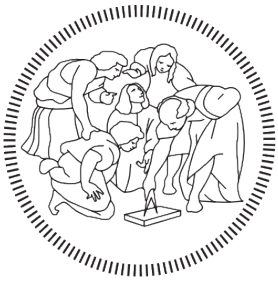


POLITECNICO
MILANO 1863

School of Industrial and Information Engineering
Master's Degree in Space Engineering

Design, prototyping and testing of a reaction wheel assembly for an air-bearing spacecraft attitude simulator

Supervisor: Prof. Francesco Topputo
Co-supervisor: Gianfranco Di Domenico

Candidate:
Corsi Giulio
944379

Department of Aerospace Science and Technology
Academic Year 2020-2021

Abstract

In the last two decades, the number of small satellites launched per year has grown at nearly exponential rates. However, despite the increasing number of launches, a problem yet to be addressed in order to establish themselves as an even more disruptive technology is their high failure rate. This can be explained by both the impossibility of testing complex systems in their operational environment, the space, and the cheapness and affordability of the employed technologies, especially for small satellites developed by universities or smaller agencies. A methodology to allow additional testing capabilities at relatively low costs is represented by Hardware-In-the-Loop (HIL) simulations, a technique for performing system-level testing of embedded systems in a comprehensive, cost-effective, and repeatable manner, increasing the chances of success of the mission. In this framework, the EXTREMA project, which aims at achieving autonomous guidance and navigation on CubeSats, will exploit the possibilities of HIL simulations through the development of a testing facility, the EXTREMA Simulation Hub (ESH), in which integrated simulations of guidance, navigation, and control (GNC) systems for deep-space CubeSats will be performed. A key component of the ESH is an attitude simulation platform that will mimic the attitude dynamics of a CubeSat during an interplanetary transfer. The main focus of this thesis is to develop a system, based on reaction wheels, to control the attitude behavior of the simulator, allowing it to reproduce the spacecraft orientation during the transfer. To do so, the reaction wheels will be sized as to fulfill the mission requirements and minimizing their power demand. Moreover, a mathematical and numerical model of the platform will be developed in Simulink to assess the performances of the control system, designed to control the orientation along three orthogonal axes. Furthermore, the hardware components will be selected in the attempt to reduce the costs and size of the system, and will then be integrated to obtain the first prototype of a single reaction wheel assembly. A series of tests will be carried out with the chosen components in order to assess their performances and validate the mathematical model.

Keywords: reaction wheel, attitude simulator, air-bearing, hardware-in-the-loop

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the last two decades, the space sector has seen a sharp growth: in fact, after the decline in the 1990s and early 2000s, the number of satellites launched has started to grow again at rapid rates [1] and, according to forecasts, will continue also in the next years. This incredible growth is certainly linked to the development of small satellites: indeed, in these decades the number of small satellites launched per year has grown at nearly exponential rates [2]. In particular, the great majority of them are composed of CubeSats, small satellites characterized by a compact, standardized form factor based on units of 10 cm x 10 cm x 10 cm (1U), which have made access to space available to a much wider audience. This innovation has benefited above all the commercial and educational sectors, which together account for over 80% of small satellite launches in 2018 [3]. Anyway, after two decades of launches, a problem they have yet to address to establish themselves even more as a disruptive technology is their high failure rate [1]. This issue can be related to the impossibility of testing complex systems in their operational environment, as happens for satellites in space. While important space missions involving large satellites can exploit costly testing facilities, which represents only a fraction of the total mission costs, that is not the case for CubeSats and other small satellites, usually developed by universities (or private industries) for their cheapness and affordability.
In this framework, a methodology that can enhance the testing possibilities with relatively low costs is represented by Hardware-In-the-Loop (HIL) simulations, which is a technique for performing system-level testing of embedded systems in a comprehensive, cost-effective, and repeatable manner, according to [2]. This type of simulation improves the effectiveness of classical numerical simulations (linked to mathematical models of the system under study) by involving real hardware to simulate the more complex aspects of the mathematical model, which would require too much computational effort, or even be impossible to emulate on a computer. HIL simulations require the development of a real-time software simulation that models some parts of the system under test, together with the hardware simulating the remaining parts of the system and all the significant interactions with the operational environment. The software simulation produces some outputs that will be fed to the embedded system, which in turn will use these data to generate some outputs to send again to the software, in a simulation loop involving also some hardware, hence the name.

[1] Number of satellites launched from 1957 to 2019, `https://www.statista.com/statistics/896699/number-of-satellites-launched-by-year/`

[2] Nanosatellites launches, `https://www.nanosats.eu/`

[3] Small satellite launches worldwide in 2018 and 2030, by application sector, `https://www.statista.com/statistics/1086625/global-small-sat-market-applications/`

1.1 Motivations

As explained above, HIL simulations are essential for testing small satellites before they are launched into space, increasing the chances of success of the mission. This is one of the main aspects on which is based the EXTREMA project (Engineering Extremely Rare Events in Astrodynamics for Deep-Space Missions in Autonomy), which is currently being carried out by the Deep-space Astrodynamics Research & Technology (DART) Group [4] at Politecnico di Milano. EXTREMA is an ERC-funded project which aims to enable interplanetary CubeSats to perform guidance, navigation, and control (GNC) procedures autonomously, revolutionizing the way deep-space missions are carried out [5]. The goal is to enable self-driving CubeSats, capable of traveling in deep space without requiring any control from ground. The project is built up on three pillars: the first, autonomous navigation, envisions the development of an optical navigation technique that extracts the line of sight of the celestial bodies to infer the state of the deep-space spacecraft; the second, autonomous guidance and control, deals with the development of a lightweight, robust closed-loop guidance algorithm; the third, ballistic capture, addresses the definition of the corridors for ballistic capture, an extremely rare phenomenon that allows for planetary capture without any energetic effort. The outcomes from the three pillars will be integrated into a fully-functional testing and validation facility, the EXTREMA Simulation Hub (ESH), developing a framework that will enable the testing of autonomous guidance laws and algorithms in a lab environment, where the space conditions that the interplanetary CubeSat will face will be reproduced accurately. As traditional space missions last months if not years, a ground-breaking accelerating approach will be employed to guarantee faster simulation times. A complete overview of the project and in particular of this accelerating approach is given in [3]. Regarding the other operative conditions, some effects, like that of the main propulsion system, will be monitored through a HIL simulation approach, while the evolution of the spacecraft attitude will be reproduced through an air-bearing spherical joint, which is the means for reproducing the torque-free conditions of a satellite in space. At the beginning of this work, the air-bearing was already present in the laboratory, but it lacked any actuation system for controlling its attitude behavior: this was the main focus in developing this thesis.
While many attitude actuators are available, reaction wheels are a common choice for spacecraft simulator attitude control: they offer precision torque generation with relatively low weight and high sensitivity when compared to thrusters. For accuracy and moderately fast maneuverability, reaction wheels are the preferred attitude control system because they allow continuous and smooth control while inducing the lowest possible disturbance torques [4].

1.2 Thesis objectives

Taking into account what has been said so far, the main research question was conceived as follows:

- **Research question**. How to develop an attitude actuation system based on reaction wheels capable of controlling the attitude dynamics of the simulator platform?

To answer this question, the following operative objectives are formulated:

- **Objective I**. Perform the sizing and design of the reaction wheels that will be employed by the attitude control system. This will include the architecture of the actuators along with a mathematical model to simulate the performances of the system.

[4] DART Group website, `https://dart.polimi.it/`

[5] European Research Council, *Engineering Extremely Rare Events in Astrodynamics for Deep-Space Missions in Autonomy*, `https://cordis.europa.eu/project/id/864697`

– **Objective II**. Implement the attitude control system, including both hardware and software necessary to operate the system. At the end of this phase, all components must be connected to the simulator and fully functional.

– **Objective III**. Testing of the system through experiments, in order to validate the model and check the correctness of the overall design and implementation of the attitude control system.

1.3 Outline

Chapter 2 describes the background of this work. First, the fundamental concepts of attitude dynamics are introduced: angular momentum, Euler's equations, and how to account for the presence of reaction wheels. Then, the spacecraft attitude simulators technology is presented, with a particular focus on air bearings. After that, the components already available in the laboratory are shown.
Chapter 3 illustrates the mathematical model of the system. The sizing process for the reaction wheels is discussed. Once the dimensions of the wheels are chosen, it is possible to develop a complete scheme in Simulink, which includes the reaction wheels and air-bearing simulator in the system module, the sensors module for the measurement, the reference trajectory generation module, and the control module. In the end, some results from the simulation are shown.
Chapter 4 is related to the hardware implementation of a first prototype of the system: both structural and electronic components are described, along with the reasons that led to their selection.
Chapter 5 gives details of the testing and validation phases. The software implementation to run the first experiments is explained, and the components are tested. Finally, the first prototype is tested to validate the results from the mathematical model.
Chapter 6 draws conclusions from the work and discusses some possible future developments.

Chapter 2

Background

2.1 Spacecraft attitude dynamics fundamentals

Spacecraft attitude dynamics represents the study of spacecraft orientation in orbit and its evolution over time. One of the fundamental field for the study of spacecraft dynamics is the rotational motion of a rigid body or system of rigid bodies. This will be the starting point of this section, where it will be presented the fundamental aspects of describing the orientation, angular momentum, energy and differential equations of motion of a rigid body. After discussing the rigid body kinematics, the inertia properties of a body about arbitrary reference points are developed. The angular momentum and kinetic energy of a rigid body are critical to discussing Euler's equations of motion. Then, the presence of the reaction wheels and the modification that they impose to the equations of motion are discussed.

2.1.1 Rotating coordinate frames

The starting point will be the angular rotation and orientation coordinates used to describe the motion of a rigid body. A coordinate frame can be fixed to a rigid body: in this case, the study of the rigid body attitude evolution coincides with the behaviour of the coordinate frame orientation.

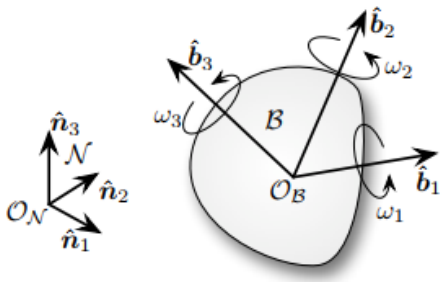


Figure 2.1: Coordinate frames representation.[14]

Let $\mathcal{N} : \{\mathcal{O}_{\mathcal{N}}, \hat{\mathbf{n}}_1, \hat{\mathbf{n}}_2, \hat{\mathbf{n}}_3\}$ be a fixed inertial (non accelerating) coordinate frame, defined through its origin $\mathcal{O}_{\mathcal{N}}$ and the unit vectors $\hat{\mathbf{n}}_1$, $\hat{\mathbf{n}}_2$, and $\hat{\mathbf{n}}_3$ that form an orthonormal frame, which satisfies $\hat{\mathbf{n}}_1 \times \hat{\mathbf{n}}_2 = \hat{\mathbf{n}}_3$. In the same way, we can define the body frame $\mathcal{B} : \{\mathcal{O}_{\mathcal{B}}, \hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2, \hat{\mathbf{b}}_3\}$ with origin $\mathcal{O}_{\mathcal{B}}$ and unit vectors $\hat{\mathbf{b}}_1$, $\hat{\mathbf{b}}_2$, and $\hat{\mathbf{b}}_3$. The body frame $\mathcal{B}$ is attached to a rigid body, so that describing the orientation of the rigid body is equivalent to study the relative attitude between the body fixed frame and the inertially fixed frame, where the attitude of the rigid-body

$A_{\mathcal{B}/\mathcal{N}}$ is defined as the relative orientation between the body fixed frame $\mathcal{B}$ and the inertial frame $\mathcal{N}$. Moreover, for the study of attitude dynamics, the translation of the rigid body is not of interest, since the orbital and attitude dynamics are generally decoupled, so that it is possible to consider just the relative orientation as:

$$\mathcal{B} = A_{\mathcal{B}/\mathcal{N}}\mathcal{N} \tag{2.1}$$

where $A_{\mathcal{B}/\mathcal{N}}$ is an orthonormal matrix also known as Direct Cosine Matrix (DCM).
Besides, it is possible to write the angular velocity of the two frames in their basis as $\boldsymbol{\omega}_{\mathcal{N}} = \omega_1\hat{\mathbf{n}}_1 + \omega_2\hat{\mathbf{n}}_2 + \omega_3\hat{\mathbf{n}}_3$ and $\boldsymbol{\omega}_{\mathcal{B}} = \omega_1^{'}\hat{\mathbf{b}}_1 + \omega_2^{'}\hat{\mathbf{b}}_2 + \omega_3^{'}\hat{\mathbf{b}}_3$. Additionally, the angular velocity of the frame $\mathcal{N}$ can be expressed in the body frame thanks to the attitude matrix as $\boldsymbol{\omega}_{\mathcal{N}}^{\mathcal{B}} = A_{\mathcal{B}/\mathcal{N}}\boldsymbol{\omega}_{\mathcal{N}}$.
At this point, the relative angular motion between the two frames can be described through the angular velocity vector

$$\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} = \boldsymbol{\omega}_{\mathcal{B}} - \boldsymbol{\omega}_{\mathcal{N}}^{\mathcal{B}} = \boldsymbol{\omega}_{\mathcal{B}} - A_{\mathcal{B}/\mathcal{N}}\boldsymbol{\omega}_{\mathcal{N}} \tag{2.2}$$

This vector is the instantaneous angular rotation vector of body $\mathcal{B}$ relative to $\mathcal{N}$, and is typically expressed in body frame vector components. If only these two frames are considered, the $\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}$ vector is often written simply as $\boldsymbol{\omega}$.
The last notion to be defined in relation to rotating frames is the *transport theorem*, which concerns the differentiation of a vector expressed in the body frame. When describing the time evolution of a vector, it is mandatory to specify an observer frame: the transport theorem is used to map a time derivative with respect to a frame $\mathcal{B}$ into the equivalent derivative with respect to another frame $\mathcal{N}$ as

$$\frac{^{\mathcal{N}}d\mathbf{r}}{dt} = \frac{^{\mathcal{B}}d\mathbf{r}}{dt} + \boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} \times \mathbf{r} \tag{2.3}$$

where $\mathbf{r}$ is a general vector, for example a position vector.

2.1.2 Parameters for attitude representation

In the previous section, the DCM has been introduced: thanks to this matrix, we are able to switch from one reference frame to another one. Being able to define the orientation of a frame with respect to another one is an important task in attitude dynamics, that can be carried out in several ways. To describe the three-dimensional orientation of rigid bodies a minimum of three coordinates, or attitude parameters, are required. Anyway, sometimes redundant parameters are used either to improve the physical insight into the transformation or to avoid some singularity issues. In this section, the attitude parameters mainly used in the following chapters are presented.

Direction Cosine Matrix

The rotation matrix $A_{\mathcal{B}/\mathcal{N}}$ is a fundamental way to express the orientation of $\mathcal{B}$ with respect to $\mathcal{N}$. According to [14], the parameters of the matrix are determined as

$$A_{\mathcal{B}/\mathcal{N}} = \begin{bmatrix} \hat{\mathbf{b}}_1 \cdot \hat{\mathbf{n}}_1 & \hat{\mathbf{b}}_1 \cdot \hat{\mathbf{n}}_2 & \hat{\mathbf{b}}_1 \cdot \hat{\mathbf{n}}_3 \\ \hat{\mathbf{b}}_2 \cdot \hat{\mathbf{n}}_1 & \hat{\mathbf{b}}_2 \cdot \hat{\mathbf{n}}_2 & \hat{\mathbf{b}}_2 \cdot \hat{\mathbf{n}}_3 \\ \hat{\mathbf{b}}_3 \cdot \hat{\mathbf{n}}_1 & \hat{\mathbf{b}}_3 \cdot \hat{\mathbf{n}}_2 & \hat{\mathbf{b}}_3 \cdot \hat{\mathbf{n}}_3 \end{bmatrix} \tag{2.4}$$

As it can be seen, every component is expressed with a dot product, which returns the cosine of the angle between the two unit directions inside the dot product, from which the name Direction Cosine Matrix. The DCM is an orthonormal matrix, with a determinant equal to 1 and the inverse given by the transpose operator

$$A_{\mathcal{B}/\mathcal{N}} A_{\mathcal{B}/\mathcal{N}}^T = I_{3\times 3}\,; \qquad \det(A_{\mathcal{B}/\mathcal{N}}) = 1 \tag{2.5}$$

where $I_{3\times 3}$ is the 3 by 3 identity matrix. These properties express in a convenient matrix form the fact that the nine parameters of the DCM have six constraints: three expressing orthogonality between the axes and three expressing the invariability of magnitude of the unit vectors. In fact, the rotation will not affect the magnitude of the vectors. The principal use of the DCM is to realize three-dimensional coordinate transformations

$$\mathbf{r}_{\mathcal{B}} = A_{\mathcal{B}/\mathcal{N}} \mathbf{r}_{\mathcal{N}} \tag{2.6}$$

Knowing the relative angular velocity between two frames $\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}$, it is possible to define how the DCM changes with time through the differential kinematic equation [15]

$$\dot{A}_{\mathcal{B}/\mathcal{N}} = -[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}] A_{\mathcal{B}/\mathcal{N}} \tag{2.7}$$

where the tilde matrix notation represents a skew symmetric matrix defined as

$$[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{2.8}$$

One last important remark about the DCM concerns the addition property [14], which allows us to obtain directly the final matrix of rotation even if multiple rotations are carried out. For example, we already saw that $A_{\mathcal{B}/\mathcal{N}}$ express the rotation between the frame $\mathcal{N}$ and $\mathcal{B}$. If we define another frame $\mathcal{C}$, the rotation between $\mathcal{B}$ and $\mathcal{C}$ will be expressed by the DCM $A_{\mathcal{C}/\mathcal{B}}$. At this point, with the addition property, it is possible to obtain the attitude of $\mathcal{C}$ relative to $\mathcal{N}$ simply multiplying these DCMs with each other:

$$A_{\mathcal{C}/\mathcal{N}} = A_{\mathcal{C}/\mathcal{B}} A_{\mathcal{B}/\mathcal{N}} \tag{2.9}$$

Euler's angles

Euler's angles are a different type of attitude parameters that describe the rotation between two frames through three sequential one-axis rotations, indicated as ψ, θ, and ϕ. The Euler's angles have a clear physical interpretation and are a minimal representation because they use only 3 parameters. Anyway, being a minimal orientation description, it contains attitudes where the description or the associated differential kinematic equations become singular. Twelve different combinations of rotation exist, according to the different sequences used for the three single-axis rotations. Half of them have all different indexes, like the 3-2-1 sequence, while the other half, instead, contains one repetition in the indexes, like the 3-1-3 sequence for example. The singularities vary in accordance with the sequence: when all indexes are different, the singularity

condition is $\theta = (2n+1)\frac{\pi}{2}$, while in the other case the singularity occurs when $\theta = n\pi$. From the Euler's angles, it is possible to write the DCM thanks to its addition property

$$A_{\mathcal{B}/\mathcal{N}} = A_{313}(\phi, \theta, \psi) = A_3(\psi)A_1(\theta)A_3(\phi) \tag{2.10}$$

where

$$A_3(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}; \quad A_1(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}; \quad A_1(\theta) = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

Multiplying them, it results:

$$A_{313} = \begin{bmatrix} \cos\psi\cos\phi - \sin\psi\sin\phi\cos\theta & \cos\psi\sin\phi + \sin\psi\cos\phi\cos\theta & \sin\psi\sin\theta \\ -\sin\psi\cos\phi - \cos\psi\sin\phi\cos\theta & -\sin\psi\sin\phi + \cos\psi\cos\phi\cos\theta & \cos\psi\sin\theta \\ \sin\phi\sin\theta & -\cos\phi\sin\theta & \cos\theta \end{bmatrix} \tag{2.12}$$

On the other hand, the inverse transformations from the DCM to the Euler's angles are:

$$\begin{cases} \phi = -\tan^{-1}\left(\dfrac{A_{31}}{A_{32}}\right) & (2.13) \\ \theta = \cos^{-1}\left(A_{33}\right) & (2.14) \\ \psi = \tan^{-1}\left(\dfrac{A_{13}}{A_{23}}\right) & (2.15) \end{cases}$$

As it can be seen, the transformations from Euler's angles to DCM and viceversa depend strictly on the sequence chosen. Equations from 2.12 to 2.15 are valid only for the sequence 3-1-3. The differential kinematic equations relate the 3-1-3 Euler's angles rates to the body angular velocity vector $\boldsymbol{\omega}$ through [15]

$$\begin{Bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{Bmatrix} = \begin{bmatrix} \frac{\sin\psi}{\sin\theta} & \frac{\cos\psi}{\sin\theta} & 0 \\ \cos\psi & -\sin\psi & 0 \\ -\sin\psi\frac{\cos\theta}{\sin\theta} & -\cos\psi\frac{\cos\theta}{\sin\theta} & 1 \end{bmatrix} \begin{Bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{Bmatrix}_{\mathcal{B}} \tag{2.16}$$

It is important to notice that for the other sequences of rotations, different sets of kinematic equations are obtained.

Euler axis/angle

This representation is founded upon Euler's rotation theorem, which states that any rotation can be indicated thanks to an axis, which remains fixed during the rotation, and an angle that expresses the magnitude of the rotation. So, instead of using three different rotation (as in the Euler's angles case), it is possible to rotate from the frame $\mathcal{N}$ to the frame $\mathcal{B}$ using only a single rotation around a specific axis. The axis (Euler axis), indicated by $\hat{\mathbf{e}}$, and the angle (Euler angle), indicated by Φ, are respectively the eigenvector and the eigenvalue of the transformation [16]. Euler axis/angle parameters are useful since they are only four parameters, plus a constraint condition given by the normalization of $\hat{\mathbf{e}}$. Moreover, this representation is also useful to introduce another attitude parameter called quaternion.

Quaternions

In general, a quaternion is a vector, indicated by $\mathbf{q}$, with four components q_1, q_2, q_3, and q_4. Usually, a quaternion is divided into a vector part q with three components, plus a scalar part, which can be the first or last component. In this work, the fourth component is chosen as the scalar part, so that the quaternion can be written as:

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad , \quad q_4 \tag{2.17}$$

Quaternions are used to parameterize rotations: in fact, a quaternion can be related to an Euler axis/angle as:

$$\begin{cases} q_1 = \hat{\mathbf{e}}_1 \sin\dfrac{\Phi}{2} & (2.18) \\ q_2 = \hat{\mathbf{e}}_2 \sin\dfrac{\Phi}{2} & (2.19) \\ q_3 = \hat{\mathbf{e}}_3 \sin\dfrac{\Phi}{2} & (2.20) \\ q_4 = \cos\dfrac{\Phi}{2} & (2.21) \end{cases}$$

In this case, the quaternion is a unit quaternion, so that $q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$. It is important to note that, since there are four parameters in total, there is a duality in representing an orientation, with $\mathbf{q}$ and $-\mathbf{q}$. In this case, the scalar part q_4 is fundamental: if q_4 is greater than zero, the quaternion is describing a short rotation, smaller than 180 degrees, while if q_4 is negative, the rotation is greater than 180 degrees. From the quaternion's notation, it is possible to evaluate the DCM according to:

$$A_{\mathcal{B}/\mathcal{N}} = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \tag{2.22}$$

The inverse transformation have no singular condition and is represented as:

$$\begin{cases} q_1 = \dfrac{1}{4q_4}(A_{23} - A_{32}) & (2.23) \\ q_2 = \dfrac{1}{4q_4}(A_{31} - A_{13}) & (2.24) \\ q_3 = \dfrac{1}{4q_4}(A_{12} - A_{21}) & (2.25) \\ q_4 = \pm\dfrac{1}{2}(1 + A_{11} + A_{22} + A_{33})^{\frac{1}{2}} & (2.26) \end{cases}$$

The quaternion rates relate to the body angular velocity vector $\boldsymbol{\omega}$ through:

$$\begin{Bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \\ \dot{q_4} \end{Bmatrix} = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \begin{Bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{Bmatrix} \tag{2.27}$$

Quaternions have no physical meaning and, as we already saw, they are also non-unique. However, they are a minimal global representation, which means they are singularity free. Furthermore, [16] states that it is more efficient to specify rotations with quaternions rather than attitude matrix, because they only have four components instead of nine, and have only one constraint (the unitary norm), on the contrary of the constraints imposed on the attitude matrix by orthogonality.

2.1.3 Inertia matrix

This section will deal with the definition and properties of the inertia matrix. Recalling the two coordinate frames $\mathcal{N}$ and $\mathcal{B}$ described before, we can remember that the frame $\mathcal{B}$ is attached to a rigid body. Moreover, we can now assume that this rigid body has its center of mass coincident with the origin $\mathcal{O}_\mathcal{B}$, as shown in Fig. 2.2. The position of the center of mass in the inertial frame is defined by $\mathbf{R}_c$, while $\mathbf{r} = [r_1, r_2, r_3]^T$ defines the position of an infinitesimal point mass dm in the body frame. In addition, $\boldsymbol{\omega}$ represent the relative angular velocity between the two frames.

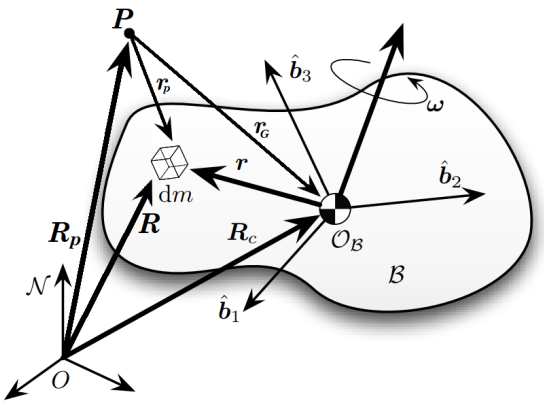


Figure 2.2: Body attached to reference frame $\mathcal{B}$ [14]

From these quantities, it is possible to define the inertia matrix I_c of the body about its center of mass as

$$I = \int_\mathcal{B} -[\mathbf{r}][\mathbf{r}]\mathrm{dm} \qquad \text{, where} \quad [\mathbf{r}] = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix} \tag{2.28}$$

Developing the calculations, leads to the following form of the inertia matrix

$$I = \int_\mathcal{B} \begin{bmatrix} r_2^2 + r_3^2 & -r_1 r_2 & -r_1 r_3 \\ -r_1 r_2 & r_1^2 + r_3^2 & -r_2 r_3 \\ -r_1 r_3 & -r_2 r_3 & r_1^2 + r_2^2 \end{bmatrix} \mathrm{dm} \tag{2.29}$$

In Eq. (2.29), the terms in diagonal are called *moments of inertia* and they are always positive, since they are defined as a sum of square quantities. Identifying these three quantities with the symbols I_{11}, I_{22}, and I_{33}, it is easy to check that they are subjected to the constraints $I_{11} + I_{22} \geq I_{33}$ and $I_{11} - I_{22} \leq I_{33}$. On the other hand, the terms out of the diagonal are called *products of inertia* and they can be positive, negative or null: these terms are a measure of the

imbalance in the mass distribution [17]. In general, the inertia matrix reflects how the mass of a rigid body is distributed, which is extremely important when analyzing the body response to applied torques since it is not the mass alone but how that mass is distributed that characterizes the body's rotary motion. For example, if the $\hat{\mathbf{b}}_1\hat{\mathbf{b}}_2$ plane is a plane of symmetry, then the products of inertia with the index 3 will be null. In fact, considering I_{13}, which is given by the integral of $-r_1 r_3$, the symmetry plane implies that for any value of r_1 there exist two identical infinitesimal masses located at $+r_3$ and $-r_3$, so that these two elements cancel out themselves. The same can be said in relation to the planes $\hat{\mathbf{b}}_2\hat{\mathbf{b}}_3$ and $\hat{\mathbf{b}}_1\hat{\mathbf{b}}_3$. It follows that if the body has two planes of symmetry relative to its body frame, then all three products of inertia vanish, and I becomes a diagonal matrix such that,

$$I_{diag} = \begin{bmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{bmatrix} \tag{2.30}$$

In this case, I_1, I_2, and I_3 are called the principal moments of inertia, and the body frame axes are the body's principal axes of inertia. For a general three-dimensional body, it is always possible to find three mutually orthogonal axis for which the products of inertia are zero, and the inertia matrix takes a diagonal form. If I is the non-diagonal inertia matrix in a general body frame, it can be diagonalized such that

$$CIC^T = I_{diag} \tag{2.31}$$

where the rows of C are given by the eigenvectors of I.

2.1.4 Angular momentum

Referring again to Fig. 2.2, the infinitesimal mass element dm has a linear momentum that can be written as $\dot{\mathbf{R}}\,dm$. Additionally, it is possible to define the moment of the linear momentum of dm about a generic point P as $\mathbf{r}_p \times \dot{\mathbf{R}}\,dm$, that is the cross product between the distance between P and dm, and the linear momentum of the mass element. The moment of momentum, or *angular momentum*, of the entire body is the integral of this cross product over all of its mass elements. That is, the total angular momentum of the body relative to point P is [18]

$$\mathbf{H}_P = \int_{\mathcal{B}} \mathbf{r}_p \times \dot{\mathbf{R}}\, dm \tag{2.32}$$

From Fig. 2.2, we can rewrite the vectors inside Eq. (2.32) as $\mathbf{r}_p = \mathbf{r}_G + \mathbf{r}$ and $\dot{\mathbf{R}} = \dot{\mathbf{R}}_c + \dot{\mathbf{r}}$, so that the total angular momentum becomes:

$$\mathbf{H}_P = \int_{\mathcal{B}} (\mathbf{r}_G + \mathbf{r}) \times (\dot{\mathbf{R}}_c + \dot{\mathbf{r}})\, dm \tag{2.33}$$

that can be extended to

$$\mathbf{H}_P = \int_{\mathcal{B}} \mathbf{r}_G \times \dot{\mathbf{R}}_c\, dm + \int_{\mathcal{B}} \mathbf{r}_G \times \dot{\mathbf{r}}\, dm + \int_{\mathcal{B}} \mathbf{r} \times \dot{\mathbf{R}}_c\, dm + \int_{\mathcal{B}} \mathbf{r} \times \dot{\mathbf{r}}\, dm \tag{2.34}$$

Since $\mathbf{r}_G$ and $\dot{\mathbf{R}}_c$ represent quantities that do not vary along the body, they can be taken outside the integral in the following way

$$\mathbf{H}_P = \mathbf{r}_G \times \dot{\mathbf{R}}_c \int_{\mathcal{B}} dm + \mathbf{r}_G \times \int_{\mathcal{B}} \dot{\mathbf{r}}\, dm + \int_{\mathcal{B}} \mathbf{r}\, dm \times \dot{\mathbf{R}}_c + \int_{\mathcal{B}} \mathbf{r} \times \dot{\mathbf{r}}\, dm \tag{2.35}$$

At this point, it is possible to take advantage of the definition of center of mass, which states that $\int_{\mathcal{B}} \mathbf{r}\, dm = 0$, and so also $\int_{\mathcal{B}} \dot{\mathbf{r}}\, dm = 0$. Doing so, the second and third term in Eq. (2.35) result to be null, leading to

$$\mathbf{H}_P = \underbrace{\mathbf{r}_G \times M\, \dot{\mathbf{R}}_c}_{\mathbf{H}_{cm}} + \underbrace{\int_{\mathcal{B}} \mathbf{r} \times \dot{\mathbf{r}}\, dm}_{\mathbf{H}_b} \tag{2.36}$$

where M is the total mass of the body. This equation is composed of two terms: the first one, $\mathbf{H}_{cm}$, represents the angular momentum of the body's center of mass about the point P, while the second one, $\mathbf{H}_b$, is the angular momentum of the body about its own center of mass. This distinction can be useful if we want to rewrite the total angular momentum about another point different from P, like for example the origin $\mathcal{O}$: the only thing that should be changed in Eq. (2.36) is the distance $\mathbf{r}_G$, which should be replaced with $\mathbf{R}$. Furthermore, taking into account that the body is rotating with an angular velocity $\boldsymbol{\omega}$, we can recall the transport theorem from Eq. (2.3) to evaluate the derivative $\dot{\mathbf{r}}$. Doing so, the angular momentum of the body becomes

$$\mathbf{H}_b = \int_{\mathcal{B}} \mathbf{r} \times (\boldsymbol{\omega} \times \mathbf{r})\, dm \tag{2.37}$$

Finally, this formulation can be simplified recalling the definition of inertia matrix given in Eq. (2.29). In fact, developing the calculations related to the cross products, it is possible to write the angular momentum of the body as

$$\mathbf{H}_b = I\boldsymbol{\omega} \tag{2.38}$$

where I is the 3×3 inertia matrix and $\boldsymbol{\omega}$ is the 3×1 angular velocity vector. If we want to study only the attitude orientation of a satellite, it is sufficient to account for the rotational motion of the rigid body and focus on $\mathbf{H}_b$, without considering the spacecraft translation, and so the angular momentum of the center of mass $\mathbf{H}_{cm}$. If a *principal* coordinate frame $\mathcal{B}$ is chosen, then the angular momentum of the body can be written as

$$\mathbf{H}_b = \begin{Bmatrix} I_1\, \omega_1 \\ I_2\, \omega_2 \\ I_3\, \omega_3 \end{Bmatrix} \tag{2.39}$$

2.1.5 Kinetic energy of a rigid body

The total kinetic energy of a rigid body is the integral over the whole body of the kinetic energy $\frac{1}{2}\dot{\mathbf{R}} \cdot \dot{\mathbf{R}}\, dm$ of the infinitesimal mass dm, it is a scalar value and is indicated with T.

$$T = \frac{1}{2} \int_{\mathcal{B}} \dot{\mathbf{R}} \cdot \dot{\mathbf{R}}\, dm \tag{2.40}$$

This equation can be simplified in a similar way to what was done previously for the angular momentum. First, we substitute $\dot{\mathbf{R}} = \dot{\mathbf{R}}_c + \dot{\mathbf{r}}$ inside Eq. (2.40), and developing the scalar products we find

$$T = \frac{1}{2}\left(M\dot{\mathbf{R}}_c \cdot \dot{\mathbf{R}}_c + 2\dot{\mathbf{R}}_c \cdot \int_{\mathcal{B}} \dot{\mathbf{r}}\, dm + \int_{\mathcal{B}} \dot{\mathbf{r}} \cdot \dot{\mathbf{r}}\, dm \right) \tag{2.41}$$

As already done before, we can cancel the second term in this equation using the definition of center of mass. In addition, we can also use the transport theorem to substitute $\dot{\mathbf{r}}$ in the last term

$$T = \frac{1}{2}\left(M\dot{\mathbf{R}}_c \cdot \dot{\mathbf{R}}_c + \int_{\mathcal{B}} \dot{\mathbf{r}} \cdot (\boldsymbol{\omega} \times \mathbf{r})\, dm \right) \tag{2.42}$$

At this point, it is possible to rewrite Eq. (2.42) exploiting the property of the scalar triple product which states that the order of the operands can be shifted according to the rule $A \cdot (B \times C) = B \cdot (C \times A)$. Applying this leads to

$$T = \underbrace{\frac{1}{2} M\dot{\mathbf{R}}_c \cdot \dot{\mathbf{R}}_c}_{T_{trans}} + \underbrace{\frac{1}{2}\int_{\mathcal{B}} \boldsymbol{\omega} \cdot (\mathbf{r} \times \dot{\mathbf{r}})\, dm}_{T_{rot}} \tag{2.43}$$

As it can be seen, the kinetic energy is composed by two terms T_{trans} and T_{rot}, that are related respectively to the translational kinetic energy and rotational kinetic energy. Again, as already done for the angular momentum, it is possible to ignore the translational term because we are interested only in the study of the attitude. Analyzing the rotational kinetic energy, we can see that the angular velocity $\boldsymbol{\omega}$ can be taken outside the integral, since it does not change along the body. At this point the integral can be evaluated recalling Eq. (2.36) and Eq. (2.38)

$$T_{rot} = \frac{1}{2}\boldsymbol{\omega} \cdot I\boldsymbol{\omega} = \frac{1}{2}\boldsymbol{\omega}^T I \boldsymbol{\omega} \tag{2.44}$$

If a principal coordinate system is chosen for $\mathcal{B}$, the energy is written using the principal inertias as

$$T_{rot} = \frac{I_1}{2}\omega_1^2 + \frac{I_2}{2}\omega_2^2 + \frac{I_3}{2}\omega_3^2 \tag{2.45}$$

2.1.6 Euler's rotational equations

In this section, the equations for the rotational motion will be described. The quantities presented are still referred to Fig. 2.2. Each infinitesimal mass element dm feels a net external force $d\mathbf{F}_{net}$ and a net internal force $d\mathbf{f}_{net}$, so that Newton's second law can be written as

$$dm\ddot{\mathbf{R}} = d\mathbf{F}_{net} + d\mathbf{f}_{net} \tag{2.46}$$

where $\ddot{\mathbf{R}}$ is the absolute acceleration of the mass element in the inertial frame $\mathcal{N}$. Using again point P, the moment about P of the forces on mass element dm is

$$d\mathbf{M}_P = \mathbf{r}_p \times d\mathbf{F}_{net} + \mathbf{r}_p \times d\mathbf{f}_{net} \tag{2.47}$$

Collecting the term $\mathbf{r}_p$, we can write the right-hand side of the equation as $\mathbf{r}_p \times (d\mathbf{F}_{net} + d\mathbf{f}_{net})$. Then, substituting Eq. (2.46) and integrating over all the mass elements of the body yields

$$\mathbf{M}_{P_{net}} = \int_{\mathcal{B}} \mathbf{r}_p \times \ddot{\mathbf{R}}\, dm \tag{2.48}$$

Using the derivation rule, the integrand in Eq. (2.48) can be written as

$$\mathbf{r}_p \times \ddot{\mathbf{R}} = \frac{d}{dt}(\mathbf{r}_p \times \dot{\mathbf{R}}) - \dot{\mathbf{r}}_p \times \dot{\mathbf{R}} \tag{2.49}$$

which substituted in Eq. (2.48) leads to

$$\mathbf{M}_{P_{net}} = \frac{d}{dt}\int_{\mathcal{B}} (\mathbf{r}_p \times \dot{\mathbf{R}})\, dm - \int_{\mathcal{B}} \dot{\mathbf{r}}_p \times \dot{\mathbf{R}}\, dm \tag{2.50}$$

The first integral on the right-hand side of Eq. (2.50) is the total angular momentum (about point P) $\mathbf{H}_P$, as already shown in Eq. (2.32). Moreover, looking at Fig. 2.2, we notice that $\dot{\mathbf{r}}_p$ can be expressed as $\dot{\mathbf{r}}_p = \dot{\mathbf{R}} - \dot{\mathbf{R}}_p$. Using this formulation, the second term in Eq. (2.50) becomes $+\int_{\mathcal{B}} \dot{\mathbf{R}}_p \times \dot{\mathbf{R}}\, dm$, where $\dot{\mathbf{R}}_p$ can be taken outside the integral and it is possible to use the definition of center of mass, $\int_{\mathcal{B}} \dot{\mathbf{R}}\, dm = M\dot{\mathbf{R}}_c$, to write

$$\mathbf{M}_{P_{net}} = \dot{\mathbf{H}}_P + \dot{\mathbf{R}}_p \times M\dot{\mathbf{R}}_c \tag{2.51}$$

This equation represent the net moment of the forces about an arbitrary point P. Making some assumptions, the equation can be simplified: in fact, in some special cases the second term results to be null. This happens if the point P is at rest in the inertial frame, if the center of mass of the body is at rest (for example, the body is rotating around its center of mass), if P and the center of mass have parallel velocities ($\dot{\mathbf{R}}_p \parallel \dot{\mathbf{R}}_c$), or also if the point P is coincident with the center of mass. In all these cases, Eq. (2.51) reduces to

$$\mathbf{M}_{P_{net}} = \dot{\mathbf{H}}_P \tag{2.52}$$

This equation is known as Euler's equation for rotational motion. Usually, the moment of the forces and the angular momentum are taken about the center of mass. In this case, Euler's equation was developed taking into account only a single body, but it can be extended also for a system containing N rigid bodies, where the angular momentum expression $\mathbf{H}$ must be the total angular momentum, which can be composed by the momentum of the spacecraft plus the other components, such an attached fly-wheel. Also, the time derivative of the angular momentum must be taken as seen by an inertial coordinate frame. Applying the transport theorem leads to

$$\mathbf{M}_{P_{net}} = \frac{^{\mathcal{N}}d}{dt}(\mathbf{H}_P) = \frac{^{\mathcal{B}}d}{dt}(\mathbf{H}_P) + \boldsymbol{\omega} \times \mathbf{H}_P = \frac{^{\mathcal{B}}d}{dt}(I\boldsymbol{\omega}) + \boldsymbol{\omega} \times I\boldsymbol{\omega} \tag{2.53}$$

where I is the fully populated inertia matrix, in a generic reference frame. Since the body is rigid, the derivative of I is zero and the equation becomes

$$I\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times I\boldsymbol{\omega} = \mathbf{M}_{P_{net}} \tag{2.54}$$

Expanding the expression of the angular momentum we obtain a system of first order differential equations, nonlinear, and strongly coupled:

$$\begin{cases} I_{11}\dot{\omega}_1 + I_{12}\dot{\omega}_2 + I_{13}\dot{\omega}_3 - I_{12}\omega_1\omega_3 + (I_{33} - I_{22})\omega_2\omega_3 - I_{23}\omega_3^2 + I_{13}\omega_1\omega_2 + I_{23}\omega_2^2 = M_1 & (2.55) \\ I_{21}\dot{\omega}_1 + I_{22}\dot{\omega}_2 + I_{23}\dot{\omega}_3 - I_{23}\omega_1\omega_2 + (I_{11} - I_{33})\omega_1\omega_3 - I_{13}\omega_1^2 + I_{12}\omega_3\omega_2 + I_{13}\omega_3^2 = M_2 & (2.56) \\ I_{31}\dot{\omega}_1 + I_{32}\dot{\omega}_2 + I_{33}\dot{\omega}_3 + (I_{22} - I_{11})\omega_1\omega_2 - I_{13}\omega_2\omega_3 - I_{12}\omega_2^2 + I_{23}\omega_3\omega_1 + I_{12}\omega_1^2 = M_3 & (2.57) \end{cases}$$

Choosing a principal body fixed coordinate system, the inertia matrix I is diagonal and Eq. (2.54) can be expanded as

$$\begin{cases} I_1\dot{\omega}_1 + (I_3 - I_2)\omega_2\omega_3 = M_1 & (2.58) \\ I_2\dot{\omega}_2 + (I_1 - I_3)\omega_1\omega_3 = M_2 & (2.59) \\ I_3\dot{\omega}_3 + (I_2 - I_1)\omega_1\omega_2 = M_3 & (2.60) \end{cases}$$

where M_i are the body frame $\mathcal{B}$ vector components of the moment of the forces vector $\mathbf{M}_{P_{net}}$, which is also known as the vector of the total external torque $\mathbf{T}$.

2.1.7 Reaction wheels

In the previous section, the general equations for the rotational motion of a rigid body in space were presented. These equations form the basis for the modeling of the motion of satellites: anyway, they do not take into account the presence of external disturbance torques, which can alter the motion of the satellite causing it to deviate from its orbit. For this reason, the Attitude Determination and Control System (ADCS) is a fundamental subsystem of every satellite, which is used to stabilize and orient the vehicle in the presence of external disturbance torques. As it can be guessed by the name, the ADCS is encharged of determining the attitude of the vehicle with respect to a fixed inertial reference frame through a set of sensors. The number and types of sensors can vary depending on the mission requirements. If the measured orientation is different from the desired one, the ADCS has to control it through another type of component, the actuators: their task is to provide a controlled force or torque to the satellite, in order to counteract external disturbances. Some kinds of actuators are capable of producing a net torque, changing the total angular momentum of the satellite, like thrusters and magnetorquers. However, these actuators are not always available, since the thrusters require an expendable fuel source that may be exhausted after some time, while magnetorquers require an external magnetic field, such as that of the Earth. Another family of actuators is called momentum exchange devices because they can only exchange momentum with the satellite, without producing a net torque. This type of actuators is based on spinning rotors that have fixed inertia, but can change their angular momentum by varying their angular velocity. These rotors are called reaction wheels (RW) when the nominal angular velocity is zero and inertia wheel (IW) when the nominal angular velocity is different from zero. Also control moment gyroscopes (CMG) belong to this family of actuators: in this case, the rotation velocity is constant and their angular momentum is modified thanks to the principle of gyroscopes [19]. The total angular momentum of a satellite equipped with momentum exchange devices can be written as

$$\mathbf{h}_{tot} = \mathbf{h}_{sat} + A\mathbf{h}_a = I\boldsymbol{\omega} + A\mathbf{h}_a \tag{2.61}$$

where the subscript a stands for actuators. In absence of external torques the total angular momentum $\mathbf{h}_{tot}$ is constant, as expressed in Eq. (2.52), and can only be exchanged between the satellite and the actuators. Moreover, the angular momentum of the satellite can be written as $I\boldsymbol{\omega}$ in accordance with Eq. (2.38), while the angular momentum of the actuators $\mathbf{h}_a$ is written around each actuator's own spin axis. For this reason, since the actuator's axes are different from the axes of the reference frame under analysis, $\mathbf{h}_a$ is multiplied by A, which is a rotational matrix with three rows and as many columns as the number of actuators, and it is used to rotate the angular momentum from the actuator's axes to the axes of the reference frame. Assuming a total external torque $\mathbf{T}$, Eq. (2.61) can be derived to write the dynamics

$$I\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times I\boldsymbol{\omega} + \dot{I}\boldsymbol{\omega} + \dot{A}\mathbf{h}_a + A\dot{\mathbf{h}}_a + \boldsymbol{\omega} \times A\mathbf{h}_{\boldsymbol{a}} = \mathbf{T} \tag{2.62}$$

which can be simplified in the case of reaction wheels, since the terms $\dot{I}\boldsymbol{\omega}$ and $\dot{A}\mathbf{h}_a$ exist only for CMGs. Considering only reaction wheels, the dynamics becomes

$$I\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times I\boldsymbol{\omega} + A\dot{\mathbf{h}}_a + \boldsymbol{\omega} \times A\mathbf{h}_{\boldsymbol{a}} = \mathbf{T} \tag{2.63}$$

Here, the terms related to the angular momentum of the wheels can be grouped to write the control torque $\mathbf{T}_c$ as $\mathbf{T}_c = -A\dot{\mathbf{h}}_a - \boldsymbol{\omega} \times A\mathbf{h}_{\boldsymbol{a}}$. In this way, the dynamics can be written in a compact way as

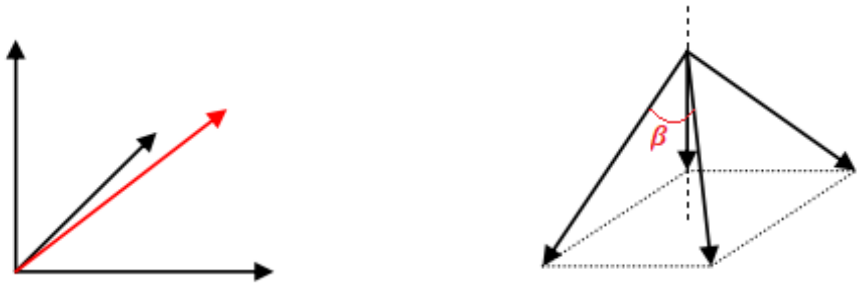
$$I\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times I\boldsymbol{\omega} = \mathbf{T}_c + \mathbf{T} \tag{2.64}$$

and the angular momentum of the reaction wheels is

$$\dot{\mathbf{h}}_a = I_a\dot{\boldsymbol{\omega}}_a \tag{2.65}$$

In general, three wheels, for example one for each axis, are enough to control the satellite. Anyway, satellites often mount four wheels for redundancy, otherwise the failure of a single wheel could cause the fail of the entire mission. There are two main configurations usually adopted in satellite's ADCS, depicted in Fig. 2.3: one is with three wheels aligned with the principal axes and the fourth with equal components along the three axes, while the other is a full pyramid configuration with no wheel aligned with any of the principal axes.

In the pyramidal configuration, the orientation of the wheels can be decided in accordance with the disturbance torques expected: for example, if the greater torque is expected in the z-axis, it is possible to select a small value for the angle β in Fig. 2.3b so that the angular momentum of the reaction wheels has a greater component in the z direction. For instance, assuming an angle of 45 degrees for the pyramidal configuration, four wheels can provide $4 \cdot \frac{\sqrt{2}}{2}$ times more angular momentum than a single wheel. In general, considering the actuators oriented with equal components along the principal axes, the A matrix is



(a) Reference frame **(b)** Pyramidal

Figure 2.3: Different configurations of the reaction wheels. [15]

$$A = \begin{bmatrix} -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{bmatrix} \tag{2.66}$$

Looking at Eq. (2.61), there is no limit in the momentum that can be exchanged between satellite and reaction wheels. However, this is not the case: in fact, the angular velocity of a wheel is usually controlled by an electric motor, which can provide a torque that is quite constant in a certain speed range (in relation to the individual operational curve of the motor). Once the velocity falls out of this range, the motor is no longer able to provide the torque, which means there is a limit in the speed and so in the angular momentum that can be stored by the wheel. Once this limit is reached, it is necessary to reduce the speed of the wheel to make it operative again: this operation is called desaturation of the wheel. The problem is that reducing the angular momentum of the wheel, will increase the one of the satellite, according to Eq. (2.61). To avoid this, an external net torque must be applied, so that the desaturation requires additional actuators like thrusters or magnetorquers. The saturation limit is an important characteristic of a wheel that must be accounted for during the design phase, in particular when the disturbance torques defined in the mission requirements have a secular component that tends to continuously increase the level of saturation of the wheel. On the other hand, in a slew maneuver, the satellite starts and ends the maneuver with zero angular velocity, and so with zero angular momentum. In this case, the angular momentum of the wheel will increase until it reaches a maximum and then return to zero (like the satellite) at the end of the maneuver. Therefore, it will only be necessary to check that the maximum value of the stored momentum does not exceed the saturation limit, but there is no limit to the number of maneuvers that can be carried out. In addition, if the maneuver is too demanding in terms of angular momentum stored by the wheel, it is always possible to subdivide the main maneuver into two (or more) smaller maneuvers.

2.2 Spacecraft attitude simulators

The greatest difficulty in simulating satellites, or in general systems conceived to work in a space environment, is related to the main difference between such an environment and usual testing facilities situated on the ground: gravity. In fact, the free-fall situation underwent by a satellite in orbit causes it to experience what is commonly called weightlessness or, more appropriately, microgravity. Since the beginning of the Space era, different ways to simulate this condition have been developed and used to test components and train astronauts. NASA used a swimming pool to train some of its astronauts, taking advantage of the neutral buoyancy condition [5]. Another way exploited by NASA uses an aircraft flying in an arc at a specific

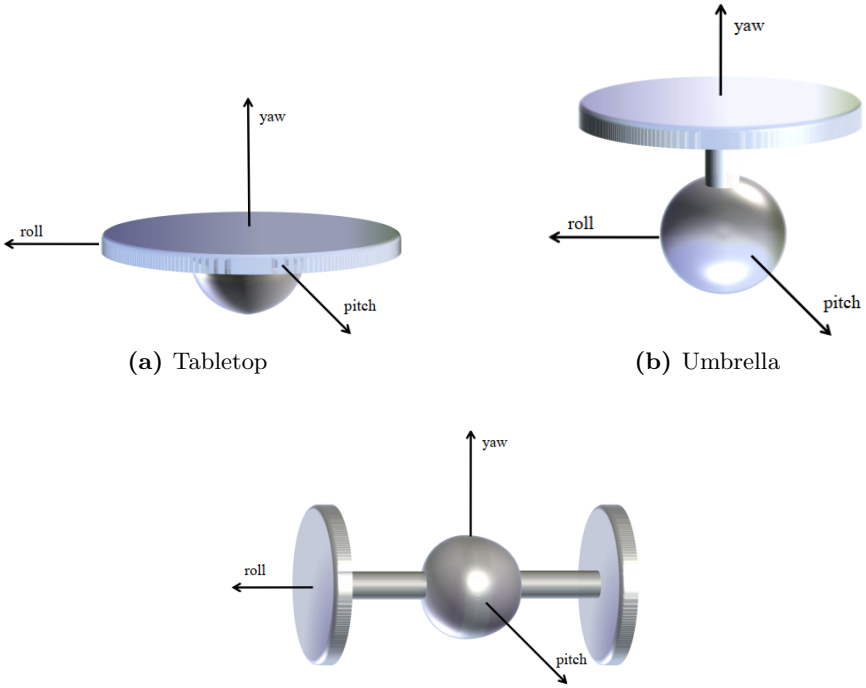
angular rate, which makes objects in the aircraft float for a small amount of time [6]. Other ways are magnetic suspension [7], which uses magnetic forces to make the component levitate, and the drop from a tower [8]. A good description of the pros and cons of each method is presented by Boynton in [9], where is also stated that none of those are suitable for testing satellites. On the other hand, a method of simulating weightlessness that can be used for satellite testing, consists in floating the satellite on an air bearing. Such a bearing offers a nearly torque-free environment, and for this reason, it is the preferred technology for ground-based research in spacecraft attitude dynamics and control.

2.2.1 Air bearings

Here, a brief introduction to air bearings will be presented. First of all, different types of air bearings exist, and depending on the type, some combination of torque-free rotational motion and force-free translational motion can be achieved. Anyway, almost all air bearings are composed of a fixed part, the stator, which has some orifices through which flows pressurized air, and a moving part, the rotor, which is supported by the very thin film of air established between rotor and stator. So, the air film acts as the lubricant and imparts nearly no shear between the two sections: in this way, the response of the moving part is characterized solely by its mass and moment of inertia. The satellite, or every other component that needs to be tested, is placed over the rotor to exploit this feature, as long as the center of gravity of the moving part is coincident with the center of rotation of the bearing. However, an air bearing allowing full range of motion in six degrees of freedom (DOFs) does not exist. Concerning DOFs, air bearings can be divided into two main types:

- **flat air bearings**: the rotor and stator are flat surfaces, so they can allow free motion in a horizontal plane (two DOFs) plus a rotation along the vertical axis (one DOF) for a total of three DOFs;
- **spherical air bearings**: in this case rotor and stator are portions of concentric spheres, allowing the complete rotation around the vertical axis plus limited rotation along the other two horizontal axes, for a total of three DOFs, simulating all the rotational modes.

These single bearings can be assembled together to simulate motion in five DOFs, while the sixth one, which is vertical translation, is more difficult to add. For example, the system can be attached to a counterweighted beam or can be suspended from a long spring. Obviously, both these methods introduce some drawbacks that make the simulation of all six DOFs quite tricky. The most common type of bearing used in spacecraft attitude dynamics is the spherical one because, ideally, they provide unconstrained rotational motion. As already stated, that is not completely true since rotation around horizontal axes is usually constrained to angles smaller than 90 degrees. That is the case of tabletop and umbrella-style platforms, depicted in Fig. 2.4a and 2.4b. The tabletop consists of a flat plate mounted directly on the rotor, so that it is the easiest one, but also the most constrained in motion. The umbrella-style adds a beam between the flat plate and the hemispherical part of the rotor: in this way, the stator can be extended, allowing a wider range of motion in roll and pitch. The last possible style is the dumbbell configuration, shown in Fig. 2.4c, which is composed of two beams. This configuration greatly reduces structural interference within the rotation space of the payload and thereby provides unconstrained motion in both the roll and yaw axes [10].
There are three main factors that characterize a spherical air bearing: the weight capacity, the height of the rotational center, and the maximum tilt angle. These parameters define the quality of an air bearing and must be taken into account when choosing an air bearing.



(a) Tabletop **(b)** Umbrella

(c) Dumbbell

Figure 2.4: Possible configurations of air bearings n.

Weight capacity

This parameter refers to the maximum load that the bearing can withstand. As already described, the rotor of the bearing is suspended on a thin film of air coming from orifices in the stator. The film of air works like a spring: by placing a weight on top of the bearing, the film will become thinner. If the weight capacity is exceeded, the air pressure will no longer be able to support the rotor, which will come into contact with the stator. The value of the weight capacity depends on the dimensions of the bearing, in particular the surface area, and on the maximum air pressure. In theory, every load could be counterbalanced by very high pressure, but, in reality, this causes the bearing to be unstable.

Height of rotational center

In order for a spherical air bearing to simulate weightlessness, there must be no restoring forces applied to the system, so that the moving elements remain in any position when brought to rest. This means that its center of rotation must be coincident with the center of gravity of the moving part of the bearing, which includes the rotor plus everything on top of it. So, putting a large satellite over the bearing will cause the center of gravity to move towards up: the greater the equipment to be tested on the bearing, the higher is the center of gravity, the higher should be the rotational center. For this reason, a high rotational center is preferable. This can be achieved with a large radius of the spherical bearing, However, increasing the radius will reduce the depth of the cup, limiting the resistance to side forces. In fact, the side load capability of a bearing is related to the cross sectional area of the bearing in a vertical plane. So there is always a trade off between height of the rotational center and side load capability.

Maximum tilt angle

As already seen, all the spherical air bearings have at least one rotation which is limited. The maximum tilt angle represents the maximum angle that can be achieved by constrained rotation. Exceeding this limit makes the airflow unstable and prevents the rotor from returning to its original position. This parameter is influenced by the dimensions of the stator 'cage' that encapsulate the rotor, and so it depends again on the radius. A small radius makes it possible to have a stator that is quite a complete sphere, allowing a larger maximum tilt angle, but decreasing the height of the rotational center. From this consideration, is easy to understand that the tilt angle is related to the side load capability, sharing also the same relation with the height of the center of rotation.

2.2.2 Balancing methods for air bearings

As already said, an air bearing is capable of replicating the conditions of a satellite in orbit, in particular the cancellation of the gravity effects and the virtual absence of frictional resisting torques, obtained thanks to the air film of the air bearing. However, these effects cannot be perfectly canceled. Since the simulator is composed of real physical components, a lot of different sources of disturbance exist, so that those effects can only be minimized as much as possible. So, the goal of the simulator should not be that of completely eliminating every external torque, but should be that of minimizing them to such a low level that they can be neglected in the description of the system. The level to be achieved depends on the application: for example, [11] states that a nanosatellite in Low Earth Orbit can be subjected to an external torque of about 10^{-6} Nm. Taking this value as reference, a great effort should be put in the minimization of the distance between the center of rotation and the center of gravity, as also in the minimization of the other disturbances.
A survey on all possible disturbance torques acting on an air bearing based simulator has been carried on by [12]. In the document, the disturbances are collected into four different groups:

- **Torques from the platform.** These torques depend directly upon the construction of the air bearing platform. This group contains the unbalance caused by gravity effects. They can be minimized by building a very stiff platform and moving the center of gravity in coincidence of the rotational center.
- **Torques from the bearing.** These disturbances arise from the bearing itself, in particular from the airflow channels, if they are not perfectly symmetric or clean. They can be minimized in the manufacturing of the bearing itself.
- **Torques from the environment.** This group contains torques external to the platform itself, so that they are more difficult to be controlled. The most important ones come from the interaction between the external air and the platform, which can act as a source of damping.
- **Torques from the test system.** The torques of the last group depend on the particular components utilized for the simulation and their configuration. One example could be mass unbalance torques arising from the discharge of batteries.

A complete list of the disturbing torques described by [12] is given in Table 2.1.

As it can be seen, the first five disturbance torques from the first group are related to the offset between the rotational center of the air bearing and the center of gravity of the overall rotating part of the platform, that will be represented by $\mathbf{r}_{\text{off}}$. Minimizing this distance is essential for the success of the simulation, since in this way many disturbing components are eliminated. Assuming that the system is subjected only to the gravity torque due to the offset $\mathbf{r}_{\text{off}}$, the

Table 2.1: Disturbing torques on an air bearing platform.

platform	Torques from the bearing	environment	test system
Static unbalance	Aerodynamic turbine effect	Air damping	Electrical wires to base
Dynamic unbalance	Exhaust air impingement	Air currents	Mass shifting in bearings and loose fits
Anisoelasticity		Magnetic fields	Battery discharge
Material instability		Vibration	Reaction jet supply discharge
Stress-Temperature Humidity-Evaporation		Radiation pressure	Replacement of components
Gravity gradient			
Equipment motion			
Solenoids - Relays			

rotational dynamics of the simulator with respect to the center of rotation is given by Euler's equation [13]:

$$J\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times J\boldsymbol{\omega} = \mathbf{r}_{\text{off}} \times m_{s/c}\mathbf{g}_b + \boldsymbol{\tau}_c \tag{2.67}$$

where J is the simulator inertia matrix, $\boldsymbol{\omega}$ is the simulator angular velocity, $m_{s/c}$ is the simulator mass and $\boldsymbol{g}_b$ is the gravitational acceleration in the simulator coordinate system. Moreover, $\boldsymbol{\tau}_c$ can be present or not, and it indicates the control torque. If a constant torque input cannot be provided, then the only way to eliminate the disturbance due to the gravity is to minimize $\mathbf{r}_{\text{off}}$, which means that the center of gravity becomes coincident with the rotational center.
The first way to reduce the offset is to manually balance. First of all, all the components on the platform should be arranged as symmetrically as possible, or in such a way as to balance the offset in the horizontal plane. After that, the center of gravity can also be balanced in the vertical direction. This method is the simplest one, but can be time consuming and the accuracy reached is often not sufficient. Anyway, it can always be used as a first step to reduce the offset.
The second way is the automatic balancing. This method involves the actuation of shifting masses, mounted along the three orthogonal axes of the simulator reference system. Moving the masses along these axes will change the position of the center of gravity of the platform, while the rotational center remain fixed, thus reducing the offset $\mathbf{r}_{\text{off}}$. To do so, a control law should be designed in order to actuate the masses with the right gain, according also to the offset. So, automatic balancing involves also the estimation of the offset $\mathbf{r}_{\text{off}}$ and of the system inertia. If actuators are available, the offset can be estimated by measuring the response of the system to known input torques. Instead, when actuators are not available, several solution have been proposed. A good and updated literature review of the solutions adopted both for the estimation and the balancing is given in [11].

2.3 Available components

In this section, the components already available in the laboratory will be presented. The laboratory is located inside the buildings of Politecnico di Milano and the hardware described later on is part of the EXTREMA Simulation Hub that is developing as a testing and validation facility in the context of the EXTREMA ERC project, already introduced in chapter 1.

2.3.1 Air bearing

The main component used to simulate weightlessness is a tabletop style spherical air bearing from Specialty Components. The model is the SRA250-R30 [1] and is depicted in Fig. 2.5. A good description of the bearing, containing also a description of the auxiliary systems necessary to operate the air bearing, is given in a previous thesis work by Luca Mariani [20].

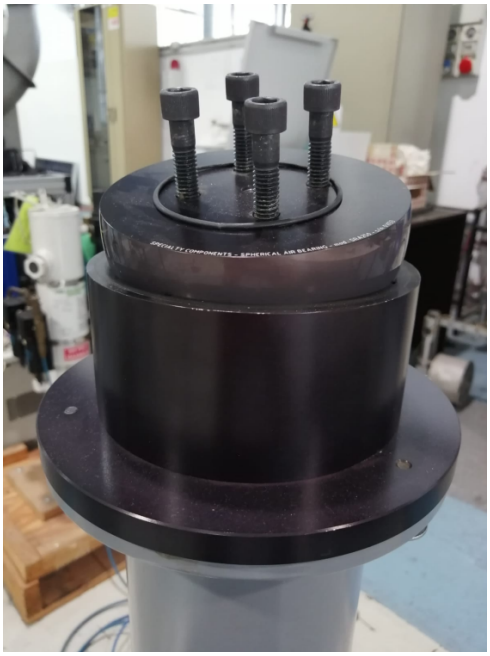


Figure 2.5: SRA250-R30 spherical air bearing

The main characteristics are summarized in Table 2.2: being a tabletop style air bearing, the rotation of the joint is unlimited only in one direction, which is the one orthogonal to the ground. In the other two directions, the rotation is possible only in the range [-30, 30] degrees. The load capacity is high enough not to be a restricting parameter for the design of any kind of actuator, since it is in the order of hundreds of kg.

Moreover, an indication about the aerodynamic drag coefficient can be obtained from the drag torque at 30 rpm: however, this parameter could depend largely on the degree of wearing of the internal surfaces of the bearing and does not take into account the aerodynamic moment due to the interaction of the above platform with the air. Similarly, the value of the CR-CG offset is related only to the bearing, and could change drastically according to the other components mounted on the bearing itself. In this regard, other projects related to the simulator were also underway during the development of this thesis, including the implementation of moving masses for platform balancing and the study of algorithms for automatic balancing by the motion of the masses themselves. Fig. 2.6 shows the CAD of the platform and all the other components

[1] Specialty Components, *Spherical Air Bearings SRA250-R30*, `https://www.specialtycomponents.com/Products/sra250-r30/`

Table 2.2: SRA250-R30 characteristics

Rotor mass	1.81 kg
Load capacity	230 kg
Maximum tilt angle	30 deg
CR-CG offset	17.56 mm
Drag torque at 30 rpm	25 μ Nm

mounted on it: the image shows a final configuration, developed in parallel with this work. As it can be seen, the bearing is equipped with two plates, one at the top and the other more below, in order to lower the center of gravity. The plates represent the main structure to which are attached the other components, like the moving masses, the batteries, and the other electrical components, which are not accounted for in the CAD model. Of course, the presence of a large number of components affects the total inertia of the platform, increasing it.

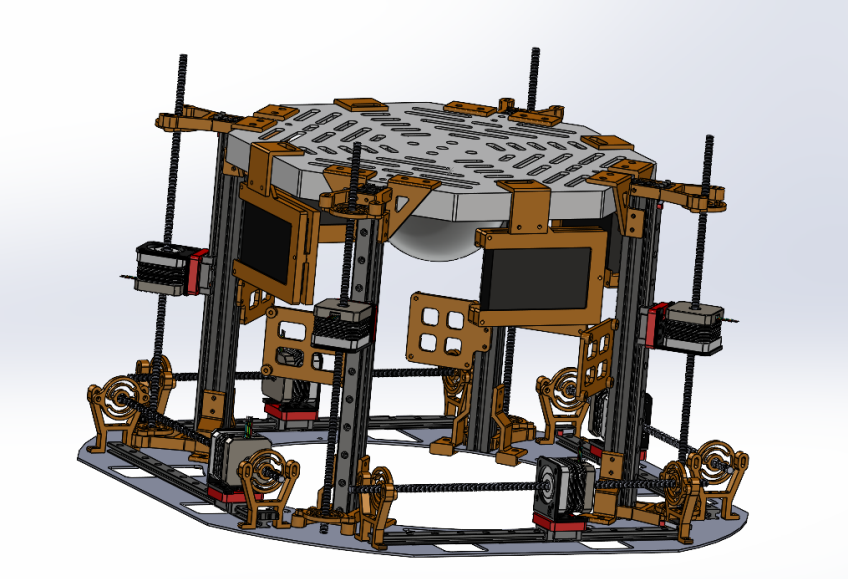


Figure 2.6: CAD model of the overall platform [21]

2.3.2 IMU sensor

Another component already equipped on the platform was an inertial measurement unit (IMU) sensor by Bosch: the BNO055 Intelligent 9-Axis Absolute Sensor [2]. The unit integrates a triaxial 14-bit accelerometer, a triaxial 16-bit gyroscope, and a triaxial geomagnetic sensor, all in a very limited space, as depicted in Fig. 2.7. In addition, Table 2.3 summarizes the principal characteristics of the sensor.

The gyroscope and the accelerometer have integrated digital low-pass filters, with a bandwidth that can be selected by the user. The resolution of the gyroscope and the accelerometer depends on the selected range, while the magnetometer has a single range and resolution. Furthermore, several power modes are available, like power-saving, standby, or high accuracy modes. The outputs fused sensor data consist of quaternion, Euler angles, rotation vector, linear acceleration,

[2]Bosch, *BNO055 Intelligent 9-Axis Absolute Sensor*, Datasheet, `https://cdn-shop.adafruit.com/datasheets/BST_BNO055_DS000_12.pdf`

Table 2.3: Bosch BNO055 characteristics

	Gyroscope	**Accelerometer**	**Magnetometer**
Range	$\pm$ 125 to $\pm$ 2000 $\frac{deg}{s}$	$\pm$ 2 to $\pm$ 16 g	$\pm$ 2500 μ T (z-axis)
Resolution	16-bit	14-bit	0.3 μT
Low-pass filter bandwidths	523 Hz - 12 Hz	1 kHz - 8 Hz	-

gravity, and heading. The outputs can be obtained at different data rates, with a maximum frequency of 100 Hz. As a last remark, for optimum system integration, the BNO055 is equipped with digital bidirectional I^2C and UART interfaces.

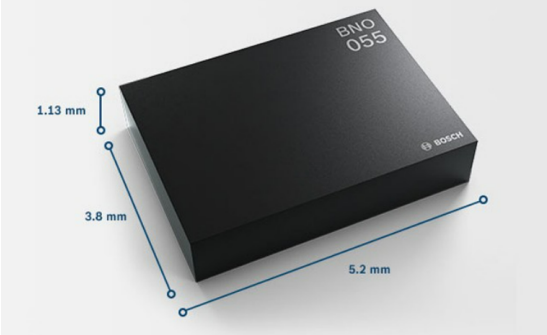


Figure 2.7: Bosch BNO055 IMU sensor

Chapter 3

Testbed model and numerical simulation

This chapter will focus on the development of a mathematical model for the overall system composed by the air-bearing simulator plus the reaction wheels linked to it, which shall then be used to simulate the behavior of the system itself. Numerical modeling and simulation represent usually the first step in the design process because it can give useful insights on the phenomenon under study, that lead to a better understanding of all the parameters involved, reducing the development time and costs. Moreover, once that all the components will be implemented in the laboratory, the predicted response provided by the simulation can be used to validate the actual response of the air-bearing testbed during testing scenarios.

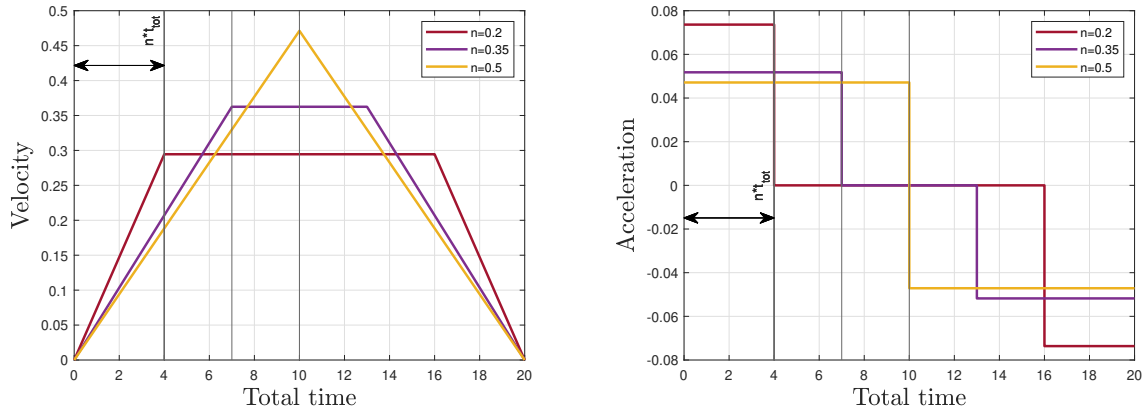
The chapter will be divided as follows: in the first section, starting from a simplified version of the coupled dynamics of the simulator and of the reaction wheels, the principal characteristics of the latters will be derived, which will lead to the sizing of a single wheel.
In the second section, a refined model of the overall system, developed in the Matlab Simulink environment, will be presented and used both to check the correctness of the reaction wheels sizing and to guide the choice of proper values for the design parameters.

3.1 Reaction wheels sizing

In this section, the process that led to the sizing of the reaction wheels will be presented. The goal is to define the main properties of the reaction wheel, that will be mandatory to build the Simulink model, like its rotational inertia. The basic idea developed in this section is described in [22] and relies on a set of assumptions on the dynamics of the wheel. In addition, the referenced document refers to a single reaction wheel whose axis of rotation coincides with the axis of rotation of the vehicle.
In the beginning, all the parameters affecting the design will be described, along with the major assumptions. Then, the integration of the equations of motion will give a way of analyzing the torque and angular momentum needed by the simulator to complete the maneuver under study. From these, the features of the reaction wheel will be derived thanks to the conservation of angular momentum.
The slew maneuver taken into account, which is a typical maneuver for the mission envisioned in the EXTREMA project, refers to a rotation of the spacecraft to take orthogonal measurements in space. In the worst case, when the Sun is between the initial and final position, the rotation could be 270 degrees to prevent the sensor from being burned by the Sun. Also, the temporal limit selected for the maneuver is 20 seconds.

3.1.1 Torque and maximum angular momentum

As already stated, this simplified analysis will make use of different assumptions: the first and most important one is related to the velocity profile of the wheel. The main objective is that of realizing a slew maneuver, which means that the reaction wheel starts rotating from a null velocity, as the simulator, and continue until the simulator has reached the final desired angle θ_{tot} in the total time t_{tot}, with a final velocity equal to zero for both the wheel and the simulator. For this reason, the velocity profile of the wheel will be characterized by an acceleration phase at the beginning and a deceleration phase just before the end of the maneuver. The main assumption consists of assuming that the acceleration time is equal to the deceleration time and is indicated by the parameter n, which indicates the fraction of the total time spent in the acceleration (or deceleration) phase. So, $n\,t_{tot}$ represents the acceleration (or deceleration) time and the remaining time, $(1-2n)\,t_{tot}$, is spent at constant velocity. From these considerations, it is obvious that the parameter n can vary between the values of 0 and 0.5, where $n = 0$ corresponds to a maneuver with an instantaneous acceleration phase, then a constant velocity for t_{tot}, and again an instantaneous deceleration phase, while $n = 0.5$ corresponds to a maneuver with an acceleration phase for half of the total time and a deceleration phase for the other half, without the constant velocity phase.
In addition to the assumption about the acceleration and deceleration times, another assumption is related to the torque acting on the wheel, which is constant. This means that the angular acceleration $\ddot{\theta}$ (and deceleration) will be constant and that the angular velocity $\dot{\theta}$ will vary linearly with time, while the displacement θ will be a quadratic function of time.



(a) Velocity **(b)** Acceleration

Figure 3.1: Velocity and acceleration for three different values of n.

Fig. 3.1 displays on the left the velocity profile and on the right the acceleration profile, according to the aforementioned assumptions, for three different values of n, respectively 0.2, 0.35, and 0.5. Looking at the acceleration profile, we notice that when n increases, the constant value of the acceleration decreases, but the time required for the acceleration increases. The consequences are that when n is greater, the larger time for the acceleration allows the velocity to reach a larger maximum value than the case with smaller n, where the velocity will arrive at a lower maximum in less time, anyway with a greater slope. For this reason, the product of the average velocity and total time is the same for all the cases, i.e., the areas under the three velocity-time curves are equal. This is another way of stating that the impulses are equal. [22]

With these assumptions, it is possible to write the equations of motion for both the simulator and the reaction wheel during the three different phases: acceleration phase, constant velocity phase and deceleration phase.

Acceleration phase

For this phase, the equations expressing the dynamics are:

Simulator:	Wheel:
$I_s\ddot{\theta}_s = T_0$	$I_w\ddot{\theta}_w = -T_0$

with null initial conditions

$$\begin{cases} \dot{\theta}_s(t=0) = 0 & (3.1) \\ \theta_s(t=0) = 0 & (3.2) \end{cases}$$

where T_0 is the constant torque, I represents the inertia and the subscripts s and w define respectively variables related to the 'simulator' or to the 'wheel'.

Integrating the equations with the initial conditions leads to the maximum rotational velocity reached during the acceleration phase:

$$\dot{\theta}_{s,a} = \frac{T_0}{I_s}\, n\, t_{tot} \tag{3.3}$$

and to the total displacement during the acceleration phase:

$$\Delta\theta_s = \theta_s(t=t_a) = \frac{1}{2}\frac{T_0}{I_s}\,(n\, t_{tot})^2 \tag{3.4}$$

Constant velocity phase

In this case, the equations expressing the dynamics are:

Simulator:	Wheel:
$I_s\ddot{\theta}_s = 0$	$I_w\ddot{\theta}_w = 0$

where the subscript c stands for 'constant velocity'.

Again, it is possible to integrate the equations, but this time the initial conditions are:

$$\begin{cases} \dot{\theta}_s(t=t_a) = \dot{\theta}_{s,a} & (3.5) \\ \Delta\theta_s^{'}(t=t_a) = 0 & (3.6) \end{cases}$$

In fact, the velocity is the final velocity of the previous phase, while the initial condition expressed in Eq. (3.6) is referred to the relative displacement during this phase, and not the total displacement, such that $\Delta\theta_s^{'}(t) = \theta_s(t) - \Delta\theta_s$.

Then, remembering the assumptions at the beginning of this section, which can be expressed in a mathematical form as

$$\begin{cases} t_a = t_d = n\, t_{tot} & (3.7) \\ t_c = (1-2n)t_{tot} & (3.8) \end{cases}$$

and integrating the equations of motion, the total relative displacement during this phase is evaluated as:

$$\Delta\theta'_{s,max} = \Delta\theta'_s(t = t_a + t_c) + \dot{\theta}_{s,a}\,(1 - 2n)\,t_{tot} \tag{3.9}$$

Deceleration phase

Finally, in the deceleration phase the equations become:

Simulator:	Wheel:
$I_s\ddot{\theta}_s = -T_0$	$I_w\ddot{\theta}_w = T_0$

Integrating with the initial conditions

$$\begin{cases} \dot{\theta}_s(t = t_a + t_c) = \dot{\theta}_{s,a} & (3.10) \\ \Delta\theta''_s(t = t_a + t_c) = 0 & (3.11) \end{cases}$$

and substituting Eq. (3.3) leads to:

$$\Delta\theta''_{s,max} = \Delta\theta_s \tag{3.12}$$

Hence, the relative displacement in the acceleration and deceleration phase is the same.

At this point, it is possible to calculate the overall displacement throughout all the maneuver as $\theta_{s,tot} = \Delta\theta_s + \Delta\theta'_{s,max} + \Delta\theta''_{s,max}$. Substituting Eq. (3.12), Eq. (3.9), and Eq. (3.4) leads to:

$$T_0 = \frac{\theta_{s,tot}\, I_s}{t_{tot}^2\,(n - n^2)} \tag{3.13}$$

that is the constant torque that must be provided to the wheel in order to achieve a rotation of the angle $\theta_{s,tot}$ in the time t_{tot} with the parameter n, for a simulator with total inertia around the rotational axis I_s. These parameters represent the input data, that are defined by the maneuver under study.

Furthermore, it is possible to calculate the maximum angular momentum of the simulator during the slew maneuver as:

$$h_{max} = I_s\,\dot{\theta}_{s,a,max} = T_0\,n\,t_{tot} \tag{3.14}$$

While T_0 is a constant value, the angular momentum of the vehicle h changes with the angular velocity and Eq. (3.14) gives only the maximum value, in correspondence of the maximum velocity.

Considering Eq. (3.13) and Eq. (3.14), it is useful to study the behavior of T_0 and h_{max} in relation to n. Indeed, the other inputs data are known once selected a certain maneuver ($\theta_{s,tot}$, t_{tot}) and the simulator inertia I_s, while n is a parameter that can be adjusted from time to time.

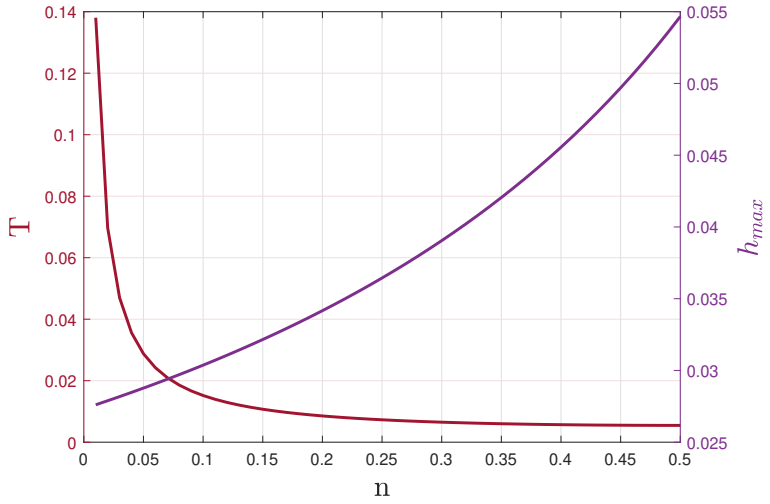


Figure 3.2: Torque T and maximum angular momentum h_{max} against n.

Fig. 3.2 shows the behaviour of the torque and the maximum value of the angular momentum against n. As it can be seen, the torque decreases with n: this is reasonable, since in the limit case of $n = 0$ the acceleration phase is instantaneous and so the torque should be infinite, while with a large value of n the acceleration time is greater and allows to have a smaller torque. On the other hand, the maximum angular momentum increases with n because of the angular velocity. In fact, the angular momentum is related to the inertia and to the maximum value of the velocity: the inertia is constant, while the maximum velocity reached by the simulator increases with n, as shown in Fig. 3.1, and so does h_{max}.

3.1.2 Considerations on inertia and power consumption of the wheel

Up to now, the main focus was put on the simulator. Exploiting the conservation of angular momentum, as stated in Eq. (2.61), the focus will switch on the reaction wheel.

$$I_s \, \dot{\theta}_s + I_w \, \dot{\theta}_w = 0 \tag{3.15}$$

In this case, the matrix A in Eq. (2.61) becomes a scalar with value 1, because we are considering a one dimensional problem (wheel and simulator acting on the same axis). According to this equation, the maximum angular momentum h_{max} calculated for the simulator must be provided by the reaction wheel, so

$$h_{max} = I_w \, \dot{\theta}_{w,max} \tag{3.16}$$

At the same time, the mechanical power needed to rotate the wheel can be estimated as [22]:

$$P = T_0 \, \dot{\theta}_{w,max} \tag{3.17}$$

Since T_0 and h_{max} are already known, three unknowns remain from the two equations 3.16 and 3.17. In order to have an estimate of the power consumption and inertia of the wheel, the easiest way is that of selecting a fixed value for the maximum angular velocity of the wheel. With this value, it is possible to study the behavior of the inertia and of the power consumption in relation to n, as shown in Fig. 3.3.

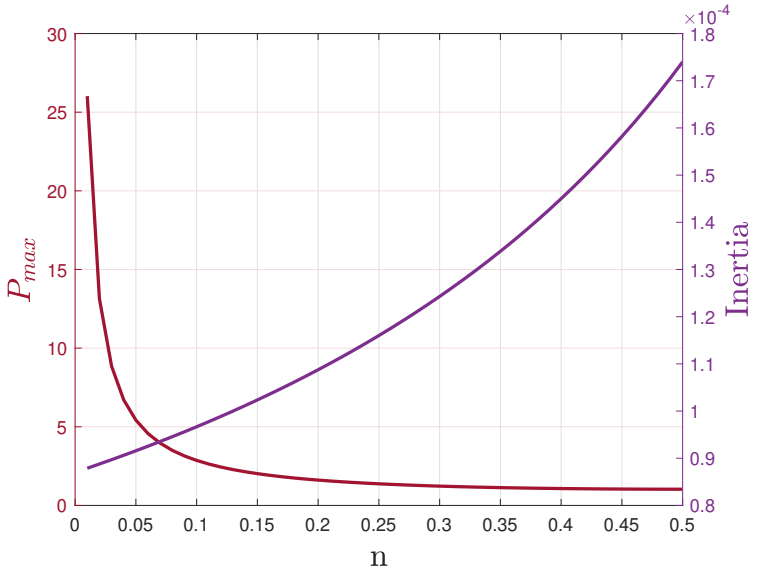


Figure 3.3: Max power P_{max} and wheel inertia I_w against n.

Comparing Fig. 3.3 and Fig. 3.2 it can be immediately noticed a considerable similarity. This also emerges from the equations because, once the wheel velocity $\dot{\theta}_{w,max}$ is fixed, the inertia becomes proportional to the maximum angular momentum and the power becomes proportional to the torque. So, this analysis comes out to be useful for the inertia, since from the maximum angular momentum and the maximum rotational velocity one can obtain the minimum value for the inertia of the wheel to select in order to not overcome the maximum rotational velocity of the wheel itself. However, that is not the case for the power consumption. In fact, Eq. (3.16) can be substituted inside Eq. (3.17) to express the power as a function of both the torque and the angular momentum:

$$P = T_0 \frac{h_{max}}{I_w} \tag{3.18}$$

Substituting Eq. 3.13 and 3.14, it is possible to show the relation between P and n

$$P = \frac{(\theta_{s,tot}\, I_s)^2}{I_w\, t_{tot}^3\, n\,(1-n)^2} \tag{3.19}$$

When studying the power in relation to n it is not possible to fix a value for the wheel velocity (as done previously), because this would mean that both h_{max} and I_w would change with n to maintain the same value for the velocity, while in reality the inertia of the wheel will be fixed and so the evolution of the power with n will be determined by the two contrasting behaviors of T_0 and h_{max}. For this reason, a fixed value of the wheel inertia I_w was selected (the one with $n = 0.5$) and the results are shown in Fig. 3.4.

In this case, the power is not continuously decreasing with n, but there is a global minimum. In fact, for high values of n, T_0 will be small but h_{max} will become too large and vice-versa for small values of n. The aftermath is that there is a minimum value for the power consumption of the wheel, which can be calculated from Eq. (3.19). Analytically, the minimum is found for a value of $n = \frac{1}{3}$, as can be also seen in Fig. 3.4.

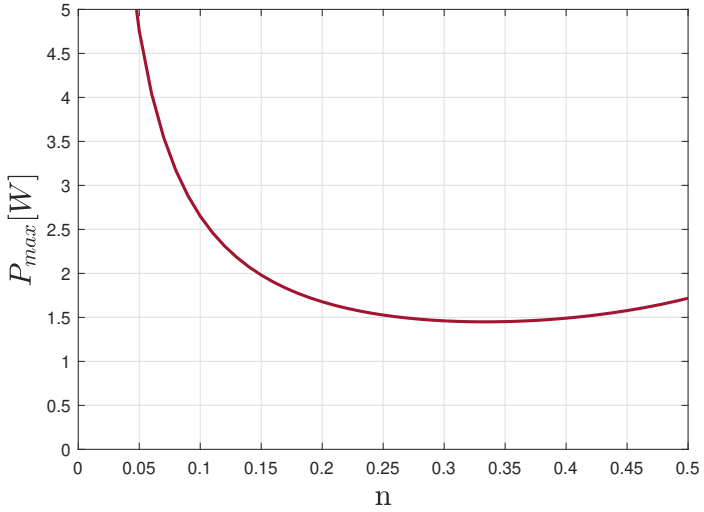


Figure 3.4: Power consumption of the wheel against n, for a fixed value of the wheel inertia.

3.1.3 Sizing process for the reaction wheel

At this point, after the analysis of the principal characteristics affecting the system, it is clearer the role and the importance of every one of them, in particular of the torque and angular momentum that must be provided by the wheel. Therefore, the primary focus of the sizing process is that of choosing the inertia for the wheel and the power consumption, which is related to the motion of the wheel and so to how the wheel achieves its main goal of rotation, that is expressed by the parameter n.
Concerning the inertia, it has already been discussed the relation between this parameter and the maximum angular momentum that the wheel can provide. So, the choice shall take into account that once the inertia will be fixed, also the maximum rotation angle achievable by the wheel with a single maneuver will be fixed. On the other hand, the power consumption depends on n, which can be chosen to minimize the power for every maneuver, since n is not related to the physical configuration of the system but to the on-board software that will govern the motion of the wheel.

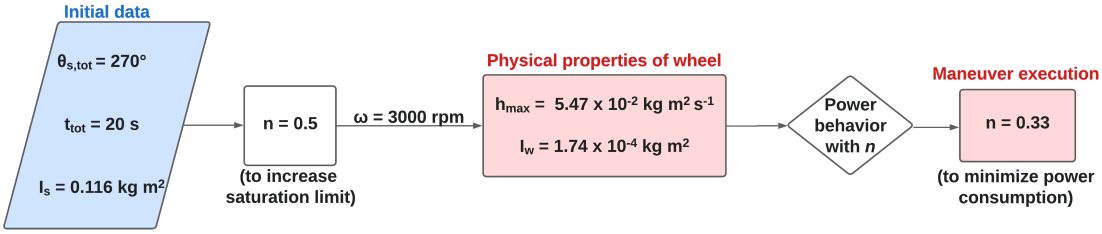
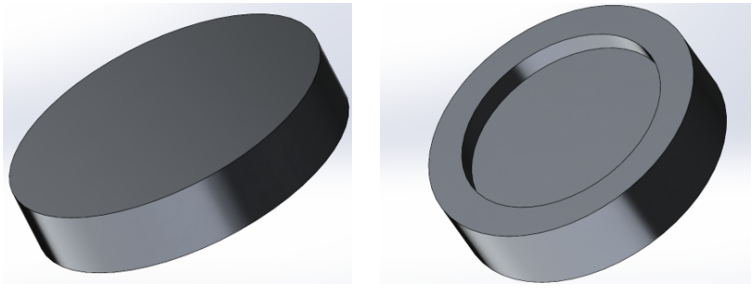


Figure 3.5: Block diagram of the decision process for the sizing of the wheel.

The inertia is chosen from Eq. (3.16), where the maximum allowable velocity for the wheel is set to 3000 rpm and h_{max} depends on n. Since it was found a minimum power consumption for $n = 0.33$, one could think to select this value to calculate h_{max}, but doing so there will be a lack of flexibility for the system because then it would not be possible to perform any other maneuver with values of n greater than 0.33, since the wheel would reach its saturation limit. In fact, with a greater value, the maximum angular momentum will increase and the wheel would be forced

to rotate at a velocity larger than 3000 rpm. For this reason, the value of the inertia is chosen calculating h_{max} with $n = 0.5$. In this way, with every other value of n, the maximum angular momentum of the wheel will be smaller and so attainable. Moreover, this choice is referred to a single wheel rotating along the axis of the maneuver, while in the final system there will be 3 or 4 wheels, so that the total angular momentum to be provided to the simulator will be split between all the wheels, with a coefficient related to the orientation.
Once that the inertia is selected, the typical maneuver will be performed with a velocity profile characterized by $n = 0.33$ to reduce the power consumption. Anyway, the parameter n remains flexible so that can be modified at any time, even after the laboratory implementation. The overall decision process is summed up in Fig. 3.5, where the red blocks represent the final chosen values.



(a) Cylinder wheel **(b)** Hollow cylinder wheel

Figure 3.6: Different geometries for the reaction wheel.

Now that the inertia of the wheel is selected, it is possible to sketch a first prototype of the wheel. Usually, reaction wheels are simply made as solid discs like in Fig. 3.6a. Considering aluminum as material, the rotational inertia needed is attained with a radius of 40 mm, a height of 16.2 mm, and a total weight of 220 g. However, a more refined geometry can lead to a weight reduction, as in Fig. 3.6b. Here, the material is concentrated away from the center of rotation, so that the same rotational inertia is obtained with less material. Indeed, the new geometry has the same radius of 40 mm, but a height of 20 mm in the external cylinder and of only 8 mm in the central part, leading to a total weight of 180 g, which is 18 % less than the previous case.

3.2 Simulink model

Now, it is possible to define a more accurate model of the overall system composed by the simulator and the reaction wheels. These two coupled components will be referred to as the 'System'. In addition, other components such as sensors and a control algorithm should be added. As a starting point, the complete model should resemble the structure of a classical feedback control system, as shown in Fig. 3.7.

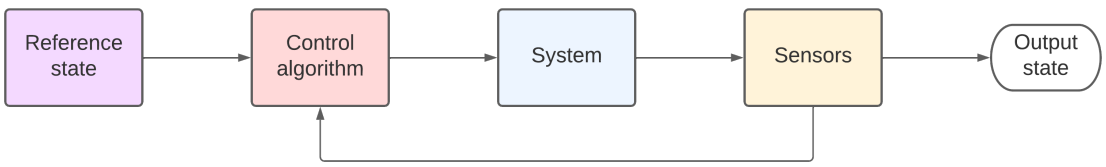


Figure 3.7: Classical feedback control system.

To do so, the Matlab Simulink environment was used because of its straightforward way of implementing many equations grouped in different structures called 'blocks', which are easy to

recognize and manipulate. The block structure allows also to easily turn on and off certain parts, like sensor measurements. In order to guarantee maximum fidelity with the real simulator, both continuous blocks and discrete time blocks have been used: in fact, while the simulator and the wheels would respond in a continuous way (to simulate the real system), the sensors and the control system would act only at some defined time steps, because even in real life the sensors run at a finite frequency. Using a continuous model of the system dynamics and a discrete model of the sensors and control algorithm allows the simulation to provide the best estimate of the output state. Obviously, the discretization is necessary to create data arrays within the Matlab environment, even in the case of the continuous model of the system, but it should occur at a much higher frequency than the discrete control algorithm to accurately represent the continuous information.
In Fig. 3.8 is represented the final complete model of the overall system in Simulink. Since this is the most outer layer of the model, it lacks of all the little details, but it is easy to recognize the classical structure of a feedback control system.

3.2.1 System module

As it can be seen in Fig. 3.8, all the blocks of the model are grouped into different 'modules'. The first one to be described will be the 'System module', which represents the physical system and is composed by the reaction wheels block, the simulator dynamics block and the simulator kinematics block. The module takes as input only the voltages that power up the wheels motors and gives as outputs the state of the wheels (which is composed by the velocities and the currents), the angular velocities and Euler angles of the simulator. This module is the only one simulated in a continuous way.

Reaction wheels block

This block represents the model of the reaction wheels and the motors attached to each wheel. There are several types of motors that can be used for this type of application. In our case, it was chosen to use a BrushLess Direct Current (BLDC) motor because of the advantages it offers over brushed DC motors and induction motors, which includes better speed versus torque characteristics, higher efficiency, higher dynamic response, and higher torque delivered to motor size [23]. Furthermore, the absence of brushes means that less maintenance is required and that the reversal of polarity is performed electronically by semiconductor switches, which are operated accordingly with rotor position, obtained with measurement devices such as Hall effect sensors. A complete mathematical model of a three-phase BLDC motor is presented in [24], based on the physical model of Fig. 3.9. This model is taken as reference and rewritten in a state-space representation in order to develop an 'RW block' in Simulink, which simulates a single motor plus reaction wheel. Then, the 'Reaction wheels block' is simply composed of three identical 'RW blocks'.
Applying Kirchhoff's voltage law to the electrical circuit of the motor it is possible to write, for the three phases

$$V_a = R\, i_a + L\, \frac{di_a}{dt} + e_a \tag{3.20}$$

$$V_b = R\, i_b + L\, \frac{di_b}{dt} + e_b \tag{3.21}$$

$$V_c = R\, i_c + L\, \frac{di_c}{dt} + e_c \tag{3.22}$$

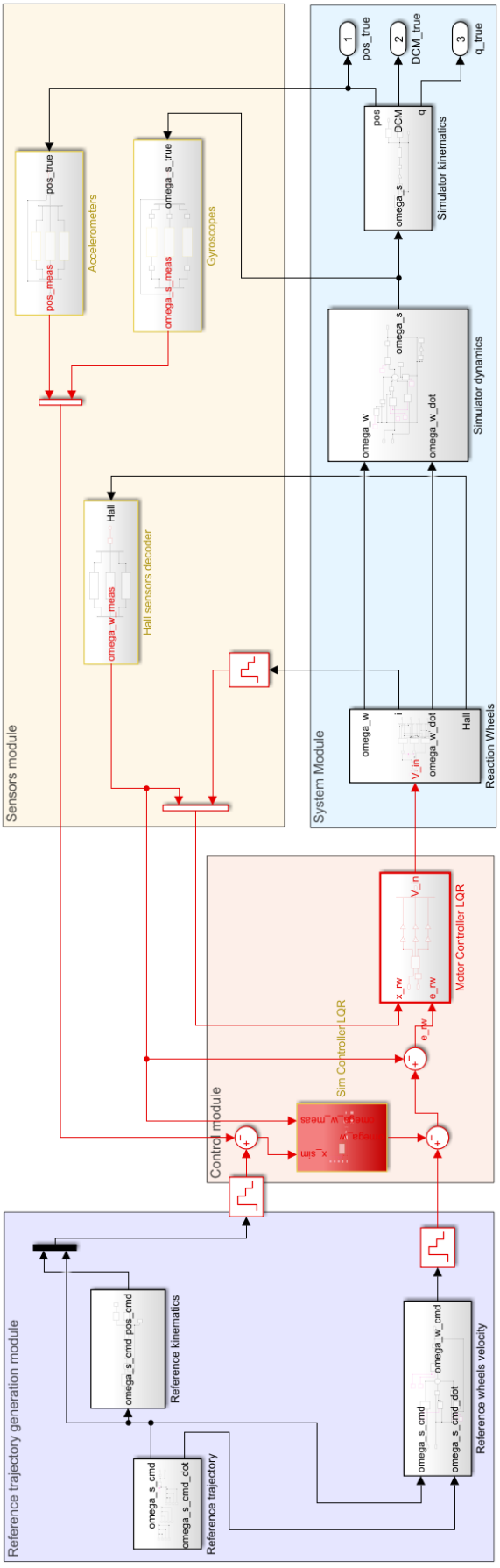


Figure 3.8: Scheme of the complete Simulink model.

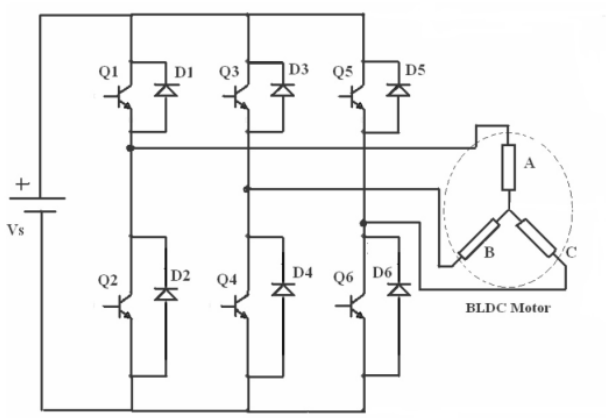


Figure 3.9: Physical model of a BLDC motor[24].

where V_A, V_B, V_C are the source voltages, i_a, i_b, i_c are the currents and e_a, e_b, e_c are the back-electromotive forces. Moreover, $R\,i$ represents the voltage drop caused by the resistance R and $L\,\frac{di}{dt}$ represents the voltage drop caused by the inductance L. R and L are assumed to be equal in the three phases.
An important characteristic of BLDC motor is the number of poles p, usually expressed as pole pairs number $\frac{p}{2}$, because each pole pair is composed by a magnetic north and magnetic south pole. The number of poles p is strictly related to the electronic commutation through the relation between the electrical rotor angle θ_e and the mechanical rotor angle θ_m, which can be written as

$$\theta_e = \frac{p}{2}\theta_m \tag{3.23}$$

The mechanical rotor angle represents the actual rotation of the motor shaft and its related to the angular velocity (of the shaft and of the wheel), while the electrical rotor angle is used to calculate the back-EMF reference function $f(\theta_e)$, which has trapezoidal shape and maximum magnitude of ± 1 and can be represented by

$$f(\theta_e) = \begin{cases} 1, & \text{if } 0 \leq \theta_e < 2\pi/3 \\ 1 - \frac{6}{\pi}(\theta_e - \frac{2\pi}{3}), & \text{if } 2\pi/3 \leq \theta_e < \pi \\ -1, & \text{if } \pi \leq \theta_e < 5\pi/3 \\ -1 + \frac{6}{\pi}(\theta_e + \frac{5\pi}{3}), & \text{if } 5\pi/3 \leq \theta_e < 2\pi \end{cases} \tag{3.24}$$

Knowing the back-EMF reference function, it is possible to write the back-EMF forces, which are out of phase of 120 degrees each, as

$$e_a = K_e\, f(\theta_e)\, \omega_w \tag{3.25}$$

$$e_b = K_e\, f(\theta_e - 2\pi/3)\, \omega_w \tag{3.26}$$

$$e_c = K_e\, f(\theta_e + 2\pi/3)\, \omega_w \tag{3.27}$$

As it can be seen, the back-EMF forces are proportional to the mechanical angular velocity ω_w of the shaft (and of the reaction wheel) through a constant called K_e, typical of the motor

under analysis, and of the back-EMF reference function. In a similar way, the torque of each phase is proportional to the current i through a constant typical of the motor called K_t, and the total electromagnetic torque output can be represented as the summation of each phase's torque

$$T_a = K_t\, f(\theta_e)\, i_a \tag{3.28}$$

$$T_b = K_t\, f(\theta_e - 2\pi/3)\, i_b \tag{3.29}$$

$$T_c = K_t\, f(\theta_e + 2\pi/3)\, i_c \tag{3.30}$$

Then, the mechanical characteristics of the wheel are expressed starting from Newton's second law for rotational bodies, taking into account the torque T generated by the motor, and a dissipation torque T_b proportional to the damping ratio b and the wheel velocity ω_w, so that the equation of motion can be written as:

$$J\frac{d\omega_w}{dt} = T - T_b = T_a + T_b + T_c - b\,\omega_w \tag{3.31}$$

where J is the rotational inertia of the wheel plus the motor along the axis of rotation of the wheel itself. So, the four equations 3.20, 3.21, 3.22, and 3.31 are four differential equations coupled through the currents i and the velocity ω_w, given the input voltages. In this case, the voltage will be given by the motor controller. In fact, substituting the expression of the back-EMF forces and the torques derived earlier, the four equations can be rearranged as:

$$\begin{cases} \dfrac{d\omega_w}{dt} = \dfrac{K_t}{J}(i_a\, f_a + i_b\, f_b + i_c\, f_c) - \dfrac{b}{J}\omega_w & (3.32) \\ \dfrac{di_a}{dt} = \dfrac{V_a}{L} - \dfrac{R}{L}i_a - \dfrac{K_e}{L}f_a\,\omega_w & (3.33) \\ \dfrac{di_b}{dt} = \dfrac{V_b}{L} - \dfrac{R}{L}i_b - \dfrac{K_e}{L}f_b\,\omega_w & (3.34) \\ \dfrac{di_c}{dt} = \dfrac{V_c}{L} - \dfrac{R}{L}i_c - \dfrac{K_e}{L}f_c\,\omega_w & (3.35) \end{cases}$$

where $f_a = f(\theta_e)$, $f_b = f(\theta_e - 2\pi/3)$, and $f_c = f(\theta_e + 2\pi/3)$. At this point, defining the state $\mathbf{x}_{rw}$ and the input $\mathbf{V}$ as

$$\mathbf{x}_{rw} = \begin{Bmatrix} \omega_w \\ i_a \\ i_b \\ i_c \end{Bmatrix} \qquad\qquad \mathbf{V} = \begin{Bmatrix} V_a \\ V_b \\ V_c \end{Bmatrix} \tag{3.36}$$

it is possible to write the system in state space form as:

$$\begin{cases} \dot{\mathbf{x}}_{rw} = A_{rw}\mathbf{x}_{rw} + B_{rw}\mathbf{V} & (3.37) \\ \mathbf{y}_{rw} = C_{rw}\mathbf{x}_{rw} & (3.38) \end{cases}$$

Eq. (3.39) gives the state matrix A_{rw} as a function of the components of the electrical and mechanical dynamic equations:

$$A_{rw} = \begin{bmatrix} -\frac{b}{J} & \frac{K_t}{J} f_a & \frac{K_t}{J} f_b & \frac{K_t}{J} f_c \\ -\frac{K_e}{L} f_a & -\frac{R}{L} & 0 & 0 \\ -\frac{K_e}{L} f_b & 0 & -\frac{R}{L} & 0 \\ -\frac{K_e}{L} f_c & 0 & 0 & -\frac{R}{L} \end{bmatrix} \tag{3.39}$$

Eq. (3.40) gives the input matrix B_{rw} as a function of the components of the electrical dynamic equation.

$$B_{rw} = \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{L} & 0 & 0 \\ 0 & \frac{1}{L} & 0 \\ 0 & 0 & \frac{1}{L} \end{bmatrix} \tag{3.40}$$

Since the mechanical dynamic equation is not a function of the voltage input, the input matrix does not apply the voltage input to the mechanical portion of the state.
Finally, the desired outputs of the state space model are the angular velocity and the currents, but also the angular acceleration of the wheel, obtained from the output matrix

$$C_{rw} = \begin{Bmatrix} \omega_w \\ i_a \\ i_b \\ i_c \\ \dot{\omega}_w \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{b}{J} & \frac{K_t}{J} f_a & \frac{K_t}{J} f_b & \frac{K_t}{J} f_c \end{bmatrix} \tag{3.41}$$

The last step is related to the determination of the input voltages V_a, V_b, and V_c: indeed, the reaction wheels block will take as input the value of the voltage V coming from the control module. Anyway, a function will be needed to assign that scalar value of voltage to the three phases, according to the rotor position. That is, the voltages V_a, V_b, and V_c are related one to the other: at every instant, one phase will be assigned to the value V, one phase to the value $-V$ and the last phase will have 0 voltage. To do this, three Matlab Functions are used inside the Simulink block, each of which implements a truth table according to Table 3.1.

Table 3.1: Six step switching sequence for commutation [24]

Rotor position [deg]	$\mathbf{H_a}$	$\mathbf{H_b}$	$\mathbf{H_c}$	**Switches closed**
0 - 60	1	0	0	Q_1 Q_4
60 - 120	1	1	0	Q_1 Q_6
120 - 180	0	1	0	Q_3 Q_6
180 - 240	0	1	1	Q_3 Q_2
240 - 300	0	0	1	Q_5 Q_2
300 - 360	1	0	1	Q_5 Q_4

The first Matlab Function check the rotor position, θ_e, and assigns the signal of the Hall sensors, that switch every 60 degrees. The second function decides which switch to close, according to the signal coming from the Hall sensors. The last function take as input the switches and assigns the voltage values V, $-V$, or 0 to V_a, V_b, and V_c, according to the closed switches at the time. The relation between the switches and the phases of the circuit can be seen in Fig. 3.9: if

one of the top switches is closed, then the relative phase will have a positive voltage V. On the other hand, if one of the bottom switches is closed, then the relative phase will have a negative voltage $-V$, while the last phase will remain at zero voltage.
Fig. 3.10 provides an overview of the 'RW block'. As it can be seen, the state space is implemented in the Matlab Function called 'Motor state space', which needs as input the back electromotive forces vector F. After the integration of the dynamics, the output is divide into the velocity ω_w, the currents i (composed of i_a, i_b, and i_c), and the acceleration $\dot{\omega}_w$. The mechanical angle is retrieved integrating ω_w and then transformed to the electrical angle θ_e, which is used in the 'Back-EMF' function to get F, which in turn is fed back to the state-space equations. At the same time, θ_e is used for the commutation by the three truth tables connected in series, from which also the Hall sensors signal is extracted. As a last remark, two other quantities are extracted from the block: the electrical power P_{el} and the mechanical power P_{mec}. The electrical power is obtained multiplying the voltage by the current, for each phase, while the mechanical power is obtained according to Eq. (3.17). Moreover, 'RMS' (root mean square) blocks are used to filter the output of currents and voltages.

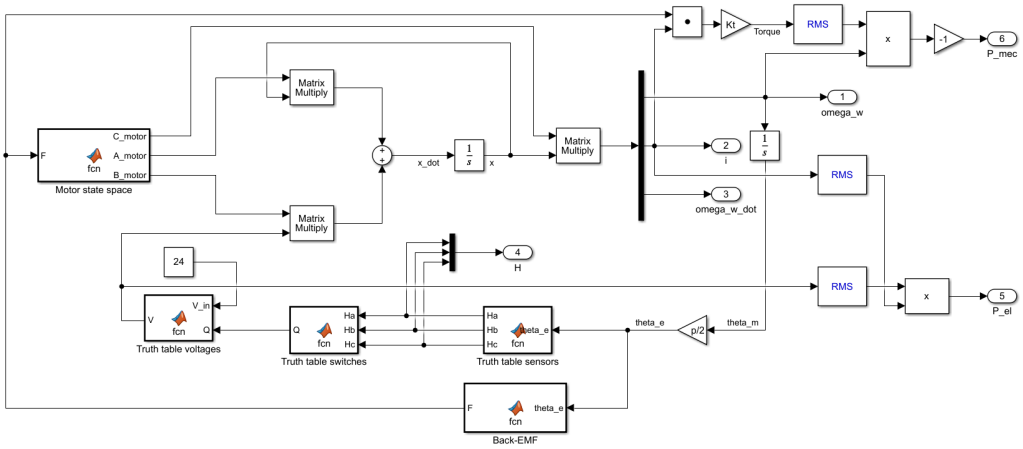


Figure 3.10: Simulink diagram of one RW block.

Simulator dynamics block

The dynamics of the simulator obeys to the Euler's equation already presented in Sec. 2.1.6, where the only source of torque are the reaction wheels, which do not provide external torques, and is assumed to be no external disturbance torques on the system. Therefore, the vectorial equation describing the dynamics of the simulator is Eq. (2.62), which in this case can be written as:

$$I_s\dot{\boldsymbol{\omega}}_s + \boldsymbol{\omega}_s \times I_s\boldsymbol{\omega}_s + AI_w\dot{\boldsymbol{\omega}}_w + \boldsymbol{\omega}_s \times AI_w\boldsymbol{\omega}_w = 0 \tag{3.42}$$

where I_s is the inertia matrix of the simulator, I_w is the inertia matrix of the wheel (which is a diagonal matrix with J as value), and A is the matrix that allows to pass from the reference system of the wheels to the body frame, as already explained in Sec. 2.1.7. In this block, the input is arriving from the reaction wheels block and is represented by the wheels velocities $\boldsymbol{\omega}_w$ and accelerations $\dot{\boldsymbol{\omega}}_w$, while the output is the vector of angular velocities of the simulator $\boldsymbol{\omega}_s = [\omega_{s,x} \quad \omega_{s,y} \quad \omega_{s,z}]^T$. The velocity vector $\boldsymbol{\omega}_s$ can be retrieved from the integration of $\dot{\omega}_s$, which in turn can be retrieved rearranging Eq. (3.42) as

$$\dot{\boldsymbol{\omega}}_s = I_s^{-1}(-\boldsymbol{\omega}_s \times I_s\boldsymbol{\omega}_s - AI_w\dot{\boldsymbol{\omega}}_w - \boldsymbol{\omega}_s \times AI_w\boldsymbol{\omega}_w) \tag{3.43}$$

To do so, the inertia matrix I_s was assumed to be constant: actually, even considering all the components on the simulator as rigid bodies, the inertia could change over time, because of the presence of loose cables and of the moving masses used to balance the system. For this reason, it was assumed that the balancing procedure had already been carried out and therefore that the position of the moving masses is fixed during the maneuver.
Fig. 3.11 shows the Simulink block diagram for the integration of Eq. (3.42): the inputs arrive from the previous block and the integrated output is fed back into the equation.

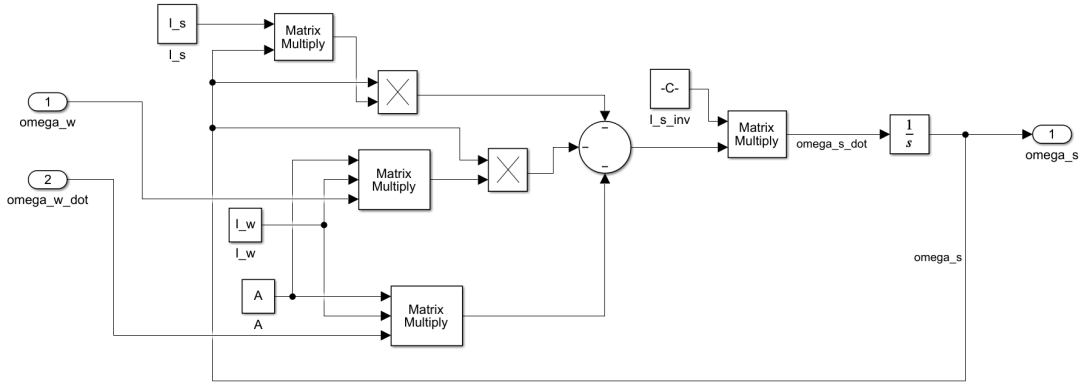


Figure 3.11: Simulink diagram of the simulator dynamics block.

Simulator kinematics block

This block represents the kinematics equations and is used to obtain the Euler angles and the direction cosine matrix from the angular velocities of the simulator. The parameters chosen to describe how attitude changes are the quaternions: the main advantages of this representation are that a quaternion is a vector of only four parameters (the direction cosine matrix needs nine parameters) and that they have no singular condition. In some cases it is convenient to divide the quaternion into a vector part and a scalar part: in this case, the fourth entry is chosen as the scalar part, so that:

$$\mathbf{q} = \begin{Bmatrix} q_1 \\ q_2 \\ q_3 \end{Bmatrix} \quad , \quad q_4$$

Having stated that, the kinematics equation for the quaternions can be expressed as

$$\dot{\mathbf{q}} = \frac{1}{2}\Omega q \tag{3.44}$$

where Ω is a skew symmetric matrix composed by the angular velocities of the simulator, as follows

$$\Omega = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \tag{3.45}$$

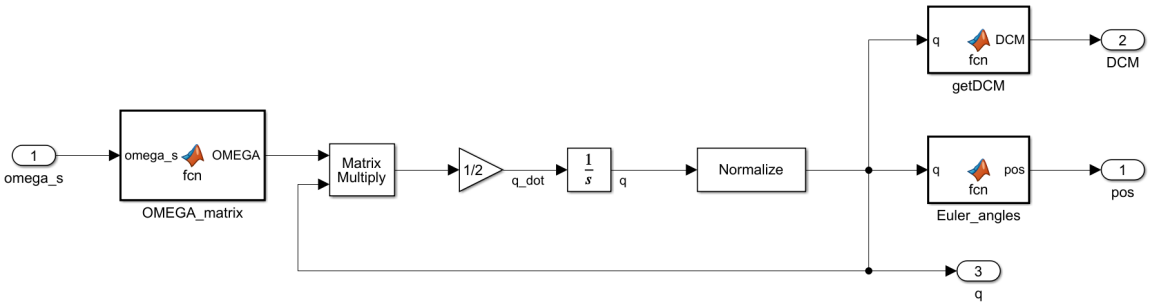


Figure 3.12: Simulink diagram of the simulator kinematics block.

As shown in Fig. 3.12, the angular velocities taken from the input are assembled in the matrix Ω thanks to a Matlab function and then integrated. At every step, after the integration, the quaternion needs to be normalized. As a final step, the quaternion in converted into the Euler angles and the direction cosine matrix, thanks again to two Matlab functions which implements respectively the following equations:

$$\begin{Bmatrix} \phi \\ \theta \\ \psi \end{Bmatrix} = \begin{Bmatrix} \tan^{-1}\left(\frac{2(q_1q_4+q_2q_3)}{1-2(q_1^2+q_2^2)}\right) \\ \tan^{-1}\left(2(q_4q_2-q_3q_1)\right) \\ \tan^{-1}\left(\frac{2(q_4q_3+q_1q_2)}{1-2(q_2^2+q_3^2)}\right) \end{Bmatrix} [25] \tag{3.46}$$

$$DCM = \begin{bmatrix} q_1^2-q_2^2-q_3^2+q_4^2 & 2(q_1q_2+q_3q_4) & 2(q_1q_3-q_2q_4) \\ 2(q_1q_2-q_3q_4) & -q_1^2+q_2^2-q_3^2+q_4^2 & 2(q_2q_3+q_1q_4) \\ 2(q_1q_3+q_2q_4) & 2(q_2q_3-q_1q_4) & -q_1^2-q_2^2+q_3^2+q_4^2 \end{bmatrix} [26] \tag{3.47}$$

3.2.2 Sensors module

The aim of this module is to simulate the signals coming from the sensors that will be mounted on the platform. Since the control strategy implemented is of feedback type, it requires knowledge of the characteristics of the system at every time step. The sensors module takes as input the signals coming from the system module, which simulates the real continuous characteristics of the system (like orientation and angular velocity) and gives as output the measured characteristics, which are discretized according to the sampling time of the sensors and subjected to noise. Simulating the stochastic properties of the signals coming from the sensors is an important step that improves the model and increases its fidelity to the real system.

Gyroscopes block

The first block to be described is the gyroscopes block, which takes as input the angular velocities of the simulator and outputs the measured velocities $\boldsymbol{\omega}_{meas}$. The block is composed of three subsystems, one for each axis. Each subsystem models one gyro and its noise according to the following equation:

$$\omega_{meas} = \omega_{real} + n_g + b_g \tag{3.48}$$

where the measured velocity ω_{meas} is obtained from the real velocity of the system ω_{real} plus n_g, which is known as angular random walk (ARW), plus b_g, known as rate random walk (RRW). The ARW is modeled as white Gaussian noise with zero mean and standard deviation σ_n, while the RRW is slightly different because it is its derivative, $\dot{b}_g$, to be modeled as white Gaussian noise with zero mean and standard deviation σ_b. So, the ARW acts as a Gaussian noise directly on the real velocity, while the white noise of the RRW needs to be integrated before being added to the real velocity. Because of the integration, the RRW does not oscillate around a mean value, but rather increases over time. Luckily, sensors can be calibrated in order to reduce as much as possible this contribution. Fig. 3.13 shows the Simulink block of a single gyroscope, where the addition of the ARW and RRW are visible at the beginning, on the left. All the numerical values are taken from the datasheet of the BNO055, already presented in Sec. 2.3.2.

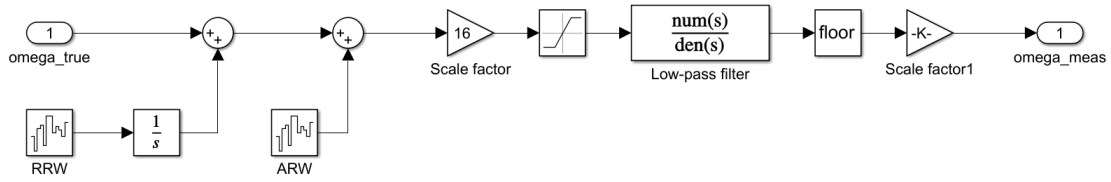


Figure 3.13: Simulink diagram of one gyroscope subsystem.

After the addition of the noises, the signal is multiplied by a scale factor which transforms the signal from an angular velocity to a number of counts: this value is typical of the sensor used and is related to its sensitivity. After that, a saturation block is used to model the range of the sensor: indeed, if the velocity is too high, the sensor will fall out of its range and gives back a constant value. As the last step, a low-pass filter is applied to the signal, in order to filter out the higher frequencies oscillations. Low-pass filters are always present in this type of sensor: in this case, the only information in the datasheet about the filter is that the bandwidth can be selected between 523 and 12 Hz. Without further information, it was decided to implement a Butterworth filter because this type of filter has a flat pass-band, but a poor roll-off rate [27]. Since the filters that are in the sensors have usually good performances, it was decided to use a third-order filter, instead of a simple first-order one. The transfer function for a third-order Butterworth filter is:

$$H(s) = \frac{\omega_c^3}{s^3 + 2\omega_c s^2 + 2\omega_c^2 s + \omega_c^3} \tag{3.49}$$

where ω_c is the cutoff frequency of the filter, which was chosen empirically to be 40 Hz. Anyway, as already said, the bandwidth of the filter in the sensor can be controlled by the user. With this data, it is possible to plot the bode diagram of the filter, as shown in Fig. 3.14.

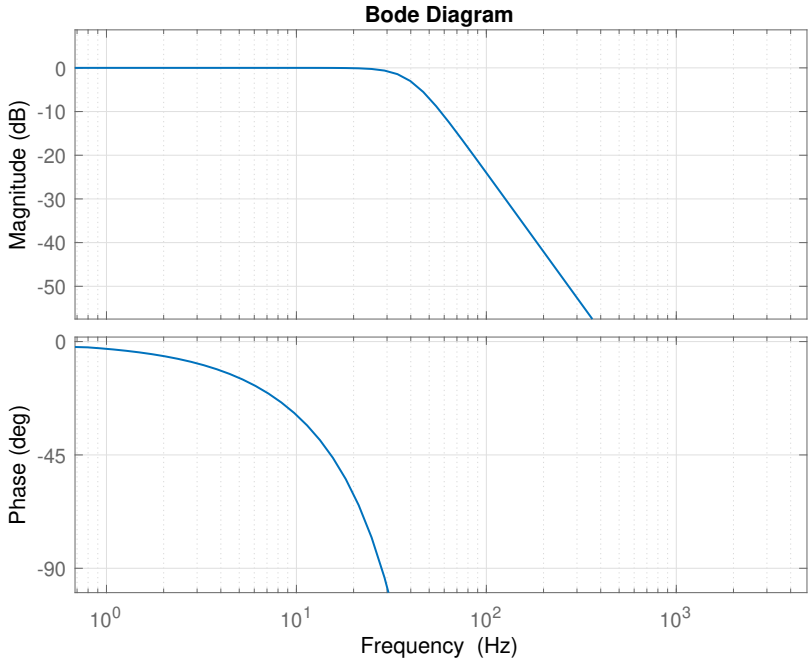


Figure 3.14: Third-order Butterworth filter bode diagram ($\omega_c = 40$ Hz).

Then, in order to verify the validity of the filter obtained, it was decided to check information about other similar sensors. In particular, in the datasheet of the ITG-3205 by InvenSense [1], there is plotted the bode diagram of Fig. 3.15. Here, the numbers correspond to different bandwidths: the curve marked with the number three refers to a bandwidth of 42 Hz. As it can be seen, the third-order Butterworth filter implemented approximates well the behavior of a typical gyroscope low-pass filter.

After the filter, the count is approximated to an integer and converted back to a spin rate value through the scale factor.
The last step of the gyroscope block consists of a zero-order hold, which discretizes the signal according to the sampling rate of the sensor. According to the datasheet of the BNO055, the sampling rate was selected as 100 Hz.

Accelerometers block

This block describes how the accelerometers are modeled. Accelerometers, as the gyroscopes, are Micro Electro-Mechanical Systems (MEMS) used to measure the gravity vector. The main idea of the block should be to take as input the gravity vector in the simulator reference frame g_s (obtained premultiplying $g = [00 - 9.81]$ by the DCM from the kinematics block) and add some noise to get a measured value of the gravity vector g_{meas}. From this, the error in the orientation of the simulator can be obtained comparing the measured gravity vector g_{meas} with a reference gravity vector g_{ref} properly generated. Anyway, this implementation has a major problem: the unobservability around the z-axis. In fact, since the maneuver under study involves a rotation around the z-axis, the gravity vector will remain fixed during the entire maneuver, making it useless for measuring the error in orientation. To overcome this problem, sensors implement other types of measurement systems, like magnetometers, to be used in statistical methods for attitude determination. However, the sensor mounted on the simulator, the BNO055, is capable

[1] InvenSense ITG-3205 datasheet, `http://dl.btc.pl/kamami_wa/itg3205.pdf`

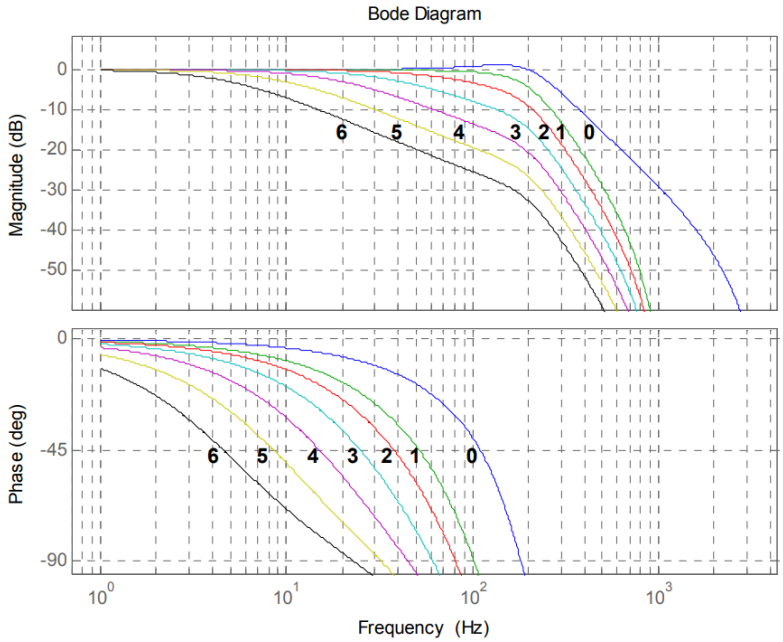


Figure 3.15: ITG-3205 filter bode diagram.

of giving as output already the Euler angles (or quaternions) and the datasheet gives information about the noise expected. Taking everything into account, it was decided to only simulate the effect of the noise on the outputted Euler angles, since the presence of noise could influence the control algorithm. For this reason, for the accelerometers block there were used the same subsystems of the gyroscopes block, as in Fig. 3.13. The only difference is represented by the absence of the ARW, since a more precise calibration process for the accelerometers makes it possible to neglect this contribution, while the low-pass filter remains the same. In this case, the data are taken from the accelerometers data of the BNO055.
In this way, the effect of the noise on the Euler angles is accounted for, while at the same time reducing the complexity and the computational effort of the simulation.

Hall sensors decoder block

This block takes as input the signal from the Hall sensors coming from the reaction wheels block and uses it to calculate the velocity of the wheels. As for the reaction wheels block, this block is composed of three identical subsystems, one for each wheel. The signal from the Hall sensors of a single wheel has three components (H_a, H_b, and H_c) that change between 0 and 1 as a function of the rotor position, according to Table 3.1: as it can be seen, each change corresponds to a rotation of 60 degrees. The main idea for decoding the signal is that of evaluating the time between one change and the following and then calculating the velocity as 60 degrees divided by the time elapsed.
Fig. 3.16 displays the model of one sensors decoder. The input signal is used to trigger two subsystems: the first one is triggered by the first change in the signal (both rising and falling type), while the second subsystem is triggered with a delay. In this way, there is always one step of difference between them, so that their subtraction gives the time elapsed between one change and the following. Then, the time is put at the denominator and multiplied by a gain of $\frac{\pi}{3}\frac{2}{p}$, which represents 60 degrees of the electrical angle θ_e. Right before this, there is a switch that activates only in the first step, to avoid putting 0 in the denominator. At this point, the velocity is calculated.

Anyway, the signal outputted in this way results to be too much noisy, in particular at high

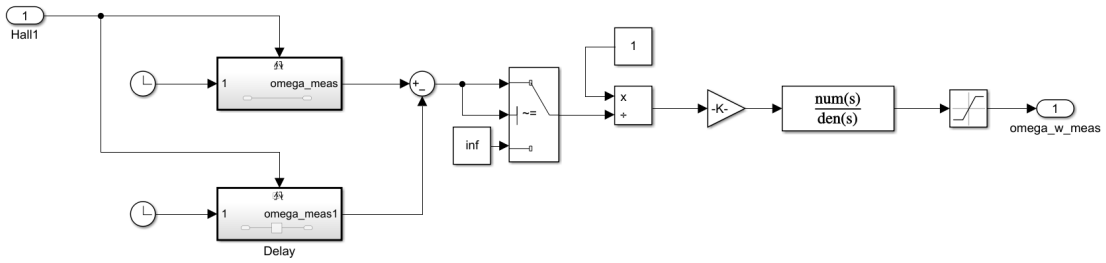


Figure 3.16: Simulink diagram of one Hall sensors decoder subsystem.

rotational speeds. For this reason, the same low-pass filter applied in the gyroscope block was added at the end of this block, in addition to a saturation block, to limit the maximum value of the output. Furthermore, a zero-order hold is placed just before the output of the block, with the same sampling time of the gyroscopes and accelerometers.

3.2.3 Reference trajectory generation module

This module provides the reference state to be tracked by the control algorithm: in this way, the user can define the motion of the simulator and the control of the system will move the wheels in order to achieve the specified motion. The inputs of the module can be selected by the user: in our case, the inputs from the maneuver described in Sec. 3.1 have been assumed, which are the total angle of the maneuver, the total time, and the parameter n (as will be explained in the following section). From these inputs, the module will provide the reference Euler's angles, the reference velocities of the simulator $\omega_{s,ref}$, and the reference velocities of the wheels $\omega_{w,ref}$ that will go into the control module.

Reference trajectory block

This block simply concatenates the inputs along x, y, and z to provide the reference trajectory. The trajectory to be tracked can be provided in terms of angles, velocities or accelerations. For the slew maneuver under study, the block generates both the velocity and acceleration profiles. In Sec. 3.1, the motion of the wheel and of the simulator was already studied considering three different phases: constant acceleration, constant velocity, and constant deceleration. These profiles, shown in Fig. 3.1, represent the reference trajectory to be generated. To do so, ramp inputs were used in the Simulink block: to define them, it is required the knowledge of the slope of the ramp and of the times t_0, t_1, t_2, and t_3 that defines each phase. These parameters (times and slope) were calculated starting from the inputs recalled at the beginning of the previous section: $\theta_{s,tot}$, t_{tot}, and n. In fact, recalling that $t_{acc} = n\, t_{tot}$ and $t_{dec} = t_{acc}$, the times are obtained as

- $t_0 = 0$
- $t_1 = t_0 + t_{acc}$
- $t_2 = t_1 + (1 - 2n)\, t_{tot}$
- $t_3 = t_2 + t_{acc}$

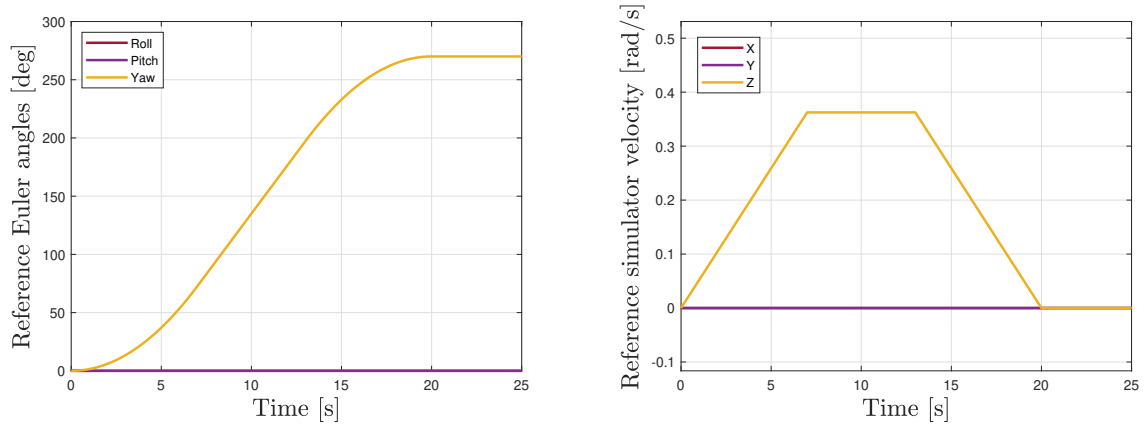
The slope of the velocity, instead, is obtained from Eq. (3.13), taking into account that the slope is $\frac{T_0}{I_s}$ (as can be checked from Eq. (3.3)) and is:

$$\omega_{slope} = \frac{\theta_{s,tot}}{t_{tot}^2 \left(n - n^2\right)} \tag{3.50}$$

These inputs data are calculated in a Matlab file called `input_preload.m`, which is automatically launched when opening Simulink.

Reference kinematics block

This block is the same already described in Sec. 3.2.1, which is the 'Simulator kinematics block'. Their function is the same: obtaining Euler's angles from the velocity of the simulator. To distinguish between the two cases, the output of this block are called the 'reference Euler's angles' ϕ_{ref}, θ_{ref}, and ψ_{ref}. In this way, all the reference values of the simulator (angles, in addition to velocities and accelerations) are calculated. Below, the reference angles and velocities are shown for a maneuver of $\theta_{s,tot} = 270\,\text{deg}$ in $t_{tot} = 20\text{s}$ with the parameter n equal to 0.35.



(a) Reference Euler's angles

(b) Reference velocities of the simulator

Figure 3.17: Reference trajectories for a maneuver of 270 deg in 20 seconds with $n = 0.35$.

Fig. 3.17b displays the reference velocities of the simulator, as calculated in the previous 'Reference trajectory block'. As it can be seen, the behavior is increasing until t_1 (which is 7s for this maneuver), then constant, and then decreasing, as expected. After 20s, the command is stopped. Moreover, the velocity along z (which is the axis of rotation) reaches a value of 0.3625 rad/s. This represents the profile of the velocities that the simulator should reproduce in order to complete the specified maneuver.
The kinematics block gives the reference Euler's angles seen in Fig. 3.17a. Unlike the reference velocities, the reference angles are smooth. This is because integrating the ramps in Fig. 3.17b produce the parabolas connecting the linear portions of Fig. 3.17a. Once again, the maneuver is completed in 20s, with a rotation of 270 deg along the yaw axis.
So, this module is very flexible and can be changed from time to time to check the performance of the system with different maneuvers. The only limits are the physical limits of the air bearing in roll and pitch.

Reference wheels velocity block

This is the last block of the reference trejecotry generation module and its function is that of calculating the velocity at which each wheel needs to rotate in order for the simulator to achieve its reference trajectory. The reference velocity vector $\omega_{w,ref}$ can be calculated from Eq. (3.42):

indeed, it is possible to reverse the equation, taking as inputs the velocities and accelerations of the simulator. In this way, the equation becomes:

$$\dot{\omega}_{w,ref} = I_w^{-1} A^* (-I_s \dot{\omega}_{s,ref} - \omega_{s,ref} \times I_s \omega_{s,ref} - \omega_{s,ref} \times A I_w \omega_{w,ref}) \tag{3.51}$$

where A^* is the Moore–Penrose inverse (or pseudo-inverse) of matrix A, that, in the case of 3 reaction wheels, is simply the inverse. The reference wheels velocity block is the Simulink implementation of Eq. (3.51) and it is shown in Fig. 3.18.

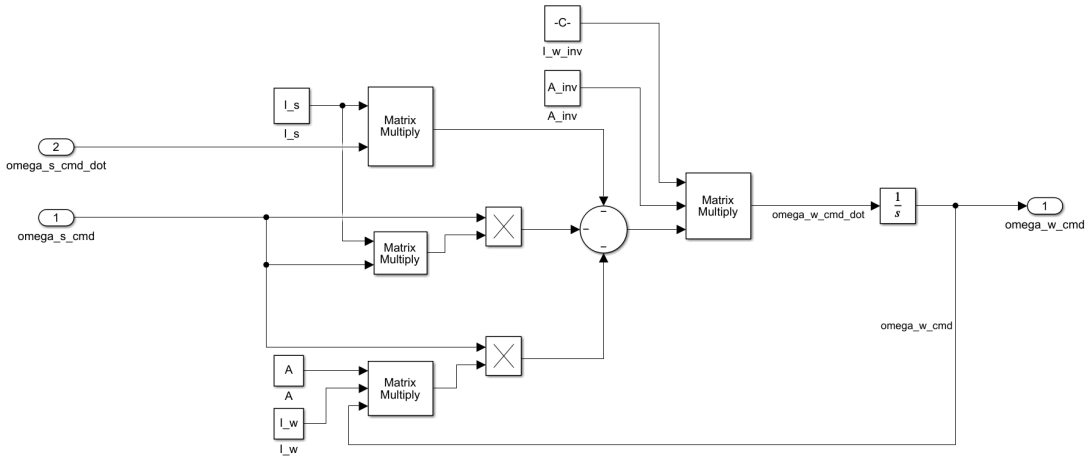


Figure 3.18: Simulink diagram of the reference wheels velocity block.

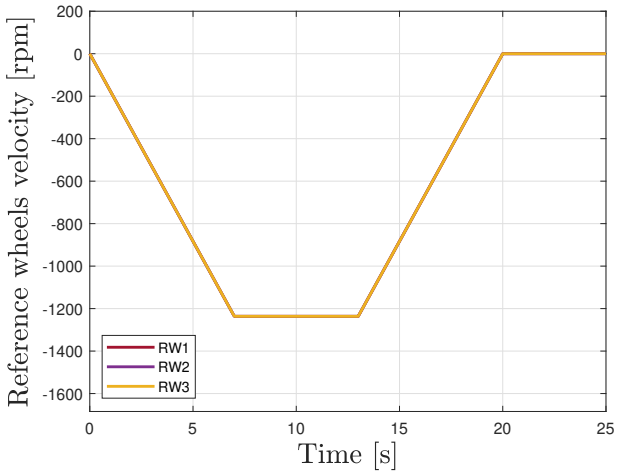


Figure 3.19: Reference velocities of the wheels.

Fig. 3.19, instead, shows the velocities of the wheels in *rpm*. Comparing it with Fig. 3.17b, it is clear that the behavior is very similar but opposite, as it should be expected because of the conservation of angular momentum of the system. Anyway, in this case, all the three wheels are moving with the same speed: in fact, given the pyramidal configuration, it is mandatory that all the three wheels rotate so that the components of torque along x and y eliminate each other, while the components along z are summed up. This is one of the greatest advantages of this configuration, because it allows the torque to be distributed among the wheels, so that each wheel is responsible only for a fraction of the total torque.

Finally, an important consideration should be done. This block gives the reference velocity needed by the wheels to achieve the desired rotation of the simulator. Putting this $\omega_{w,ref}$ into Eq. (3.42), the desired velocity (and so orientation) of the simulator will be obtained. However, this is not enough to assure that the real system will behave properly. The reason is that Eq. (3.42) is only a simplified model of the overall *real* system, written with many assumptions, without taking into account disturbances or noise. That is why a control algorithm will be needed, to ensure the right orientation of the simulator even in presence of unmodeled dynamics and other errors.

3.2.4 Control module

The control module is probably the most important one in the whole Simulink model. Its goal is to assure that the simulator will move in accordance with the motion specified by the user, achieving the required orientation with good accuracy in a reasonable time. The only input that can be fed by the control system is the velocity of the reaction wheels: the final orientation of the overall system should be attained changing these velocities in an accurate way, established by the control algorithm. In Sec. 3.2.3, the reference wheels velocity block was presented, from which the desired velocity of the wheels $\omega_{w,ref}$ is calculated. However, this block is not inserted in the control module because that velocity is only a reference that should be tracked by the control algorithm. For this reason, a first feedback loop should be defined about the reaction wheels equations. This first loop is needed to be sure that the velocity of the wheels ω_w will track the reference velocity $\omega_{w,ref}$. Doing so, the correct wheels' velocity will enter in the simulator's equations, ensuring that the simulator reaches the required orientation. Unfortunately, this is true only in the Simulink model, where the evolution of the simulator is governed purely by Eq. (3.42). In the real system, instead, the evolution of the simulator will depend also by other factors neglected in the equations, as already mentioned in Sec. 3.2.3. These factors are mainly represented by disturbance torques acting on the system, like the misalignment of the center of mass with the center of rotation, the friction inside the air bearing, and the atmospheric friction outside, but also by unmodeled dynamics. Furthermore, the coefficients used for the system, like the inertias, the constants of the motor and so on, are all estimates of quantities that can be slightly different in the real physical world.
For all these reasons, also a second loop was conceived, that will check the actual orientation of the system and compare it with the reference orientation, calculating a control effort that will then be added to the reference wheels'velocity $\omega_{w,ref}$. In this way, $\omega_{w,ref}$ will account also for the effects neglected in the equations.

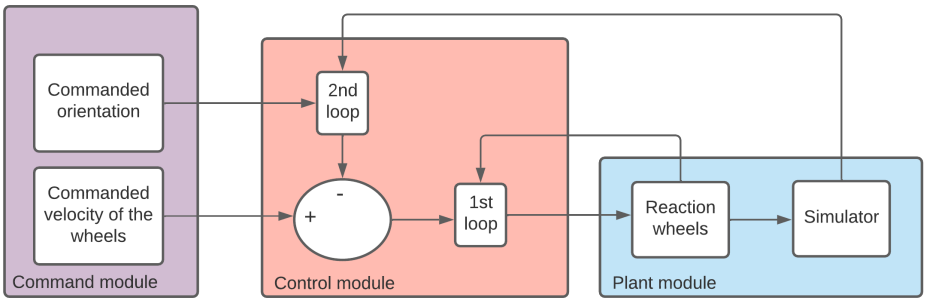


Figure 3.20: Scheme of the control algorithm.

Fig. 3.20 shows a scheme of the control algorithm, highlighting the two closed loops. From the figure, it can also be seen how the control effort from the second loop is added to the reference

velocity of the wheels $\omega_{w,ref}$ before entering in the first loop.

LQR technique

In the following subsections it will be described how the two controllers (inner and outer) were designed. Both control loops are of the type multiple input - multiple output: so, to design the controllers, the linear quadratic regulator (LQR) technique was adopted. The LQR is a state-space design technique which relys on a full-state feedback law: taking into account the fundamental state-space system

$$\begin{cases} \dot{\mathbf{x}}(\mathbf{t}) = A\mathbf{x}(t) + \mathbf{u}(t) & (3.52) \\ \mathbf{z}(\mathbf{t}) = C_z\mathbf{x}(t) + D_z\mathbf{u}(t) & (3.53) \end{cases}$$

the dynamics of the system is function of the state $\mathbf{x}(t)$ and of the input $\mathbf{u}(t)$, while $\mathbf{z}(t)$ represent the performance vector. Using a full-state feedback law means that the input $\mathbf{u}(t)$ can be expressed as a function of the state $\mathbf{x}(t)$, that is:

$$\mathbf{u}(t) = -G\,\mathbf{x}(t) \tag{3.54}$$

where G is the gain matrix that, once computed, provides the optimal solution to the control problem. It is important to notice that the full-state feedback requires the knowledge of all the variables of the state, or at least of an estimate of them. Knowing the gain matrix and the state, the closed-loop dynamics can be expressed as

$$\dot{\mathbf{x}}(t) = A\,\mathbf{x}(t) + B\,\mathbf{u}(t) = A\,\mathbf{x}(t) - B\,G\,\mathbf{x}(t) \tag{3.55}$$

The main idea behind the LQR technique is that the optimal solution is related to the minimization of a quadratic cost function, which can be written as a function of the system input and performance as

$$J = \frac{1}{2}\int_0^\infty (\mathbf{z}(t)^T W_{zz}\mathbf{z}(t) + \mathbf{u}(t)^T W_{uu}\mathbf{u}(t))dt \tag{3.56}$$

Anyway, it is usually preferred to rewrite the cost function in relation to the system input and state. Manipulating Eq. (3.56), it is possible to write the cost function as

$$J = \frac{1}{2}\int_0^\infty (\mathbf{x}(t)^T Q\mathbf{x}(t) + 2\mathbf{x}(t)^T S\mathbf{u}(t) + \mathbf{u}(t)^T R\mathbf{u}(t))dt \tag{3.57}$$

where

- $Q = C_z^T W_{zz} C_z$
- $S = C_z^T W_{zz} D_z$
- $R = W_{uu} + D_z^T W_{zz} D_z$

As it can be seen, the matrix S is related to the disturbance matrix D_z. Since in our case the matrix D_z is not taken into account in any state-space system, it is possible to neglect the matrix S in the definition of the cost function. In this way, only the matrices Q and R remain, which are related respectively to the deviation of the state from the origin and to the control effort. The corresponding solution will then be strongly affected by the weighting matrices, which therefore play a fundamental role in the design process. In fact, since J is to be minimized, increasing Q with respect to R will mean that $\mathbf{x}(t)$ will have more importance in the minimization process than $\mathbf{u}(t)$, and vice versa. Moreover, this is true also for each components of the two matrices: if one of the states is more important to control than the others, a large value can be placed in the corresponding position within the matrix Q. This will place a larger penalty on that particular state and drive the cost function to minimize that state more than the others. The same holds for the input weighting matrix R.
Once that the weighting matrices are specified by the designer, it is possible to solve the algebraic Riccati equation (Eq. (3.58)) to find the symmetric positive matrix P.

$$P(A - BR^{-1}S^T) + (A^T - SR^{-1}B^T)P - PBR^{-1}B^TP + Q - SR^{-1}S^T = 0 \tag{3.58}$$

Substituting P into Eq. (3.59) gives the optimal gain matrix for the system. The matrix G is constant, so it does not change over time and can be calculated only one time. Multiplying it with the time-dependent state $\mathbf{x}(t)$ gives the optimal control input at every time, according to Eq. (3.54).

$$G = R^{-1}(B^TP - S^T) \tag{3.59}$$

Eq. (3.59) and Eq. (3.58) are solved automatically by Matlab thanks to the built-in function `lqr`, which takes as input the matrices of the system A and B, plus the weighting matrices Q and R. The calculations are done at the beginning of the simulation by a Matlab file, launched automatically by Simulink, called `data_preload.m`, which contains all the initial data for the simulation.
The LQR technique, with some modifications, was used to design both the controllers of the model. The exact design will be presented in the following sections. Anyway, the control module, as the reference one, is quite flexible and can be modified at any time, even during the testing process, according to the different scenarios under study. The simplest modification would be to change the weighting matrices Q and R, which would consequently change the gain matrix G.

Motor controller LQR block

This block implements the LQR that will control the velocity of the wheels. As previously described, the LQR is able to provide a certain control input $\mathbf{u}(t)$ related to the state $\mathbf{x}(t)$, so that it works when the state is different from zero and acts to bring it back to the origin. However, in this case, the main focus is to follow a certain reference profile, rather than bring the state to zero. For this reason, the LQR needs some modifications to be able to track the reference assigned. Two common methods for adding steady-state tracking capabilities to full-state feedback laws are the *feedforward input* and the *integral action* [27]. For this particular case, robustness is of great relevance since the Simulink model has some parameters which are only estimated (like the inertia of the simulator). So, the integral action is more suited because it is a method more robust with respect to any change in the parameters of the system.
The method consists in adding the integral of the system error to the dynamic equations.

Considering the output of the state-space system $\mathbf{y}(t)$ and the reference signal $\mathbf{r}(t)$, the tracking error is

$$\mathbf{e}(t) = \mathbf{r}(t) - \mathbf{y}(t) \tag{3.60}$$

The control law $\mathbf{u}(t)$ is related not only to the state $\mathbf{x}(t)$, but also to the integral of the error expressed as $\mathbf{x}_I(t)$, as follows

$$\mathbf{u}(t) = -G\mathbf{x}(t) - G_I \int \mathbf{e}(t)dt = -\begin{bmatrix} G & G_I \end{bmatrix} \begin{Bmatrix} \mathbf{x}(t) \\ \mathbf{x}_I(t) \end{Bmatrix} \tag{3.61}$$

So, $\mathbf{x}(t)$ and $\mathbf{x}_I(t)$ will form a new augmented state $\mathbf{x}_{aug}(t)$. The dynamics of $\mathbf{x}(t)$ remains the same of the basic system, while the dynamics of the new state $\mathbf{x}_I(t)$ is

$$\dot{\mathbf{x}}_I(t) = \mathbf{e}(t) = \mathbf{r}(t) - \mathbf{y}(t) = \mathbf{r}(t) - C\mathbf{x}(t) \tag{3.62}$$

The new augmented system will be

$$\dot{\mathbf{x}}_{aug}(t) = A_{aug}\mathbf{x}_{aug}(t) + B_{u,aug}\mathbf{u}(t) - B_{r,aug}\mathbf{r}(t) \tag{3.63}$$

where the augmented matrices are

$$A_{aug} = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix} \qquad B_{u,aug} = \begin{bmatrix} B \\ 0 \end{bmatrix} \qquad B_{r,aug} = \begin{bmatrix} 0 \\ I \end{bmatrix} \tag{3.64}$$

At this point, it is possible to define the weighting matrices Q and R and design a classic LQR using the augmented state instead of the basic one. Doing so, the state will track the reference $\mathbf{r}(t)$ in place of tracking the origin.
Having said that, it is clear why the motor controller block has two inputs, which are the state of the reaction wheels system $\mathbf{x}_{rw}$, which contains the velocities and the currents of the three wheels, and the error to be tracked $\mathbf{e}_{rw}$, which contains the difference between the reference velocities and the actual velocities of the wheels. Inside the block, after the integration of the error, a Matlab function is used to rearrange the components of $\mathbf{x}_{rw}$ and $\mathbf{e}_{rw}$, in order to create the augmented state $\mathbf{x}_{rw,aug}$ of each wheel. For instance, the augmented state of the third wheel is expressed as

$$\mathbf{x}_{rw3,aug} = \begin{Bmatrix} \omega_3 \\ i_{a3} \\ i_{b3} \\ i_{c3} \\ e_{rw3} \end{Bmatrix} \tag{3.65}$$

where the 3 indicates the third wheel. Multiplying the augmented state by the gain matrix G_{aug}, with a minus sign according to Eq. (3.54), it is obtained the nominal voltage to be applied to the wheel, independently for each wheel. After that, the three voltages are concatenated into a single vector $\mathbf{V}_{in}$, which is the output of the block.
Now, the reference to be tracked by the motor controller is represented by the profile of the

reference velocity of each wheel $\omega_{w,ref3}$. This means that the reference r_{rw3} and so the error e_{rw3} will be scalar values. For this reason, it is mandatory to write a new output matrix $C_{rw,e}$ for the reaction wheel state-space system that gives as outputs only the velocity of the wheel. This new matrix $C_{rw,e}$ is defined simply as the first row of matrix C_{rw} in Eq. (3.41).
On the other hand, the input is $\mathbf{V} = [V_a \quad V_b \quad V_c]$: anyway, the voltages V_a, V_b, and V_c can have a value of either V, -V, or 0, according to the switches that are closed, as defined in Sec. 3.2.1. For this reason, it is possible to define a single input V by multiplying the B_{rw} matrix for the vector [1 -1 0], which represents the values of the switches at the initial time. After this, the augmented reaction wheels system can be written as

$$A_{rw,aug} = \begin{bmatrix} A_{rw} & 0 \\ -C_{rw,e} & 0 \end{bmatrix} = \begin{bmatrix} -\frac{b}{J} & \frac{K_t}{J} f_a & \frac{K_t}{J} f_b & \frac{K_t}{J} f_c & 0 \\ -\frac{K_e}{L} f_a & -\frac{R}{L} & 0 & 0 & 0 \\ -\frac{K_e}{L} f_b & 0 & -\frac{R}{L} & 0 & 0 \\ -\frac{K_e}{L} f_c & 0 & 0 & -\frac{R}{L} & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.66}$$

$$B_{rw,aug} = \begin{bmatrix} B_{rw} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{L} \\ -\frac{1}{L} \\ 0 \\ 0 \end{bmatrix} \tag{3.67}$$

At the same time, the weighting matrices for the LQR are defined as

$$Q_{rw} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 0 & 0 \\ 0 & 0 & 30 & 0 & 0 \\ 0 & 0 & 0 & 30 & 0 \\ 0 & 0 & 0 & 0 & 100 \end{bmatrix} \qquad R_{rw} = \begin{bmatrix} 50 \end{bmatrix} \tag{3.68}$$

These matrices are chosen with a trial-and-error procedure because they guarantee a good result in terms of dynamics response, but they can be modified easily through the Matlab file `data_preload.m`, which could be very helpful during the testing phase. Anyway, it is important to note how the values are assigned: while the entries related to the currents and voltages are respectively 30 and 50, the entry related to the error is assigned a bigger penalty of 100, in order to minimize more the error. Moreover, the entry related to the velocity is 1 since the velocity of the wheel does not have to be minimized: on the contrary, it will reach high values, to provide the desired angular momentum. Finally, the gain matrix G_{aug} is obtained through the Matlab function `lqr`, specifying as inputs the matrices $A_{rw,aug}$, $B_{rw,aug}$, Q_{rw}, and R_{rw}.

With this controller, it is possible to close the loop formed by the reference and system module, and so simulate the response of the system. Fig. 3.21 shows the comparison between the reference wheels velocity $\boldsymbol{\omega}_{w,ref}$ coming from the reference trajectory generation module and the reaction wheels velocity $\boldsymbol{\omega}_w$ coming from the system module, which simulate the real velocity of the reaction wheels. It must be noticed that this simulation does not take into account any disturbances or any noise coming from the sensors, since it is assumed the perfect knowledge of the full state: the only purpose is to check the validity of the gain matrix obtained from the LQR and to see if the wheels are able to track the velocity profile assigned. As it can be seen from the figure, the tracking is good, as there is no overshoot and the delay is only about half a second.

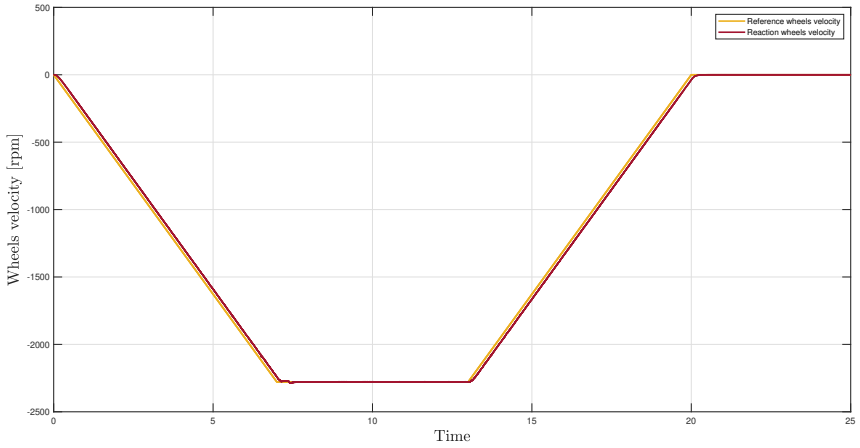


Figure 3.21: Comparison between the reference wheels velocity and the simulated reaction wheels velocity.

Sim controller LQR block

This block implements the controller for the second (outer) loop of the control module: as the first one, also this controller was designed with the LQR technique. However, Euler's equation for the simulator is nonlinear, on the contrary of the equations for the reaction wheels, so that writing the state-space system will be not as simple as before. The method used to write the state-space system is the linearization of Euler's equations, as presented in [28]. The starting point is Eq. (3.42), which can be rewritten in scalar form (along x) as:

$$\dot{\omega}_{sx} = \frac{1}{I_{sxx}}\Big(-(I_{szz} - I_{syy})\omega_{sy}\omega_{sz} - J(A_{11}\dot{\omega}_{w1} + A_{12}\dot{\omega}_{w2} + A_{13}\dot{\omega}_{w3}) + \\ + \omega_{sz}J(A_{21}\omega_{w1} + A_{22}\omega_{w2} + A_{23}\omega_{w3}) - \omega_{sy}J(A_{31}\omega_{w1} + A_{32}\omega_{w2} + A_{33}\omega_{w3})\Big) \quad (3.69)$$

Along the other directions, the equations are very similar, so they will be omitted for the sake of clarity. The equations, now, need to be linearized. To do so, the partial derivatives with respect to each variable are evaluated and calculated at the equilibrium point. Then, each derivative must be multiplied by its respective variable, so that at the end every variable appears with, at most, a linear dependence. This procedure is summarized in Eq. (3.70), with nine partial derivatives.

$$\dot{\omega}_{sx,lin} = \frac{\partial\dot{\omega}_{sx}}{\partial\omega_{sx}}\bigg|_{eq}\omega_{sx} + \frac{\partial\dot{\omega}_{sx}}{\partial\omega_{sy}}\bigg|_{eq}\omega_{sy} + \frac{\partial\dot{\omega}_{sx}}{\partial\omega_{sz}}\bigg|_{eq}\omega_{sz} + \frac{\partial\dot{\omega}_{sx}}{\partial\omega_{w1}}\bigg|_{eq}\omega_{w1} + \\ + \frac{\partial\dot{\omega}_{sx}}{\partial\omega_{w2}}\bigg|_{eq}\omega_{w2} + \frac{\partial\dot{\omega}_{sx}}{\partial\omega_{w3}}\bigg|_{eq}\omega_{w3} + \frac{\partial\dot{\omega}_{sx}}{\partial\dot{\omega}_{w1}}\bigg|_{eq}\dot{\omega}_{w1} + \frac{\partial\dot{\omega}_{sx}}{\partial\dot{\omega}_{w2}}\bigg|_{eq}\dot{\omega}_{w2} + \frac{\partial\dot{\omega}_{sx}}{\partial\dot{\omega}_{w3}}\bigg|_{eq}\dot{\omega}_{w3} \quad (3.70)$$

Using these partial derivatives, the linearized equation of motion can be found about any equilibrium point. Considering a generic equilibrium point $\bar{\mathbf{x}}$, the nine partial derivatives are written as:

$$X_1 = \frac{\partial\dot{\omega}_{sx}}{\partial\omega_{sx}}\bigg|_{\bar{x}} = 0 \quad (3.71)$$

$$X_2 = \left.\frac{\partial \dot{\omega}_{sx}}{\partial \omega_{sy}}\right|_{\bar{x}} = \frac{1}{I_{sxx}}(-(I_{sz} - I_{sy})\bar{\omega}_{sz} - J(A_{31}\bar{\omega}_{w1} + A_{32}\bar{\omega}_{w2} + A_{33}\bar{\omega}_{w3})) \tag{3.72}$$

$$X_3 = \left.\frac{\partial \dot{\omega}_{sx}}{\partial \omega_{sz}}\right|_{\bar{x}} = \frac{1}{I_{sxx}}(-(I_{sz} - I_{sy})\bar{\omega}_{sy} + J(A_{21}\bar{\omega}_{w1} + A_{22}\bar{\omega}_{w2} + A_{23}\bar{\omega}_{w3})) \tag{3.73}$$

$$X_4 = \left.\frac{\partial \dot{\omega}_{sx}}{\partial \omega_{w1}}\right|_{\bar{x}} = \frac{J}{I_{sxx}}(A_{21}\bar{\omega}_{sz} - A_{31}\bar{\omega}_{sy}) \tag{3.74}$$

$$X_5 = \left.\frac{\partial \dot{\omega}_{sx}}{\partial \omega_{w2}}\right|_{\bar{x}} = \frac{J}{I_{sxx}}(A_{22}\bar{\omega}_{sz} - A_{32}\bar{\omega}_{sy}) \tag{3.75}$$

$$X_6 = \left.\frac{\partial \dot{\omega}_{sx}}{\partial \omega_{w3}}\right|_{\bar{x}} = \frac{J}{I_{sxx}}(A_{23}\bar{\omega}_{sz} - A_{33}\bar{\omega}_{sy}) \tag{3.76}$$

$$X_7 = \left.\frac{\partial \dot{\omega}_{sx}}{\partial \dot{\omega}_{w1}}\right|_{\bar{x}} = -\frac{JA_{11}}{I_{sxx}} \tag{3.77}$$

$$X_8 = \left.\frac{\partial \dot{\omega}_{sx}}{\partial \dot{\omega}_{w2}}\right|_{\bar{x}} = -\frac{JA_{12}}{I_{sxx}} \tag{3.78}$$

$$X_9 = \left.\frac{\partial \dot{\omega}_{sx}}{\partial \dot{\omega}_{w3}}\right|_{\bar{x}} = -\frac{JA_{13}}{I_{sxx}} \tag{3.79}$$

To clarify the discussion, the partial derivatives of $\dot{\omega}_{sx}$ are called $X1$, $X2$, ... $X9$. In the same way, the derivatives of $\dot{\omega}_{sy}$ and $\dot{\omega}_{sz}$ are represented respectively by $Y1$, $Y2$, ... $Y9$ and $Z1$, $Z2$, ... $Z9$.
With all the equations linearized, it is possible to define the state-space representation of the simulator. The state $\mathbf{x}_{sim}$ and the input $\mathbf{u}_{sim}$ can be defined in many different ways: the one chosen for this model is the following

$$\mathbf{x}_{sim} = \begin{Bmatrix} \phi \\ \theta \\ \psi \\ \omega_{sx} \\ \omega_{sy} \\ \omega_{sz} \end{Bmatrix} \qquad \mathbf{u}_{sim} = \begin{Bmatrix} \omega_{w1} \\ \omega_{w2} \\ \omega_{w3} \\ \dot{\omega}_{w1} \\ \dot{\omega}_{w2} \\ \dot{\omega}_{w3} \end{Bmatrix} \tag{3.80}$$

where the first three entries represent the Euler's angles: in fact, linearizing the equations around an equilibrium point, it is possible to write the angular velocities as the derivatives of Euler's angles, without the need of the kinematics equations. Once that the state and the input are defined, it is possible to write the complete system as:

$$\dot{\mathbf{x}}_{sim} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & X1 & X2 & X3 \\ 0 & 0 & 0 & Y1 & Y2 & Y3 \\ 0 & 0 & 0 & Z1 & Z2 & Z3 \end{bmatrix}}_{A_{sim}} \mathbf{x}_{sim} + \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ X4 & X5 & X6 & X7 & X8 & X9 \\ Y4 & Y5 & Y6 & Y7 & Y8 & Y9 \\ Z4 & Z5 & Z6 & Z7 & Z8 & Z9 \end{bmatrix}}_{B_{sim}} \mathbf{u}_{sim} \tag{3.81}$$

$$\mathbf{y}_{sim} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{C_{sim}} \mathbf{x}_{sim} \tag{3.82}$$

As it can be seen, the first three columns of the state matrix A_{sim} are null because are related to the first three entries of $\mathbf{x}_{sim}$, which are the Euler's angles that do not appear in Eq. (3.70). The input matrix B_{sim} is not populated in the upper half because the input has no influence about the angular velocities, but only about the angular accelerations $\dot{\boldsymbol{\omega}}_s$. At last, the output matrix C_{sim} is simply the identity matrix because the final output is exactly the state.
Usually, when an equation is linearized, the equilibrium point is chosen as a particular point like zero (null state). In this case, the simulator will rotate from 0 to 270 degrees, and also the velocity will change and reach values far from zero: given this large range of values, linearizing around a single equilibrium point would stretch too much the assumptions of the linearizing process, since this approximation is most effective near the equilibrium point. For this reason, it was decided to take as equilibrium point $\bar{\mathbf{x}}$ the actual state $\mathbf{x}_{sim}$. Since the state change at every iteration, this method would need to evaluate the gain matrix G at every iteration. In order to reduce the computational effort, it was selected a sample time of $0.1s$ for the sim controller LQR block, so that the equilibrium point is updated every 0.1 seconds. In this way, a state-space model based on the linearized Euler's equations was developed and can now be used to design the LQR controller.
As already done for the motor controller LQR block, the gain matrix G_{sim} is obtained through the Matlab function `lqr`, specifying as inputs the matrices A_{sim}, B_{sim}, Q_{sim}, and R_{sim}, where the weighting matrices are defined as:

$$Q_{sim} = \begin{bmatrix} 15 & 0 & 0 & 0 & 0 & 0 \\ 0 & 15 & 0 & 0 & 0 & 0 \\ 0 & 0 & 15 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix} \qquad R_{sim} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix} \tag{3.83}$$

As it can be seen, the higher weights are given to the Q matrix, while the R matrix, related to the control effort, has lower values. Now, it must be noticed that the state $\mathbf{x}_{sim}$ fed to the LQR is not composed of the Euler angles and simulator velocities coming from the system module, but rather it is composed of the difference between the reference quantities and the real quantities from the system module, as it can be checked in Fig. 3.8. In this way, the gain is multiplied by the state error rather than the state itself. This produces input commands relating to the state error regardless of the reference state. These inputs due to error are eventually added to the reference wheels velocity from the reference trajectory generation module to produce the total reference wheel velocity that will be used by the motor controller LQR.

3.2.5 Simulation results

At this point, all the modules composing the Simulink model of the system were presented and it is possible to perform a simulation to check the performances of the reaction wheels in

controlling the simulator platform. Moreover, the Simulink model will be useful also in the following chapters, because it can be exploited to select the right components and to compare the results from the actual implementation. As already mentioned, the data for the simulation are uploaded thanks to two Matlab files which are pre-uploaded when Simulink starts, and they are `input_preload.m` and `data_preload.m`. The first one contains the data related to the maneuver under study: for this simulation, the maneuver consists of a rotation around the z-axis of 270 degrees in a total time of 40 seconds, with the parameter n equal to 0.35. The total time was doubled with respect to the initial time of 20 seconds because the simulator inertia estimate was nearly doubled. The inertia value, as also the other data for the simulation, are contained in `data_preload.m`: in particular, the simulator inertia was estimated from the CAD model (shown in the next chapter) as:

$$I_s = \begin{bmatrix} 0.1838 & 0.0114 & 0.0029 \\ 0.0114 & 0.2234 & 0.0072 \\ 0.0029 & 0.0072 & 0.1747 \end{bmatrix} \text{ kg m}^2 \tag{3.84}$$

Concerning the inertia of the reaction wheel, it was set as $J = 1.74 \,\text{x}\, 10^{-4} + 1.35 \,\text{x}\, 10^{-5} \text{kg m}^2$: the first term is the inertia of the flywheel as estimated from the CAD model, while the second term represents the inertia of the motor that will be attached to the wheel. This value, as all the other characteristics of the motor, are taken from the datasheet of the motor selected for the prototype implementation, which will be described later on.

Table 3.2: Motor data

Characteristic	**Symbol**	**Value**
Back-EMF constant	K_e	$0.0335 \, \frac{V}{rad/s}$
Torque constant	K_t	$0.0335 \, \frac{Nm}{A}$
Inductance	L	$4 \,\text{x}\, 10^{-4} \, H$
Resistance	R	$0.88 \, \Omega$
Friction coefficient	b	$1.05 \,\text{x}\, 10^{-6} \, \frac{Nm}{rad/s}$
Motor poles	p	16

Table 3.2 summarizes the motor data. The only data not available in the datasheet is the friction coefficient b, which was selected checking the value of similar motors. The last data to set is the A matrix related to the position of the reaction wheels: for this simulation, the wheels are assumed to be in a pyramidal configuration, so that the matrix can be written as

$$A = \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & \sqrt{\frac{2}{3}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{bmatrix} \tag{3.85}$$

Concerning the solver, it was selected a fixed-step solver, in particular ode4, with a step size of $1 \,\text{x}\, 10^{-4}$ seconds, which was quicker with respect to a variable-step solver: the better performance of the fixed-step solver are due to the fact that some modules of the model (sensors and control modules) are simulated in a discrete way, with a fixed time step of 0.01 seconds. With these parameters, the simulation outputted the following results. Fig. 3.22 shows the Euler angles of the simulator: the behavior tracks well the reference trajectory (Fig. 3.17a) and the

final value reached is 272.8 degrees. So, the simulator reaches its target with a steady-state error which is about 1 % of the nominal final value. The control performances are good in terms of overshoot, which is absent, and settling time, while the steady-state error is the only performance which could be improved.

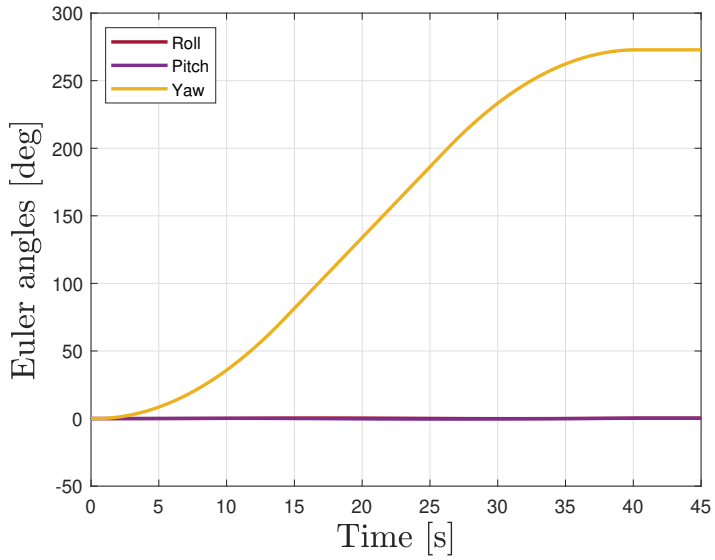


Figure 3.22: Euler angles of the simulator.

Then, Fig. 3.23 displays the speed of the three wheels. As it can be seen, their profiles are similar, which is in accordance to the reference velocities to be tracked. At the beginning, there is a certain amount of oscillation, after which the behavior is more stable. The speed profile follows the one of Fig. 3.19, and the speed during the constant phase is around 1150 rpm, which is not comparable to the one of Fig. 3.19 since the total time and the simulator inertia are different. In this case, a small overshoot can be observed before reaching the constant phase value, around 0.2 %. Moreover, the speed profile has a delay of about half second with respect to the reference wheel velocity.

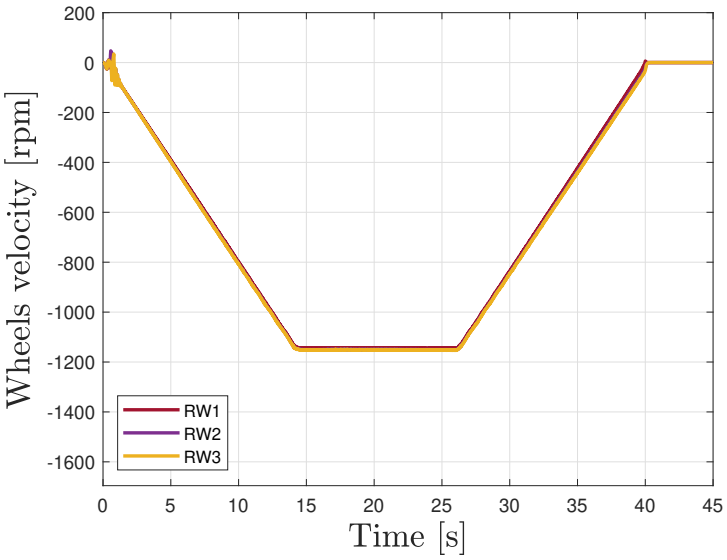


Figure 3.23: Reaction wheels velocities.

The last result, shown in Fig. 3.24, represents the power consumption of a single wheel, which will be similar to the other two wheels. The electrical power required by the motor is related to the current flowing in the windings, which in turn depends on the speed of the motor. In fact, the behavior is the same of the reaction wheels velocities: in this case, the maximum value is about 2.62 W. As it will be explained later, the power consumption represents an important parameter for the selection of the motor and should not exceed 20 W. Considering all the three reaction wheels, the total power required by the actuators system will be about 7.86 W, which is well below the limit imposed.

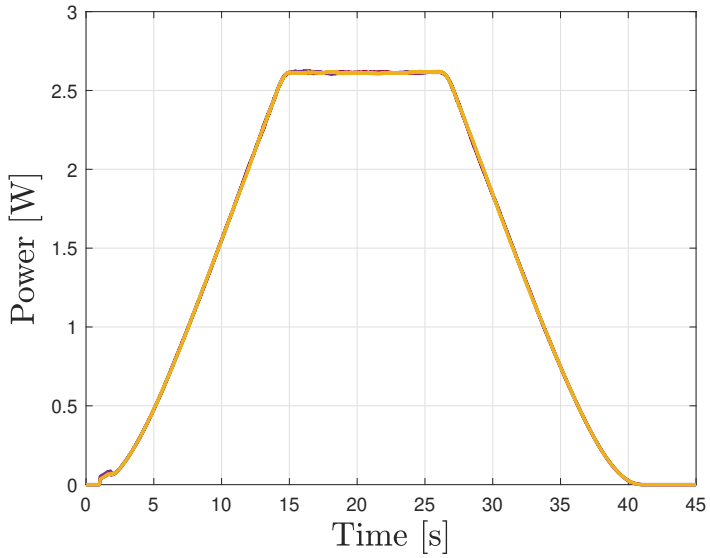


Figure 3.24: Power consumption of a single wheel.

Chapter 4

Prototype components and implementation

This chapter deals with the hardware implementation of the model described before: the goal is to have a working system that is able to verify and validate the results obtained in the previous chapter. For this reason, the chapter will describe the phases related to the choice of the components and their integration on the simulator.
The model described earlier refers to a complete attitude control system, composed of three reaction wheels, capable of controlling the simulator in the three-axis. Anyway, since the goal is to build an hardware that is able to validate the numerical results, it was decided to first develop a prototype of a single wheel assembly. In this way, it is possible to check the validity of the model with minimal effort in terms of costs and time. Then, if the prototype results to be effective, it will be easy to use the same hardware also for the other two wheels. Since the maneuver under study is related to a rotation around the z-axis, it was decided to develop for the first prototype a reaction wheel along the same axis.
The chapter will deal with the components choice, their procurement, and their actual implementation on the simulator. The first thing will be the choice of the motor for the reaction wheel assembly. After that, the structural components designed to connect the motor, the wheel, and the simulator will be described, before of the electronic components. Eventually, the final implementation of all the components on the simulator will be described.

4.1 Brushless DC motor

As stated before, the first component to be chosen was the brushless DC motor. This represents the main component of the wheel assembly, since it must be able to rotate the wheel at the required velocity and with the desired accuracy. Moreover, it is mandatory to know the dimensions of the motor to design all the structural components. At the same time, also the electronic components will depend on the choice of the motor. For all these reasons, the motor must be selected first.
In order to choose a suitable motor for this work, its main characteristics should be defined: first of all, given the results presented in chapter 3, the motor should be able to reach at least 2500 rpm. This value was then increased to 3000 rpm, since sometimes the values shown in the datasheets contain errors up to 10 % of the nominal value. In any case, the higher the velocity that can be reached, the better. Another characteristic taken into account was the size of the motor: since the final actuator will be composed of three wheels to be mounted on the simulator, it would be good to have a compact design for the wheel assembly. For this reason, the choice of the motor took into account flat BLDC motors. Another consideration is related to the power input of the motor, which will be mounted on the simulator and rotate with it: this makes it impossible to have cables powering up the motor, since they would interfere with the rotation

of the platform. So, the power needs to be generated on the simulator itself, and this poses limitations in terms of power consumption. Considering the presence of additional systems on the platform, an upper limit of 15 to 20 W was selected. Moreover, it would be preferable for the motor to be able to reach the target velocity without overheating and reliably. As a last remark, it is important to note that BLDC motors are available in two different configurations: with and without Hall sensors. A motor without Hall sensors needs external devices, such as encoders, to read its velocity. These devices are usually more precise, but also more bulky and complicated with respect to the Hall sensors already mounted on the motor. For this project, it was decided to select a BLDC motor with built-in Hall sensors because their precision is enough and they offer an advantage in terms of compactness of the overall assembly.
Taking into account all of the above, the first motor to be selected was the EC 45 flat (part number 339276) by maxon motor [1], a reliable company specialized in electrical motors. This motor is a flat BLDC motor with an external diameter of 50 mm, rated power of 12 W, and it has built-in Hall sensors. The specifications of the motor are summarized in Table 4.1: as it can be seen, the power required is less than the maximum allowable and the nominal speed is more than enough. Moreover, the currents are smaller than 1 A and the motor is extremely compact, with a depth of less than 30 mm and a total weight of 57 g. This motor represents one of the tiniest motors available online that fulfills all the needs specified above. Unfortunately, there was no motor left in stock and the company said that a new motor would have to be assembled and then shipped, for a total overhead time of several months. For this reason, the EC 45 flat motor was discarded.

Table 4.1: Comparison between motors specifications.

	EC 45 flat maxon	**DF45M024053-A2 Nanotec**
Rated power [W]	12	50
Nominal voltage [V]	24	24
Speed no load/nominal [rpm]	7310/4390	6700/5260
Current no load/nominal [A]	0.0476/0.766	<0.4/2.36
Nominal torque [Ncm]	2.7	8.4
Weight [g]	57	120

The final choice fell on a motor by Nanotec, the DF45M024053-A2 [2], which was immediately available. As the former, also this motor has built-in Hall sensors and it is shown in Fig. 4.1,while its specifications are shown in Table 4.1: it has the same nominal voltage of the motor by maxon, but it is more powerful, having higher nominal values for torque, current, and speed, as also the weight. Concerning the dimensions, it is thicker but it has an external diameter of 43 mm. The only out-of-scope parameter is the power, since the motor on the platform cannot be fed by 50 W. Anyway, this motor was selected with the idea of using it underpowered: this means that it will not be able to reach its maximum specifications, but only lower values. To understand if the motor can be used in this way, it is necessary to check that the motor can reach the speed required by the maneuver while being powered only with a maximum of 20 W. To do so, the Simulink model was used, entering the motor specifications from [3].

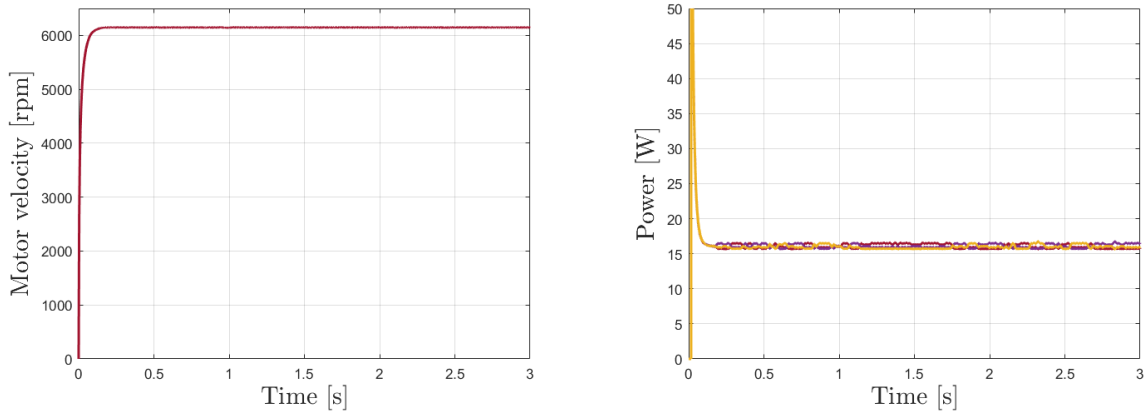
[1] maxon motor website, `https://www.maxongroup.com/maxon/view/product/339276`

[2] Nanotec website, `https://en.nanotec.com/products/1789-df45m024053-a2-brushless-dc-motor-with-connecting-wires`

[3] EC 45 flat datasheet, `https://en.nanotec.com/fileadmin/files/Datenblaetter/BLDC/DF45/DF45M024053-A2.pdf`

Figure 4.1: DF45M024053-A2 motor by Nanotec.

At first, only a single reaction wheel block was simulated, like the one of Fig. 3.10. The voltage was set to its nominal value of 24 V, and no additional inertia sources were added apart from the one of the motor itself. The friction coefficient b was taken as 1.05×10^{-6}. All the other values were taken from the datasheet of the motor. Fig. 4.2 shows the results obtained from the simulation. On the left, the speed of the motor reaches a steady-state value of about 6150 rpm, which is in accordance with the value of the zero-load rated speed on the datasheet, 6700 rpm $\pm 10\%$. Instead, on the right, the power consumption has an initial spike, after which it stabilizes around 16 W. This value is lower than the imposed limit of 20 W: moreover, it is related to the motor running at full speed, while in our case the speed will be less than half, and so also the power will be lower. Anyway, it is still possible to run the motor at full speed with less than 20 W.

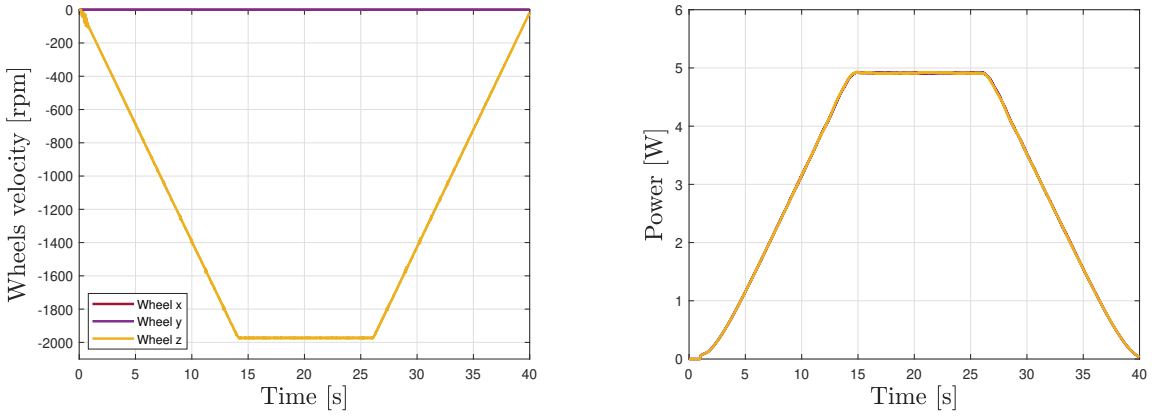


(a) Velocity **(b)** Power

Figure 4.2: Velocity and power consumption with no load.

These results show that the reaction wheels block does a good job in simulating the dynamics of the motor, and can be used in order to check the power consumption during the slew maneuver. To this aim, another simulation was run, in this case involving the overall Simulink model presented in Fig. 3.8. Anyway, as already stated at the beginning of the chapter, the prototype that will be implemented includes only a single wheel, mounted along the z-axis. So, the Simulink model will need some modifications to take into account the presence of a single wheel instead of three. Actually, analyzing a single wheel along the z-axis corresponds to analyzing a set of three

reaction wheels in a cartesian configuration (one wheel for each cartesian axis). In fact, with this configuration, a maneuver that involves a rotation around a single axis will only activate a single wheel. This means that it is possible to emulate the effect of the single wheel prototype with the same Simulink model described before: the only difference is related to the A matrix inside Eq. (3.42). Using the identity matrix of dimension three, each wheel is related to a cartesian axis, and a rotation along z will activate only the third wheel. After changing the A matrix, a simulation was performed and the results are shown in Fig. 4.3.



(a) Velocity **(b)** Power

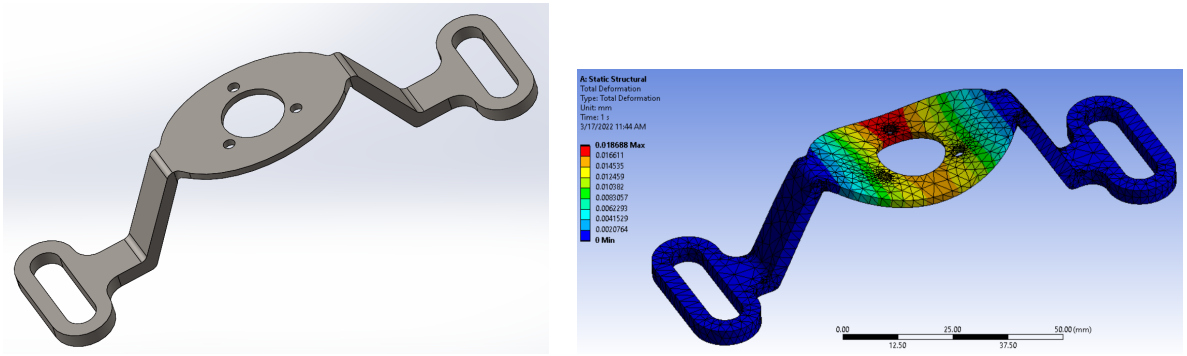
Figure 4.3: Velocity and power consumption during the slew maneuver.

As it can be seen, the wheels related to the x and y axes remain still, while the wheel along the z-axis reaches a velocity of about 2000 rpm. This value is greater than the velocity reached in Fig. 3.23, but this was expected since a single wheel gives a smaller contribution in terms of angular momentum with respect to three wheels in pyramidal configuration. More importantly, on the right, it is possible to see that the power consumption behavior is strictly related to the velocity, and reaches values well below the maximum limit of 20 W. Even considering some degree of error in the model, it is probable that the power in Fig. 4.3b is overestimated, as explained before. So, taking everything into account, it was decided to definitely select the DF45M024053-A2 motor by Nanotec because, even though the rated power is listed at 50 W, it was checked that for our maneuver the power will not surpass the imposed limit of 20 W.

4.2 Structural components

This section describes all the structural components that will form the reaction wheel assembly: this term refers to the assembly composed of the reaction wheel (Fig. 3.6b), the motor (Fig. 4.1), a flange that connects the wheel to the shaft of the motor, and a structure to fix all these components on the platform. While the first two have been already presented, the latter two are described in the following, starting from the structure between the motor and the platform.
The starting point is the motor: from Fig. 4.1, is easy to recognize the stator of the motor, which is the part in front, thinner and shinier, to which are attached the cables, and the rotor, made up of the shaft, but also of the rear part. The structure will connect the platform to the stator, where are visible three threaded holes, designed just for this purpose. Then, the wheel will be fixed to the shaft, in order to rotate at the same velocity. This configuration could be the source of potential vibrations, in particular if the shaft is placed along a direction perpendicular to the gravity direction: in this case, in fact, the shaft could be represented as a clamped beam with a load on the free end. This could generate an important bending moment, and so an eccentricity along the nominal direction of rotation, which, especially at high speeds, could lead

to not negligible vibrations. To mitigate this problem, a bearing could be placed just before the wheel, in order to decrease the displacement of the shaft at the free end. Anyway, since in this first prototype the shaft will be along the gravity direction, it was decided not to add additional bearings. In this case, indeed, the weight of the wheel will act as a compression force for the shaft, exploiting the axial rigidity of the latter, which, for a circular cross-section, is higher than the bending rigidity. Nevertheless, this is something to take into account when designing the full set of three wheels.
Having said so, the structure will be the only link between the wheel assembly and the platform, and so it needs to guarantee a high enough rigidity, especially at high speeds, when a small eccentricity could lead to high vibrations. Moreover, it is important for the structure, and for the overall assembly, to be as low as possible. In fact, the assembly will be placed on top of the platform, causing its overall center of gravity to move up. As already explained in chapter 2, it is important for the stability of the platform that its center of gravity is below the center of rotation of the air bearing. So, the lower the center of gravity, the better. Considering these requirements, different configurations of the structure were designed and tested through finite element (FE) simulations, performed using Ansys Workbench [4]. After some iterations, it was developed the final structure, shown in Fig. 4.4a. As it can be seen, on the sides there are holes that will be used to fix the structure to the platform through bolts, while in the center there is a flat elliptic portion raised with respect to the sides to house the motor, with three holes in correspondence of the three threaded holes of the stator of the motor. The sides and the central portion are linked by two arms.



(a) CAD model of the structure **(b)** Static analysis - total deformation

Figure 4.4: Structure connecting the motor to the platform.

The first simulation carried out with Ansys was a static analysis of the structure. The geometry was imported from Solidworks, where the selected material was aluminum, and the mesh was generated automatically. The Adaptive Sizing was used to generate a finer mesh where the curvature of the geometry is smaller. A fixed support was selected for the holes on the sides, where the structure is supposed to be fixed by the bolts, and a force orthogonal to the central portion was applied on the three central holes, to emulate the forces shared between the motor and the structure. The value of the force was of 50 N, corresponding roughly to the weight of 5 kg of mass, way higher than the mass of the motor and the wheel. The results in terms of total deformation are shown in Fig. 4.4b: the deformations occur almost all in the central portion, with the greatest values in correspondence of the holes. The maximum deformation is lower than 0.02 mm, which is an acceptable value. So, the weight of the assembly should not be a problem for the structure.
Anyway, during the maneuver, the wheel and the motor will be rotating at speeds up to 3000 rpm, which corresponds to a frequency of 50 Hz. To check that this frequency will not resonate with the natural frequencies of the structure, a modal analysis was carried out. The geometry

[4]Ansys Workbench website, `https://www.ansys.com/it-it/products/ansys-workbench`

and the mesh were the same of the static analysis, as also the fixed support, while the force is not required for the modal analysis. The solver was configured to find the first twelve modes. Fig. 4.5 shows the first vibration mode, corresponding to a first natural frequency of 2013.6 Hz. Table 4.2, instead, lists the first six natural frequencies of the structure. As it can be seen, the first frequency is way higher than the frequency of the motor.

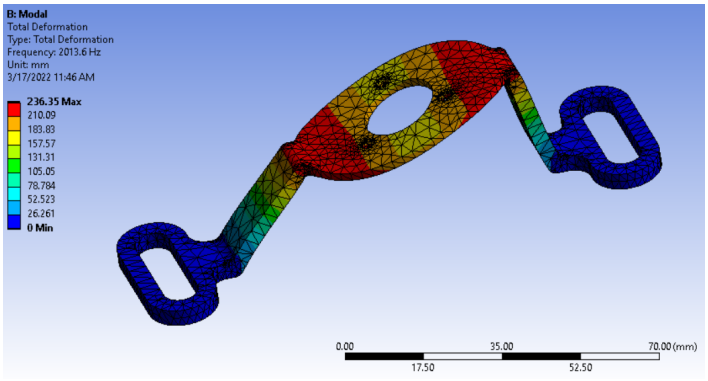
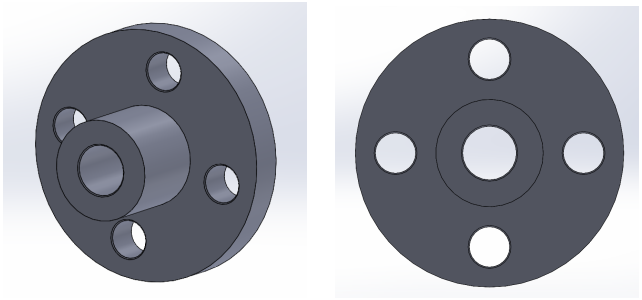


Figure 4.5: Modal analysis - First natural frequency

Table 4.2: Natural frequencies

Mode	Frequency [Hz]
1	2013.6
2	2224.1
3	2886.3
4	3788
5	3926.8
6	8193.4

Up to now, it was assumed that the shaft of the motor and the wheel were connected together. There are different ways to accomplish a shaft-hub connection: the classical method involves a keyway on the shaft, where it must be inserted a key in order to transmit the stress between the two pieces, or a splined shaft connection. Anyway, these solutions are usually applied with bigger shafts, where the presence of the keyway will not cause problems to the structural integrity of the shaft. In our case, the diameter of the shaft is only 4 mm, and the producer does not offer the possibility of modifications on the shaft. So, after hearing some advice from the engineers of maxon motor italia s.r.l., it was decided to realize the connection with press-fit, which consists of linking a hub with a nominal diameter smaller than the nominal diameter of the shaft through interference. Anyway, in order to simplify the assembly and avoid damage to the wheel, it was designed a flange to be connected to the shaft with press-fit, and then to the wheel thanks to four bolts. Some views of the CAD model of the flange are reported in Fig. 4.6. Particular attention should be paid regarding the central hub of the flange, whose cylindricity should be checked to avoid misalignment with the shaft of the motor, and regarding the flatness and perpendicularity of the surface in contact with the wheel.



(a) 3D view **(b)** Bottom view

Figure 4.6: Views of the flange.

All these components linked together will form the reaction wheel assembly, which is shown in Fig. 4.7. All the components, with the exception of the motor, will be produced inside the laboratories of Politecnico di Milano. The wheel and the flange will be made of aluminum 6061 and produced with CNC machining, giving their axial symmetry. On the other hand, the

structure is more complicated to be produced in the same way, so it was decided to use 3D printing.

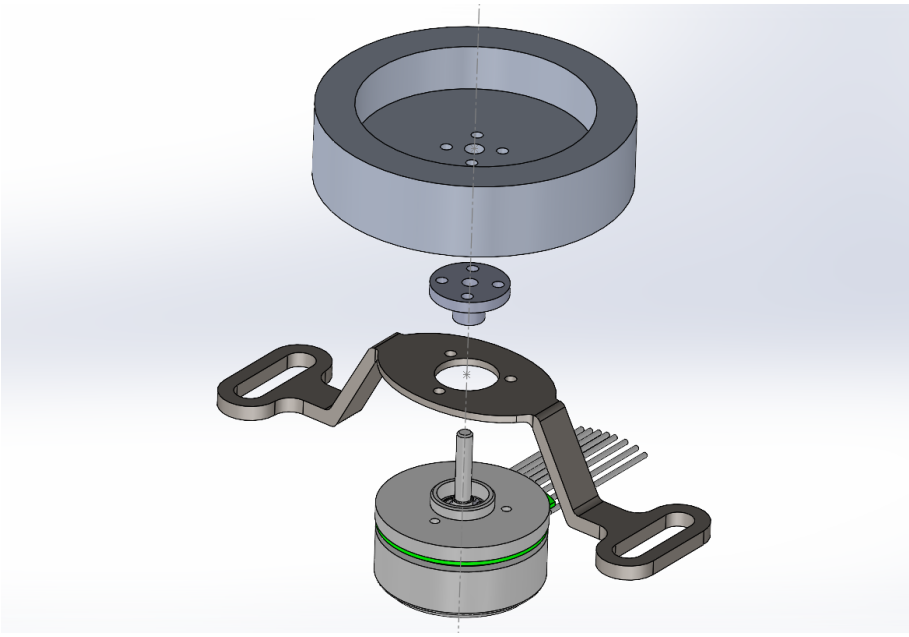


Figure 4.7: Exploded view of the wheel assembly.

4.3 Electronics

This section deals with the description of the electronic components that will be needed to operate the reaction wheel assembly, in particular to control the motor and to elaborate the data from the sensors in order to produce the proper control input for the motor.
The first component to be presented is the microcontroller, which represents the brain of the system: in fact, it will be in charge of connecting all the other devices, reading the measurements from the sensors, processing the data and sending the inputs to the driver of the motor. The microcontroller used for the first prototype implementation is an ESP32-DevKitC [5], shown in Fig. 4.8: it is a development board produced by Espressif, based on the ESP32-WROOM-32E module. The choice fell on this board because it was already available in the laboratory and meets all the requirements for this project, in particular the ability of generating Pulse-width modulation (PWM) signals and the possibility of communicating wirelessly with Simulink (installed on an external computer). As stated in the datasheet, at the core of the module there is the ESP32-D0WD-V3 chip, which has two CPU cores that can be individually controlled, and the CPU clock frequency is adjustable from 80 MHz to 240 MHz. One of its main feature is the connectivity, with the PCB antenna enabling Bluetooth and 2.4 GHz Wi-Fi functions, in addition to a rich set of peripherals, ranging from capacitive touch sensors, Hall sensors, SD card interface, Ethernet, high-speed SPI, UART, I2S and I2C. The board has a total of 34 General Purpose Input/Output (GPIO) pins, 28 of which can generate PWM signals.
The microcontroller is able to execute the operations that are uploaded on it from the computer: this can be performed through the ARDUINO IDE, connecting the board with a micro-USB cable or even over the air (OTA), which means without cables. Anyway, the first configuration must be done with the cable. Once that the code is uploaded, the board only needs its power supply to be operative, and can exchange data with the sensors and with the computer while

[5]Espressif website, `https://www.espressif.com/en/products/devkits/esp32-devkitc`

being mounted on the simulator, thanks to TCP/IP protocols. This is fundamental in order for the board to work without cable connections, that would interfere with the rotation of the simulator.

Table 4.3: ESP32 specifications

Connectivity	Memory	Power supply
2.4 GHz Wifi Bluetooth Micro-USB	4 MB Flash	3.0 $\sim$ 3.6 V

Figure 4.8: Microcontroller ESP32

The second most important component, after the microcontroller, is the driver of the motor. This component is in charge of setting the right voltage difference between the poles of the motor, enabling the currents to flow in the three phases of the rotor's winding, controlling the motion of the motor. The most common type of motor drivers is based on H-bridge circuits, that are able to switch the polarity of the voltage applied, allowing the motor to run forwards and backwards. So, it is possible to build your own driver starting from small electrical components like H-bridges and MOSFETs or, in alternative, several boards are available on the market. In our case, since the complete development of the electrical circuit for the driver is out of scope for this work, it was decided to select a ready to use board, and the choice fell on the Brushless 12 Click [6] from Mikroe, which is shown in Fig. 4.9. Different driver boards are available on Mikroe website, mainly subdivided into two categories: sensorless drivers and Hall sensors drivers. The first category contains drivers that do not use signals from the Hall sensors of the motor, and rely only on internal algorithms to govern the current flowing in the windings. On the other hand, the second category of drivers checks the signal from the Hall sensors to understand when and how the currents need to be adjusted. The Brushless 12 Click belongs to the second category, which assures a better precision in the control of the motor. Another characteristic to verify when selecting a motor driver is the voltage supply: some boards have the same power supply both for the board itself and for driving the motor. This solution is adopted only with small motors, running at maximum at a voltage difference of 5 V. The Brushless 12 Click was selected also because the voltage difference for the motor can be supplied by an external source in the range from 8 to 48 V, since the DF45M024053-A2 motor has a nominal voltage of 24 V.

Table 4.4: Brushless 12 click specifications

Hall sensors decoding	Yes
Logic voltage	5 V
Max output current	2.8 A
External voltage supply	8 $\sim$ 48 V

Figure 4.9: Motor driver Brushless 12 click

The Brushless 12 Click board is based on the L6235 motor driver, which includes a 3-phase DMOS bridge, an OFF-TIME PWM current controller, and the decoding logic for single-ended Hall sensors that generate the required sequence for the power stage, in addition to other features

[6] Mikroe website, `https://www.mikroe.com/brushless-12-click`

for safe operation and flexibility, like the Over Current Detection (OCD) that allows protection against short circuits. By default, the motor will modify its torque to maintain the speed at a constant value: this is known as Speed mode. At the same time, the VREF switch allows to select the Torque mode, where the speed will be modified to maintain the torque value. The board can communicate with the microcontroller thanks to several GPIO pins on the bottom side, while on the top side there are five GPIO pins for the signals coming from the Hall sensors of the motor. Moreover, additional connectors are present, two for the external voltage supply and three to be connected to the three phases of the motor. Table 4.4 summarizes the main specifications of the motor driver. As it can be seen, the maximum output current that can be provided is higher than the nominal current of the DF45M024053-A2 motor.
After the description of the microcontroller and of the motor driver, it is possible to talk about the power supply of the system. The choice of the hardware for the power supply will take into account the limit stated before of 20 W as maximum input power for the reaction wheel assembly. In particular, the motor will be the most demanding component in terms of power, but also the boards (microcontroller and driver) will need some power. Moreover, the boards operate with a lower voltage difference (5 V) with respect to the motor, which needs 24 V. So, the easiest way would be that of using two different voltage sources, one for the motor and one for the boards. Anyway, it was decided to use the same power supply for all the components because of two main reasons:

- using a single power supply will require only one battery pack, which means fewer components, fewer cables, less weight, less space, and in general a more compact reaction wheel assembly;
- the power required by the boards is way lower than that required by the motor. This would lead to an oversized power supply for the boards, capable of working for many hours. Nonetheless, the power supply of the motor will discharge faster, causing the system to stop.

In the laboratory, there are different power banks already available. In particular, for the first prototype it was used a power bank from Shenzhen Kangda Industrial, shown in Fig. 4.10a. It has a maximum current of 2.4 A from the USB channel and a capacity of 10000 mAh. The output voltage is 5 V, which is perfect for the boards, but not for the motor. For this reason, it was decided to use the step-up converter shown in Fig. 4.10b. This component takes as input a voltage difference between 2 and 24 V and gives as output a voltage difference between 5 and 28 V. Adjusting the potentiometer it is possible to obtain 24 V in output from the 5 V in input given by the power bank. Moreover, the internal circuit of the battery isolates the two power lines, preventing motor oscillations from interfering with the electronics.

(a) Power bank **(b)** Step-up converter

Figure 4.10: Power supply components.

The chip on the converter has a high conversion efficiency, up to 93%, and it has a maximum output current of 2 A. This value is lower than the nominal current of 2.36 A required by the motor. Anyway, that value is referred to a power consumption of 50 W: since we will drive the motor with less than half of that power, the current in the motor will never reach 2 A, otherwise we will exceed the power limit imposed.
The last components to be described are the sensors. The angular velocities and the Euler angles of the platform will be measured by the IMU sensor already presented in Sec. 2.3.2. The angular velocity of the wheel, instead, will be obtained from the Hall sensors equipped on the motor, whose signals will be decoded by the microcontroller. The last sensor used in the reaction wheel assembly is the current sensor shown in Fig. 4.11, which is a ready to use board based on the chip INA219. The module can sense a maximum voltage of 32 V and a maximum current of 2 A, with a resolution adjustable in relation to the maximum current expected and a maximum precision of about 1%. The sensor communicates with the microcontroller through the I2C interface and requires an operating voltage between 3 and 5.5 V.

Figure 4.11: Current sensor.

4.4 Hardware implementation

At this point, all the components constituting the reaction wheel assembly have been introduced, and they have been divided between the structural components and the electronics. Regarding the structural components, it has already been described how they will be integrated with the motor, as it can be seen in Fig. 4.7. In this section, it will be described the connection of all the electronic components to each other and to the motor, and the final implementation of the reaction wheel assembly on the platform of the simulator.
The first step was that of designing the circuit composed of the electronic components: to do so, it was used the program KiCad [7], a free software suite for electronic design automation (EDA). KiCad integrates different tools, offering the possibility of realizing schematic capture, printed circuit board (PCB) layout, manufacturing file viewing, SPICE simulation, and engineering calculation. In this way, it is possible to develop the complete design of an electronic circuit, starting from the first circuit diagram and ending with all the files needed for the manufacturing of the specific PCB. In our case, KiCad was used at the beginning of the electronic design process to develop the first schematic capture of the circuit, in order to understand which were the pins to be connected. In our case, all the electronic components are ready to use boards that needed no additional welding. For this reason, it was decided to test the schematic capture developed in KiCad directly in the laboratory with the real components, a breadboard, and some wires. In

[7]KiCad website, `https://www.kicad.org/`

this way, it was possible to iterate the process, developing the schematic capture in KiCad and testing it with the actual components mounted on the breadboard. After a couple of iterations, it was reached the final version of the schematic capture, shown in Fig. 4.12.

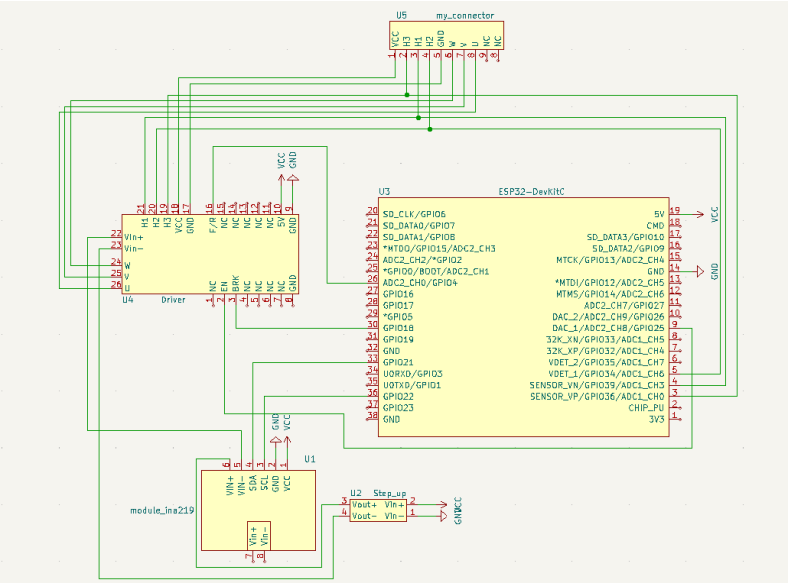


Figure 4.12: Schematic capture of the electronic circuit.

As it can be seen, all the components in the schematic are connected to the flags representing the ground (GND) and the 5 V (VCC) logic voltage, apart from the components called 'my_connector' at the top. This component was added only to simulate the cables coming from the motor, which are three cables for the currents (U, V, and W) and five cables for the Hall effect sensors (three for the signals and two for the power supply). The GND and VCC flags will be connected to the two ends of the power banks, which is not present in the schematic. Anyway, the motor needs 24 V to operate: so, the step-up at the bottom is connected to the power bank to increase the voltage difference, which is then connected to the 'Vin+' and 'Vin-' pins of the driver. In this way, the driver uses this voltage difference to set the right currents that will drive the motor through the cables U, V and W. Moreover, the current sensor was inserted between the driver and the step-up, to get estimates of the voltage difference, current, and electrical power used by the motor and the driver. At the same time, the current sensor send the data to the microcontroller thanks to the typical I2C pins 'SCL' and 'SDA', which transmit respectively the serial clock and the serial data. On the other hand, the signals from the Hall sensors is sent both to the driver, which will use them in its internal algorithm, and to the microcontroller, from which it is possible to have an estimate of the wheel velocity. The last connections are the ones between the pin of the driver and the microcontroller, which are set in accordance to the datasheet of the Brushless 12 Click. As a last remark, it is important to pay attention on the 'F/R' pin of the driver, which should be connected to a pin of the microcontroller capable of sending PWM signals.
The connections just presented were tested using the breadboard, and a working prototype of the electric circuit implementing the schematic capture of Fig. 4.12 is shown in Fig. 4.13. Before connecting everything together, some tests were made to check each component: first, the motor was connected to the driver and microcontroller, with the power supply coming from a voltage generator available in the laboratory, and tested with a constant velocity. Then, the current sensor was tested in a simple circuit with a resistor and a LED, to verify the measurements given by the sensor. Lastly, the step-up converter was connected to the power bank, adjusting the potentiometer until it was reached an output voltage of 24 V, measured by a multimeter.

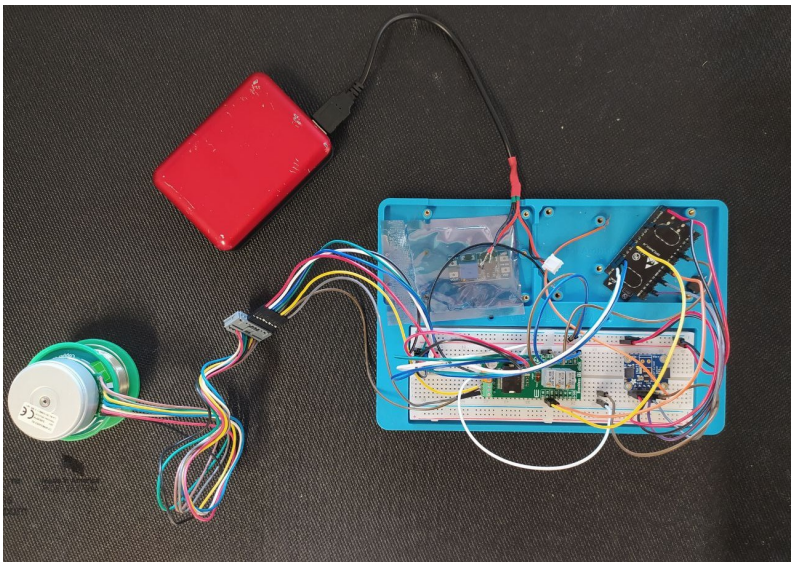
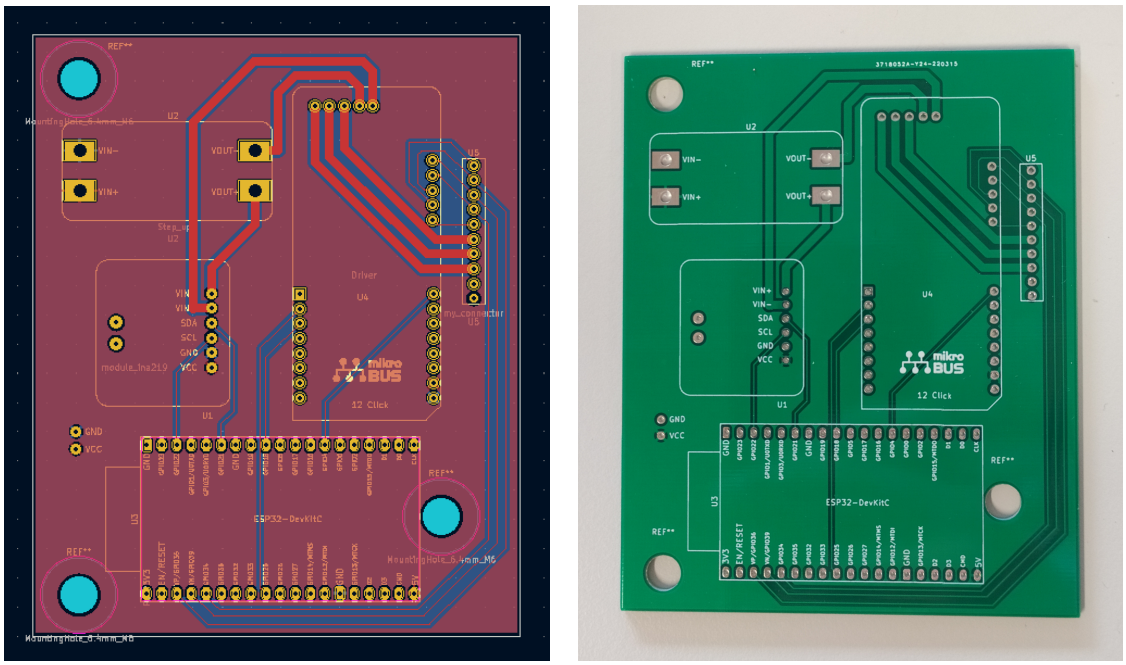


Figure 4.13: Prototype of the electric circuit.

At this point, all the components and their connections were checked and it was verified that everything was working as designed. Anyway, this prototype based on a breadboard has some disadvantages. The first is related to the breadboard, which has relatively large parasitic capacitance, high inductance in some connections, and relatively high contact resistance. This makes the breadboards a good alternative for fast prototyping and verification, but not ideal for a reliable implementation. Moreover, the boards and the wires are not soldered, so that the connections are not very stable, and could unplug if mounted on the simulator. In addition, there are a lot of cables, long and twisted, that could create problems when rotating on the platform. For all these reasons, it was decided to design a specific PCB in KiCad, from the scheme of Fig. 4.12.



(a) Scheme of the PCB **(b)** Real PCB

Figure 4.14: Comparison between the PCB scheme in KiCad and the real component.

The PCB scheme is depicted in Fig. 4.14a: here, it is possible to see the footprints of the components, including the connector on the right, that will be connected to the motor cables, and the power supply pins on the left, that are to the other GND and VCC pins through power planes. As it can be noted, the traces related to the current flowing in the motor wires are wider than the standard. The width was calculated online to allow a current of 2 A. Then, the PCB was ordered online, and the real board is shown in Fig. 4.14b.
In the end, it is possible to mount all the structural components of the reaction wheel assembly on the platform, as well as the PCB and the power bank. The only cables used for connections will be from the power bank to the PCB, and from the PCB to the motor. A CAD model of the final implementation is represented in Fig. 4.15, which was used to check the space occupied by the components and to place in the correct position the mounting holes on the PCB.

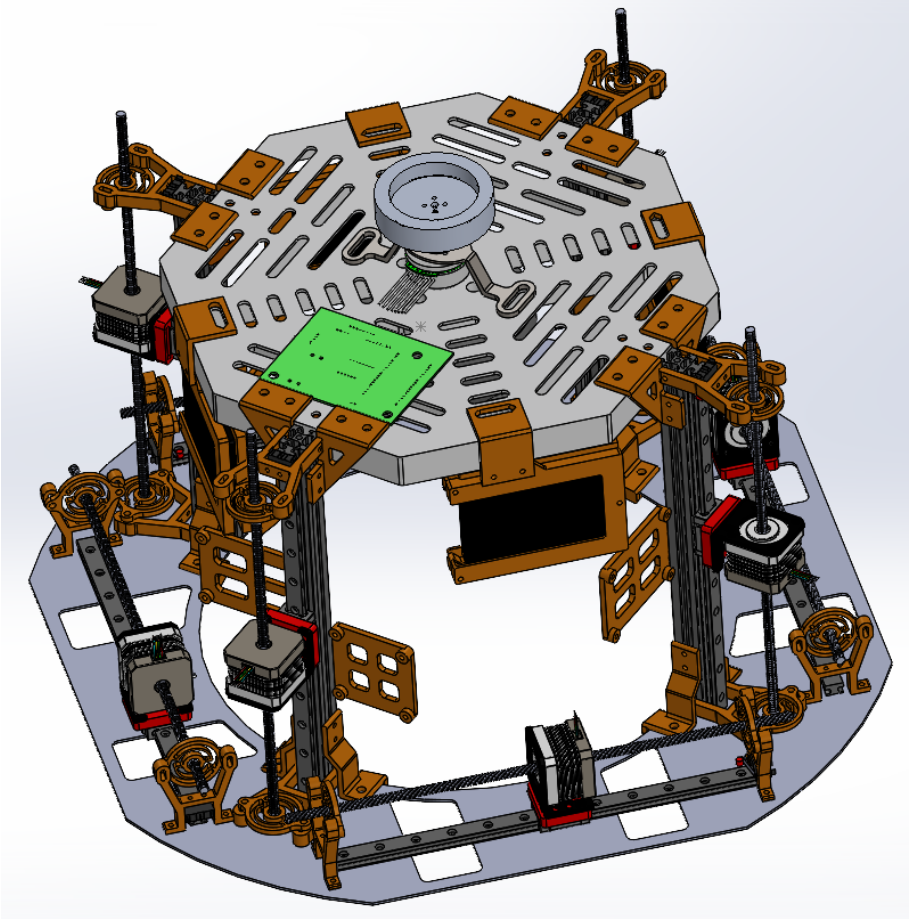


Figure 4.15: CAD representation of the final implementation.

Finally, the actual implementation realized in the laboratory is shown in Fig. 4.16: the electronic board, the battery, and the reaction wheel assembly are mounted on the simulator platform. The only difference with respect to the CAD model is related to the position of the wheel, which is not centered on the platform, but a bit shifted. This was necessary because the four screws that connect the air-bearing to the upper plate, not included in the CAD model, were interfering with the motor. Anyway, this problem was overcome by simply shifting the reaction wheel assembly: in fact, since the wheel and the platform exchange a moment between them, it is important only that the wheel axis is parallel to the z-axis of the platform. In other words, Euler's equations do not require that the axes are coincident, but only parallel.

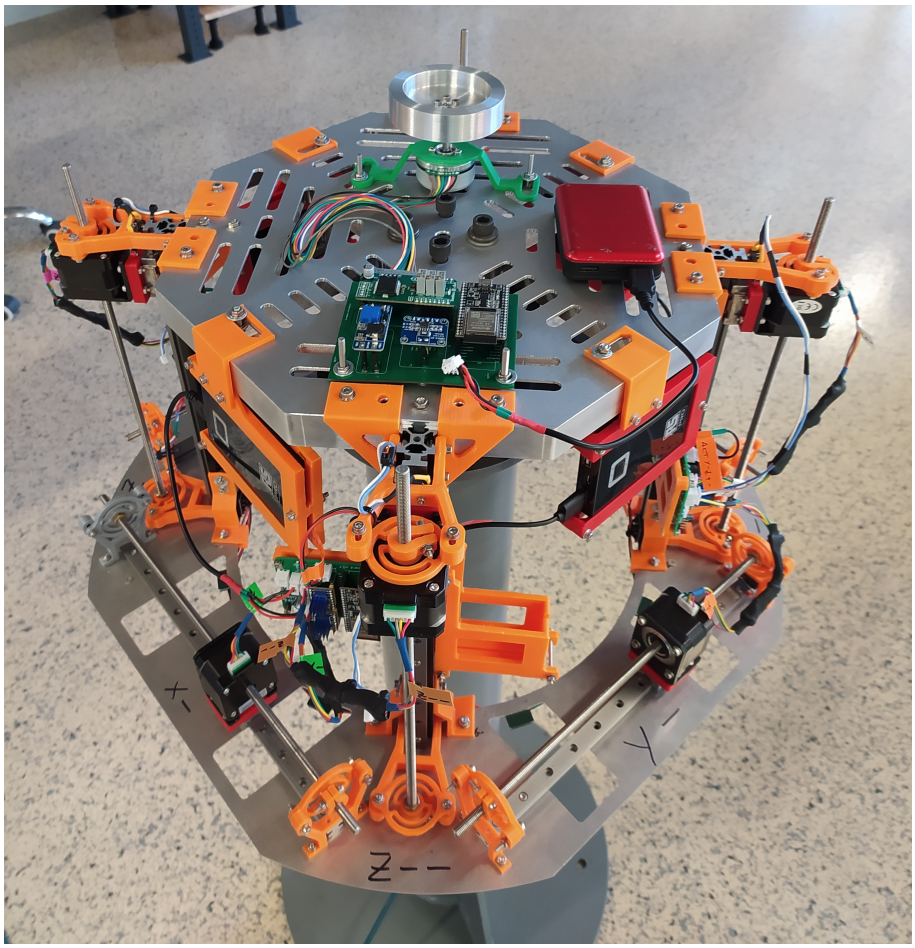


Figure 4.16: Final implementation on the platform.

Chapter 5

Tests and validation

Up to now, this work described in chapter 3 the development through Simulink of a numerical model, which was used to simulate the motion of the simulator platform as actuated by a set of three reaction wheels both in pyramidal and cartesian arrangement. Then, in chapter 4, it was presented the implementation of a first prototype of the system using a single wheel along the z-axis. In this chapter, the hardware implemented will be tested in different configurations, in order to verify the performances of the components and to validate the results of the numerical model, comparing the results obtained with the single wheel prototype with the results from the numerical model in cartesian arrangement.
In the beginning, some tests were carried out using only the motor (no load condition), to check its performances in terms of PWM signal, velocity control, and power consumption. After that, the flywheel was connected to the motor, as to test the performances of the whole reaction wheel assembly. Finally, the assembly was mounted on the platform over the air bearing, to test the ability of the reaction wheel to control the motion of the platform and to validate the results from the numerical analysis.

5.1 Motor tests

The first tests are related to the BLDC motor in a no load condition. The hardware configuration used was that depicted in Fig. 4.12 and 4.13. These tests were carried out as part of the iterative process already described in Sec. 4.4: on the one hand the connections of the components were tested from a hardware point of view, and on the other hand the control software was developed. The latter is mainly composed of a code written via Arduino IDE and loaded on the ESP32 microcontroller, apart for the final validation, in which a connection with Matlab via TCP/IP protocol is also used.
The first task was to understand how to operate the motor via the ESP32 and the driver. As for the rotation velocity, it is estimated starting from the hall sensors, whose signal is processed in a subsequent phase. For this reason, the evaluation of the motor drive is initially performed visually. To actuate the motor, the microcontroller will send the signals to the motor driver, that will ensure that the motor will reach the velocity specified by the microcontroller. Three pins are available to exchange information between the ESP32 and the driver: the enable (EN) pin, which enables the chip, the brake (BRK) pin, which implements the brake function, and the forward/reverse (F/R) pin, which selects the direction of the rotation. All these three pins can be used for speed control of the motor: anyway, the easiest and most effective choice is to use only the F/R pin connected to a PWM signal coming from the microcontroller, as stated in [29]. In this configuration, the PWM controls both speed and direction: a PWM duty cycle of greater than 50% will cause the motor to run forward and a duty cycle of less than 50% will cause the motor to run in the reverse direction.

To generate the PWM signal, it was used the LED Control (LEDC) peripheral API already available for the ESP32. Thanks to it, it is possible to define all the main characteristics of the PWM signal outputted by the microcontroller, like the frequency and the duty cycle resolution. Concerning the frequency, several values were tried: with higher frequencies, the rotation of the motor resulted to be smoother. Moreover, for values between 5k and 18k Hz, a high-pitched beep could be heard coming from the motor. Taking into account this information, the frequency was set to a value of 50k Hz, which ensures smooth and quiet operations. On the other hand, the resolution was selected as 10 bit, meaning that the duty cycle can be set to a value between 0 and 1023, where 512 represents a value for which the motor is still. Obviously, a higher resolution would give more values for the duty cycle, leading to more precise control of the speed. Anyway, the resolution is strictly related to the frequency of the PWM signal: higher resolutions would mean lower frequencies and viceversa. In our case, higher resolutions with a frequency of 50k Hz caused errors in the microcontroller, so that 10 bit is the highest resolution that can be set with a frequency of 50k Hz. Apart from this, the LEDC peripheral allows also to select other characteristics for the PWM signal, such as the channel and timer that will be used to set and update the duty cycle at each iteration. For a complete description of the peripheral, see [1].

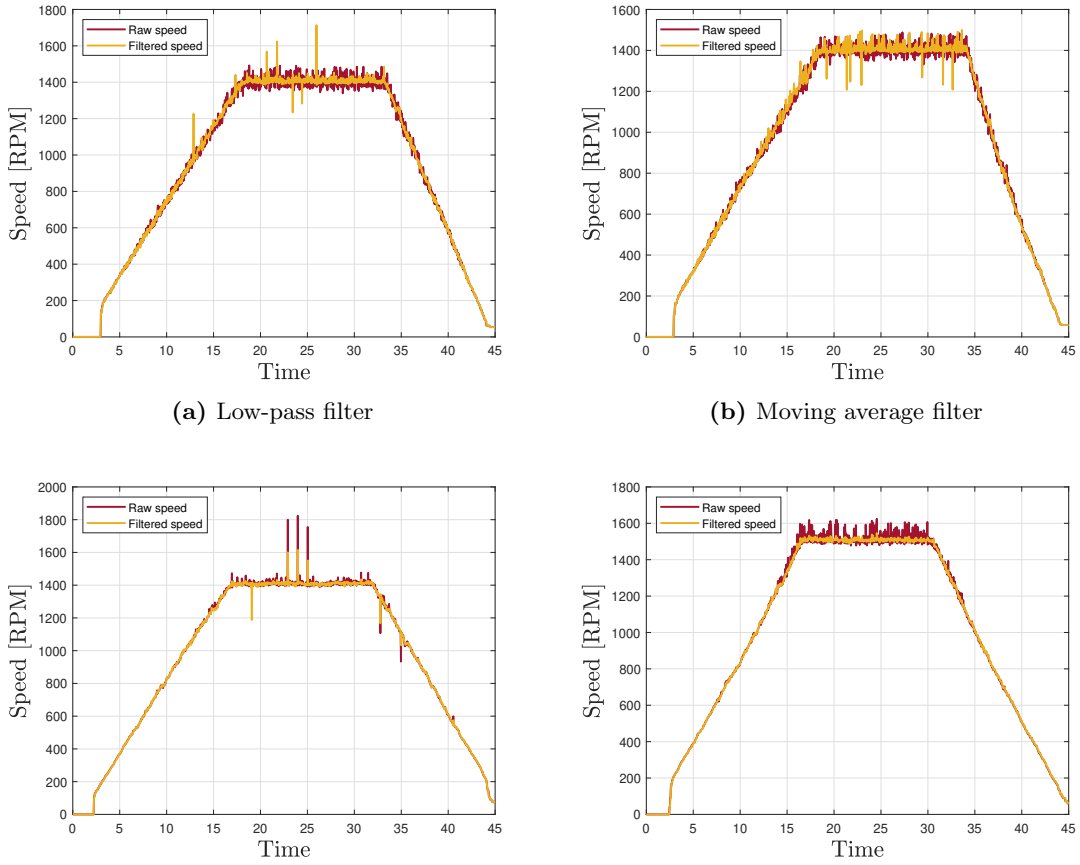
5.1.1 Velocity estimation

After understanding how to operate the motor, the next step is to obtain an estimate of its rotation speed through the signal provided by the Hall sensors. In fact, a good estimate of the speed of the wheel is essential to control the wheel itself and consequently the movement of the platform, which is the final goal. To do so, the signals from the Hall sensors is sent to the microcontroller, where it is decoded and manipulated to get the speed of the wheel. Since the raw speed obtained from the sensors resulted to be noisy, some tests were performed with different filters, in order to get the better estimate.
The first step involved the decoding logic for the signals from the Hall sensors, which is similar to what already described in Sec. 3.2.2. The code written in the Arduino IDE checks at each iteration the sequence coming from the Hall sensors and saves the elapsed time between two different sequences. Since there are 3 different sequences and eight pole pairs, each change of sequence corresponds to an increment in the angle of one twenty-fourth of a complete rotation. Knowing the angle and the time elapsed, it is possible to calculate the absolute value of the angular speed. Then, since a single sequence of the Hall sensors signal changes eight times in a full rotation (due to the pole pairs), it was added a counter to update the velocity once every eight sequence changes: in this way, the sequence is updated only one time for each revolution. This has the advantage of reducing the sampling frequency, and so the noise, at high speeds, while could lead to a poor estimate at low speeds. Anyway, it was decided to adopt this solution, given the fact that the motor will rotate for the most at high speeds. The speed estimation obtained in this way is shown in blue in Fig. 5.1a. The four subfigures in Fig. 5.1 show a comparison between filtered data in red and unfiltered data in blue, for different filters. The speed profile was attained setting manually the PWM in such a way to resemble the profile of the reaction wheel velocity during the maneuver.
In Fig. 5.1a, the raw speed was filtered using a third-order low-pass filter, similar to the one used in the sensor module of the Simulink model. To develop the discrete filter, the signal processing toolbox in Python was exploited, using the transfer function from Eq. (3.49), a pole frequency of 40 Hz and a sampling frequency of 100 Hz, the toolbox gives as output the coefficients a_i and b_i, that will be used to get the filtered speed as:

$$v_{f,k} = a_0 v_{f,k-1} + a_1 v_{f,k-2} + a_2 v_{f,k-3} + b_0 v_k + b_1 v_{k-1} + b_2 v_{k-2} + b_3 v_{k-3} \tag{5.1}$$

[1] Espressif documents, `https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/ledc.html`

where v_f is the filtered speed, v the unfiltered one, and k indicates the time step. As it can be seen, the filtered data result to be less noisy, but there are still some spikes that are not ideal considering that this speed will be used in a feedback loop. The second filter applied was a moving average filter, which averages the data of the last eight values. The results are similar to the low-pass filter, and are represented in Fig. 5.1b. The figures at the bottom, instead, are obtained using the same filters, low-pass on the left and moving average on the right, but modifying the starting data coming from the decoding of the Hall sensors signals. In fact, while previously the speed was updated at each sequence change using all the signals from the three sensors, now an estimate of the speed is calculated for each Hall sensor, obtaining three estimates v_1, v_2, and v_3. Then, the final speed is the average of the three values. This average was used to pre-filter the data. Comparing the bottom figures, it is clear that the best result is achieved employing the sensors average in addition to the moving average: in this case, the filtered speed is less noisy and lacks completely the spikes present in the other cases. Furthermore, the moving average filter can be easily modified, increasing the steps that are averaged.



(a) Low-pass filter

(b) Moving average filter

(c) Low pass and sensors average

(d) Moving and sensors average

Figure 5.1: Different ways of filtering Hall sensors data.

Concerning the speed profile, it must be noted that in every case the motor starts when it reaches a speed of about 100 rpm. This problem is related to the required initial torque: when the duty cycle has a value near half of the resolution, of about 512 ± 10, the motor remains still, so that is not possible to control it for speeds lower than 100 rpm.
At this point, having a reliable estimate of the motor speed, it is possible to evaluate some of the performances. The first thing was to check the relation between the duty cycle and the speed reached by the motor. To do so, some tests were performed setting the value of the duty cycle at

intervals of 50 (so 400, 450, 512, 574, 624, and so on). The speed profile was similar to the one of Fig. 5.1: from this, only the data between 4 and 6 seconds were taken, in order to be in the constant velocity phase, and the average speed was calculated, to have a single value of speed for each value of duty cycle. The data obtained in this way are represented by the circles in Fig. 5.2, while the lines simply connect linearly the data. As it can be seen, the relation is quite perfectly linear.

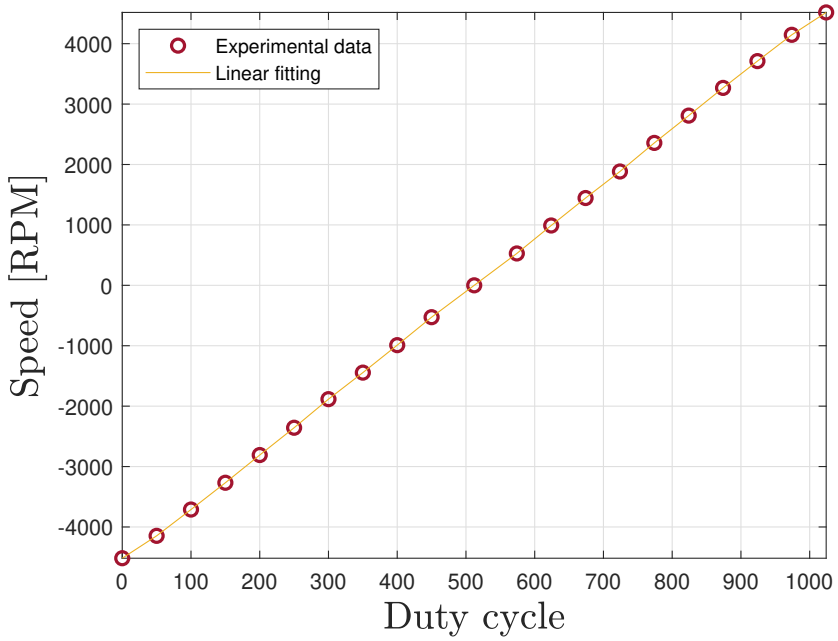


Figure 5.2: Relation between PWM duty cycle and motor speed.

The maximum speed reached, with the maximum duty cycle, is about 4500 rpm: anyway, it must be noticed that these tests were carried out with the step-up converter which was not well calibrated, and was outputting a voltage of only 16 V. For this reason, the step-up was then calibrated to 24 V, and the maximum duty cycle was tested again. In this case, the maximum speed reached was about 6100 rpm.

5.1.2 Motor LQR implementation

The next step consists of implementing the first LQR loop to be able to control the motor. In the previous section, the speed profile was set manually by imposing in advance the duty cycle of the motor. Now, instead, we want to implement the LQR control defined in Sec. 3.2.4 to allow the motor to track an external profile. To achieve this goal, it was developed in the microcontroller's code not only the LQR, but also the reference trajectory generation, in particular for the wheel speed. In this way, at each iteration it is possible to compare the reference speed to the speed estimate, to obtain the error e_{rw}. Moreover, the gain matrix was taken from the Simulink model and written in the microcontroller's code as:

$$G = \begin{Bmatrix} 0.1271 & 0.269 & -0.269 & 0.01 & -1.4142 \end{Bmatrix} \tag{5.2}$$

At this point, the control input u is obtained as $u = -G\,\mathbf{x}_{rw}$, where the state vector $\mathbf{x}_{rw}$ is composed of the wheel velocity, the motor currents and the velocity error, as defined in Eq. (3.65). In the Arduino implementation, the state vector is populated with the estimate of the speed

and the error defined previously, while the currents are assumed to be zero, since the currents estimate is still not available. Anyway, this should not be a problem, since the currents are much smaller than the velocity and have a smaller weight than the error, so that they do not have a strong influence on the final control input u. To be sure, a simulation in Simulink was carried out, putting the currents in the LQR to zero, and it was checked that the results in terms of wheel velocity and Euler angles of the simulator were not affected. At this point, the control input u was used to define the duty cycle as:

$$\text{dutyCycle} = 512 + u\,\frac{512}{24} \tag{5.3}$$

In fact, in the Simulink model, the control input u defines the voltage between 0 and 24: Eq. (5.3) is used to map the control input from the voltage to the duty cycle. The results from the reference tracking can be seen in Fig. 5.3. From the figure it is clear that the tracking is good, with less noise in the transient phases than the constant phase. It is important to note that the moving average filter was modified using 2500 steps for the average: this reduced greatly the noise magnitude. Also, the reference speed (in blue), does not have the 'start-up problem', while the motor speed (in red) does. Nonetheless, when the motor starts, the LQR is able to quickly regain the difference in speed, with a little oscillation at the beginning. The main difference between the two profiles is a certain amount of delay in the motor speed. This delay was even larger in the firsts trials, but it was reduced by increasing the gain multiplied by the error. The results in Fig. 5.3, in fact, were obtained increasing the fifth component of the gain matrix G by 5. In this way, the oscillations in the constant phase have a magnitude of at most ± 30 rpm, which, with respect to the nominal speed of 2000 rpm, represent an error smaller than 1.5 %.

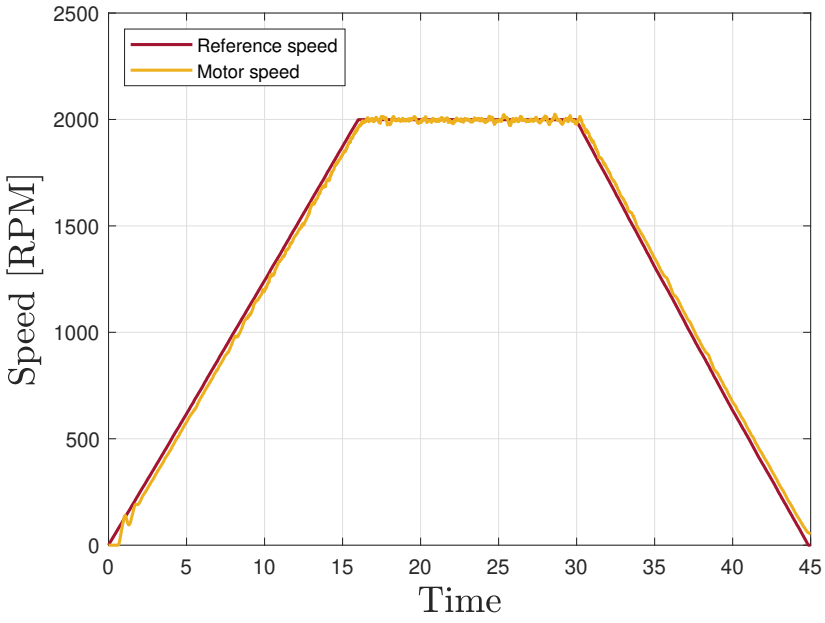


Figure 5.3: Reference speed tracking.

5.1.3 Power estimation

Among the electrical components, the last one remaining to be testes is the current sensor, which is connected to the circuit according to Fig. 4.12. Checking the connections, it is possible

to see that the current from the power bank is splitted: some current flows between the VCC and GND poles to power up all the electrical components, while the remaining current is adjusted by the driver and sent to the motor. The position of the current sensor implies that it will be able to measure only the current going into the motor. This is fine because the motor is the most critical component in terms of required power, while the microcontroller and driver can use limited amounts of current, and so power, which are therefore considered negligible.
As explained before, the sensor uses the I2C interface to communicate with the microcontroller: anyway, there is no need of developing a specific code, since the Adafruit library already offers some functions to evaluate the load voltage, current, and power, in addition to bus voltage and shunt voltage, which are related to the characteristics of the sensor. Moreover, three different range for the measurement can be selected: 32V/2A, 32V/1A, and 16V/0.4A. For our case, the second range was selected, since the voltage range should include the nominal voltage of 24 V, but the current range can be reduced to 1 A, which guarantees higher precision on the measured current.
To test the sensor, the same reference velocity of Fig. 5.3 was used, and the results are shown in Fig. 5.4. In this case, the total time is 45 seconds because 5 idle seconds were added at the beginning, so that the motor starts rotating a $t = 5$ s. The most accurate measure is related to the voltage, which is not reported in the figure since it was very stable at around 24.18 V, with little oscillations of ± 0.01 V. On the other hand, the measure of the current and power resulted to be noisy, so the profiles in Fig. 5.4 are filtered thanks to a moving average of 100 steps. As it can be seen, their behavior is exactly the same, since the power is estimated from the current. In the beginning, both start from zero because the power bank was turned off. After 2 seconds, the power was turned on and both power and current arrives at idle values of respectively 2.775 V and 0.114 A. These values, between 2 and 5 seconds, are due to the idle current that flows in the motor even when it is still. Then, the behavior is similar to the one of the speed, with an increase, a constant phase, and a decrease. In the constant phase, the power required by the motor to rotate at 2000 rpm with no load oscillates around a value of 3.9 W, which derives from a current of 0.16 A.

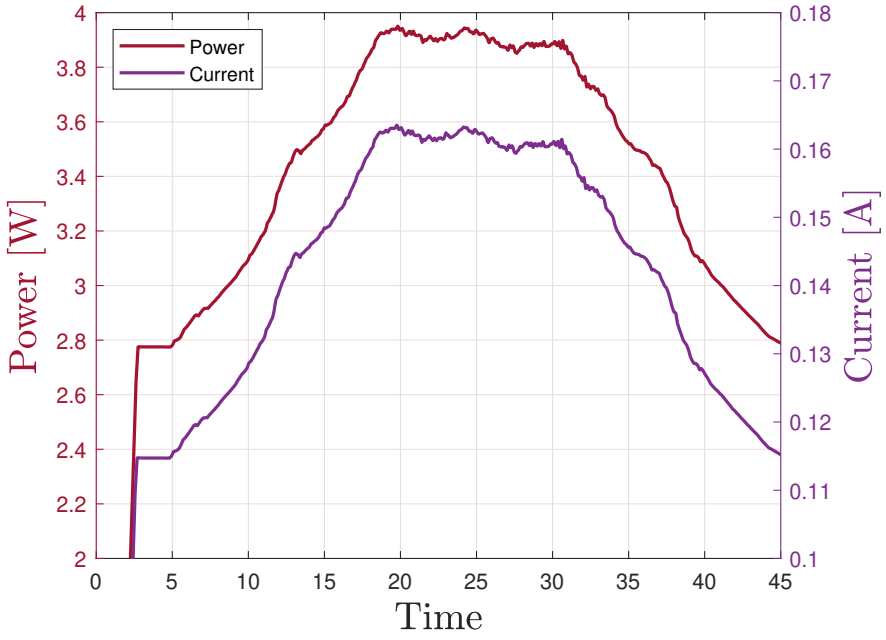


Figure 5.4: Power and current estimation from the tests with no load applied on the motor.

5.2 Reaction wheel assembly tests

At this point, all the components have been tested. Starting from the motor, the signals from the Hall sensors have been decoded and manipulated to get an estimate of the speed, which was then used to implement the first feedback control loop based on the LQR. Furthermore, the relation between the PWM duty cycle and the motor speed was investigated and then an estimate of the power consumption and current of the motor was obtained. All these tasks were carried out with the motor in a no load condition, so free to rotate. This section will present the results of the tests performed with the same configuration and software described up to now, but with the addition of the flywheel attached to the motor shaft.
The evolution of the motor speed is displayed in Fig. 5.5. As it can be seen, the reference speed is the same described earlier, apart from the first 5 seconds where the motor is still, as explained in Sec. 5.1.3, while the yellow line is the speed of the motor, which tries to keep up with the reference speed thanks to the LQR control. This graph is similar to the one of Fig. 5.3, with the same initial oscillation, but at the same time it is noisier: in fact, in the constant phase, the oscillations have a magnitude of about ± 50 rpm, which is 2.5 % of the nominal value. This increase in the noise could be linked to the connection between the flywheel, the flange, and the motor shaft. In particular, the flange was clamped to the motor by overheating the former and inserting it into the latter (shrink fit). Then, the perpendicularity of the plane in which the flywheel lies with respect to the motor axis (z-axis) was checked with the help of a level. Some eccentricity could be introduced in the system by this connection: anyway, the increase in noise is restrained and should not affect the final performances of the prototype.

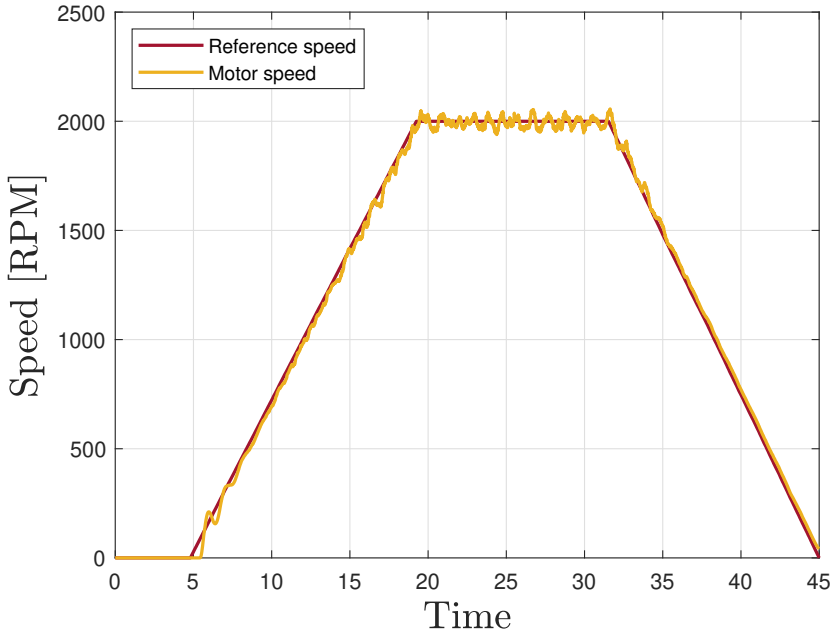


Figure 5.5: Speed estimation with the flywheel attached to the motor.

On the other hand, Fig. 5.6 shows the power consumption and the current flowing in the motor during the test. In this case, both quantities are higher than in the no load case, with a mean value for the power of 4 W in the constant phase and peaks reaching a value of 4.8 W. As for the speed, also the current, and so the power, results to be noisier because of the speed estimation which is used to generate the control input. In particular, the oscillations are evident during the acceleration and deceleration phases, while are more contained in the constant phase. Anyway, given that the power consumption is not used in any control loop, but only to check

that the wheel does not require too much power, the behavior is acceptable: even in the highest peak, the power consumption is below 5 W, so well below the 20 W limit.

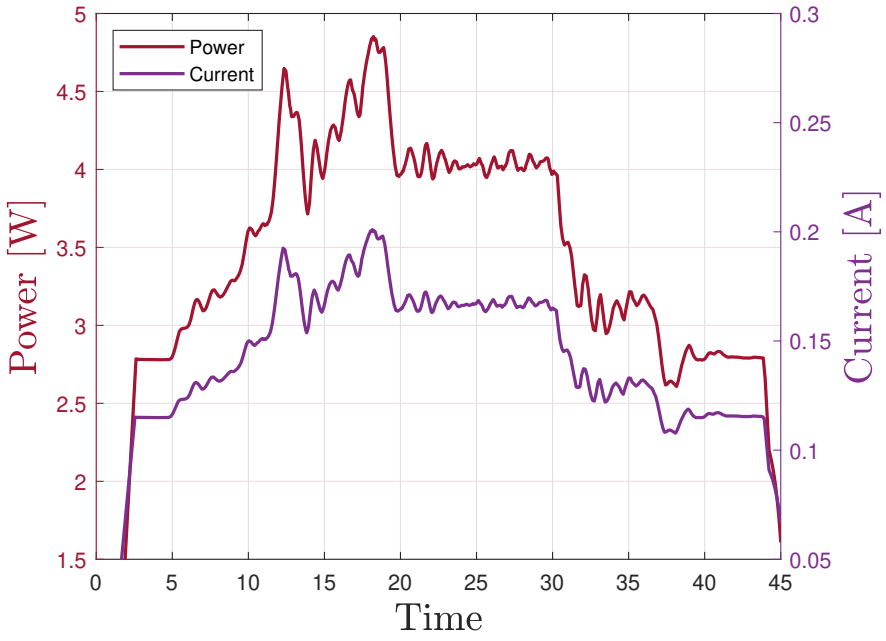


Figure 5.6: Power and current estimation with the flywheel attached to the motor.

Moreover, the results from these tests are in accordance with what we expected for the motor performance. Comparing the Figs. 5.5 and 5.6 with Fig. 4.3, it is possible to verify that the predictions from the mathematical model of the motor are accurate in terms of absolute values. About the power, the behavior measured from the real motor is noisier and lower during the constant phase: anyway, it was demonstrated that it is possible to run the motor at power levels way lower than that listed on the datasheet, with good accuracy in the speed profile.

5.3 Platform tests

After the tests of the reaction wheel assembly are completed, it is possible to mount all the components on the platform to test if the wheel is able to actuate the simulator. This final configuration has been already presented in chapter 4 and is the one depicted in Fig. 4.16, which represents the actual implementation of the complete Simulink model shown in Fig. 3.8. In the previous chapters, it has already been discussed how the control module works and why it is composed of two LQRs: the inner one is related to the control of the wheel velocity, while the outer one is linked to the control of the simulator's position and velocity. Even if the two feedback loops use both the LQR method, there is a difference to take into account for their implementation. In particular, in the first loop the gain matrix is calculated at the beginning of the maneuver, and then the control input is derived at every step as $u(t) = -G_{aug}\, x(t)$, where $x(t)$ is the speed of the wheel. In this case, the control logic has been implemented in the microcontroller, as described in Sec. 5.1.2. Anyway, the same thing cannot be done for the outer loop: in this case, the linearization of Euler's equations requires that the gain matrix is updated regularly, in relation to the new position and velocity reached by the simulator. In Simulink, the updating time was selected as 0.1 seconds, so that at this time interval the control module calculates again the gain matrix from the weighting matrices and from the state vector measured at that time, using the built-in Matlab function `lqr`. However, an analogous of this function does

not exist in the Arduino IDE, so the second loop cannot be implemented on the microcontroller like the first one. To overcome this problem, it was decided to establish a connection between the microcontroller and the computer with which the simulations were carried out, which replaces the on-board computer of the spacecraft. The basic idea is the following: the microcontroller will obtain the measurements from the sensors and compare them with the reference values of the reference trajectory generation module, as already done for the speed of the wheel. Then, at intervals of 0.1 seconds, it will send the data related to the position of the simulator, the speed of the simulator, and the speed of the wheel to the computer, which will use them to calculate the new values of the gain matrix to be sent back to the microcontroller. In this way, it is possible to develop the control logic for the two loops in a similar way, deriving the input as $u(t) = -G(t)\,x(t)$ inside the ESP32: the only difference is that the gain matrix of the outer loop is updated through Matlab at regular intervals.
To establish a connection between the microcontroller and the computer, it was exploited the TCP/IP protocol, which enables the communication between the two components via the WiFi network of the laboratory. In particular, the ESP32 will represent the server and Matlab the client: in this way, it is possible for both to read and write data on the server. Anyway, the data are sent by the microcontroller one byte at a time, which means one character at a time in ASCII notation. For this reason, it is necessary to convert them after the transmission: in Matlab, the data are read one by one and collected into strings, which are then converted into doubles. The same happens in the ESP32 after the reading. This allows the data exchange between the microcontroller and the computer, enabling also the outer feedback loop to be implemented. So, referring to the complete Simulink scheme of Fig. 3.8, both the control module and the reference trajectory generation module are finally implemented in the ESP32, apart from the 'Sim controller LQR block', which is developed thanks to the TCP/IP protocol. Fig. 5.7 shows a scheme of the implementation for both control loops.

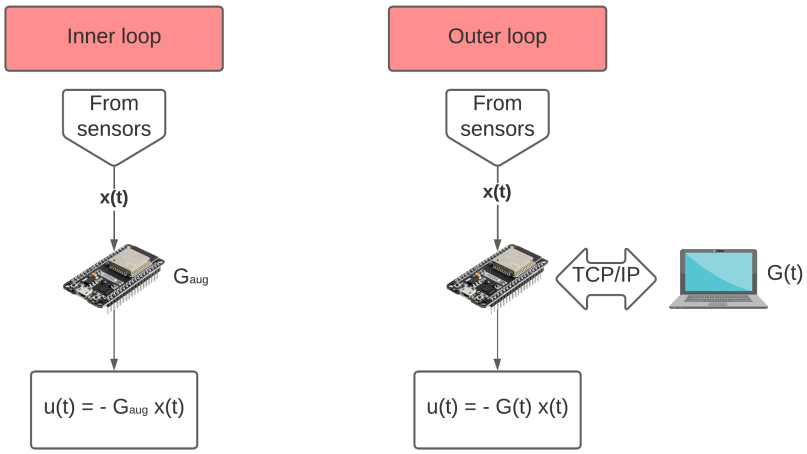


Figure 5.7: Scheme of the control loops implementation.

At this point, everything is ready for the realization of the final tests, in which the reaction wheel is tried and the rotation angle of the platform is measured. For this purpose, it is possible to measure the rotation angle through the IMU sensor mounted on the simulator: this would not represent the ideal solution, as the sensor is the same used for the estimation of the position in the control loop, while it would be desirable to use an external sensor for position verification only. In any case, considering that the rotation angle is a multiple of the right angle, it is also possible to check for any gross errors by eye. The idea was that of performing two different tests: the first test will be carried out with only the first LQR on the velocity of the wheel, so without

any control on the position of the simulator, in order to verify how much the disturbance torques, not accounted for in Eq. (3.51), influence the response of the platform. The second test, instead, will be performed with both control loops activated, to verify the usefulness of the second LQR in contrasting the disturbance torques.
Unfortunately, it was not possible to run these final tests before the deadline set because the platform was still not completely balanced. The balancing of the platform, which was already introduced in Sec. 2.2.2, is fundamental to assure the stability of the motion of the air-bearing: without this step, it is impossible to control the platform with only the reaction wheel. Moreover, the wheel cannot be used to overcome the torque generated by the gravity vector acting on the offset between the center of rotation of the bearing and the center of gravity of the platform. In fact, as shown in Eq. (2.67), the torque generated by the gravity would require a constant control torque, which in turn would lead to a constant acceleration of the reaction wheel, thus rapidly reaching the saturation limit of the latter.

Chapter 6

Conclusion

In this final chapter, the work carried out during this thesis project will be summarized, in order to assess if the objectives presented in the first chapter were fulfilled, and future developments related to this work will be investigated.
First of all, the background of the project was presented: this included the fundamentals of spacecraft attitude dynamics, with a particular focus on some aspects that will be central in the development of the mathematical model, like the parameters used for attitude representation, the concept of angular momentum, and Euler's rotational equations, that are derived both in the cases of a rigid body and in the presence of momentum exchange devices, in particular reaction wheels. Then, spacecraft attitude simulators are discussed, describing the principal characteristics of air-bearing and introducing the problem of the balancing, whose solution is fundamental for the correct functioning of the system. The description of the background is completed with an introduction to the simulator platform already available in the laboratory.
After that, the mathematical model is developed starting from the sizing of the reaction wheel, based on the parameter n defining the maneuver profile, which is chosen to reduce the power consumption. The properties of the wheels are then inserted in the Simulink model, composed of four main modules, as can be seen in Fig. 3.8. In particular, the system module contains the equations described in the background, in addition to the physical and mathematical model for the brushless DC motor, presented for the first time in chapter chapter 3. In the same way, concerning the control module, the control logic was introduced, from the LQR fundamentals to the choice of the weighting matrices. Eventually, the numerical simulation was performed, demonstrating the validity of the proposed architecture and control logic. Moreover, some results from the model were used later on for the hardware selection of the first prototype of the reaction wheel assembly, in particular for the choice of the DC motor. In addition, all the other mechanical and electronic components were designed or selected, and assembled together to obtain the final system shown in Fig. 4.16. After the hardware selection, also the software part was developed, in order to implement the control logic inside the microcontroller. As the software was being developed, some tests were carried out with the motor to check that everything was working as planned. Finally, every part of the Simulink model was set up on the microcontroller to perform the final tests including the complete platform, but unfortunately, it was not possible to perform the final tests on a working platform.

6.1 Thesis objectives

At this point, it is possible to recall the thesis objectives defined in the Introduction and discuss them.

– **Objective I**. Perform the sizing and design of the reaction wheels that will be employed by the attitude control system. This will include the architecture of the actuators along

with a mathematical model to simulate the performances of the system.
This objective was reached and its evolution is documented in chapter 3. The reaction wheel was sized in order to minimize the power consumption and for the control system of the simulator it was decided to adopt a set of three reaction wheels in pyramidal configuration. The mathematical model was developed through Matlab and Simulink, and the latter was used to simulate the response of the system.

- **Objective II**. Implement the attitude control system, including both hardware and software necessary to operate the system. At the end of this phase, all components must be connected to the simulator and fully functional.
  Regarding this objective, the implementation focused not on the complete control system, but on the first prototype of it, composed of a single reaction wheel. Anyway, both the hardware and the software parts were completed, and the final operating prototype mounted on the simulator is shown in Fig. 4.16.

- **Objective III**. Testing of the system through experiments, in order to validate the model and check the correctness of the overall design and implementation of the attitude control system.
  This objective was partially fulfilled. All the components forming the system were tested individually, as also the reaction wheel assembly. In particular, the results shown in Sec. 5.2 are in accordance with the simulation results presented in Sec. 4.1. Moreover, the motor was able to follow the reference trajectory as expected, validating the inner control loop. Unfortunately, it was not possible to validate also the outer control loop.

Regarding the research question, the answer is provided by putting together all the objectives listed above, which provide the method followed in this thesis to develop an attitude actuation system based on reaction wheels capable of controlling the attitude dynamics of the simulator platform.

6.2 Future works

As explained in the previous section, the objectives were not completely fulfilled: future works are required to be able to control thoroughly the attitude of the platform. The first thing to focus on, once that the simulator will be precisely balanced, is the validation of the outer control loop. As stated at the end of chapter 5, two different tests were envisioned: one using only the inner LQR, to check the influence of disturbance torques, and another one with both the LQRs activated, to validate also the outer control loop. Moreover, to verify the precision of the maneuver, the IMU sensor equipped to the platform can be used at the beginning, but in the future it will be better to have an external dedicated sensor. Running these tests, it will be possible to check the precision of the rotation of the simulator: should it not be accurate enough, some modifications could be required in the LQR, in particular about the weighting matrices, which can be modified in the case that the simulator does not move as expected.
Once the first prototype will be completely validated, the next step would involve the development of the overall control system composed of three or four wheels. This step can be seen as a development of the first prototype: in fact, it would be possible to develop the other wheels exactly as the first one, with the only difference of the structure connecting the wheels to the platform, which should be conceived to place the wheels in pyramidal configuration. The actuation of three wheels would require three different motors, one for each wheel, as well as three drivers. Concerning the other electronic components, instead, it could be evaluated the possibility of sharing some of them among the wheels, in order to achieve a more compact configuration.
To conclude, the next step will involve the validation of the first prototype: should the results be good, it will already be possible to rotate the platform around the z-axis, enabling the realization of the slew maneuver chosen as reference.

Bibliography

[1] M. Doyle, R. Dunwoody, G. Finneran, et al. *Mission Testing for Improved Reliability of CubeSats*, International Conference on Space Optics—ICSO 2020, Virtual conference, March-April 2021. DOI: 10.1117/12.2600305

[2] J. A. Ledin, *Hardware-in-the-loop simulation*, Embedded Systems Programming, vol. 12, no. 2, pp. 42-60, February 1999.

[3] G. Di Domenico, E. Andreis, A. C. Morelli et al. *Toward Self-Driving Interplanetary CubeSats: the ERC-Funded Project EXTREMA*, 72st International Astronautical Congress (IAC), Dubai, United Arab Emirates, 25-29 October 2021. Handle: 11311/1189306

[4] R. E. Snider, *Attitude Control of a Satellite Simulator Using Reaction Wheels and a PID Controller*, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, March 2010.

[5] L. Bernadette, L. W. Curtis, S. Douglas *Space simulation in the Neutral Buoyancy Test Facility*, ISSN, pp. 81-87, January 1993.

[6] J. W. Useller, J. H. Enders, F. W. Haise Jr. *Use of aircraft for zero-gravity environment*, NASA technical report, Lewis Research Center, May 1966.

[7] D. A. Kienholz, CSA Engineering, Inc. *Simulation of the zero-gravity environment for dynamic testing of structures*, 19th Space Simulation Conference, Baltimore, MD, October 28-31, 1996.

[8] W. A. Dittrich *Drop Tower Physics*, The Physics Teacher, Vol. 52, Issue 7, 22 September 2014. DOI: 10.1119/1.4895358

[9] R. Boynton, President of Space Electronics, Inc., *Using A Spherical Air Bearing To Simulate Weightlessness*, 55th Annual Conference of the Society of Allied Weight Engineers, Inc., Berlin, CT, USA.

[10] J. L. Schwartz, M. A. Peck, C. D. Hall, *Historical Review of Air-Bearing Spacecraft Simulators*, Journal of guidance, control, and dynamics, Vol. 26, No. 4, July–August 2003. DOI: 10.2514/2.5085

[11] A. Bahu, D. Modenini, *Automatic mass balancing system for a dynamic CubeSat attitude simulator: development and experimental validation*, CEAS Space Journal (2020) Vol. 12, pp. 597–611, April 2020. DOI: 10.1007/s12567-020-00309-5

[12] G. A. Smith, *Dynamic simulators for test of space vehicle attitude control systems*, NASA - Langley Research Center, August 1964.

[13] S. Chesi, V. Pellegrini, Q. Gong, R. Cristi, and M. Romano, *Automatic mass balancing of a spacecraft three-axis simulator: Analysis and experimentation*, Journal of Guidance, Control, and Dynamics, Vol. 37, No. 1, January–February 2014. 10.2514/1.60380

[14] H. Schaub, *Attitude Dynamics Fundamentals*, Encyclopedia of Aerospace Engineering, pp. 3181–3198, Chichester, UK 2010.

[15] J. D. Biggs, *Fundamental properties and attitude dynamics of a Rigid body*, Course Notes - Spacecraft Attitude Dynamics and Control, A.Y. 2019-2020.

[16] F. Landis Markley, J. L. Crassidis *Fundamentals of Spacecraft Attitude Determination and Control*, Springer, 2014.

[17] J. Peraire, S. Widnall, *Lecture L26 - 3D Rigid Body Dynamics: The Inertia Tensor*, MIT OpenCourseWare, Dynamics, Fall 2008.

[18] H. Curtis, *Orbital Mechanics for Engineering Students*, Butterworth-Heinemann, 3rd edition, 2014.

[19] A. D. Jacot, D. J. Liska, *Control Moment Gyros in Attitude Control*, Journal of Spacecraft and Rockets, Vol. 3, No. 9, September 1966. 10.2514/3.28653

[20] L. Mariani, *Design and development of a small satellites three axis attitude simulation platform*, Politecnico di Milano, M.Sc. Thesis, A.Y. 2018-2019.

[21] A. B. Parisi, *Design and development of an automated moveable-mass system for a CubeSat Attitude Simulator platform*, Politecnico di Milano, M.Sc. Thesis, A.Y. 2020-2021.

[22] C. R. Hayleck Jr., *Analysis of the motion of a satellite reaction wheel assembly optimized for weight and power*, NASA Technical Report, Goddard Space Flight Center, Maryland, December 1965.

[23] G. Prasad, N. Ramya, P. Prasad, and G. Tulasi, *Modelling and Simulation Analysis of the Brushless DC Motor by using MATLAB*, International Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol.1, Issue-5, 2011. 10.1109/ISIE.2011.5984365

[24] R. F. Pereira Gomes, *Development of a reliable and low cost miniaturized Reaction Wheel System for CubeSat applications*, M.Sc. Thesis, Técnico Lisboa, April 2016.

[25] A. Shirazi, M. Mirshams, *Pyramidal reaction wheel arrangement optimization of satellite attitude control subsystem for minimizing power consumption*, K.N. Toosi University of Technology, Tehran, Iran, Int'l J. of Aeronautical & Space Sci. 15(2), 190–198, 2014. 10.5139/IJASS.2014.15.2.190

[26] F. Bernelli Zazzera, *Part 1: Attitude dynamics and kinematics*, Course Notes - Spacecraft Attitude Dynamics and Control, A.Y. 2020-2021.

[27] L. Dozio, *Control system design - Introduction to state-space methods*, Lecture Notes - Dynamics and Control Spacecraft Structures, A.Y. 2019-2020.

[28] C. W. Crowell, *Development and Analysis of a Small Satellite Attitude Determination and Control System Testbed*, Massachusetts Institute of Technology, June, 2011. Handle: 1721.1/67177

[29] T. Hopkins, *Speed control using the L6235 or L6229 with a PWM output from a microcontroller*, STMicroelectronics, April, 2012.