending on the angle θ_λ

$$T_{em} = k\boldsymbol{\lambda} \wedge \boldsymbol{i} \tag{4.1}$$

where the vector $\boldsymbol{\lambda}$ is oriented on the direct axis, while the vector $\boldsymbol{i}$ represents the stator magnetic vector and k is a constant. Thus the only portion of the current that is generating torque is the one that is at 90 deg wrt the rotor flux.

4.2 FOC - Implementation in the (a, b, c) frame

At high level, the FOC algorithm works by following and repeating this three steps by means of an interrupt service routine (ISR):

1. Measure the rotor flux angle.
2. Regulate the current vector to be perpendicular to the rotor flux by adjusting the three phase voltages, thus the currents.
3. Exit the ISR.

and all this is done usually at $10kHz$ or $20kHz$. The first step can be accomplished via sensored and sensoreless techniques. The heart of the FOC lies in the second step and could be decomposed into 4 substeps, namely:

1. Measure the currents already flowing in the motor windings.
2. Compare the measured currents with the desired ones, and generate an error signal.
3. Amplify the error signal to generate a correction voltage.
4. Modulate the correction voltage onto the motor terminals.

To measure the currents, at least two sensors are needed since the third current can be easily evaluate considering kirchhoff current law. Representing each phase current on the proper magnetic axes, the net stator current vector is obtained by summing the magnetic axis components vectorially. In this way the three phase currents varying in time are represented by a rotating current vector in the space vector diagram. The space vector diagram is shown in Figure 4.4. For obtaining the MTPA, the net stator current vector has to be perpendicular to the rotor flux.

By knowing the angle of the rotor flux, then the desired phase currents that allows to obtain the MTPA can be calculate via these equations:

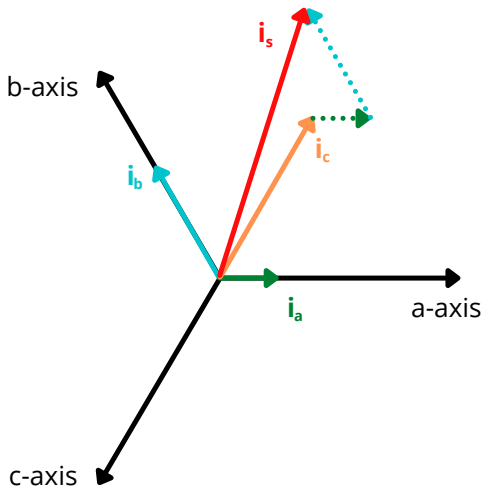


Figure 4.4: Space Vector Diagram

$$\begin{aligned} i_a^* &= -I_m sin\theta_\lambda \\ i_b^* &= -I_m sin(\theta_\lambda - 120°) \\ i_c^* &= -I_m sin(\theta_\lambda - 240°) \end{aligned} \tag{4.2}$$

where I_m is the amplitude of the phase currents and it is proportional to the motor torque that want to be generated. The simplified schematic that represents the FOC algorithm in the (a, b, c) reference frame that would make the phase currents converge to the desired values is shown in Figure 4.5.

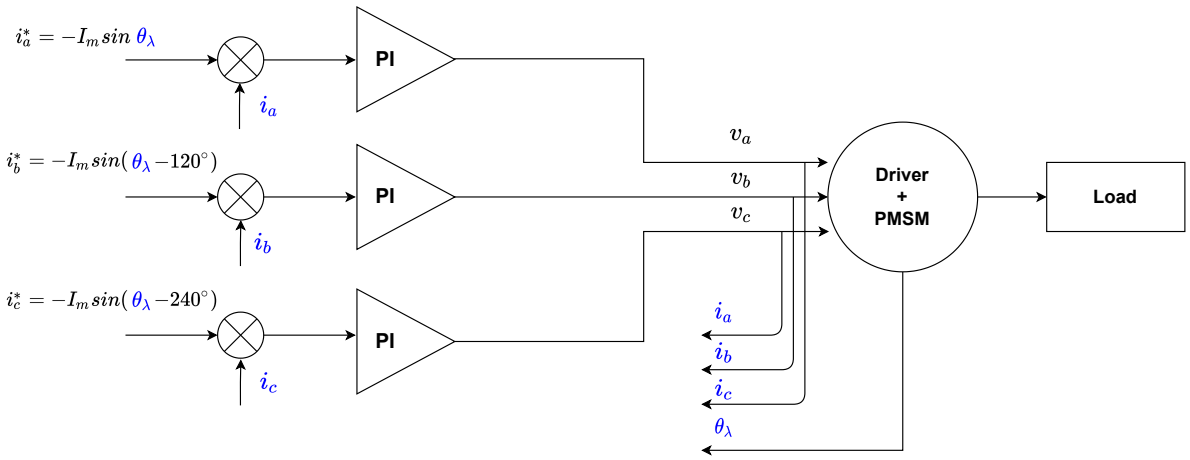


Figure 4.5: FOC in (a, b, c)

4.3 FOC - Implementation in the (d, q) frame

Until now, the principles of the FOC algorithm considering the model of the motor in the (a, b, c) frame have been discussed. For obtaining a simplified algorithm, that means simpler calculations, the current vector can be represents in a synchronous reference frame: the direct-quadrature frame through the Clarke and Park transform described in the previous chapter. This reference frame is synchronous with the rotor flux axis. Thanks to these transformations, the desired currents will be i_d and i_q, where the former is the direct components, i.e. the one parallel to the rotor flux axis, while the latter is the perpendicular component. The advantage is that the desired currents are independent form the rotor angle and should not be calculated

at each time instant, but they are fixed once the desired torque is fixed. Considering this, the direct current is set to 0 in normal conditions[1], while the quadrature represents how much torque it is needed. In this case, only two PI controllers are needed to generate the correction voltages for dq axis. The gains of the two PIs are the same just in case of non saliency. For applying the correction voltages to the motor windings, the reverse Clarke and Park transformations are needed for going back to the (a, b, c) stationary frame. Finally, the correction voltages are transformed into sinusoidal PWMs thanks to some modulation technique. One of the most prominent one is the Space vector modulation. The final schematic for the FOC algorithm into the dq frame is represented in Figure 4.6.

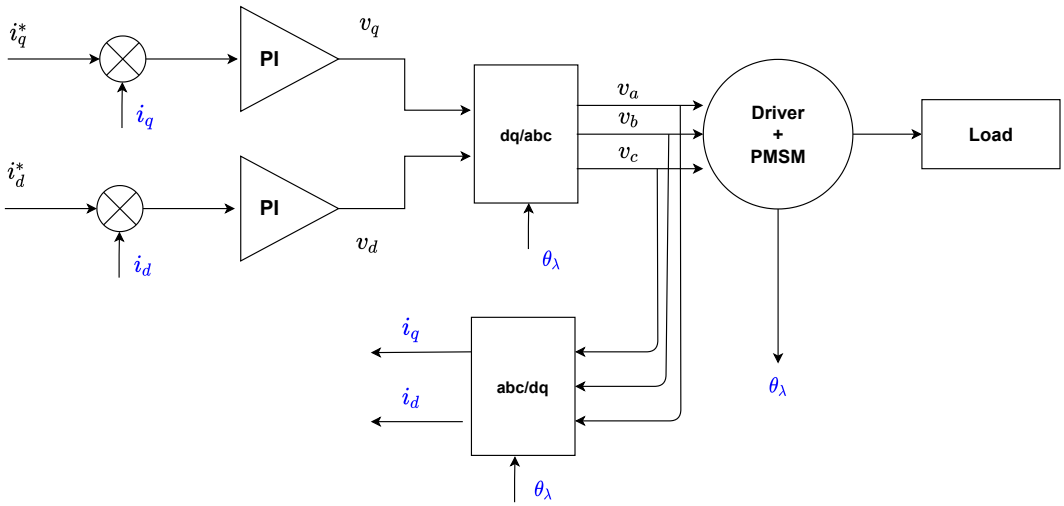


Figure 4.6: FOC in (d, q)

[1]When more speed is needed, the flux should be weakened. In this case i_d should be set different from zero.

Chapter 5

Experimental and Simulation environments

In the very first part of this work, all the experiments have not been made directly on the robot, instead an experimental set-up that integrates the same components of the real motor group discussed in Section 2.1.1 has been used. In order to make sure that the model built in Chapter 3 behaves as the real motor, a simulation environment that takes into account the real motor setup is needed. Furthermore, it is necessary to collect data on the real set up in order to perform simulation for validation purposes. In this chapter, the way the data have been collected by using the tools available, and the simulation environment created have been discussed.

5.1 Experimental environment - Collect IN-/OUT data

In this section it is illustrated the way the data have been collected, by exploiting the instruments and tools available and the way workarounds were find in order to be able to measure all the quantities needed. But before approaching to these topics, a brief overviews of the motor set-up architecture and of how the motor can be controlled are necessary.

5.1.1 Motor set-up architecture

The SPM motor of the motor set-up used in this work is a custom motor designed by MOOG and it represents the knee of iCub2.5. The motor is controlled by two boards connected in series: the EMS and the 2FOC. These two boards communicate between each other by means of a CAN communication protocol. The EMS is the board that communicates with the computer by using an ethernet bus. The communication schematic of the motor set-up is shown in figure 5.1. Furthermore, an Hall sensors at the rotor level and an absolute encorder at the joint level have been integrated to measure the rotor and joint position angle. The schematic of the logic connections among the two boards, the motor and the sensors are depicted in 5.2.

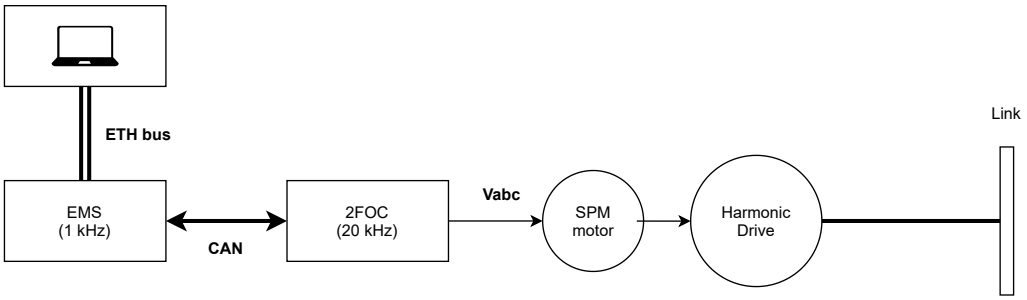


Figure 5.1: Motor set-up communications

Actuation chain: motor plus harmonic drive

As shown in figure 5.1, close after the motor in the actuation chain, the harmonic drive is the next component to transform the transmission dynamics variables, namely the angular velocity, the torque and the rotor apparent inertia with a step-down ratio ρ. The output of the harmonic drive is connected to a link that does not influence the mechanical dynamics since its torque contribution can be neglected. In figure 5.3, it is described a spinning mass m with the respective rotational inertia ${}^{M}\dot{h}_m$ and angular velocity ${}^{M}\omega_m$ both expressed on the point M of the shaft. This model can be applied to the motor set-up (and in general to an iCub main joint motor group), where G is the harmonic drive and m is the lumped mass of the fast rotating part -

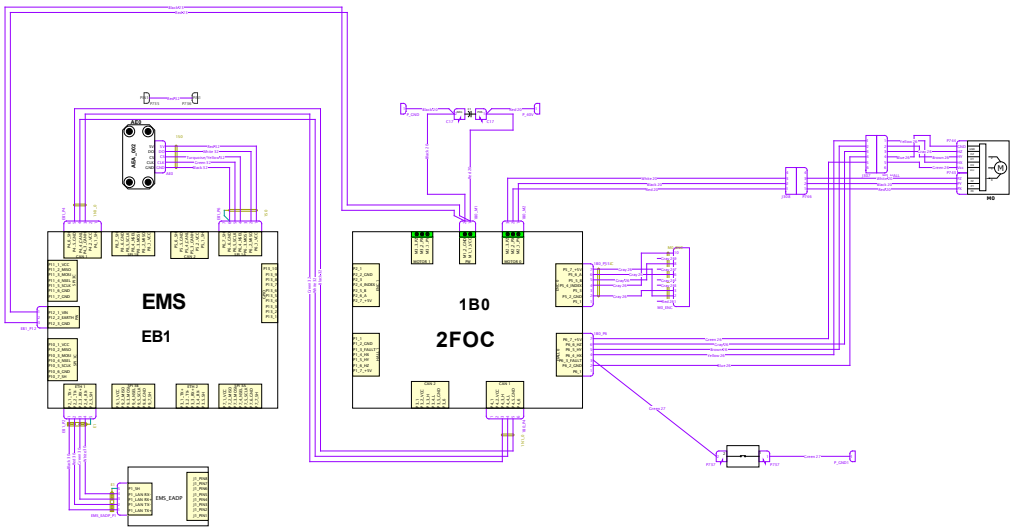


Figure 5.2: Motor set-up wiring

the rotor and the harmonic drive wave generator [1]. In particular, by writing the identity between the input-output power and by neglecting the friction losses, it is possible to state that the harmonic drive transforms the three dynamics quantities ${}^{M}\dot{h}_m$, ${}^{M}\omega_m$ and ${}^{M}\tau_m$, depending on the step-down ratio ρ, such that for its output point of view, i.e. on point G, the mass, the output torque and the rotational inertia are seen as shown in the equations 5.1.

$$\begin{aligned} {}^{G}\omega_m &= {}^{M}\ \omega_m \rho^{-1} \\ {}^{G}\tau_m &= {}^{M}\ \tau_m \rho \\ {}^{G}\dot{h}_m &= {}^{M}\ \dot{h}_m \rho^2 \end{aligned} \tag{5.1}$$

[1]Source: `https://gitlab.com/nunoguedelha/my-phd-thesis/blob/master/nuno-guedelha-cycle30-phd-thesis.pdf`

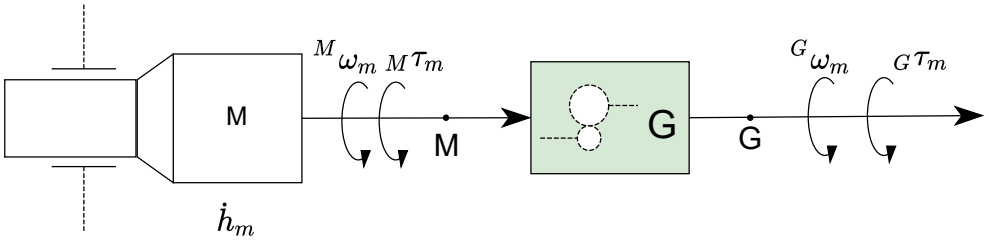


Figure 5.3: Reduction drive dynamics conversion

5.1.2 Motor Control via YARP-MATLAB Bindings

The motor has been controlled through a RemoteControlBoard Driver in the MATLAB environment. The RemoteControlBoard driver implements a set of interfaces that can be used to control the motor (e.g. IPositionControl interface) and to retrieve the readouts from the sensors mounted on the motor boards (e.g. IEncoders interface).

Figure 5.4 shows the communication diagram from the PC and the motor. The YarpMotorgui and the MATLAB script communicate with the EMS board through the YARProbotinterface.

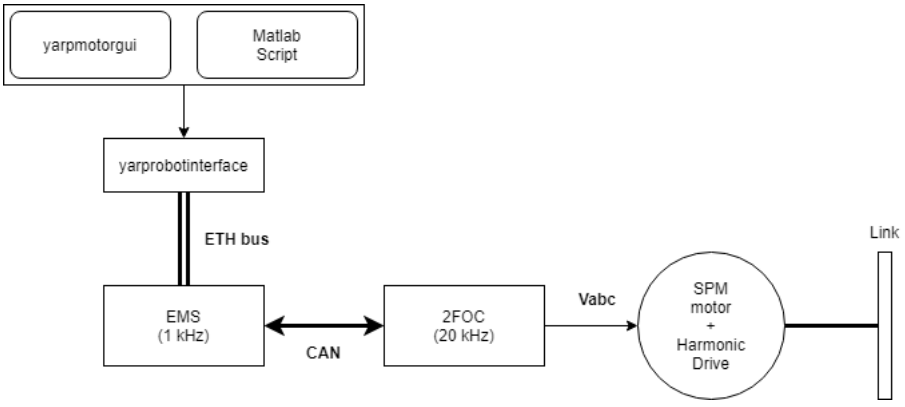


Figure 5.4: Motor Set-Up Control Tools

5.1.3 Measurements Set-Up

In the previous section, we showed how YARP can be used to retrieved the information from the motor sensors. However, since the ETH bus streams data at $1kHz$ - Fig. 5.4, we decided to exploit the CAN bus and the digital oscilloscope to increase the sampling time of the data retrieved from the motor. The following quantities are measured:

- three line voltages V_a, V_b, V_c
- two phase currents I_a, I_b ($I_c = -I_a - I_b$)
- the joint position θ_j

The electrical quantities are measured by means of two oscilloscopes, since each one has four analog channels, while the joint position has been logged via CAN.

Oscilloscope

The oscilloscopes used are the Tektronix MDO4104C and the MDO4104B-6. The digital oscilloscope works in two different modalities: triggered and roll. For acquiring continous data the roll mode is needed. The Tektronix scopes go automatically into the roll mode when the trigger mode is auto and the horizontal scale t_{hs} is set to $40ms/div$ or slower. The horizontal scale fixes the time resolution of the scope. The sampling time of the scopes can be defined by setting the horizontal scale and the record length parameters. In particular, considering the entire time window acquisition (t_W), the sampling time of the oscilloscope is defined as:

$$T_s = \frac{t_W}{RL}, \tag{5.2}$$

where $t_W = 10t_{hs}$ and RL is the record length of the scope, that corresponds to the number of samples collected in the entire time window. Considering that the three-phase inverter drives the motor with square signals at 20kHz, then the sampling time should be set accordingly. By performing some measurements, it has been noticed that the maximum sampling time for an accurate acquisition of the drive voltages (square wave signals at 20kHz) is $T_s = 2\mu s$.

CAN acquisition

The joint position has been logged at 20 kHz by means of an already implemented 2FOC firmware version, by sending start and stop logging commands to the 2FOC board through the CANREAL application.

Synchronization

For acquiring the electrical quantities, the two oscilloscopes acquisitions have been programmed via SCPI commands in MATLAB, by setting the scope start and stop acquisition simultaneously. As already said in 5.1.2, the motor has been controlled via MATLAB. In particular, the start and stop of the motor has been synchronized with the start and stop acquisition of the scopes. The CAN logging could not be synchronized automatically since the CANREAL application can not be directly interfaced with MATLAB. Thus, the CAN logged position has been manually synchronized after collecting all the data.

5.2 Simulation Environment

In this section both the simulation environment for validation and control purposes are illustrated. Firstly the motor group datasheet have been illustrated and set in the simscape model then all the subsystems describing the simulation environments have been described. Furthermore, it is important to state that the simulator for control purposes is not part of my contribution entirely.

Motor group parameters

As stated in the section 2.1.1, the motor group is composed by an SPM motor and an harmonic drive. The mechanical characteristics of the knee motor that is integrated in the motor set-up is represented in the figure 5.5, while the motor and harmonic drive parameters (see equations 3.16, 3.19, 3.20) provided by the datasheets are represented in the Tables 5.1, 5.2[2]. Those values have been set into the mask of the simscape motor model.

[2]The inertia of the motor J_m refers to the inertia without brake.
The inertia of the harmonic drive J_{HD} is the input inertia.

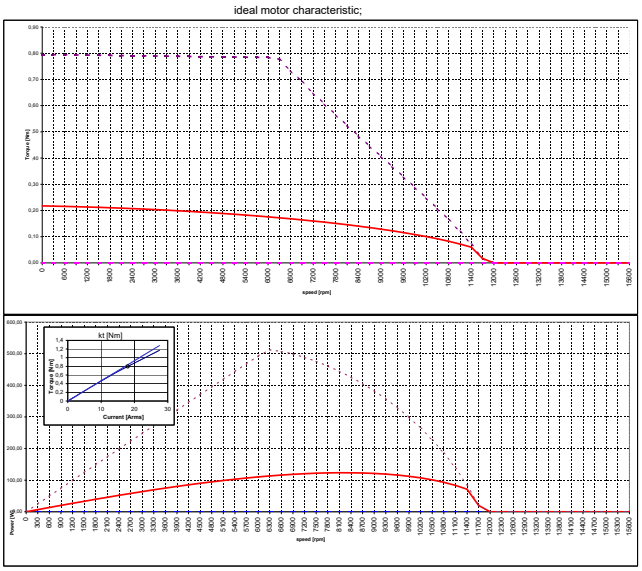


Figure 5.5: Motor Characteristics

Motor Parameters
$R_s = 0.341\Omega$
$L_d = 0.224mH$
$L_q = 0.233mH$
$\lambda_m = 0.0055Vs$
p = 4
$J_m = 8.2710^{-6}kgm^2$

Table 5.1: Motor Parameters from datasheet

Harmonic Drive parameters
$J_{HD} = 2.110^{-6}kgm^2$
$\rho = 100$

Table 5.2: Harmonic Drive Parameters from datasheet .

5.2.1 Model validation purposes

The simulation environment for validation purposes is designed using the simscape model described in 3.4.3 and considering the actuation chain composing the motor set-up (5.1.1) - Figure 5.6. A briefly illustration of the model follows.

The friction parameters of both motors and harmonic drive needs to be identified.

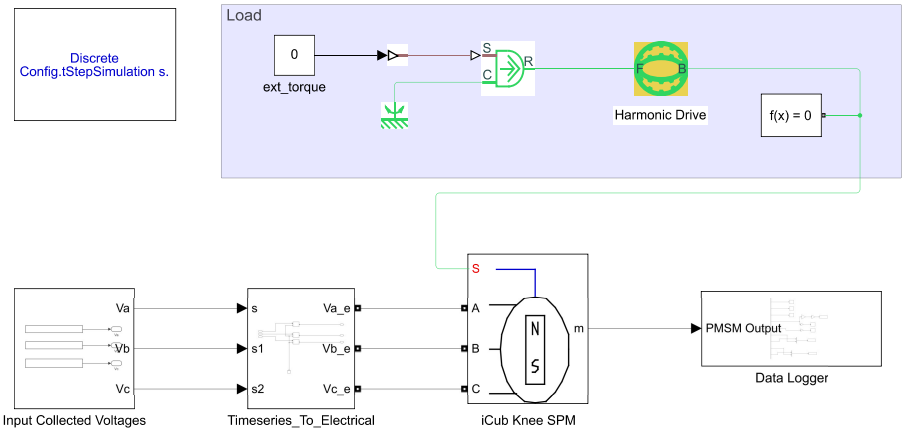


Figure 5.6: Simulink Environment - Validation purposes

Inputs

The *Input Collected Voltages* (Figure 5.6) provides as outputs the three MATLAB timeseries containing the the phase line voltages collected by means of the scopes. The *Timeseries To Electrical* block converts the timeseries into simscape electrical types. Finally these voltages have been used to feed the *iCub Knee SPM* simscape motor model describe in 3.4.3.

Load

The load is represented by the harmonic drive and an external load. The external load imposes a null torque on the mechanical shaft because, as mentioned in the previous section, the torque imposed by the link can be neglected since it is very small compared to the output torque that can be delivered by the actuation chain. The null torque should be imposed on the simscape mechanical line by means of a simulink to simscape signal converter

and an ideal torque generator, both belonging to the simscape library. The load is attached to output mechanical rotational port of the simscape motor model, that represents the rotational shaft of the machine.

Data Logger

The subsystem *Data Logger* deals with the logging of the simulated quantities into the MATLAB workspace. In particular, it has been needed to log the phase currents I_a, I_b, I_c, and the joint position θ_j.

Solver

In order to set the solver parameters, the scope acquisition setting have been taken into account. In particular, since the collected data have been sampled with a fixed sampling time, the solver has been set as discrete solver with a fixed step size equal to the sampling time. The simulation time has been set equal to the time window t_W of the scope.

5.2.2 Control purposes

To simulate the designed current control loop, an efficient simulator that takes into account the control algorithms and the SW/HW architecture of the boards is fundamental. The simulator is shown in Fig. 5.7 and here it is briefly illustrated.

The *Current Vector Controller* block implements in simulation the current vector controller designed in Chapter 7. The block designed is shown in Fig. 5.8. The design of such a controller is widely discussed in the relative Chapter. The *PWM driver* block aims at driving the *Half-bridge inverter* by modulating the voltages coming from the controller. Finally, the *Isolated Gate Bridge Driver* aims at driving the motor phase voltages between 0V and 48V.

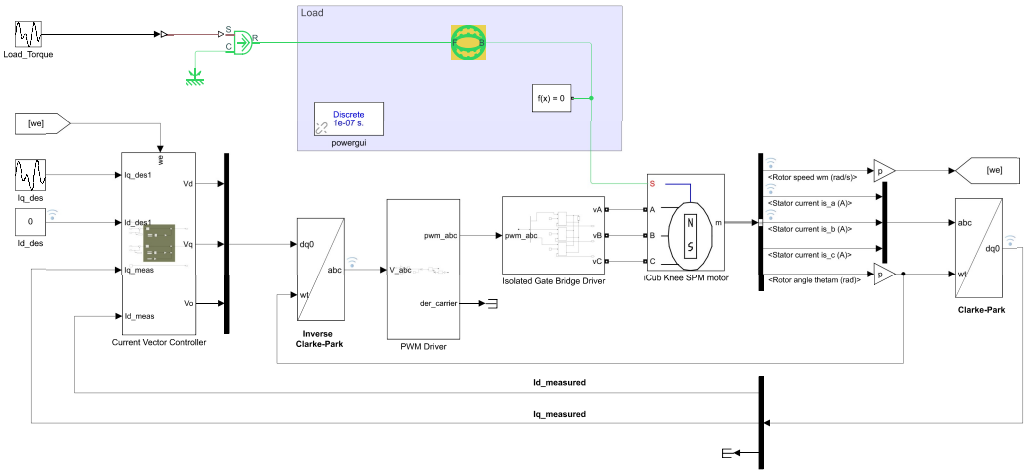


Figure 5.7: Simulink Environment - Control Purposes

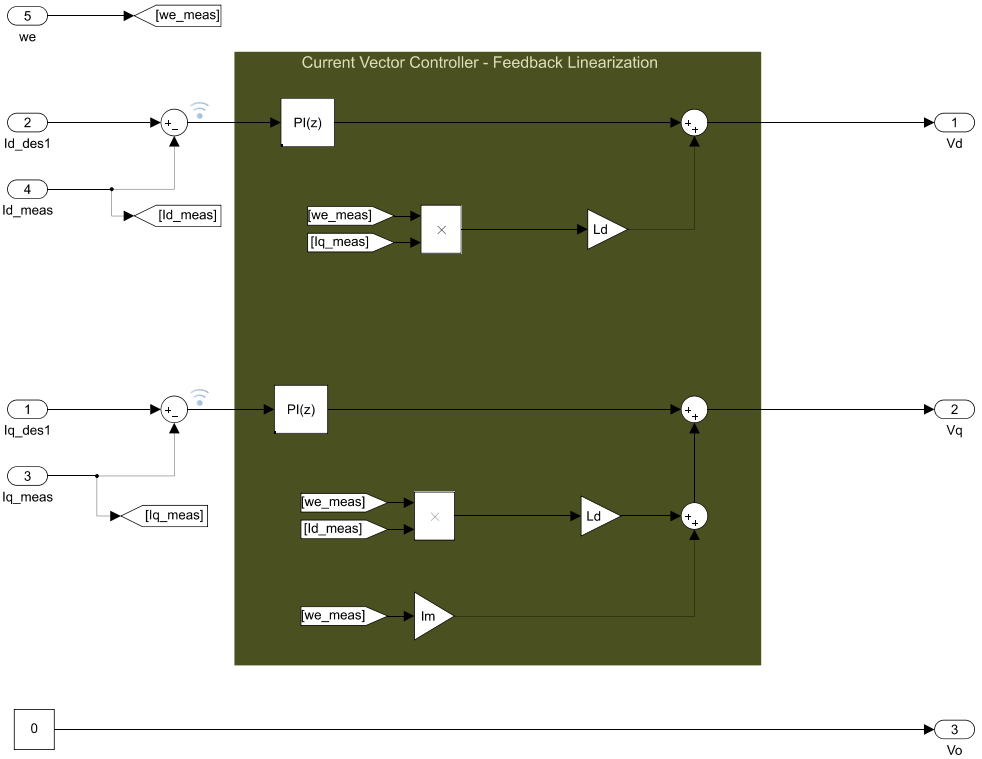


Figure 5.8: PI controller - feedback linearization

Chapter 6

Identification and Validation of Motor Parameters

6.1 Motivation

PMSM motors are characterized by nonlinear dynamics and consist of time-varying electrical parameters with high-order complex dynamics [Mohd Zaihidee et al. [2019]]. During the validation procedure of the motor model, it has been noticed that the nominal physical motor parameters, i.e. the one provided by the constructor, show good model behavior over a range of velocities (in our case $|w_j^{max}| = 50 deg/s$, where w_j is the joint velocity). When the joint velocity overcome those limits, the model does not behave as the real motor, thus the estimation of the electrical parameters becomes a crucial aspect. Such estimation has been done by means of a least square optimization problem. In this section three main different experiments illustrating such phenomena are shown. Experiments have been done by exploiting different control modes. The first two experiments were defined considering the motor true working point characterized by a smooth and slow motion. Then, a third set of experiments have been made controlling the motor in an open loop configuration, with high PWM causing the motor to rotate with a high speed.

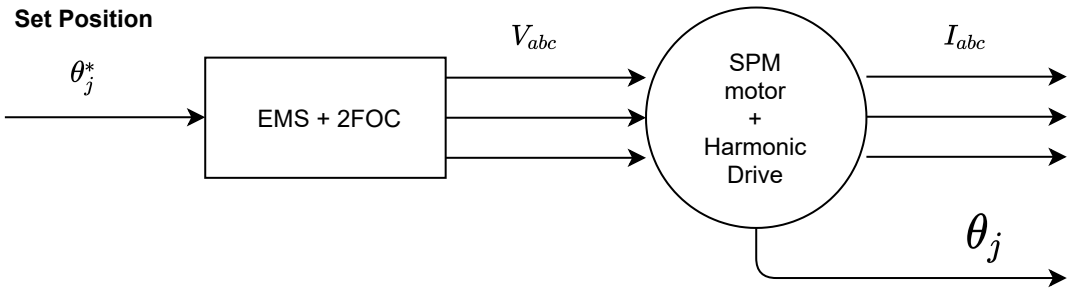


Figure 6.1: Position Control

IPos control

The schematic that illustrates at high level the position control experiment set-up is shown in Figure 6.1. It is just a schematic that In this first set of experiments, a fixed reference joint position has been set, and the minimum jerk trajectory was constrained.

Datasets

The experiments made for collecting the currents, voltages and the angle, are illustrated in the Table 6.1, where θ_{j0} is the initial joint angle and θ_j^* is the set point. Each dataset has been collected by setting as reference velocity, $\omega_{ref} = 2$ [deg/s], that represents a constraint for the minimum jerk trajectory.

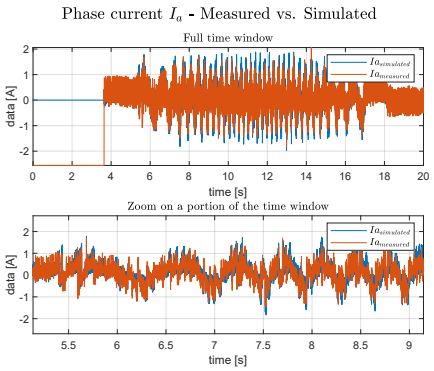
Num Experiment	θ_{j0} [deg]	θ_j^* [deg]
1	-45	-55
2	-55	-65
3	-65	-75
4	-75	-85
5	-85	-95
6	-45	-75

Table 6.1: Datasets - Position Control

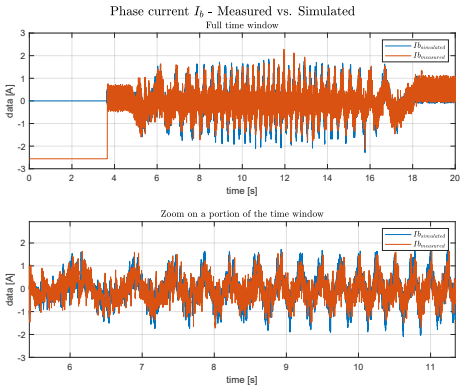
Validation

For the sake of readiness, only the validation results of one experiment have been shown here. In particular, considering the sixth dataset of the Table 6.1, in the Figures 6.2a, 6.2b, and 6.3 the two phase currents I_a, I_b measured and simulated, and the joint angle θ_j measured and simulated are shown. It can be seen that the simulated quantities are evolving like the measured one. This tells us two things:

- the simscape model is behaving as the real motor
- the datasheet parameters respect the real one



(a) Phase current Ia - Measured vs. Simulated



(b) Phase current Ib - Measured vs. Simulated

Figure 6.2: Phase currents Measured vs. Simulated

IPosDirect control

The position direct control mode is the mostly used control mode on the robot for performing walking. This control architecture has been exploit for feeding the motor with a true trajectory. In particular, during a walking test, the real knee trajectory has been collected and feeded to the motor set-up by means of the position direct control mode. Each single position has been sent to the EMS every 10 millisecond. The high level schematic representing the experiment is shown in Figure 6.4.

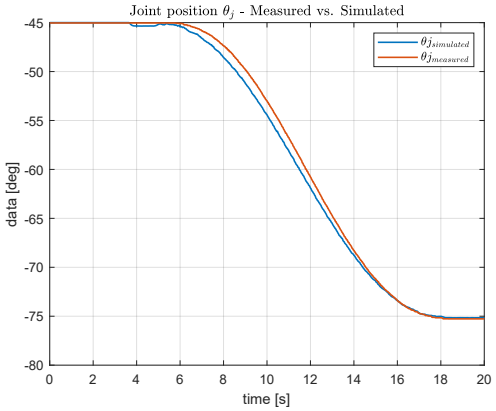


Figure 6.3: Joint Angle θ_j - Measured vs. Simulated

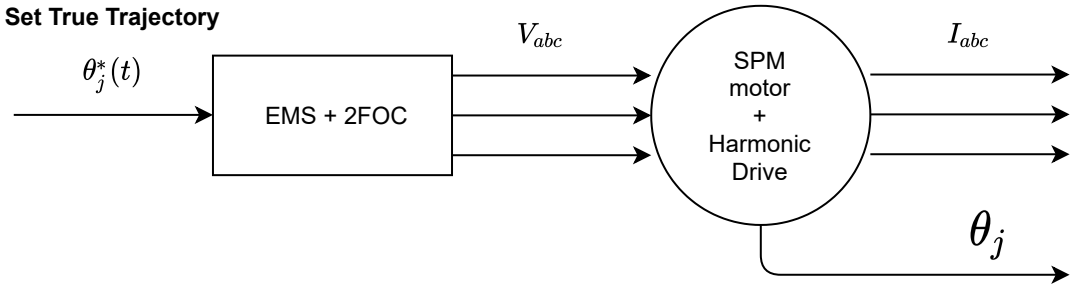


Figure 6.4: Position Direct Control

Dataset

Different trajectories have been collected and then feeded to the motor, but for the sake of simplicity only one has been discussed. The trajectory discussed here is shown in Figure 6.5.

Validation

By applying the voltages collected to the simulator described in the Subsection 5.2.1, and by performing the simulation, the results shown in Figure 6.6a, 6.6b, 6.7 have been obtained.

Also in this case, it can be said that the model is behaving as expected, as already stated in 6.1.

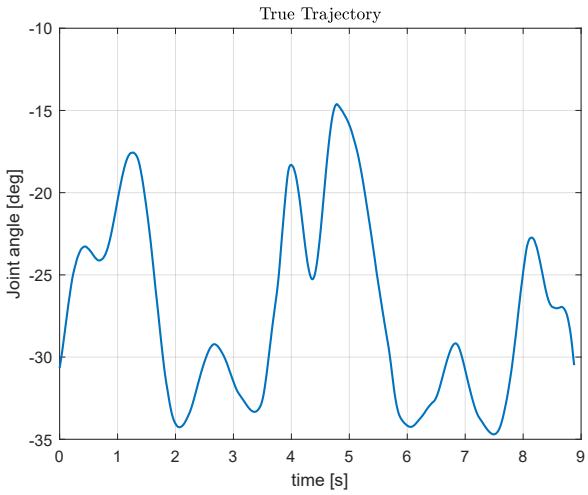
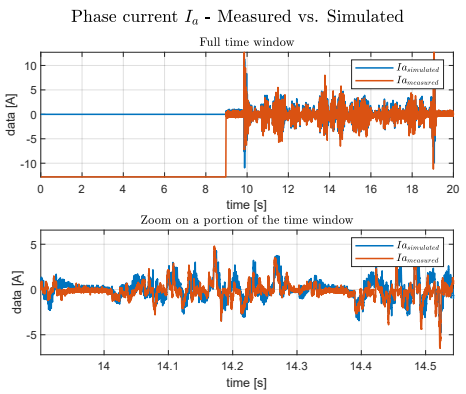
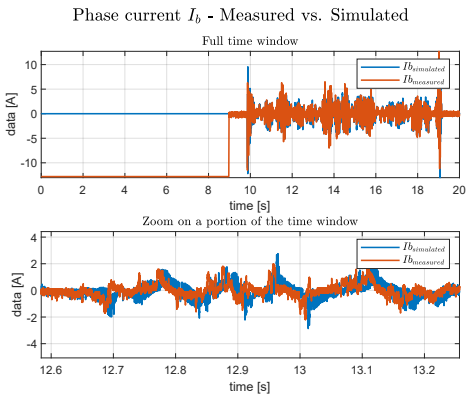


Figure 6.5: Knee True Trajectory



(a) Phase current Ia - Measured vs. Simulated



(b) Phase current Ib - Measured vs. Simulated

Figure 6.6: Phase currents Measured vs. Simulated

IPWM control

In this third set of experiments, it has been used the PWM control mode. Basically, a certain PWM is fixed as a set point, and the motor is directly driven with that fixed PWM. Setting an high PWM set point results in a

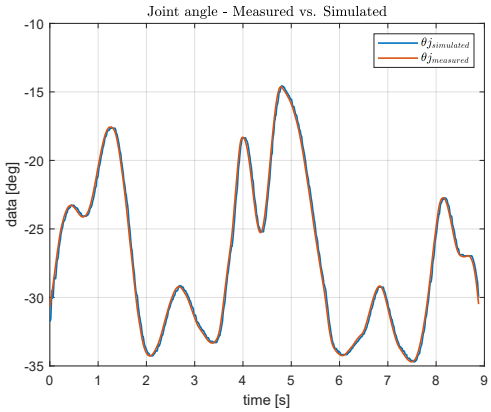


Figure 6.7: Joint Angle θ_j - Measured vs. Simulated

very fast rotation of the joint, causing high speed both at the joint and of course at motor level. The PWM ranges from [-100%, +100%], where the sign of the PWM value imposes the direction of rotation.

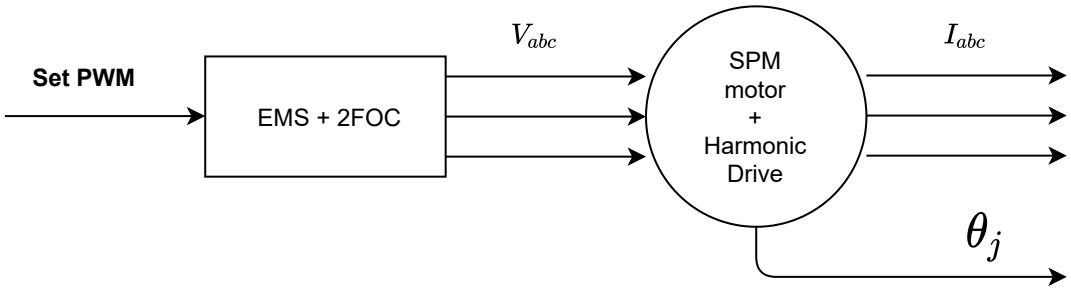


Figure 6.8: PWM Control

Dataset

Four different type of datasets have been collected and are shown in Table 6.1. In particular, four PWM set points have been set: 5%, 10%, 20% and, 50%. As regards the time duration of the experiments, the acquisitions is programmed to last until the joint angle reached -75 degrees.

Validation

By looking at the simulated vs. measured current represented in Fig. 6.9, it is evident that in this case the nominal parameters are such that the simulated

Num Experiment	PWM set point	θ_{j0} [deg]	θ_{jf} [deg]
1	5%	-45	-75
2	10%	-45	-75
3	20%	-45	-75
4	50%	-45	-75

Table 6.2: Datasets - PWM control

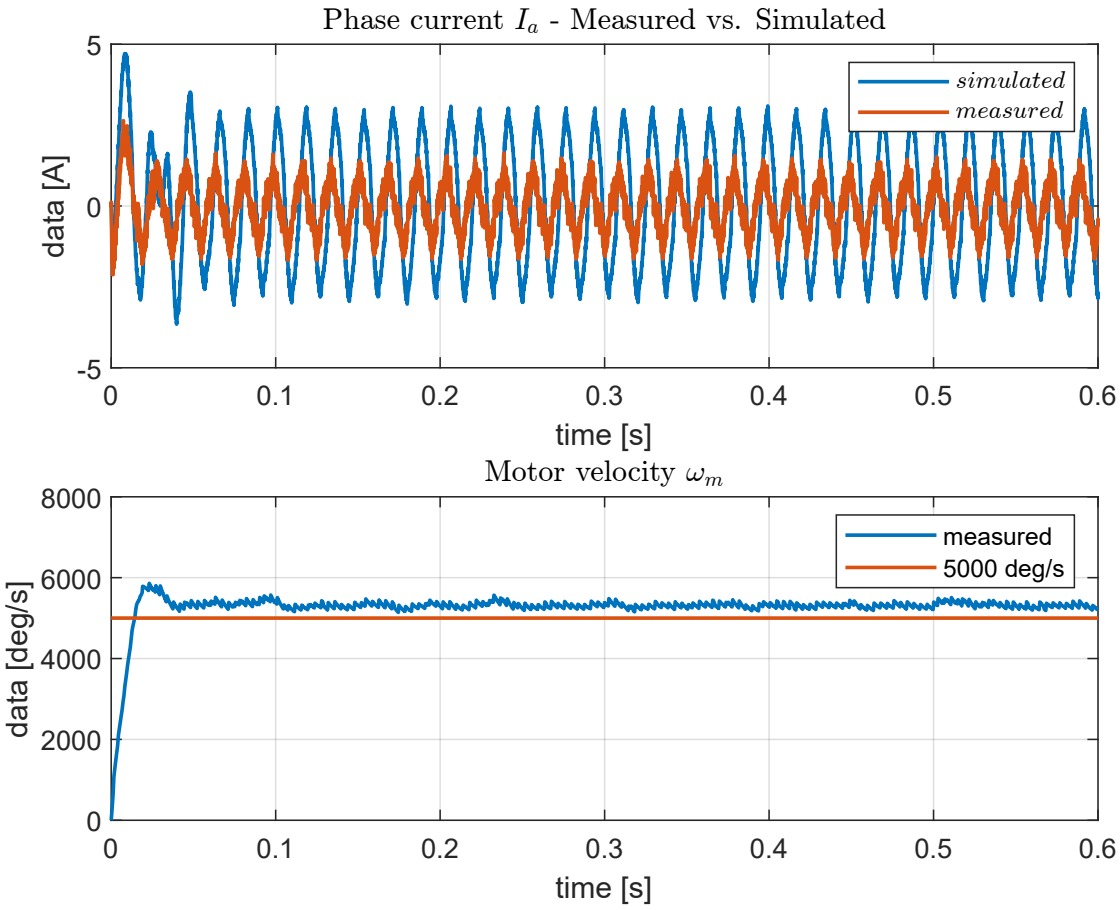


Figure 6.9: PWM Control

currents are evolving very differently from the desired ones. In fact, the mean squared error between the simulated and the measured quantities, defined as

$$MSE = \frac{\sum_{i=1}^{n}(I_i^{meas} - I_i^{s}im)^2}{n} \tag{6.1}$$

is 3A and the phase currents delay $\Delta\Phi$ is almost $70deg$. As a consequence, the estimation of the motor parameters becomes crucial.

6.2 Identification

In this section the identification procedure related to the estimation of the motor electrical parameters is illustrated. Firstly, a discrete version of the system is obtained. Then the least square optimization problem is defined. In order to be able to have good dataset for identification purposes, a suitable experiment is defined that allows to reduce as much as possible the conditioning of the problem. Finally, the results obtained that validate the model is shown.

6.2.1 Discretization

Forward Euler discretization is used in order to build the least square problem. Thus, the equations 3.16 and 3.20 describing the SPM motor dynamics are discretized in this way, by considering 3.19

$$\begin{aligned} L_d\frac{i_d^k - i_d^{k-1}}{T_s} &= v_d^{k-1} - R_s i_d^{k-1} + \omega_e^{k-1} L_q i_q^{k-1} \\ L_q\frac{i_q^k - i_q^{k-1}}{T_s} &= v_q^{k-1} - R_s i_q^{k-1} - \omega_e^{k-1} L_d i_d^{k-1} - \omega_e^{k-1}\lambda_m \\ J\frac{w_m^k - w_m^{k-1}}{T_s} &= \frac{3}{2}p\lambda_m i_q^{k-1} - F_v w_m^{k-1} - K_c sign(w_m^{k-1}) \end{aligned} \tag{6.2}$$

defining N as the number of samples of the training set, k goes from 1 to N. By considering 6.2, the matrix form can be derived as

$$\underbrace{\begin{bmatrix} v_d^{k-1} \\ v_q^{k-1} \\ 0^{k-1} \end{bmatrix}}_{\mathbf{Y}} = \underbrace{\begin{bmatrix} \frac{i_d^k - i_d^{k-1}}{T_s} & i_d^{k-1} & -p\omega_m^{k-1} i_q^{k-1} & 0^{k-1} & 0^{k-1} & 0^{k-1} & 0^{k-1} \\ p\omega_m^{k-1} i_d^{k-1} & i_q^{k-1} & \frac{i_q^k - i_q^{k-1}}{T_s} & p\omega_m^{k-1} & 0^{k-1} & 0^{k-1} & 0^{k-1} \\ 0^{k-1} & 0^{k-1} & 0^{k-1} & -\frac{3}{2}p i_q^{k-1} & \frac{w_m^k - w_m^{k-1}}{T_s} & w_m^{k-1} & sign(w_m^{k-1}) \end{bmatrix}}_{\mathbf{W}} \underbrace{\begin{bmatrix} L_d \\ R_s \\ L_q \\ \lambda_m \\ J \\ F_v \\ K_c \end{bmatrix}}_{\mathbf{\Phi}} \tag{6.3}$$

6.2.2 Optimization Problem

Considering that the parameter vector Φ is positive, i.e. all the components are positive values, then the following optimization problem is obtained:

$$\begin{aligned} \hat{\Phi} &= \underset{\Phi}{\operatorname{argmin}}\, J(\Phi) \\ &\text{subject to } \Phi > 0 \end{aligned} \tag{6.4}$$

where the cost function $J(\Phi)$ is

$$J(\Phi) = ||Y - W\Phi||_2^2 \tag{6.5}$$

that leads to:

$$||Y - W\Phi||_2^2 = (Y - W\Phi)^T(Y - W\Phi) = Y^TY - 2\Phi^TW^TY + \Phi^TW^TW\Phi$$

thus

$$J(\Phi) = \Phi^T\boldsymbol{H}\Phi + 2\boldsymbol{f^T}\Phi + const \tag{6.6}$$

Thus, by considering 6.4 and 6.6, the optimization problem requires to solve a quadratic objective function with linear constraints. Such a kind of problem can be solved easily in MATLAB by using the command *quadprog(H,f,A,b)* which finds the minumum for the problem specified in 6.4, where A and b are the constraints in matrix form $Ax \leq b$. It can be noticed that the optimization problem is linear in the parameters.

6.2.3 Informative dataset

In identification problems, the concept of identifiability is fundamental. The problem is approaching to an exclusive solution for the parameter Φ, and this depends mainly on the reliability of the model, and on the characteristics of the training set (dataset). In particular, the training set has to be informative enough to distinguish between different models[Ljung [1986]]. Thus, the definition of the experimental conditions becomes crucial. A common choice for obtaining a good system excitation is by menas of a sine-wave-based spectrally rich signal. Such signals are widely applied in identification experiments to obtain an estimation of the system parameters as close as possible to the true ones. Infact, sufficiently rich excitation signals aims

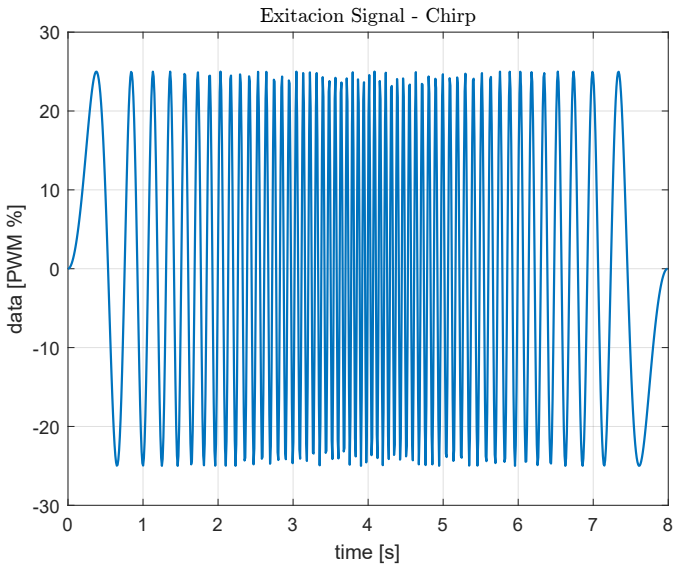


Figure 6.10: Excitation signal - Up-Chirp plus Down-Chirp

at guaranteeing informative input-output data for system identification. In this work, among the sine-wave-based signals, the chirp signal is choosen as generator of persistent excitation. Chirp signals, also known as, sweep signals, are identification excitation signals in which the frequency is swept up (up-chirp) or down (down-chirp) in one period. Chirp signals are typically divided into two groups: linear, in which the frequency of the signal varies linearly with time, and exponential/geometrical, in which the frequency varies with geometric progression. In this work, a linear chirp signal with a sinusoidal waveform is applied. The instantaneous frequency of the linear chirp signal can be calculated by

$$f(t) = f_0 + kt \tag{6.7}$$

where f_0 is the starting frequency, t is the current time, and k is the rate of frequency change, which can be obtained by

$$k = \frac{f_1 - f_0}{T} \tag{6.8}$$

where f_1 is the final frequency and T is the final time after the sweep from f_0 to f_1. The equation of the sine chirp becomes

$$u(t) = Asin(2\pi(f_0 t + \frac{k}{2}t^2)), \tag{6.9}$$

where A is the amplitude of the chirp signal. The signal applied for identification of motor parameter is a double chirp signal: an up-chirp plus a down-chirp are glued together and sent in an open loop configuration to the motor windings.

6.2.4 Problem Conditioning and Features Scaling

The regressor matrix in 6.3, is bad conditioned, in fact the conditioning number of the weighted regressor H is of the order 10^{13}. The conditioning of the problem depends on several factors, but most of all on the way the measurements are took. In order to improve the conditioning of the problem, the sampling time is increased as much as possible ($T_s = 2\mu s$), and furthermore more different chirp experiments are made and all the datasets are stacked together. These actions improved the conditioning of the problem decreasing of more than four order of magnitude the conditioning number, but it is still not enough. Thus, a scaling technique is exploited. In particular, the features scaling technique is applied. By considering 6.3, since k goes from 2 to N, I can write the regressor W as:

$$W = \begin{bmatrix} \frac{i_d^2 - i_d^1}{T_s} & i_d^1 & -p\omega_m^1 i_q^1 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{i_d^N - i_d^{N-1}}{T_s} & i_d^{N-1} & -p\omega_m^{N-1} i_q^{N-1} & 0 & 0 & 0 & 0 \\ p\omega_m^1 i_d^1 & i_q^1 & \frac{i_q^2 - i_q^1}{T_s} & p\omega_m^1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ p\omega_m^{N-1} i_d^{N-1} & i_q^{N-1} & \frac{i_q^N - i_q^{N-1}}{T_s} & p\omega_m^{N-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{3}{2}pi_q^1 & \frac{w_m^2 - w_m^1}{T_s} & w_m^1 & sign(w_m^1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \frac{3}{2}pi_q^{N-1} & \frac{w_m^N - w_m^{N-1}}{T_s} & w_m^{N-1} & sign(w_m^{N-1}) \end{bmatrix} \tag{6.10}$$

and calling each column of W as w_i, then

$$W = \begin{bmatrix} w_1 & w_2 & \dots & w_7. \end{bmatrix} \tag{6.11}$$

Features scaling leads to a new regressor matrix W_{scaled} where each column corresponds to a scaled version of the column of W. In particular, defining the i-th column w_1^* of the scaled regressor as

$$w_i^* = \frac{w_i}{\max(w_i)} \tag{6.12}$$

then

$$W_{scaled} = \begin{bmatrix} w_1^* & w_2^* & \dots & w_7^* \end{bmatrix} \tag{6.13}$$

Now, by studying the contitioning of the weighted scaled regressor, a strong decrease is obtained. In particular, the conditioning number decreases at 200.

Optimization problem with feature scaling

By applying features scaling, the scaled factor now should have to be taken into account while defining the optimization problem. In particular

$$Y = W\Phi = W_{scaled}S^{-1}\Phi = W_{scaled}\Phi_{scaled} \tag{6.14}$$

where S^{-1} is a diagonal matrix where $s_{ii}^{-1} = \max(w_i)$. Thus,

$$\begin{gathered} \hat{\Phi_{scaled}} = \underset{\Phi_{scaled}}{\operatorname{argmin}} \, ||Y - W_{scaled}\Phi_{scaled}||_2^2 \\ \text{subject to } \Phi_{scaled} > 0 \end{gathered} \tag{6.15}$$

and so, after solving the optimization problem above, the parameter vector Φ is obtained as

$$\Phi = \hat{\Phi_{scaled}}S \tag{6.16}$$

6.2.5 Estimated Parameter and Validation

The datasheet parameters provided by the constructor and the estimated ones are shown in Tab. 6.3. By substituting the estimated parameters with the datasheet ones, both the *Mean Squared Error* between measured and simulated currents and the phase delay approach to zero – Fig. 6.11. Thus,

the simulations validate the estimated quantities when the motor velocity $|\omega_m| > 5000 deg/s$.

	MOOG motor electrical parameters			
	Ld	Lq	Rs	λ_m
Datasheet	$0.233mH$	$0.224mH$	0.341Ω	$0.0055Vs$
Estimated	$0.583mH$	$0.101mH$	1.903Ω	$0.0005Vs$

Table 6.3: Motor electrical parameters

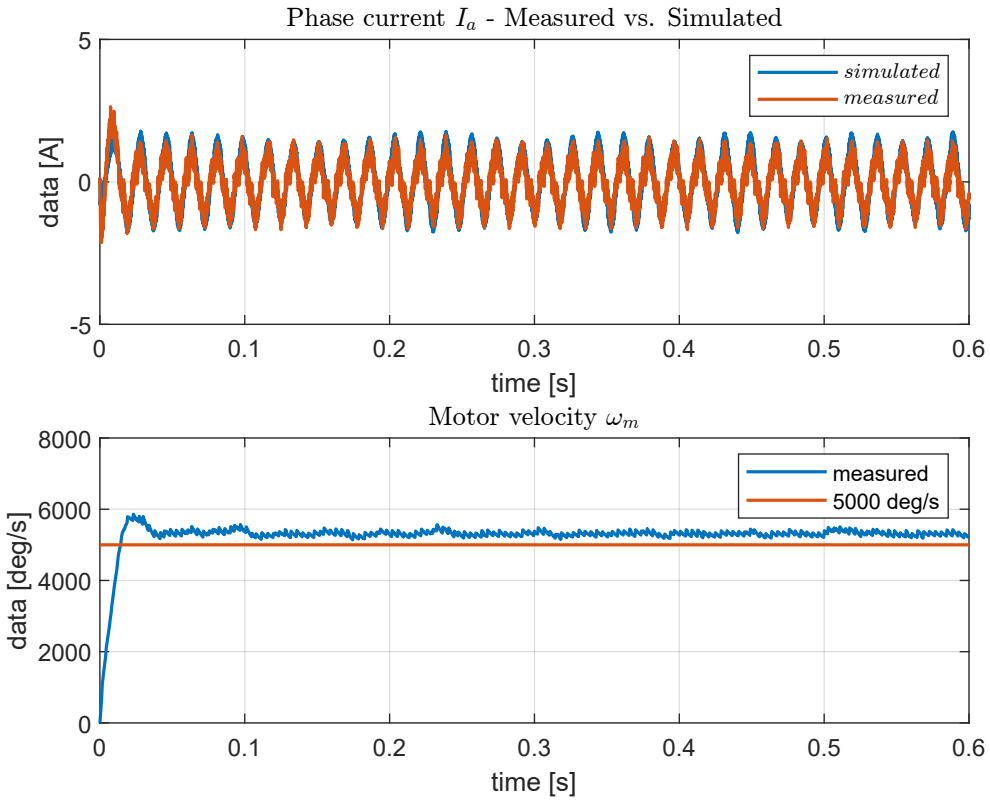


Figure 6.11: $|\omega_m| > 5000 deg/s$. Measured vs. Simulated Current.

Chapter 7

Current Vector Controller

In this Chapter the problem of designing a low level current control law is faced. The approach used in this work for stabilizing the motor electrical dynamics is the *Feedback Linearization*. Firstly, the feedback linearization technique together with the study of the *Input-Output linearization* of SISO systems are discussed. Then, these concepts are extended for the multivariable system treated in this work, and the final control law is derived. Finally, both the code implementation and the results are commented.

7.1 Feedback linearization

Feedback linearization is among the most relevant methods for controlling non linear systems. The approach of this technique is based on the transformation of a nonlinear system into a linear one, such that linear control techniques can be applied. This transformation is made by means of state feedback. Basically, this technique relates with the cancellation of the system nonlinearities, so that the closed-loop dynamics is linear. As already stated, feedback linearization is achieved by exact state transformation, rather than by linear approximation [Novara]. Before going on with the tractacions, some basic concepts on differential geometry has to be introduced, then the *Input-Output linearization* approach is discussed.

7.1.1 Differential Geometry - some definitions

Definition 7.1.1 (Smooth function)**.** Function $f(x)$ is said to be *smooth* if it has continous partial derivatives of any required order.

Definition 7.1.2 (Lie Derivative)**.** Let $h : \mathbb{R}^n \to \mathbb{R}$ be a *smooth* scalar function, and $f(x) : \mathbb{R}^n \to \mathbb{R}^n$ be a *smooth* vector field on $\mathbb{R}^n$. The *Lie Derivative* of h with respect to f is a scalar function defined by $L_f h = \nabla h f \in \mathbb{R}$.

The Lie Derivative is the derivative of h along the direction of the vector f. Thus

$$\begin{aligned} L_f^0 h &= h \\ L_f^i h = L_f(L_f^{i-1}h) &= \nabla(L_f^{i-1}h)f, i = 1,2,.. \end{aligned} \tag{7.1}$$

Considering the SISO system

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ y &= h(x) \end{aligned} \tag{7.2}$$

where $x \in \mathbb{R}^n$ is the state, and $u \in \mathbb{R}$ is the command input, $y \in \mathbb{R}$ is the output, and f, g and h are smooth functions on $\mathbb{R}^n$, the computation of $\dot{y}$ and $\ddot{y}$ is

$$\begin{aligned} \dot{y} &= \nabla h \dot{x} \\ \ddot{y} &= \nabla(L_f h)\dot{x} = L_f^2 h \end{aligned} \tag{7.3}$$

7.1.2 Input-Output linearization

The system described in Eq. 7.2 is in the *affine in u* form. The approach of the *Input-Output linearization* is based on the differentiation of the output y until the input u comes in the equation. Then, a control law that eliminates the nonlinearities of the system has to be designed. Firstly, a domain of interest $\Omega_x \in \mathbb{R}^n$ has to be defined. On this domain, the first derivative of y is

$$\dot{y} = \nabla h(x)\dot{x} = \nabla h(x)(f(x) + g(x)u) = L_f h(x) + L_g h(x)u \tag{7.4}$$

Let us suppose that the *Lie Derivative* $L_g h \neq 0$ in some region of $\Omega \in \mathbb{R}^n$, then a control law in the aforementioned domain that linearizes the nonlinear system represented in Eq. 7.2 is

$$u = \frac{1}{L_g h(x)}(-L_f h(x) + v). \tag{7.5}$$

The resulting linear system is

$$\dot{y} = v. \tag{7.6}$$

If the *Lie Derivative* $L_g h = 0$, then another differentiation is needed until, for some integer γ one obtains

$$L_g L_f^{\gamma-1} h(x) \neq 0. \tag{7.7}$$

Then in some region Ω, the following control law is got

$$u = \frac{1}{L_g L_f^{\gamma-1} h(x)} (-L^{\gamma} h(x) + v). \tag{7.8}$$

In this case the system reduces the input-output map to

$$y^{(\gamma)} = v. \tag{7.9}$$

which basically is a chain of γ integrators. γ is called the *relative degree* of the system. The corresponding state equation of the linear system defined in Eq. 7.9 is the *companion form*

$$\begin{aligned} \dot{\mu} &= A\mu + Bv \\ y &= \mu_1 \end{aligned} \tag{7.10}$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{7.11}$$

while the state vector corresponds to

$$\mu = (\mu_1, \dots, \mu_\gamma) = (y, \dot{y}, \ddot{y}, \dots, y^{\gamma-1}) \tag{7.12}$$

and it describes the *external dynamics* of the system.

The nonlinear system described in Eq. 7.2, can also be expressed in the *normal form*

$$\dot{\mu} = \begin{bmatrix} \mu_2 \\ \vdots \\ \mu_\gamma \\ a(\mu,\psi) + b(\mu,\psi)u \end{bmatrix} \qquad (7.13)$$
$$\dot{\psi} = \omega(\mu,\psi)$$
$$y = \mu_1$$

where $a(\mu,\psi) = L_f^\gamma h(x)$, and $b(\mu,\psi) = L_g L_f^{\gamma-1} h(x)$. In this way a new state (μ,ψ) is obtained and it is called the *normal state*, where μ represents the *external dynamics*, and ψ the *internal dynamics*.

Tracking control problem formulation

Here the aim is to formulate a control law for being able to track a reference signal $r(t)$. Assuming that $r(t)$ is *smooth* and *bounded*, and considering the system described in Eq. 7.2 in his *normal form* (Eq. 7.13) let us define the *tracking error* as

$$\tilde{\mu} = \mu_r - \mu \qquad (7.14)$$

where $\mu_r = (r, \dot{r}, \ddot{r}, \ldots, r^{\gamma-1})$. So the goal is to make the tracking error $\tilde{\mu}$ as small as possible and, possibly, to force it to converge to zero.
The control law represented by Eq. 7.8, allows to design a linear control law v, by means of any linear control technique.

A general control scheme that represents the *Input-Output linearization* approach is shown in Fig. 7.1:

- *Plant* described by Eq. 7.2
- *State transformation* refers to Eq. 7.12
- *Feedback Linearization* implements the Eq. 7.8
- *Linear Controller* implements some linear control law
- *h(x)* computes the output of the closed loop control system

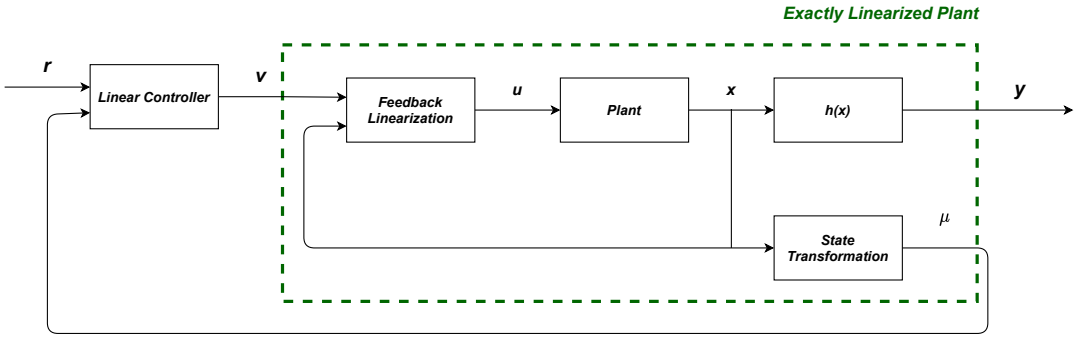


Figure 7.1: Input-Output Linearization - schematic

7.2 Current Controller

The low-level current controller aims to track the desired motor currents while rejecting external disturbances. The proposed controller computes the desired motor voltages using the motor dynamics 3.16. The motor's equations model a nonlinear dynamical system which states are $x = \begin{bmatrix} i_d & i_q & \omega_m \end{bmatrix}^\top$. The control problem is formulated by means of the feedback linearization technique – Fig. 7.1. As already said, feedback linearization is one of the most relevant methods for controlling nonlinear systems. Here, the aim is to transform the nonlinear motor model into a linear one by exact state transformation, so that linear control techniques can be applied – Fig. 7.1. In the case of a multivariable system, the feedback linearization technique is defined as the problem of finding a feedback law such that the system is reduced to a series of independent single-input single-output channels. More specifically, considering the system described by Eq. 3.16 with outputs give by $y_1 = i_d$, $y_2 = i_q$, a control law of the form

$$\begin{cases} u_d = -L_q \omega_e i_q^m + v_d \\ u_q = L_d \omega_e i_d^m + \lambda_m \omega_e^m + v_q \end{cases} \tag{7.15}$$

feedback linearizes the input/output representation of the system. v_d and v_q are additional control inputs. i_q^m, i_d^m and ω_e^m are measured quantities. Choosing v_d and v_q as

$$v_\circ = k_p(i_\circ^* - i_\circ^m) + k_i \int_0^t (i_\circ^* - i_\circ^m)\,\mathrm{d}\tau, \quad \circ = \{d,\ q\} \tag{7.16}$$

guarantees the tracking of the desired current $i_\circ^*$.

7.2.1 Exponential Filter

The aforementioned control strategy is based on the perfect measurement of motor state - namely i_q, i_d and ω_e. In a real scenario the current measurements are highly affected by sensor noise.

To mitigate the effect of the measurement perturbation, an exponential filter is implemented for i_q and i_d – Fig. 7.2.

The exponential filter is a low pass filter characterized by the parameter $\alpha \in (0,1)$

$$\hat{y}_{k+1} = \hat{y}_k \alpha + y_k^m (1 - \alpha), \tag{7.17}$$

where $\hat{y}_k$ and y_k^m are the estimated and measured values at time step k, respectively. By performing the $Z - Transform$ of (7.17), the transfer function of the filter is

$$\frac{\hat{Y(z)}}{Y^m(z)} = \frac{1 - \alpha}{1 - z^{-1}\alpha}. \tag{7.18}$$

In this work, the filter has been implemented in the ISR (interrupt service routine) of the FOC control loop. As a consequence, the filter inherits the sampling frequency of the ISR ($f_s = 20kHz$).

Exponential filters essentially assume a signal model of brownian motion/random walk: that the signal remains unchanged except for some random process noise. Then the best prediction of the next value (before seeing newer data) is the previous value. The final estimate is just a weighted average of the predicted value and a newly-observed value [Stanley].
The implementation on the currents is

$$\hat{i}_{k+1} = \hat{i}_k \alpha + i_k^m (1 - \alpha), \tag{7.19}$$

where $\hat{i}_k$ and i_k^m are the estimated and measured current at time step k, respectively.

7.2.2 Control Law

Combining (7.15), (7.16) with (7.19), the current control law is

$$\begin{cases} u_d = -L_q \omega_e \hat{i}_q + k_p (i_d^* - \hat{i}_d) + k_i \int_0^t (i_d^* - \hat{i}_d)\, \mathrm{d}\tau \\ u_q = L_d \omega_e \hat{i}_d + \lambda_m \omega_e^m k_p (i_q^* - \hat{i}_q) + k_i \int_0^t (i_q^* - \hat{i}_q)\, \mathrm{d}\tau. \end{cases} \tag{7.20}$$

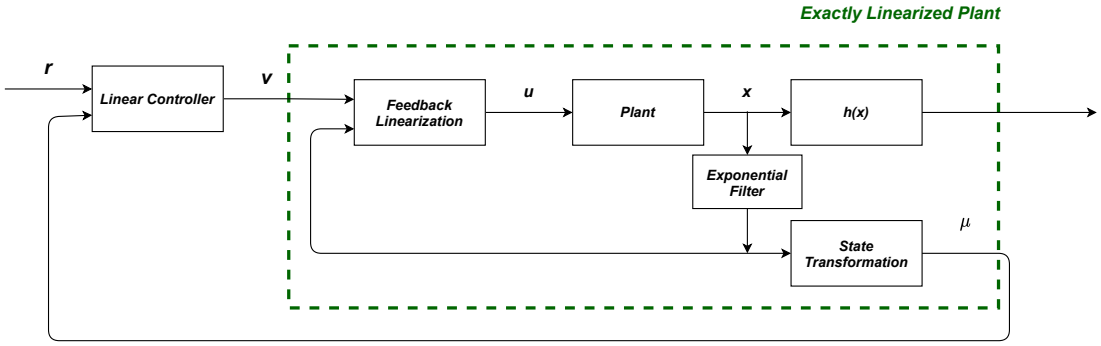


Figure 7.2: Feedback Linearization

The aforementioned control law is implemented in the 2FOC device. Such device can be programmed only by exploiting C fixed point arithmetic, and so scaling factors are used. In particular the control law (7.20) is written in the recursive form

```
1  //  IQ CONTROL LOOP
2  int iQerror = IqRef-Iq_EF;
3  long Qalpha = __builtin_mulss((__builtin_mulss(Id_EF,2))/100,(gQEVelocity/1000));
4  VqA += __builtin_mulss(iQerror-iQerror_old,IKp) + __builtin_mulss(iQerror+iQerror_old,IKi)
       + Qalpha - Qalpha_old +
5  (long)((gQEVelocity - gQEVelocity_old)/2);
6  iQerror_old = iQerror;
7  Qalpha_old = Qalpha;
8  gQEVelocity_old = gQEVelocity;
9
10 if (VqA > IIntLimit)
11     VqA = IIntLimit;
12 else if (VqA < -IIntLimit)
13     VqA = -IIntLimit;
14
15 Vq = (int)(VqA>>IKs);
16
17 //  ID CONTROL LOOP
18 int iDerror = -Id_EF;
19 long Dalpha =
       __builtin_mulss(__builtin_mulss((__builtin_mulss(Iq_EF,2))/100,(gQEVelocity/1000)),1);
20
21 VdA += __builtin_mulss(iDerror-iDerror_old,IKp) + __builtin_mulss(iDerror+iDerror_old,IKi);
22 iDerror_old = iDerror;
23 Dalpha_old = Dalpha;
24 if (VdA > IIntLimit)
25     VdA = IIntLimit;
26 else if (VdA < -IIntLimit)
27     VdA = -IIntLimit;
28
29 int Vd = (int)(VdA>>IKs);
```

where Id_{EF} and Iq_{EF} are the filtered direct and quadrature currents.

Those currents are obtained by the exponential filter, that is implemented as follow

```
long Iq_EF_shifted = __builtin_mulss(Iq_EF_old,31) +
    __builtin_mulss(I2Tdata.IQMeasured,32) - __builtin_mulss(I2Tdata.IQMeasured,31);
    //alpha = 31/32
Iq_EF = (int)(Iq_EF_shifted/32);
Iq_EF_old = Iq_EF;

long Id_EF_shifted = __builtin_mulss(Id_EF_old,31) +
    __builtin_mulss(I2Tdata.IDMeasured,32) - __builtin_mulss(I2Tdata.IDMeasured,31);
    //alpha = 31/32
Id_EF = (int)(Id_EF_shifted/32);
Id_EF_old = Id_EF;
```

In order to be able to obtain a fine tuning of the filter, the law is obtained by considering the filter parameter $\alpha \in (0,32)$. Furthermore, the shifting parameter for computation in fixed point is $IKs = 10$.
The quantity Id_{EF} and Iq_{EF} are in mA, the $gQEVelocity$ is in $tick/ms$, and control inputs V_{dq} are in mV. Considering this, all the parameters that multiply the states are scaled on the right measurements unit.

7.2.3 Results

The control law (7.20) together with the exponential filter (7.19) are implemented on the board controlling the humanoid robot knee motor with a sampling period of $50\mu s$. The α parameter of the filter is set to 0.9688, resulting in a 100Hz bandwidth. The proportional and integrative gains have been set to be

$$\begin{aligned} k_p &= 100 \\ k_i &= 5. \end{aligned} \tag{7.21}$$

Step Response

To test the goodness of the exponential filter, a step response experiment was defined. The quadrature current was set to be a step with amplitude $1A$, while the direct component is always set to $0A$. It is evident that the proposed control strategy gives much better performance in terms of currents tracking by filtering out the noise coming from the sensors – Fig. 7.3, Fig. 7.4.

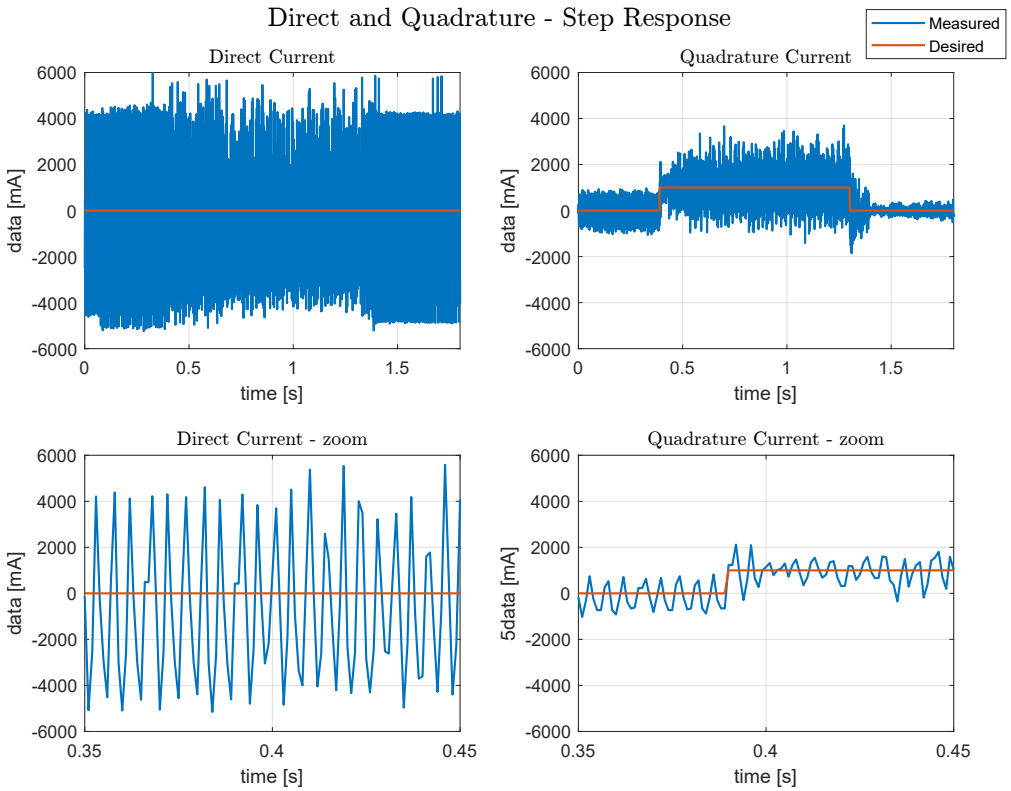


Figure 7.3: Step response - without the exponential filter.

Walking Reference Tracking

The implemented controller is tested exploiting a desired trajectory of a real walking scenario: the *whole-body QP* output torque trajectory is transformed into a current trajectory and fed to the controller. Also in this case, the controller guarantees the tracking of both direct and quadrature currents – Fig. 7.5.

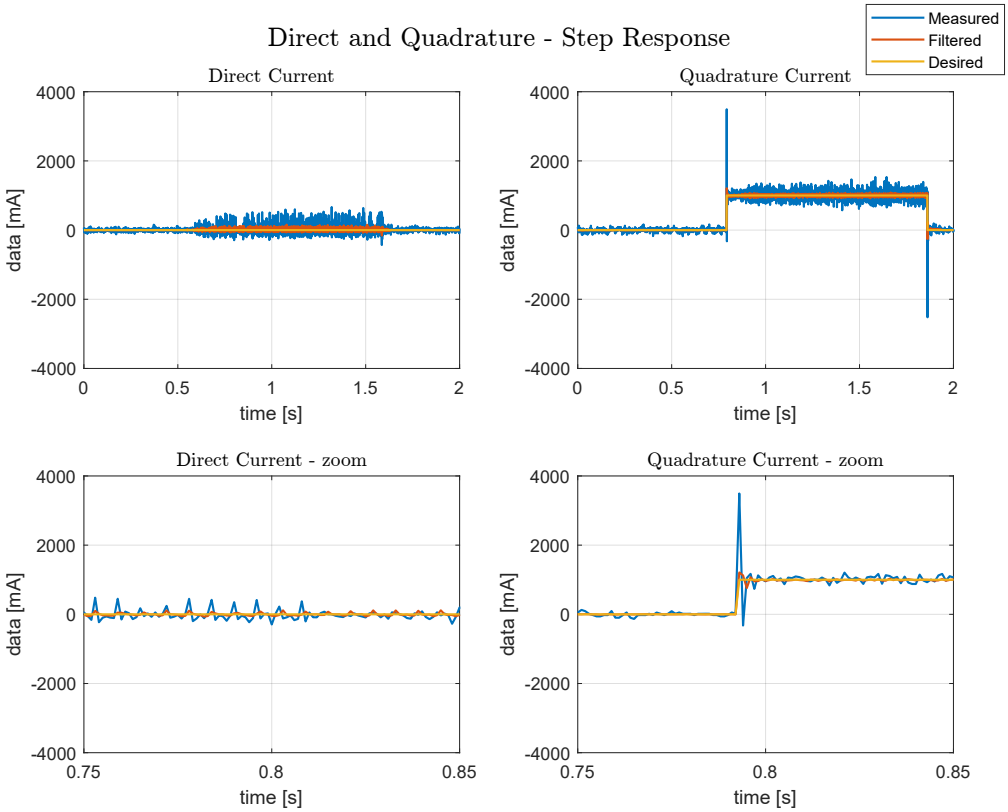


Figure 7.4: Step response - with the exponential filter.

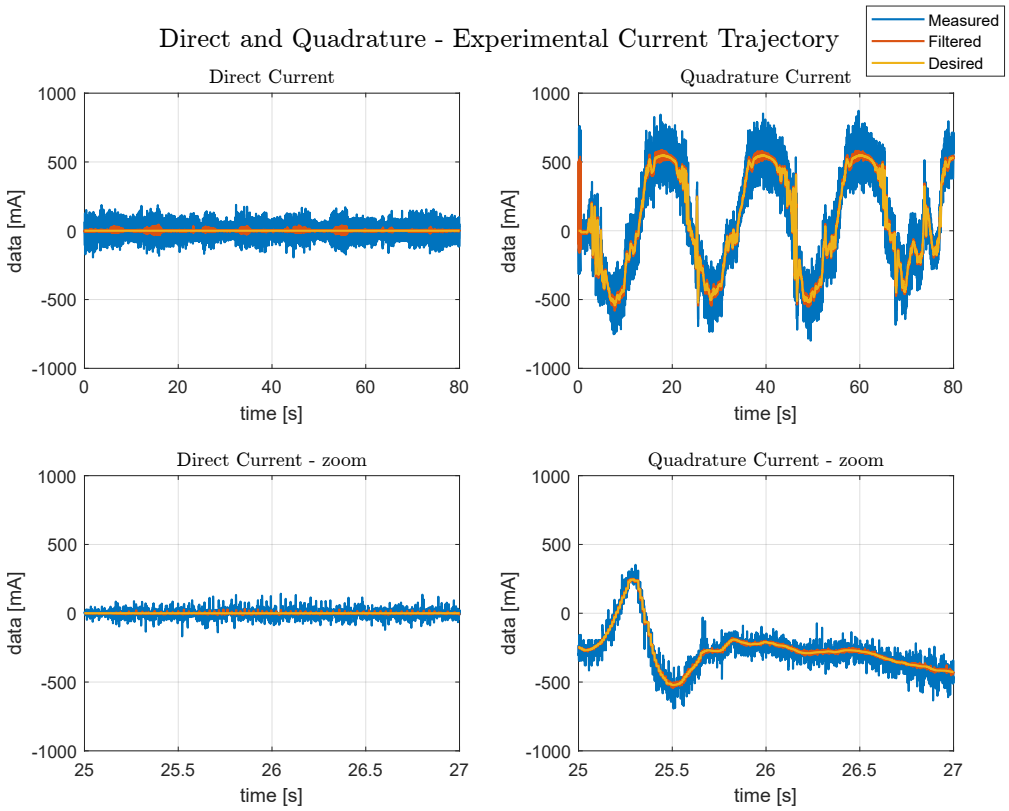


Figure 7.5: Currents Tracking Performances.

Bibliography

Matteo Fumagalli, Serena Ivaldi, Marco Randazzo, Lorenzo Natale, Giorgio Metta, Giulio Sandini, and Francesco Nori. Force feedback exploiting tactile and proximal force/torque sensing: Theory and implementation on the humanoid robot icub. *Autonomous Robots*, 33(4):381–398, November 2012. ISSN 0929-5593. doi: 10.1007/s10514-012-9291-2.

Christopher Ho Tin Lee, K.T. Chau, Liu Chunhua, and Ching Chan. Overview of magnetless brushless machines. *IET Electric Power Applications*, 12, 08 2017. doi: 10.1049/iet-epa.2017.0284.

Lennart Ljung. *System Identification: Theory for the User*. Prentice-Hall, Inc., USA, 1986. ISBN 0138816409.

Giorgio Metta, Paul Fitzpatrick, and Lorenzo Natale. Yarp: Yet another robot platform. *International Journal of Advanced Robotic Systems*, 3(1): 8, 2006. doi: 10.5772/5761. URL `https://doi.org/10.5772/5761`.

Giorgio Metta, Lorenzo Natale, Francesco Nori, Giulio Sandini, David Vernon, Luciano Fadiga, Claes von Hofsten, Kerstin Rosander, Manuel Lopes, José Santos-Victor, Alexandre Bernardino, and Luis Montesano. The icub humanoid robot: An open-systems platform for research in cognitive development. *Neural Networks*, 23(8):1125–1134, 2010. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2010.08.010. URL `https://www.sciencedirect.com/science/article/pii/S0893608010001619`. Social Cognition: From Babies to Robots.

Fardila Mohd Zaihidee, Saad Mekhilef, and Marizan Mubin. Robust speed control of pmsm using sliding mode control (smc)—a review. *Energies*, 12: 1669, 05 2019. doi: 10.3390/en12091669.

C. Novara. *Nonlinear Control and Aerospace Applications: lecture notes. Politecnico di Torino, 2017.*

Giulio Romualdi, Stefano Dafarra, Yue Hu, Prashanth Ramadoss, Francisco Javier Andrade Chavez, Silvio Traversaro, and Daniele Pucci. A benchmarking of dcm-based architectures for position, velocity and torque-controlled humanoid robots. *International Journal of Humanoid Robotics*, 17(01):1950034, 2020. doi: 10.1142/S0219843619500348. URL `https://doi.org/10.1142/S0219843619500348`.

G. Stanley. *Exponential Filter - https://gregstanleyandassociates.com/whitepapers/FaultDiagnosis/Filtering/Exponential-Filter/exponential-filter.htm.*

N. G. Tsagarakis, G. Metta, G. Sandini, D. Vernon, R. Beira, F. Becchi, L. Righetti, J. Santos-Victor, A. J. Ijspeert, M. C. Carrozza, and D. G. Caldwell. icub: the design and realization of an open humanoid platform for cognitive and neuroscience research. *Advanced Robotics*, 21(10):1151–1175, 2007. doi: 10.1163/156855307781389419. URL `https://doi.org/10.1163/156855307781389419`.