UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE CIENCIAS

Modelado matemático y simulación de un prototipo de nanosatélite CubeSat

TESIS

QUE PARA OBTENER EL GRADO DE MAESTRÍA EN CIENCIAS
PRESENTA:

LEONARDO OCIEL ESPINOZA ZEPEDA

Asesora: Dra. Eloísa del Carmen García Canseco

ENSENADA B.C. ENERO DEL 2022

AUTONOMOUS UNIVERSITY OF BAJA CALIFORNIA

FACULTY OF SCIENCES

Mathematical model and simulation of a CubeSat nanosatellite prototype

THESIS

TO OBTAIN THE MASTER'S DEGREE IN SCIENCES PRESENTS:

LEONARDO OCIEL ESPINOZA ZEPEDA

Supervisor: Dr. Eloísa del Carmen García Canseco

ENSENADA B.C. JANUARY 2022

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE CIENCIAS

Modelado matemático y simulación de un prototipo de nanosatélite CubeSat

TESIS DE MAESTRÍA

QUE PRESENTA

LEONARDO OCIEL ESPINOZA ZEPEDA

APROBADO POR:

Dra. Eloísa del Carmen García Canseco
Directora de Tesis

Dr. Miguel Ángel Alonso Arévalo
Sinodal

Dra. Priscilla Elizabeth Iglesias Vázquez
Sinodal

M.C. Saúl Alonso Zavala Ortiz
Sinodal

ENSENADA B.C. ENERO DEL 2022

RESUMEN de la Tesis de Leonardo Ociel Espinoza Zepeda presentada para la obtención del grado de maestro en ciencias. Ensenada, Baja California, México, enero del 2022.

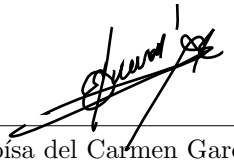
Modelado matemático y simulación de un prototipo de nanosatélite CubeSat

En los últimos años se ha incrementado el desarrollo de satélites tipo CubeSat en la industria y educación debido a su bajo costo comparado con los satélites usuales. En general los componentes de los satélites, como sensores, actuadores, computadoras y fuentes de voltaje, deben de ser probados antes de ser enviados al espacio con la finalidad de identificar posibles riesgos que comprometan el éxito de la misión. Para realizar dichas pruebas existen diversos tipos de plataformas que verifican el buen funcionamiento de los componentes. Estas plataformas pueden llegar a simular los movimientos de orientación del satélite en el espacio, así como algunas perturbaciones externas. Además, ayudan a comprender la física de los satélites y como consecuencia al desarrollo de controladores para estas naves espaciales.

En este proyecto de tesis de maestría, se obtuvo un modelo dinámico de una plataforma satelital tipo CubeSat de 2 grados de libertad de una unidad. Se identificó el método de modelado que mejor se adaptaba al sistema utilizando en un inicio la teoría de robots manipuladores. Posteriormente se utilizó la teoría de dinámica del cuerpo rígido aplicando las ecuaciones de Euler que fueron obtenidas mediante la mecánica Newtoniana y Lagrangiana. Por otro lado, se realizó un modelo de diseño asistido por computadora en Solidworks para luego realizar una simulación en Simscape Multibody de Matlab. Se validó el modelo matemático mediante una simulacion utilizando un control proporcional con retroalimentación de velocidad en el cual se tomaron en cuenta los parámetros físicos de la plataforma satelital y de algunos de sus componentes.

Los resultados obtenidos del modelo y simulación podrán ser utilizados para que futuros estudiantes y trabajadores de la industria aeroespacial desarrollen conocimientos relacionados al movimiento de orientación de los satélites generando así capital humano mejor preparado para trabajos especializados en el campo espacial.

Resumen aprobado por:



Dra. Eloísa del Carmen García Canseco
Directora de Tesis

ABSTRACT of the thesis of Leonardo Ociel Espinoza Zepeda presented to obtain his master's degree in sciences. Ensenada, Baja California, Mexico, january 2022.

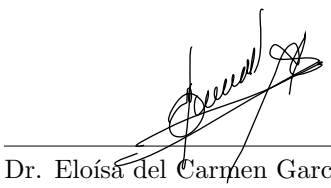
Mathematical model and simulation of a CubeSat nanosatellite prototype

In recent years, the development of CubeSat-type satellites has increased in industry and education due to their low cost compared to conventional satellites. In general, satellite components such as sensors, actuators, computers and voltage sources must be tested before they are sent into space to identify possible risks that compromise the success of the mission. To verify the proper functioning of these components there are different types of testbeds. These testbeds can simulate the orientation movements of the satellite in space as well as some external disturbances. They also allow us to understand the physics of satellites and consequently the development of controllers for these spacecraft.

In this master's thesis project, a dynamic model of a one-unit CubeSat satellite with 2 degrees of freedom was obtained. We identified the modeling method that is best suited to the system to firstly apply the robotic manipulator theory. Later we used the rigid body dynamics theory applying Euler's equations that were obtained through Newtonian and Lagrangian mechanics. On the other hand, a computer-aided design model was made in Solidworks and then simulated in Matlab's Simscape Multibody. The mathematical model was validated through a simulation using a proportional control with speed feedback in which the physical parameters of the satellite platform and some of its components were taken into account.

The results obtained from the model and simulation may be used for future students and workers of the aerospace industry to develop knowledge related to the orientation movement of satellites, thus generating a better prepared human capital for specialized jobs in the space field.

Abstract approved by:



Dr. Eloísa del Carmen García Canseco
Thesis supervisor

Acknowledgments:

I would like thank to professor Eloísa del Carmen García Canseco for allowing me to work with her, for her time, patience, and academic support that will be very helpful for my academic career.

Thanks to professors Priscilla Elizabeth Iglesias Vázquez, Miguel Ánguel Alonso Arevalo, and Saul Alonso Zavala Ortiz for their acceptance of being part of my thesis committee, their time, and guidance during my master's studies.

Thanks, CONACyT for the scholarship granted for the development of this Master's Degree thesis.

I would like to thank the Satellite Lab of CICESE for bringing us the testbed for this research

Thanks to Aravind Srinath who took the time to read my work and gave me some advice to improve it.

Thanks to Melissa Medina who has supported and motivated me to continue with my studies.

To my dog Jupiter who accompanied me (most of the time) during my research and writing of this document, thank you!

Finally, infinite thanks to my parents Margarita Zepeda and Jaime Espinoza for their unconditional support. Thanks to them I have accomplished all my goals and thanks to them I will be accomplished even more.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since the first launch of space satellites in 1957, different satellite science and technology have been developed, from the study of orbits to the improvement of sensors and actuators (Larson and Wertz, 1992). During the first years of satellite development, satellite technology used to be very expensive and not all countries could build it. Over the years, satellite technology has been miniaturized; with tinier technology, we can have smaller and cheaper satellites (Toorian et al., 2008). Small satellites can be classified according to their mass (Table 1.1) and they have all the functions and applications of a normal satellite, such as astronomy, meteorology, militia, telecommunications, medicine, education, earth sciences, astrobiology, among others (Selva and Krejci, 2012).

Table 1.1: Classifications of small satellites according to their mass. Table taken and adapted from Pelton (2020).

Classification	**Mass (kg)**
Minisatellites	100 - 500
Microsatellites	10 - 100
Nanosatellites	1-10
Picosatellites	0.1 - 1
Femtosatellites	$0.1 \leq$

A CubeSat is a particular standard of nanosatellites whose characteristics were defined in 1999 by the California Polytechnic State University and Stanford University (CalPoly, 2015). 1 unit (1U) CubeSat has a cubic shape with 10cm of edge and a mass of less than 1.33 kg. There are many other types of CubeSats that can be classified depending on the number of 1U CubeSats that make it up, for example there exist 1.5U, 2U, 3U, 6U, 27U, and more. These kinds of satellites include all the systems and subsystems such as electrical power system (EPS), attitude, determination and control systems (ADCS), structure system and telemetry, tracking, and command systems (TTC).

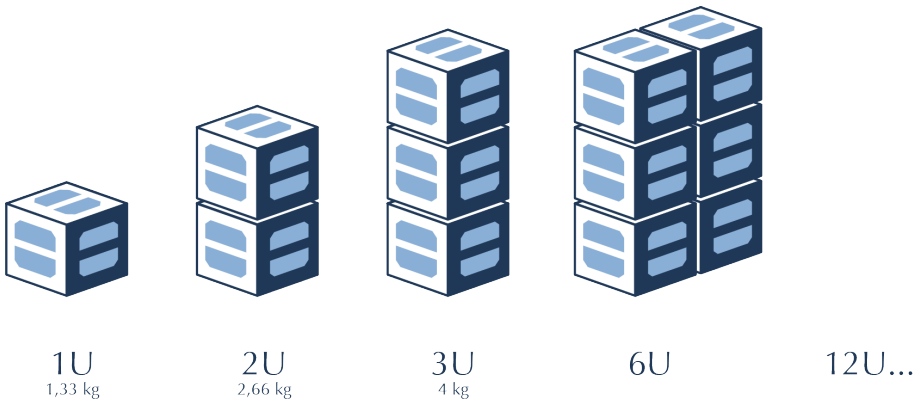


Figure 1.1: CubeSat classification according to the quantity of units they are built. Image taken from Census (2021).

CubeSats have revolutionized satellite missions due to the miniaturization and cheapening of technology. This revolution has allowed industries, universities, and emerging countries, to advance in their technology (Bouwmeester and Guo, 2010). According to the web page Nanosats Database, by August 20th, 2021, there were 1634 CubeSats launched, of which 981 have been launched by companies, and 486 by universities. Besides, there have been 2 interplanetary CubeSats launched to Mars. From all over the globe, 76 countries have launched these nanosatellites, which included first and third nations countries. Figure 1.2 shows the nanosatellites and CubeSats launched by country.

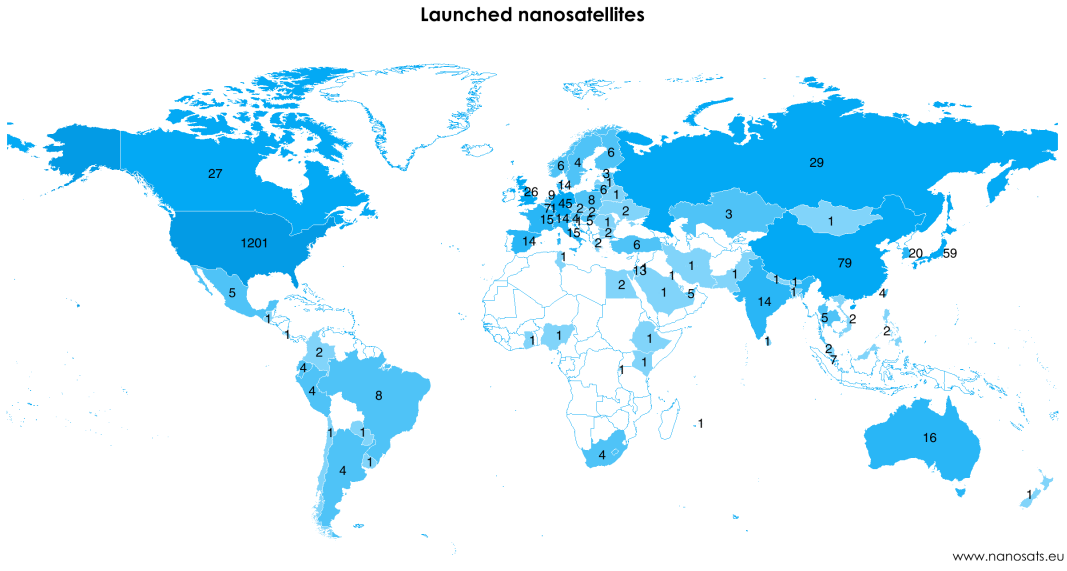


Figure 1.2: Launched nanosatellites Map. Image taken from NanoSats (2021).

Although the United States has the majority of this technology, emerging countries, such as Mexico, have begun to send CubeSats to space. Nowadays, Mexico has five nanosatellites in orbit. The first mission launched is Painani-1, which is a collaboration between CICESE and the Department of Mexican Secretariat of National Defense (SEDENA). Its main objective was to carry a low-resolution camera and to propose an S-Band downlink using frequency hopping technology to download images (de la Cerda, 2019). The second mission: AztechSat-1, is a collaboration between

the Popular Autonomous University of Puebla (UPAEP), the Mexican Space Agency (AEM), and NASA. Its objective is to communicate with the Global-Star satellite constellation to improve the transit of data to Earth (UPAEP, 2019). The National Autonomous University of Mexico (UNAM) sent NanoConect-2, which was developed for educational purposes (UNAM, 2021). A startup called Space JLTZ, in collaboration with Nanoavionics and Dragonfly Aerospace, launched the D2/AtlaCom-1 whose main purpose is to get satellite images to be processed and analyzed by students (JLTZ, 2020). These four satellites (Figure 1.3) were launched in the last two years. Finally, the University of the Mexican Army and Air Force, in collaboration with UNAM and the National Polytechnic Institute (IPN) launched in June 2021 the CubeSat Painani-2 (IPN, 2021). Currently, there are other Mexican CubeSats missions in development and they will be launched in the coming years.

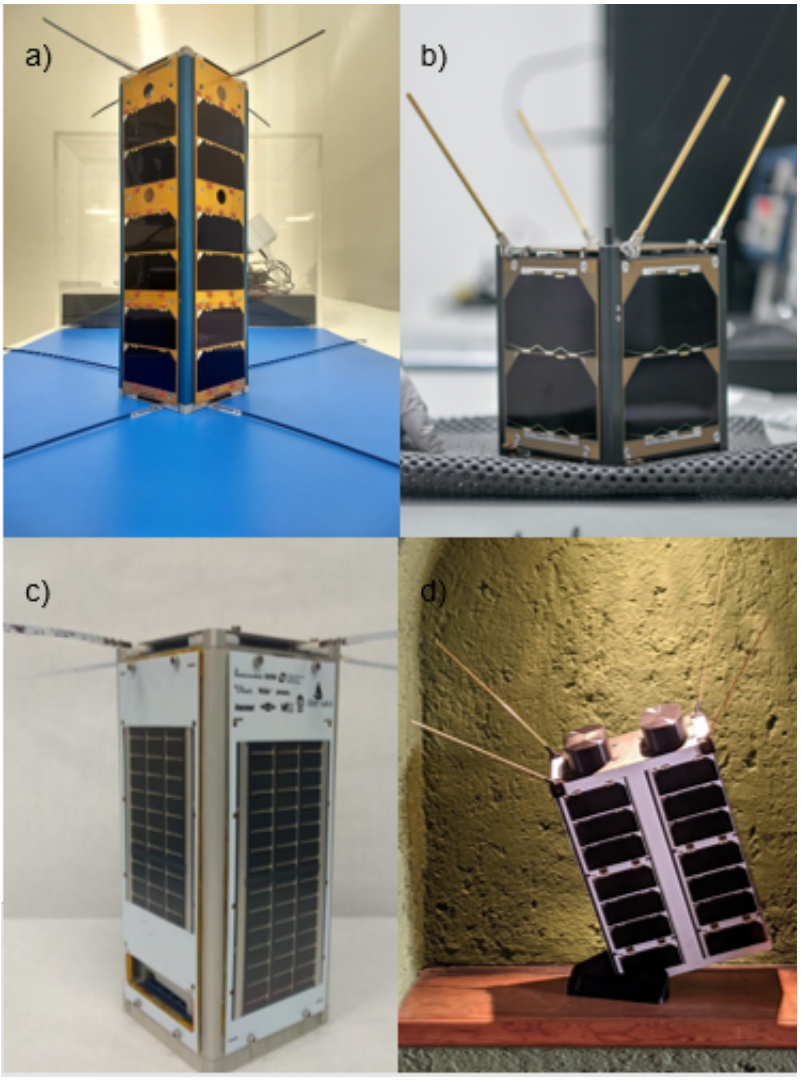


Figure 1.3: Launched Mexican CubeSats. a) Painani-1. b) AztechSat-1 c) NanoConnect-2 d) D2/AtlaCom-1. Images respectively taken and adapted from: de la Cerda (2019); UPAEP (2019); UNAM (2021); JLTZ (2020).

Regardless of the CubeSat size, it can be considered to have the same systems and subsystems of a big satellite. These systems and subsystems allow maintaining its position, orientation, temperature, battery, and communication with the Earth. These systems and subsystems can be cataloged in the following groups: structure, EPS, ADCS, Thermal control, and TTC (Figure 1.4).

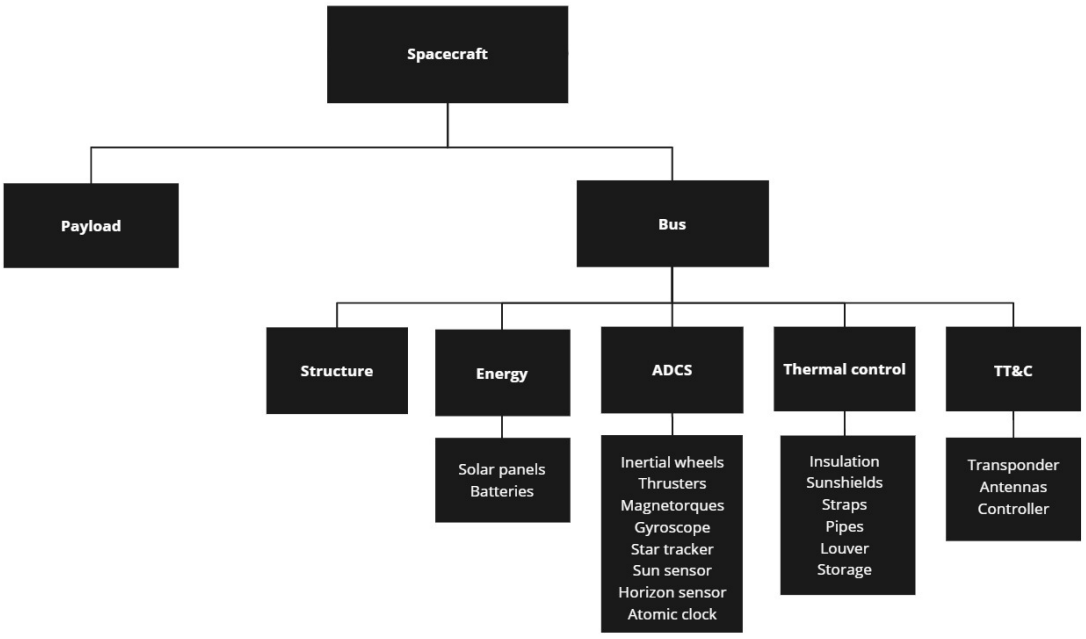


Figure 1.4: Systems and subsystems of a satellite.

The most important systems for our research are the attitude, determination, and control systems (ADCS), which are conformed by many sensors to determine the orientation and position of the satellite. These sensors can be gyroscopes, magnetometers, accelerometers, star trackers, sun sensors, or horizon sensors. To modify the satellite attitude, actuators such as reaction wheels, magnetorquers and thrusters are used. All these sensors measure data in real-time which will be processed by the satellite control actuators. Usually, we introduce the desired variable (q_d) to the system to activate the actuators and the dynamics, then we obtain a measured variable (q) that, in a closed-loop system, gives feedback to the systems (Figure 1.5).

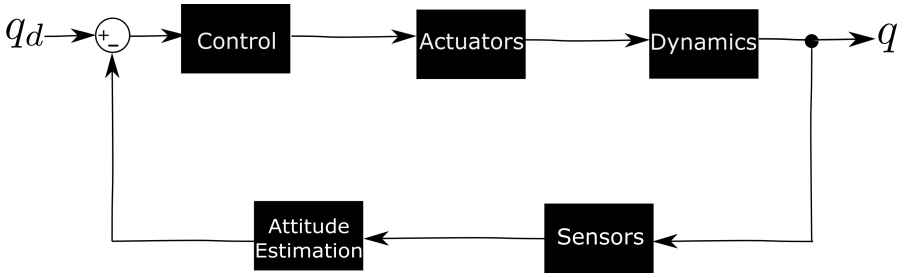


Figure 1.5: Attitude, determination and control mode (close loop).

Usually, a satellite uses an Inertial Measurement Unit (IMU) to estimate the velocity and orientation. An IMU measures using gyroscopes, accelerometers, and magnetometers. The IMU can process and administrate the data taken from these sensors and it is used in the control and stabilization, as well as navigation and course correction (Gutiérrez Medina, 2015). Gyroscopes measure the angular velocity in three dimensions. This data can be integrated and make the estimation of the angular position possible.

Reaction wheels (also called inertial wheels) are the most common method when we desire to control the orientation of the satellite (Trégouët et al., 2015). One reaction wheel can affect the rotation of the satellite on one axis, two reaction wheels affect two axes and three wheels for three

axes. If redundancy is required, we can use more reaction wheels to correct possible and small mistakes on the orientation. (Beebi, 2016).

The systems and subsystems are some of the most important components to the adequate functionality of small satellites. More information can be found in the NASA TechReport of the State of the Art of Small Spacecraft Technology (Yost et al., 2020). Nevertheless, before sending the components to space, it is important to test them on Earth. For these tests, satellite ground test systems or testbeds are required.

A testbed, also known as a ground test system, is a prototype of a real system with electromechanical components where sensors, computers, and actuators from a satellite are tested to validate their functionality and interaction of the components (Fortier and Michel, 2003). Some testbeds are usually mathematically modeled to understand the dynamics of the systems and to control their movement. If the system is well modeled, engineers and scientists can save time and money and the success of the mission is guaranteed (Schwartz et al., 2003). In this thesis project, we obtained a mathematical model and simulation of a 2-dof 1-U CubeSat ground test system (Figure 1.6). The testbed has two reaction wheels, an IMU, and a board computer. Although the dynamics allow us to understand the physics of the testbed, it is even more important if we desire to control the system in future projects.

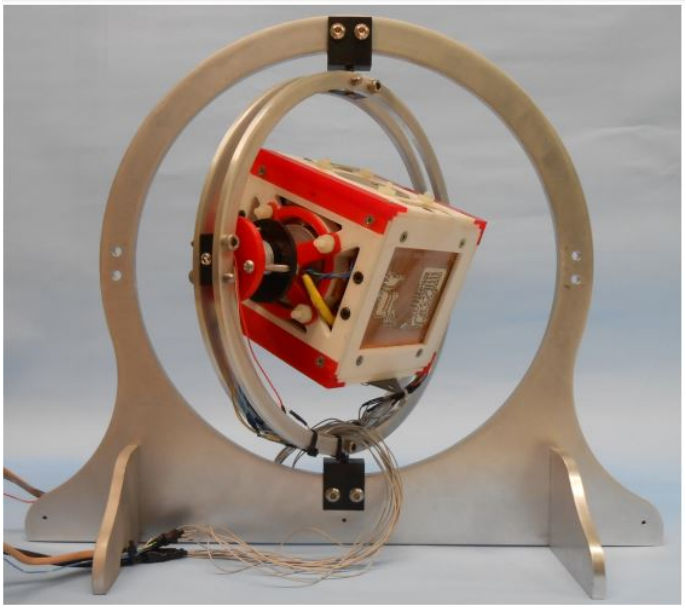


Figure 1.6: CubeSat Testbed of two degrees of freedom. Taken from Gutiérrez Medina (2015).

1.1 Objectives

The main objective of this thesis project is to obtain a dynamic model and the simulation model of the one unit CubeSat testbed of two degrees of freedom (1.6). To achieve this goal, the following specific objectives (SO) were defined:

SO1. To determine the modeling approach that best suits the requirements of the problem.

SO2. To obtain the kinematic and dynamic model using the selected modeling approach.

SO3. To validate the proposed mathematical model by a simulation.

The objectives were accomplished with the rigid body theory from Newtonian and Lagrangian mechanics. Some concepts from the robotics arm manipulator theory were also applied. The model was then computationally validated in Matlab's library Simulink Simscape Multibody with the help of a CAD model in Solidworks.

1.2 Overview

The remainder of this document is organized as follows:

Chapter 2 presents a brief history of the satellite's ground test systems, specifically those concerning small satellites and CubeSats. Besides, it analyzes some of the theories used to model those testbeds.

Chapter 3 explains the mathematical background to build the dynamic model. In particular, the Denavit- Hartenberg convention and rigid body theory will be analyzed to get Euler's equations from Newtonian and Lagrangian mechanics.

Chapter 4 shows the results of the simulations made with Solidworks, Matlab, and Simulink from the proposed model, the simulation, and the robotics perspective.

Chapter 5 discusses the results, concludes and analyzes future work and recommendations to the study of the 2-dof CubeSat testbed.

Chapter 2

State of the art of satellite testbeds

The creation of satellites brought forth the fabrication of testbeds for their validation. For a satellite testbed, the study of their movement is necessary before sending a satellite to space. These tests are extremely important because they can detect future problems in space, saving time and money.

There are many mechanisms for testbeds, which usually depend on the kind of satellite sensors or actuators that will be studied. According to (Kiesbye et al., 2019) there are two types of real simulation for satellite testing: the $software-in-the-loop$ and the $hardware-in-the-loop$.

In the $software-in-the-loop$ (SIL) real software is used together with mathematical models that replace the functionality of some components. The algorithm is usually applied to a particular mechanic or mechatronic system.

The second approach is the $hardware-in-the-loop$ (HIL) which is used with real hardware and software that usually include environment, external effects, and electrical simulations. This technique allows verifying the effectiveness of the hardware and the software.

To verify the proper functionality of testbeds, HIL and SIL simulation are applied together. The following satellite ground test systems were simulated by applying both types of simulation. However, for this thesis project, we only used SIL simulation because, at the time of this research, we did not have access to our testbed due to a sanitary emergency.

2.1 Brief history of satellite testbeds

NASA Ames Research Center was a pioneer in building testbeds. The first one was dated in 1959 and it consisted of a tabletop with air bearings with optical sensors and reaction wheels. A year before, the Marshall Space Flight Center developed a new air bearing system (Figure 2.1) which allowed $\pm 120°$ in roll and pitch and free spin in yaw. After this, many other kinds of testbeds have been developed with more technology that reproduces and simulates satellite technology more efficiently (Schwartz et al., 2003).

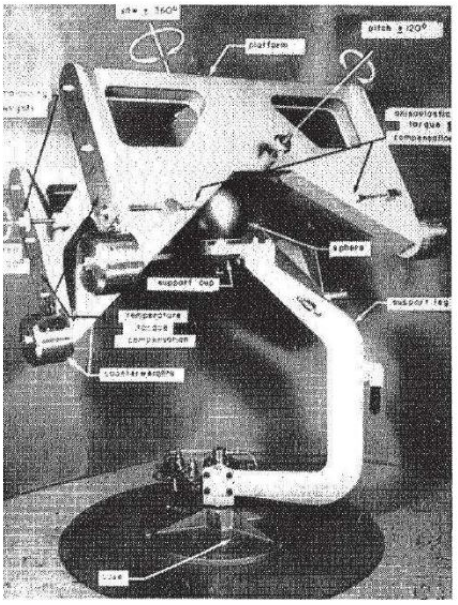


Figure 2.1: Marshall simulator. Taken from Haeussermann and Kennel (1960).

SADS

The first air-bearing platform for satellite ADCS was developed in 1995 by the Spacecraft Attitude Dynamics Control Laboratory at the Naval Postgraduate School (Figure 2.2), it was called the Satellite Attitude Dynamics Simulator (SADS). SADS was a tabletop system that included a magnetometer, three gyroscopes, a sun sensor, three reaction wheels, gas thrusters, and a microcontroller (Agrawal and Rasmussen, 2001).

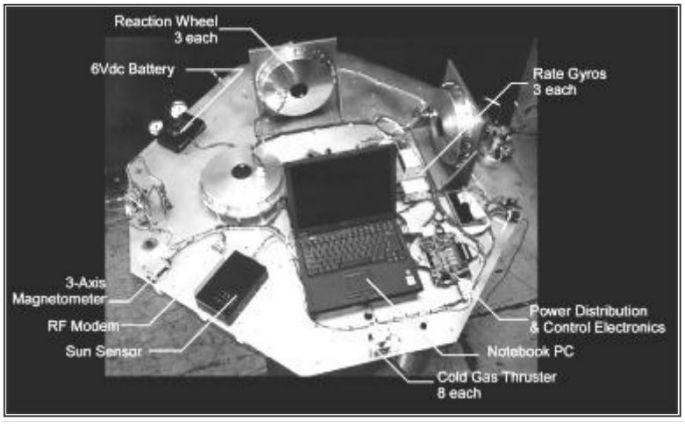


Figure 2.2: SADS Ground test system. Taken from Agrawal and Rasmussen (2001).

CalPoly Spacecraft Simulator

The university that is a pioneer in CubeSat development built its testbed in 2007. The testbed has four reaction wheels in a pyramidal configuration which allows 360° of motion in the z-axis and ±30° in x and y axes (Mittelsteadt and Mehiel, 2007). It has an IMU, camera, batteries, and gyros (Figure 2.3)

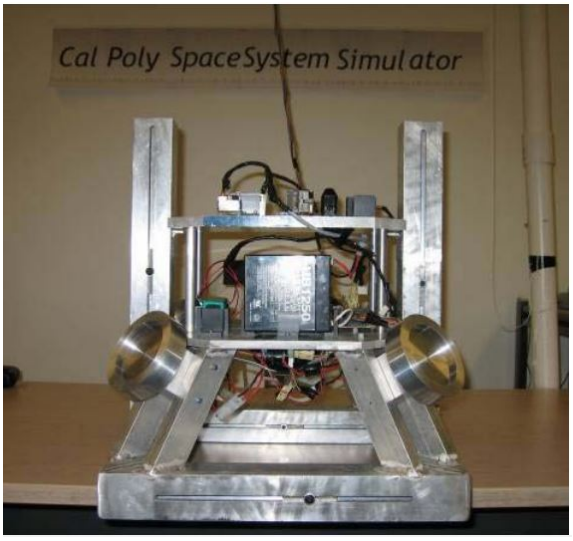


Figure 2.3: CalPolt Spacecraft Simulator. Taken from Mittelsteadt and Mehiel (2007).

2.2 Testbeds for small satellites

Due to the miniaturization of satellite technology, ground test systems had to be adapted for small satellites. There are many mechanisms for simulations, which is why the next section will describe some of them.

SIMUSAT

UNAM testbeds are planar tables that have been adapted and updated to their requirements. This testbed has a two-axis balancing system, an electronic compass, a microcontroller, and two magnetic coils (Figure 2.4). The system was linked with wireless communication to software that represents a small satellite.

Figure 2.4: Spacecraft simulator of two-axis. Taken from Prado et al. (2005).

Recently, the platform was called SIMUSAT and has had many updates (Escobedo Lugo, 2012; Flores Gómez, 2012). For example, the testbed possesses three reaction wheels and three magnetorquers (Figure 2.5) which allows better experimentation and simulation of satellite technology.

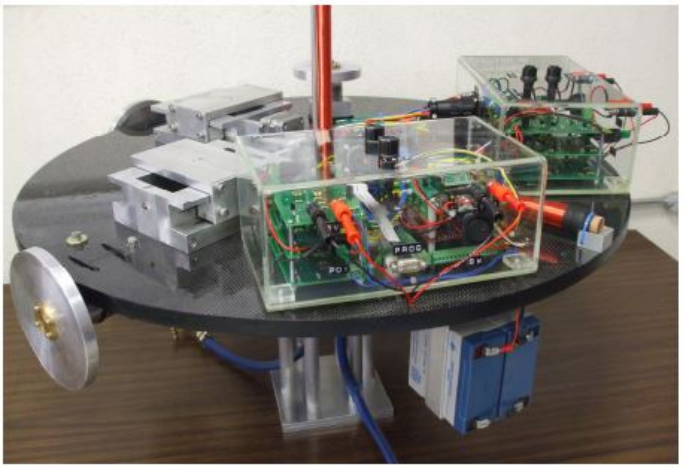


Figure 2.5: SIMUSAT 2.1 is a spacecraft simulator of three-axis. Taken from Escobedo Lugo (2012).

Angularly unbounded 3-axis spacecraft simulator

Recently, researchers from the University of California Oakland and the USA Secretary of Navy patented a spacecraft simulator that allows 3-dof using spherical air bearing, a 3-dof gimbal, motors, and two-position laser sensors (Chesi and Romano, 2020). The system permits completely free spin in 3-axes (Figure 2.6).

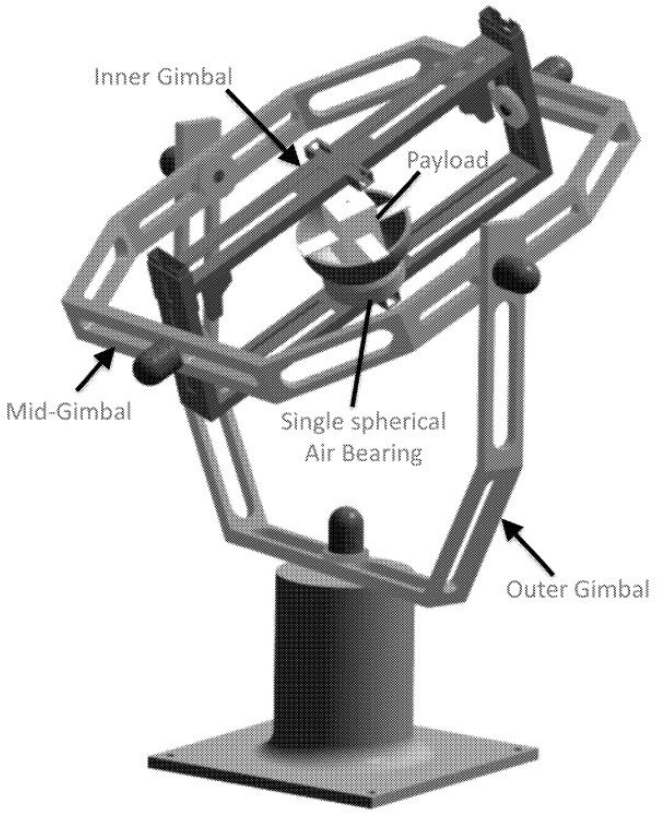


Figure 2.6: Angularly unbounded 3-axis spacecraft simulator. Taken from Chesi and Romano (2020).

2.3 CubeSat testbeds

With the increment of CubeSats developed around the world, new ground test systems have been created. Next, we briefly describe some of the CubeSat testbeds built by research centers.

CubeSat Three-Axis Simulator

The CubeSat Three-Axis Simulator (CubeTAS) was recently built by The Postgraduate School (Meissner, 2009). The current version of the platform (Figure 2.7) has a Helmholtz coil that controls the magnetic field relative to its position to Earth. CubeTAS has 3D cameras that provide attitude reference to the computer and a lamp that functions as artificial sunlight for the sun sensors. Moreover, this ground test system possesses auto-balance thanks to the gyros and motors with balancing mass. CubeTAS has included an IMU, reaction wheels, batteries, and magnetic coils and it is supported by spherical air-bearing that achieves micro-gravity conditions (Woo et al., 2011). This testbed provides only HIL simulation.

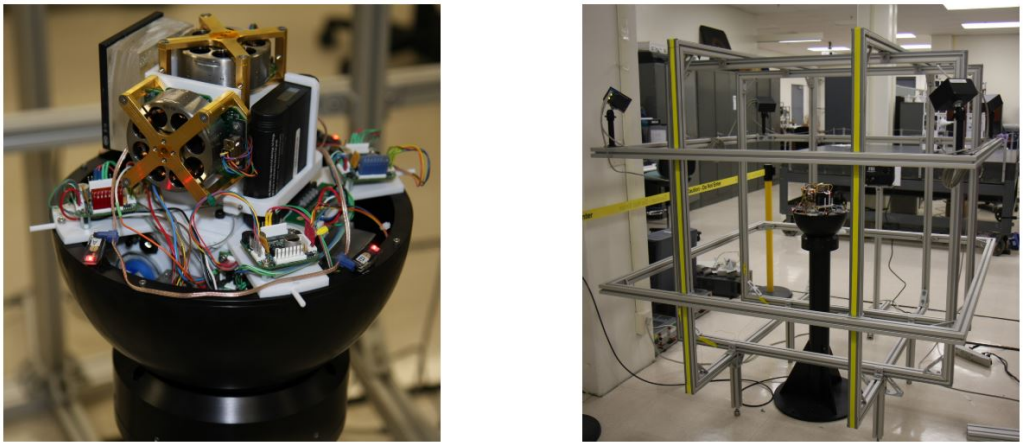


Figure 2.7: CubeSat Three Axis Simulator (CubeTAS). Taken from Woo et al. (2011).

Air Force Institute of Technology

The United States Air Force Institute of Technology has built a 6U CubeSat testbed (Figure 2.8). It has a Helmholtz cage to simulate Earth's magnetic field in the orbit and hemispherical air bearings that simulate attitude maneuvers which provides 360° of motion in the z-axis and 90° for x and y axes. The testbed includes an IMU and four reaction wheels pyramid configuration (Tibbs, 2015).

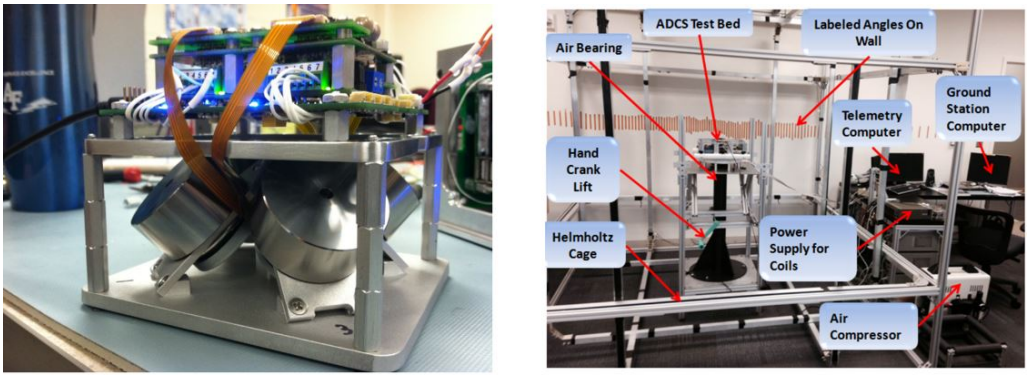


Figure 2.8: AFIT Testbed. Taken and adapted from Tibbs (2015).

LAICA's Testbed

The Laboratory of Application and Innovation in Aerospace Science (LAICA) of the University of Brasilia (UnB) has developed a testbed composed of an air bearing semi-sphere and a Helmholtz cage (Figure 2.9). This configuration allows a maximum spin of ±45° in row and pitch angles. This testbed called BRAZIL has microcontrollers, antennas, an IMU, batteries, and three motors (da Silva et al., 2019).

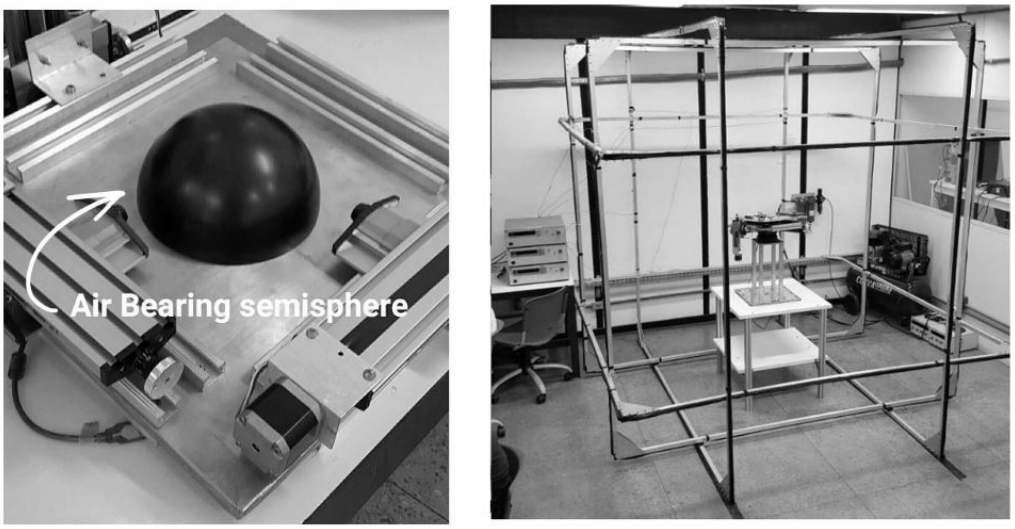


Figure 2.9: LAICA Testbed. Taken and adapted from da Silva et al. (2019).

Automatic Balancing System

One of the most recent CubeSat simulators was built by the Microsatellite and Space Microsystems Lab of the University of Bologna (Modenini et al., 2020; Bahu and Modenini, 2020). This testbed has an automatic balance system (ABS), which estimates the CM by processing data of the angular motion and compensates the unbalance with the movement of the motors. It has 3-dof due to a tabletop air bearing system. Besides, it includes a Helmholtz cage to simulate geomagnetic fields. The testbed provides 360° of spin in yaw and almost 30° for tilt angle. This system has been hardware tested and works for 3-unit CubeSat (Figure 2.10).

Figure 2.10: Automatic Balancing System for CubeSat. Taken from Bahu and Modenini (2020).

Montpellier University simulator

Another testbed was built by the University of Montpellier (Gavrilovich, 2016). This ground test system has four air bearings to generate the movement for the 3-dof for the attitude of the CubeSat. The CubeSat is connected to a robotic arm manipulator that provides 6-dof and simulates the translation around the planet (Figure 2.11). Only the star trackers of this system were SIL tested, however, the whole system was only HIL simulated.

Figure 2.11: CubeSat testbed airBall prototype with a robotic arm manipulator. Taken from Gavrilovich (2016).

Two degrees of freedom CubeSat testbed

The main purpose of this thesis project is to estimate the dynamics of a 2-dof CubeSat testbed (Figure 1.6). This system of study was developed as a master thesis project in the Applied Physics Division at CICESE (Gutiérrez Medina, 2015).

The testbed has a design with the specifications of size and weight for a CubeSat and includes two motors, two-speed handlers, and two reaction wheels to generate the movement in two axes. The reaction wheels have Hall-effect motors from the company Maxon. To balance the mass of the reaction wheels, two counterweights were added on the opposite side of each reaction wheel.

The system was HIL tested and it uses a microcomputer Beaglebone of Texas Instruments. The testbed possesses an iNEMO (iNErtial Module) model STEVAL-MKI062V2 manufactured by STMicroelectronics. The module is conformed by gyroscopes, accelerometers, magnetometers, and pressure and temperature sensors. This allows the iNEMO and the computer to estimate the orientation and angular velocity of the cube.

Even though the testbed has been analyzed and controlled with a frequency domain system identification technique (Bosselar, 2020), it has not been mathematically modeled. The importance of the dynamic model is the fact that it can be used to control the system and, in the future, it could be used to teach students or industry workers the attitude dynamics of a satellite in space.

2.4 Mathematical modeling of satellite testbeds

To obtain the dynamical model of a satellite system or a satellite test-bed different theories can be applied. Table (2.1) summarizes the different theories, such as rigid body dynamics (RBD), Euler angles (EA), or quaternions, used in the ground test systems presented in the previous section. Note that N/A (Not Apply) has been written when the testbed has been not modeled.

Table 2.1: Dynamic models used in some satellite testbeds.

Satellite testbed	**Methodology applied**
Schwartz et al. (2003)	N/A
Agrawal and Rasmussen (2001)	N/A
Mittelsteadt and Mehiel (2007)	RBD
Prado et al. (2005)	RBD & EA
Flores Gómez (2012)	N/A
Escobedo Lugo (2012)	N/A
Chesi and Romano (2020)	N/A
Meissner (2009)	RBD
Woo et al. (2011)	N/A
Tibbs (2015)	Quaternions & RBD
da Silva et al. (2019)	RBD & EA
Gavrilovich (2016)	RBD & D-H
Gutiérrez Medina (2015)	N/A

The theory used in the model depends on the control objective. One theory could be more useful than the other depending on this objective. Even so, it is very important to state that the dynamic model in these space systems could avoid potential problems in their launch, saving time and money (Toorian et al., 2008).

In a satellite or a testbed, RBD is a good approach to study its attitude representation. This theory is applied when the movement of systems of interconnected bodies under the action of external forces is required (Marion, 1965).

Euler Angles are more understandable and helpful for decomposing rotations into individual degrees of freedom. However, they are difficult to compute due to the fact that they are represented by a 3×3 matrix for each dof. So, if we have n-dof, the computation will require n matrices, which will require more computer resources. They also allow a minimum representation with just their parameters and can have some singularities when the inverse problem is required (Kluever, 2018).

Lagrangian or Newtonian mechanics are useful to easily visualize the physics of the problem and to obtain the dynamical model. On the other hand, quaternions avoid singularities and are easy to compute requiring fewer resources to the computer. The main problem with quaternions is that rotation is difficult to visualize as they are analyzed as imaginary numbers (Familton, 2015).

Finally, Gavrilovich (2016) applied D-H theory to represent the translation of the CubeSat with a robotic arm manipulator by involving two different theories to solve a problem.

Since the ground test system was built for educational purposes Gutiérrez Medina (2015), we decided to obtain our dynamical model with RBD and EA, allowing an easier understanding of the physics behind it. The theory used is presented and explained in chapter 3.

Chapter 3

Preliminaries on attitude representation and control

In general, a mathematical model can be expressed as (Borjas, 2014):

$$y = f(x, \sigma, F) \tag{3.1}$$

where y is the dependent variable which gives the state or behavior of the systems, x is the independent variable that determines the dimension of the system's behavior, the parameters σ are the properties or compositions of the system, and the force function F are the external influences that act under the system. The general model can be a simple algebraic relation or a group of differential equations.

There are three main stages to the solution of mathematical models (Zavala, 2019): mathematical formulation of the problem, solution of the equations of motions and, the interpretation of the results.

The first step is to use the physical theory that better fits our systems. Therefore, it is important to observe our system (in our case, the CubeSat testbed). The theory used should be well-founded with simulations or experiments that were done before. The equations of motion can be solved by different methods such as analytical or numerical solutions. The path to follow depends on the tools we have at that moment. Finally, when we get the results it is important to interpret the solution. Graphs, tables, or comparisons between theory and experiments can be used to interpret the results. If the results are not similar to the expectations of the theory, the mathematical model should be reformulated.

The kinematic and the dynamic model are required for the study of multibody dynamics, mechanical system simulation, and the virtual prototyping model (Goldstein et al., 2002). Kinematics is the branch of mechanics that deals with the study of motion in which the only parameters considered are displacement, velocity, and acceleration. When other parameters such as mass, forces, momentum, etc. are considered the branch of mechanics is called dynamics (Jain and Nkoma, 2019). A dynamic model of a mechanical system (with the Lagrangian approach) can be represented as a generic form that is very useful in controls, which has the form (Kelly, 2006):

$$\mathrm{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} \tag{3.2}$$

where $\mathbf{q} \in \mathbb{R}^n$ is the position vector, $\dot{\mathbf{q}} \in \mathbb{R}^n$ is the velocity vector, and $\ddot{\mathbf{q}} \in \mathbb{R}^n$ is the acceleration vector. $\mathbf{M(q)} \in \mathbb{R}^{n \times m}$ is known as the matrix of mass and includes the inertia of the system, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n \times m}$ is called the Coriolis matrix and includes the Coriolis and centripetal forces, $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^n$ contains the gravitational torques and, $\boldsymbol{\tau} \in \mathbb{R}^n$ is a vector that includes all the external

torques or forces. Note that from this section, a vector will be represented as a bold letter while a scalar with a normal letter.

In this thesis project, the kinematic model has been obtained using two methodologies. The first approach was with robotics theory. Since the position of the reaction wheels in the testbed was similar to the position of the rotational joints in a robotic arm manipulator the first approximation was done with the Denavit-Hartenberg convention (Denavit, 1955). Nevertheless, since the system is closer to a rigid body than a robotic arm, the modeling was changed to the rotational body theory. On the other hand, the dynamic model was obtained applying the Euler equations of motion for rigid body theory from Newtonian and Lagrangian dynamics. Different derivations of rigid body dynamics such as the angular momentum, kinetic energy, and rigid body equation of motion can be found in literature (Cline, 2020; Goldstein et al., 2002; Wertz, 1978; Rimrott, 1989; Thomson, 1986; Wertz, 1978; Marion, 1965). The next sections show the theory required to obtain the dynamic model of our testbed.

3.1 General dynamics of a reaction wheel

The operation of the inertial wheel Figure 3.1 is related to that of an electrical motor. A motor transforms mechanical energy into movement. Its operating principle is based on the rotation of the rotor, which is an electromagnet that spins around an axis. The electromagnet motor is brushless and it generates a variable magnetic field depending on the electrical current that enters the system. The rotor has a fixed permanent magnet that interacts with the electromagnet and generates the circular movement of the rotor (Casaday, 1966).

Equation (3.3) is the general differential equation that describes a DC electrical motor.

$$\mathrm{I}\,\ddot{\mathbf{q}} + \mathbf{K_f}\,\dot{\mathbf{q}} = \boldsymbol{\tau} \tag{3.3}$$

where $\mathrm{I} \in \mathbb{R}^{n\times m}$ is its inertia matrix, $\ddot{\mathbf{q}}$ the angular acceleration, $\dot{\mathbf{q}}$ the angular velocity, $\mathbf{K_f}$ is a diagonal matrix with the constants of friction of the motor. If the magnitude of the torque is produced by an external current i then $\tau = k\,i$ where k is a constant that depends on each motor, and i is the electric current, $k, i \in \mathbb{R}$.

Equation (3.4), summarizes the principle of dynamics of a reaction wheel. This is an electrical motor that generates a variable angular momentum on the reaction wheel $\dot{\mathbf{L}}_{r\dot{q}} \in \mathbb{R}^n$ and, by definition a torque in the same axis ($\boldsymbol{\tau}_{r\dot{q}} \in \mathbb{R}^n$). Newton's second law states that when a body exerts a force, or for our case, a torque on a second body (Figure 3.1), the second body exerts a force of the same magnitude but in a different direction ($-\boldsymbol{\tau}_s$), so the satellite spin to the opposite direction (Beebi, 2016).

$$\boldsymbol{\tau}_{r\dot{q}} = \dot{\mathbf{L}}_{r\dot{q}} = -\boldsymbol{\tau}_s \tag{3.4}$$

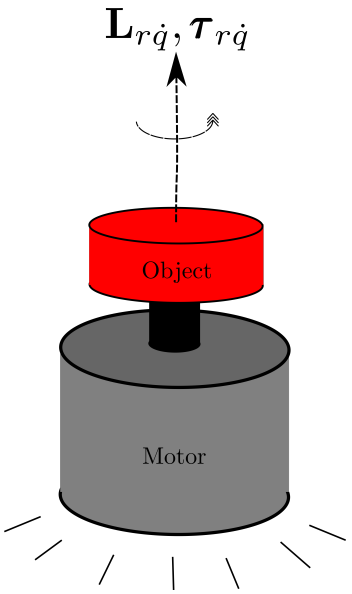


Figure 3.1: Reaction wheel torque diagram.

3.2 Attitude representation kinematics

Kinematics is a branch of physics that describes the motion of the bodies without considering the cause of the movement (Halliday et al., 2018). Several concepts of geometry are required to obtain the position, orientation, velocity, and acceleration of the body. The next section describes the concepts required to understand the kinematics of the testbed.

The attitude representation of a spacecraft is defined by its position, velocity, and orientation. The attitude motion describes the rotational motion of the system about its system of mass. There are different perspectives to represent the attitude of a space system (in our case a testbed) such as Euler angles, rotation matrix, and quaternions. Usually, the way of representing the attitude depends on the control objective. In this thesis work, we use Euler angles because they also provide easy visualization of 3D rotations. Most of this section is mainly based on literature (Cline, 2020; Wertz, 1978; Spong et al., 2006; Goldstein et al., 2002).

3.2.1 Position

A position vector can be defined as:

$$\mathbf{o} = o_x\,\mathbf{x} + o_y\,\mathbf{y} + o_z\,\mathbf{z} \tag{3.5}$$

where **x, y, z** are the unitary vectors in the reference framework and o_x, o_y, o_z are the magnitude of that component. The position of a point on the rigid body with respect to the coordinate frame O–xyz is shown in Figure (3.2) and is expressed by:

$$\mathbf{o\prime} = o'_x\,\mathbf{x} + o'_y\,\mathbf{y} + o'_z\,\mathbf{z} \tag{3.6}$$

where o'_x, o'_y, o'_z are the components of the vector $\mathbf{o\prime} \in \mathbb{R}^3$. Equation (3.6) can be expressed as a (3×1) matrix which contains the components as:

$$\mathbf{o\prime} = \begin{bmatrix} o'_x \\ o'_y \\ o'_z \end{bmatrix} \tag{3.7}$$

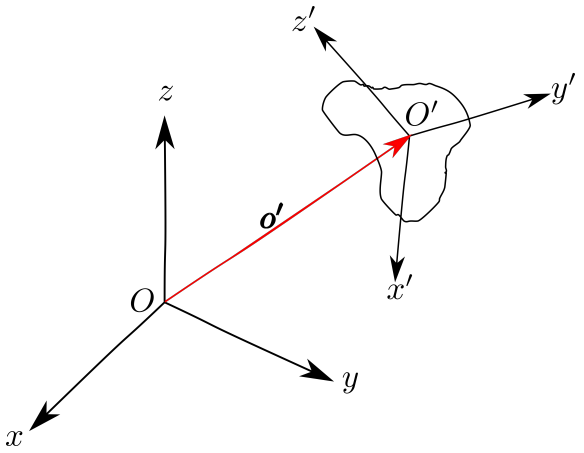


Figure 3.2: Position and orientation of a rigid body.

From this vector representation, we can express a point in space concerning a different framework.

3.2.2 Rotation matrix

In order to describe the rigid body orientation, it is convenient to consider an orthonormal $O' - x'y'z'$ frame attached to the body and to express its unit vectors with respect to the reference frame:

$$\mathbf{x}' = x'_x\mathbf{x} + x'_y\mathbf{y} + x'_z\mathbf{z} \tag{3.8}$$

$$\mathbf{y}' = y'_x\mathbf{x} + y'_y\mathbf{y} + y'_z\mathbf{z} \tag{3.9}$$

$$\mathbf{z}' = z'_x\mathbf{x} + z'_y\mathbf{y} + z'_z\mathbf{z} \tag{3.10}$$

In equations (3.8), (3.9) and (3.10), each unit vector (**x,y,z**) is a direction cosines of bodyframe $O' - x'y'z'$ with respect t $O - xyz$o.

We can define a 3×3 matrix where all the unit vectors are combined. Let's assume:

$$\mathrm{R} = \begin{bmatrix} \mathbf{x'} \\ \mathbf{y'} \\ \mathbf{z'} \end{bmatrix} = \begin{bmatrix} x'_x & y'_x & z'_x \\ x'_y & y'_y & z'_y \\ x'_z & y'_y & z'_z \end{bmatrix} = \begin{bmatrix} \mathbf{x}'^T\mathbf{x} & \mathbf{y}'^T\mathbf{x} & \mathbf{z}'^T\mathbf{x} \\ \mathbf{x}'^T\mathbf{y} & \mathbf{y}'^T\mathbf{y} & \mathbf{z}'^T\mathbf{y} \\ \mathbf{x}'^T\mathbf{z} & \mathbf{y}'^T\mathbf{z} & \mathbf{z}'^T\mathbf{z} \end{bmatrix} \tag{3.11}$$

Equation (3.11) is called a rotation matrix which is a transformation matrix that is used to rotate a vector in the Euclidean space. R is an orthogonal matrix $\in \mathbb{R}^{3\times3}$, which means that its columns and rows are orthogonal, and R is always a unitary norm.

From a geometric perspective, a rotation matrix is an operator that describes the orientation between two frameworks (Spong et al., 2006). Considering that a position vector is located at the inertial framework origin, a point in the space can be represented its components as:

$$\mathbf{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{3.12}$$

and with respect to $O - x'y'z'$ as:

$$\mathbf{p'} = \begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix}. \tag{3.13}$$

Therefore, as both vectors are the representation of the position of a point in space, we can use a transformation from one vector to another as:

$$\mathbf{p} = p'_x\mathbf{x}' + p'_y\mathbf{y}' + p'_z\mathbf{z}' = \begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \\ \mathbf{z}' \end{bmatrix} \mathbf{p}' = \mathrm{R}\mathbf{p}'. \tag{3.14}$$

From equation (3.14) we can obtain the inverse transformation:

$$\mathbf{p}' = \mathrm{R}^{\mathrm{T}}\mathbf{p}. \tag{3.15}$$

If the reference framework is rotated an angle α in the x axis then the rotation matrix is:

$$\mathrm{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{3.16}$$

To simplify notation, we define $\cos\alpha = c\alpha$ and $\sin\alpha = s\alpha$ for α or any other angle. Therefore, if the reference framework is rotated an angle β in the y axis and an angle γ in z the respectively rotation matrix are:

$$\mathrm{R}_y(\beta) = \begin{bmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{bmatrix} \tag{3.17}$$

$$\mathrm{R}_z(\gamma) = \begin{bmatrix} c\gamma & -s\gamma & 0 \\ s\gamma & c\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.18}$$

Matrices given in equations (3.16), (3.17) and (3.18) are known as elementary rotation matrices. They can be multiplied in any order, but as the multiplication of matrix is non-commutative, each order will represent a different rotation in space . Depending on the physical problem, some of the axes rotation should be done first. The geometric representation of the equations (3.16), (3.17) and (3.18) are respectively represented in Figures 3.3, 3.4 and 3.5.

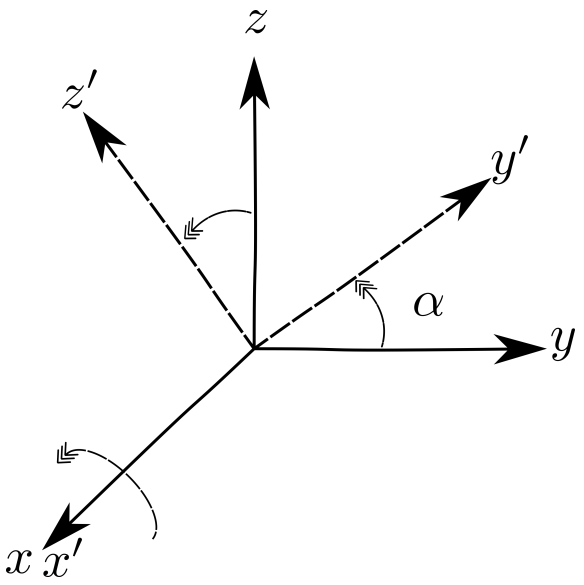


Figure 3.3: x-axis rotation.

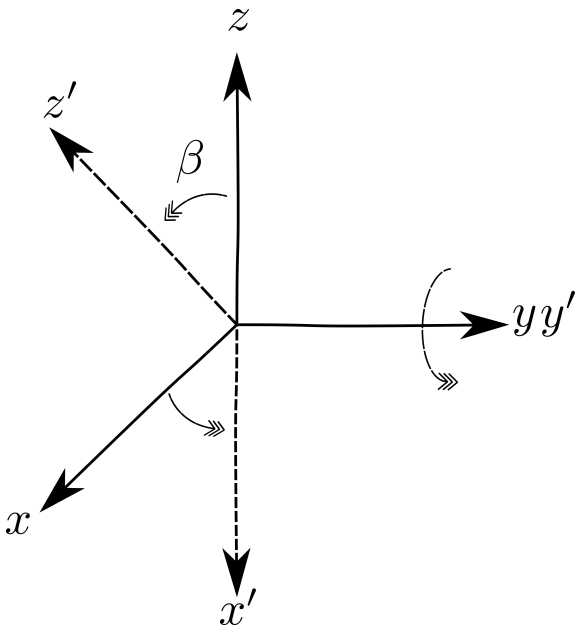


Figure 3.4: y-axis rotation.

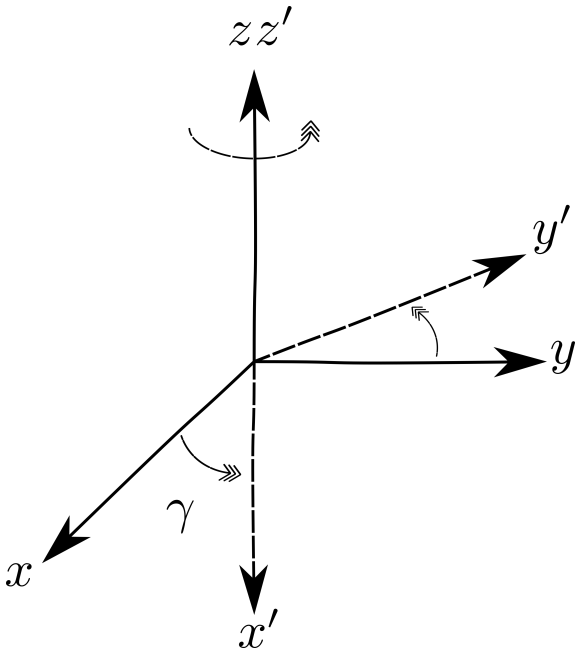


Figure 3.5: z-axis rotation.

In the example illustrated in Figures 3.3, 3.4 and 3.5, we rotated the system in the order z, y and x, that means $\mathrm{R}_z(\gamma)\mathrm{R}_y(\beta)\mathrm{R}_x(\alpha) = \mathrm{R}$ (Also known as ZYX representation) obtaining:

$$\mathrm{R} = \begin{bmatrix} c\beta\, c\gamma & -c\alpha s\gamma + s\alpha\, s\beta\, c\gamma & s\alpha\, s\gamma + c\alpha\, s\beta\, c\gamma \\ c\beta\, s\gamma & c\alpha\, c\gamma + s\alpha\, s\beta\, s\gamma & -s\alpha\, c\gamma + c\alpha\, s\beta\, s\gamma \\ -s\beta & s\,\alpha\, c\beta & c\alpha\, c\beta \end{bmatrix}. \tag{3.19}$$

Applying the rotation matrix to the vector represent by equation (3.12) we obtain:

$$\mathbf{p}' = \mathrm{R}\,\mathbf{p} \tag{3.20}$$

$$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} c\beta\, c\gamma & -c\alpha s\gamma + s\alpha\, s\beta\, c\gamma & s\alpha\, s\gamma + c\alpha\, s\beta\, c\gamma \\ c\beta\, s\gamma & c\alpha\, c\gamma + s\alpha\, s\beta\, s\gamma & -s\alpha\, c\gamma + c\alpha\, s\beta\, s\gamma \\ -s\beta & s\,\alpha\, c\beta & c\alpha\, c\beta \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}. \tag{3.21}$$

Equation (3.21) represents a general rotation of a vector in 3D space.

3.2.3 Euler angles

Since the rotation matrix requires nine parameters, it is necessary to simplify the problem assuming minimal representation such as Euler angles. In general, Euler angles locate the origin of the new mobile framework (usually the center of mass of the body) in terms of another fixed framework (Cline, 2020). The representation that we are going to use is the ZXZ representation, which is illustrated in the following example.

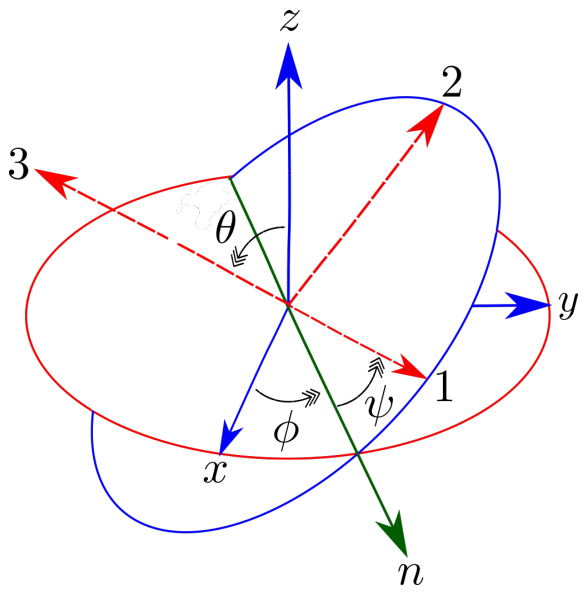


Figure 3.6: Euler angles general rotation.

The ZXZ sequence of rotations corresponds to the Euler angles (ϕ, θ, ψ) . The first rotation ϕ is about the space-fixed z axis (blue) from the x-axis (blue) to the line of nodes n (green). The second rotation θ about the line of nodes (green) is from the space-fixed z axis (blue) to the body-fixed 3-axis (red). The third rotation ψ about the body-fixed 3-axis (red) is from the line of nodes (green) to the body-fixed 1 axis (red).

Taking the rotation matrix, from equations (3.16), and (3.18), and replacing these new parameters we obtain:

$$\mathrm{R}_z(\phi) = \begin{bmatrix} c\phi & s\phi & 0 \\ -s\phi & c\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.22}$$

$$\mathrm{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\theta & s\theta \\ 0 & -s\theta & c\theta \end{bmatrix} \tag{3.23}$$

$$\mathrm{R}_z(\psi) = \begin{bmatrix} c\psi & s\psi & 0 \\ -s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.24}$$

with ϕ, θ and ψ the new rotation angles.

Again, we obtain a unique representation R by multiplying this matrix in the order ZXZ

$$\mathrm{R} = \mathrm{R}_z(\psi)\,\mathrm{R}_x(\theta)\,\mathrm{R}_z(\phi) = \begin{bmatrix} c\psi c\phi - s\psi c\theta s\phi & s\phi c\psi + c\theta s\phi c\psi & s\theta s\psi \\ -c\phi s\psi - c\theta s\phi c\psi & -s\phi s\psi + c\theta c\phi c\psi & s\theta c\psi \\ s\theta s\phi & -s\theta c\phi & c\theta \end{bmatrix}. \tag{3.25}$$

The position or velocity of any vector in the new reference framework can be obtained by multiplying the vector components by the matrix. Equation (3.25) is the final representation for 3D rotation using Euler Angles ZXZ. Figure 3.7 shows the ZXZ as single rotation.

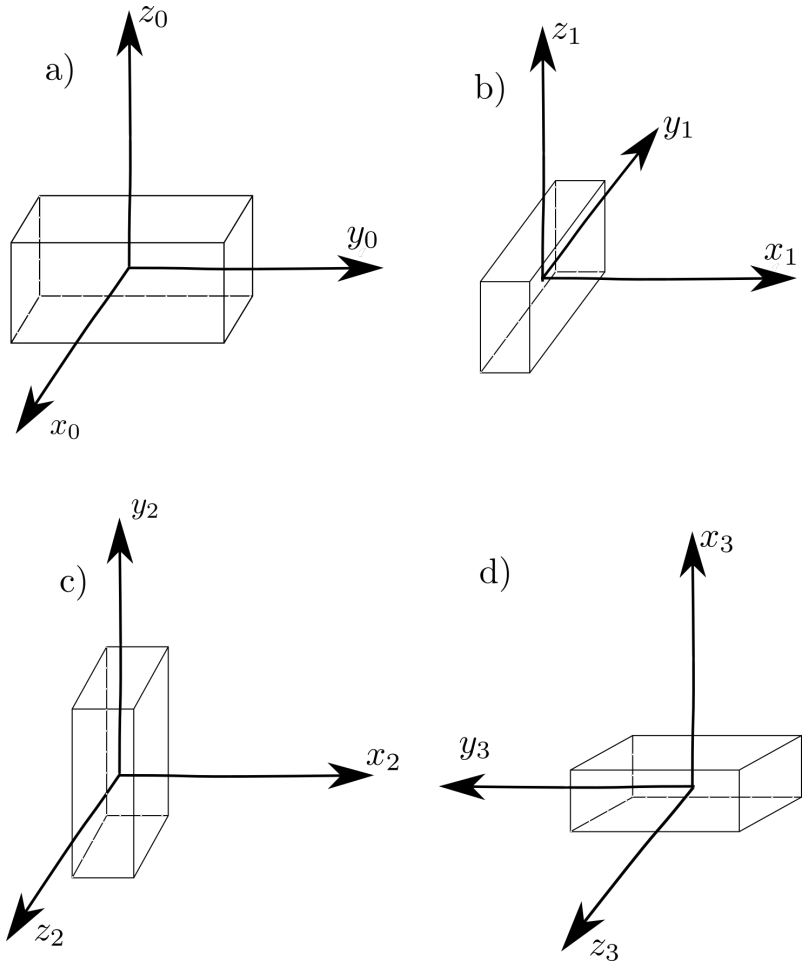


Figure 3.7: Euler angles ZXZ single rotations. a) Initial framework. b) Rotation of $\psi = \pi/2$ in the z_0-axis.c) Rotation of $\theta = \pi/2$ along the x_1-axis. d) Rotation of $\phi = \pi/2$ in the z_3-axis and shows the final orientation.

Angular velocities using Euler Angles

Since the orientation is represented by the Euler angles, the angular velocity has to be expressed in terms of these angles and their derivatives. From Figure (3.6) we observe that $\dot{\boldsymbol{\phi}}$ is about the z-axis, where 3-axis is located at angle θ to the z-axis so this vector as a component $\dot{\phi}c\theta$ in the 3-axis and in the plane $1, 2$ a component $\dot{\phi}s\theta$ and the other component along a line perpendicular to x-axis and the line of node at angle ϕ therefore $\dot{\boldsymbol{\phi}} = (\dot{\phi}s\theta c\psi, \dot{\phi}s\theta c\psi, \dot{\phi})$. For $\dot{\boldsymbol{\theta}}$ is along the x-axis and the nodes line, and as a consequence in the $1, 2$ plane (which is at an angle ψ with respect to 1-axis), having $\dot{\boldsymbol{\theta}} = (\dot{\theta}c\psi, -\dot{\theta}s\psi, 0)$. Finally the angular velocity $\dot{\boldsymbol{\psi}}$ as only a component in the 3-axis so $\dot{\boldsymbol{\psi}} = (0, 0, \dot{\psi})$.

Now we express the component of the Euler angles velocities along with the rotational frame $O - 1, 2, 3$

$$\dot{q}_1 = \dot{\phi}_1 + \dot{\theta}_1 + \dot{\psi}_1 = \dot{\phi}\, s\theta\, s\psi + \dot{\theta}\, c\psi \tag{3.26}$$

$$\dot{q}_2 = \dot{\phi}_2 + \dot{\theta}_2 + \dot{\psi}_2 = \dot{\phi}\, s\theta\, c\psi - \dot{\theta}\, s\psi \tag{3.27}$$

$$\dot{q}_3 = \dot{\phi}_3 + \dot{\theta}_3 + \dot{\psi}_3 = \dot{\phi}\, c\theta + \dot{\psi} \tag{3.28}$$

where superscript represents the order of the rotation. The angle $\dot{\phi}$ is known as precession, $\dot{\theta}$ as nutation and $\dot{\psi}$ as spin. Equations (3.26), (3.27) and (3.28) will be used to derive equations of motion from Lagrangian theory and can be expresed as a final vector:

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \begin{bmatrix} s\theta\psi & c\psi & 0 \\ s\theta c\psi & -s\psi & 0 \\ c\theta & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}. \tag{3.29}$$

3.2.4 Denavit-Hartenberg convention

For the first approach to obtain the kinematic model, we used the Denavit-Hartenberg (D-H) convention (Denavit, 1955). In robotics arm manipulators each link is represented as an independent rigid body and they are connected by joints that can be prismatic or rotational. A robot manipulator with n joints will have $n+1$ links. The links are numbered from 0 to n starting from the base. The D-H convention allows obtaining the direct kinematics of rigid robotics. In this convention, each homogeneous transformation is represented as a 4×4 matrix A_i. The final matrix A_i is a product of four basic transformations (Spong et al., 2006), with a 3×3 matrix containing the information about the orientation and a 1×3 matrix that represents the position in space of the final framework.

$$\mathrm{A}_i = \mathrm{R}_z(q_i)\mathrm{T}_z(d_i)\mathrm{T}_x(a_i)\mathrm{R}_x(\alpha_i) \tag{3.30}$$

Where T is a translational matrix.

$$\mathrm{A}_i = \begin{bmatrix} c\,q_i & -s\,q_i & 0 & 0 \\ s\,q_i & c\,q_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_i & -s\alpha_i & 0 \\ 0 & s\alpha_i & c\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_i = \begin{bmatrix} cq_i & -sq_i\,c\alpha_i & sq_i\,s\alpha_i & a_i\,c\,q_i \\ sq_i & cq_i\,c\alpha_i & cq_i\,s\alpha_i & a_i\,s\,q_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.31}$$

where q_i, α_i, d_i, and a_i the link length, the joint angle, the link twist, and the link offset respectively.

The steps to follow the the D-H convention are (Siciliano et al., 2008):

1. Find and number consecutively the joint axes; set the directions of axes $z_0, ..., z_{n-1}$.
2. Choose frame 0 by locating the origin on-axis z_0; axes x_0 and y_0 are chosen to obtain a right-handed frame. If feasible, it is worth choosing frame 0 to coincide with the base frame.

   Execute steps from 3 to 5 for $i = 1, ..., n-1$:
3. Locate the origin O_i at the intersection of z_i with the common normal to axes z_{i-1} and z_i. If axes z_{i-1} and z_i are parallel and joint i is revolute, then locate O_i so that $d_i = 0$; if joint i is prismatic, locate O_i at a reference position for the joint range, e.g., a mechanical limit.
4. Choose axis x_i along the common normal to axes z_{i-1} and z_i with direction from joint i to Joint $i+1$.
5. Choose axis y_i to obtain a right-handed frame.

   To complete:
6. Choose frame n; if joint n is revolute, then align z_n with z_{n-1}, otherwise, if joint n is prismatic, then choose z_n arbitrarily. Axis x_n is set according to step 4.
7. For $i = 1, ..., n$, form the table of parameters a_i, d_i, α_i, q_i.

8. On the basis of the parameters in 7, compute the homogeneous transformation matrices $\mathrm{A}_i^{i-1}(qi)$ for$i = 1, ..., n$.
9. Compute the homogeneous transformation $\mathrm{H}_n^0(q) = \mathrm{A}_1^0 ... \mathrm{A}_i^{i-1}$ that yields the position and orientation of Frame n with respect to frame 0.
10. Given H_0^b and H_e^n, compute the direct kinematics function as $\mathrm{H}_e^b(q) = \mathrm{H}_0^b \, \mathrm{H}_n^0 \, \mathrm{H}_e^n$ that yields the position and orientation of the end-effector frame with respect to the base frame.

Once we get the kinematics we can obtain the workspace of the systems, which is the set of points in space that can be reached by the end-effector.

3.3 Attitude dynamics

Dynamics is a branch of physics that studies the movement of an object and those physical quantities that affect its movement such as the forces and torques (Halliday et al., 2018). For this thesis project, we focus on the rigid body theory to compute its orientation.

3.3.1 Inertia tensor

In general, a rigid body rotation diagram can be represented with the Figure 3.8.

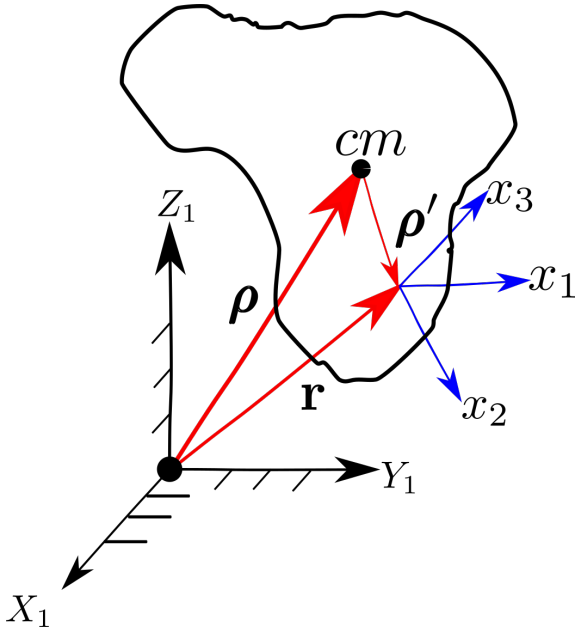


Figure 3.8: Frames involved in a Rigid Body without rotation on its center.

where cm is the center of mass $\boldsymbol{\rho} \in \mathbb{R}^3$ is the vector from the base to the cm, $\boldsymbol{\rho\prime} \in \mathbb{R}^3$ is the vector from the cm to the body frame and $\mathbf{r} \in \mathbb{R}^3$ is the vector from the cr to the body frame.

The inertia tensor is a symmetric tensor of second order that contains the information of the rotational inertia in a rigid body. It is expressed by a 3×3 symmetric matrix (Cline, 2020). This tensor is conformed by the moments of inertia according to the perpendicular axis and three products of inertia. The inertia tensor is defined as (Goldstein et al., 2002):

$$\mathbf{I} = I_{ij} = \sum_{l=1}^{n} m_l \left[\delta_{ij} (\sum_{k=1}^{3} x_{l,k}^2) - x_{l,i} \, x_{l,j} \right] \tag{3.32}$$

where the superscript l is the mass or the position of the $l-th$ particle, $\mathbf{r} = (x_{l1}, x_{l2}, x_{l3})$ which are the components of the vector $\mathbf{r}$ for the $l-th$ particle.

Or as an integral:

$$I_{ij} = \int_V \tilde{\rho}\,(\boldsymbol{\rho'})[\delta_{ij}\sum_{k=1}^{3} x^2 - x_i x_j]dV \tag{3.33}$$

Where $\tilde{\rho}$ is the density of the body and δ_{ij} is the Kronecker's delta.

$$\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{3.34}$$

The elements I_{ij} are called the moments or products of inertia. From equation (3.32):

$$I_{11} = \sum_{l=1}^{n} m_l\,[(x_{l2})^2 + (x_{l2})^2] \tag{3.35}$$

$$I_{22} = \sum_{l=1}^{n} m_l\,[(x_{l3})^2 + (x_{l1})^2] \tag{3.36}$$

$$I_{33} = \sum_{l=1}^{n} m_l\,[(x_{l1})^2 + (x_{l2})^2] \tag{3.37}$$

$$I_{12} = I_{21} = \sum_{l=1}^{n} m_l\,(x_{l1}\,x_{l2}) \tag{3.38}$$

$$I_{23} = I_{32} = \sum_{l=1}^{n} m_l\,(x_{l2}\,x_{l3}) \tag{3.39}$$

$$I_{31} = I_{13} = \sum_{l=1}^{n} m_l\,(x_{l3}\,x_{l1}) \tag{3.40}$$

It is important to note that the products of inertia are symmetric, which means that:

$$I_{ij} = I_{ji}. \tag{3.41}$$

3.3.2 Principal axes of inertia

When the coordinate system is chosen to get a real symmetric matrix, it means that all the products of inertia are zero and the diagonal terms are called principal axes or principal moments of inertia.

$$I_{ij} = I_i\,\delta_{ij} \tag{3.42}$$

or,

$$I = \begin{bmatrix} I_{11} & 0 & 0 \\ 0 & I_{22} & 0 \\ 0 & 0 & I_{33} \end{bmatrix} = \begin{bmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{bmatrix}. \tag{3.43}$$

By definition (Goldstein et al., 2002), the sum of two of the principal moments of inertia is not greater than the other, that is:

$$I_j + I_k \leq I_l \tag{3.44}$$

where $j, k, l = 1, 2, 3$

3.3.3 Diagonalization of the Inertia Tensor

If we have the inertia tensor with its nine parameters and we want to find the three principal axes of inertia, we can diagonalize the tensor. In other words, the inertia tensor is a symmetric tensor of rank 2, which means it can be diagonalized, which became an eigenvalue problem (Marion, 1965). For example, let's assume we have the next equation:

$$\mathbf{I} \cdot \dot{\mathbf{q}} = I\dot{\mathbf{q}} \tag{3.45}$$

With I the diagonal elements of the inertia tensor as the eigenvalues and $\dot{\mathbf{q}}$ the eigenvector. The solution to equation (3.45) became:

$$\sum_{j}^{n}(I_{ij} - I\delta_{ij})q_j = 0 \tag{3.46}$$

It is well known, that the solution of the equation (3.46) is not zero if and only if the determinant of the left side is zero, that is:

$$\det(\mathbf{I} - \mathbb{1}\mathrm{I}) = 0 \tag{3.47}$$

where $\mathbb{1}$ is a 3×3 identity matrix.

Finally, the inertia from equation (3.45) is:

$$\begin{bmatrix} (I_{11} - I) & I_{12} & I_{13} \\ I_{21} & (I_{22} - I) & I_{23} \\ I_{31} & I_{32} & (I_{33} - I) \end{bmatrix} = 0. \tag{3.48}$$

Equation (3.48) is the diagonalized inertia tensor.

3.3.4 Parallel axis theorem

In general, it is important to choose a coordinate system that is the center of mass of the body to reduce external torques. Nevertheless, finding the center of mass could be difficult for certain body shapes or mass distributions. As a consequence, define a new set of axis fixed X to the body but with a new origin and the same orientation as the inertial framework x_0. Then, similarly to equation (3.32), we have:

$$J_{ij} = \sum_{l=1}^{n} m_l \left[\delta_{ij} (\sum_{k=1}^{3} X_{l,k}^2) - X_{l,i} X_{l,j} \right] \tag{3.49}$$

where:

$$X_i = \rho_i + x_i \tag{3.50}$$

Substituting the equation (3.50) in (3.49) we obtain:

$$J_{ij} = \sum_{l=1}^{n} m_l \left[\delta_{ij} (\sum_{k=1}^{3} x_{l,k} + \rho_i)^2 - (x_{l,i\rho i}) \, (x_{l,j} + \rho_i) \right]$$

$$J_{ij} = \sum_{l=1}^{n} m_l \left[\delta_{ij} (\sum_{k=1}^{3} x_{l,k}^2) - x_{l,i} \, x_{l,j} \right] + \sum_{l=1}^{n} m_l \left[\delta_{ij} (\sum_{k=1}^{3} (2 x_{l,k} \, \rho_i + \rho_i^2 + (x_{l,j\rho i}) + (x_{l,i\rho j}) \, \rho_i \, \rho_j) \right] \quad (3.51)$$

Hence:

$$J_{ij} = I_{ij} + \sum_{l=1}^{n} m_l \left(\delta_{ij} (\sum_{k=1}^{3} \rho_k^2 - \rho_i \, \rho_j) + \sum_{l=1}^{n} m_l (2 \, \delta_{ij} \, (\sum_{k=1}^{3} (x_{l,k} \, \rho_k - (x_{l,j\rho i}) - (x_{l,i\rho j}) \, \rho_i \, \rho_j) \right] \quad (3.52)$$

Now, if we put our new coordinate system in the center of mass of the body:

$$\sum_{l=1}^{n} m_l \boldsymbol{\rho\prime} = \sum_{l=1}^{n} m_l x_{l,k} = 0 \quad (3.53)$$

Therefore:

$$J_{ij} = I_{ij} + \sum_{l=1}^{n} m_l \left(\delta_{ij} (\sum_{k=1}^{3} \rho_k^2 - \rho_i \, \rho_j) = \right. \quad (3.54)$$

or:

$$I_{ij} = J_{ij} - M(\rho^2 \, \delta_{ij} + \rho_i \, \rho_j) \quad (3.55)$$

This allows the estimation of the elements of the inertia tensor (with its origin in the center of mass) concerning the X_i-axes known. Equation 3.55 is also known as Steiner's parallel axis theorem.

3.3.5 Kinetic energy and angular momentum in a rigid body

From Figure 3.8, we define a vector as:

$$\mathbf{r} = \boldsymbol{\rho} + \boldsymbol{\rho}' \quad (3.56)$$

On another hand, the angular momentum can be obtained as:

$$\mathbf{L} = \mathbf{r} \times \mathbf{p} = \mathbf{r} \times m \, \mathbf{v} \quad (3.57)$$

where $\mathbf{p} \in \mathbb{R}^3$ is the linear momentum, m is the mass of a single particle and $\mathbf{v} \in \mathbb{R}^3$ is the tangential velocity. Now, for many particles we have:

$$\mathbf{L} = \sum_{i=1}^{n} \mathbf{L}_i = \sum_{i=1}^{n} \mathbf{r}_i \times \mathbf{p}_i \quad (3.58)$$

Then we have the velocity as:

$$\dot{\mathbf{r}} = \dot{\boldsymbol{\rho}} + \dot{\boldsymbol{\rho}\prime} \quad (3.59)$$

Substituting equations (3.59) and (3.58) in (3.57) we obtain:

$$\mathbf{L}_T = M\,\boldsymbol{\rho}\times\dot{\boldsymbol{\rho}} + \boldsymbol{\rho}\times\sum_{i=1}^{n}\dot{m}_i\dot{\boldsymbol{\rho}}'_i + \sum_{i=1}^{n} m_i\boldsymbol{\rho}'_i\times\dot{\boldsymbol{\rho}} + \sum_{i=1}^{n} m_i\boldsymbol{\rho}'_i\times\dot{\boldsymbol{\rho}}'_i \tag{3.60}$$

with M the total mass of the body. If we take the body framework to the center of mass of the body then $\sum_{i=1}^{n} m_i\boldsymbol{\rho}'_i = 0$ and the total angular momentum is given by:

$$\mathbf{L}_T = M\,\mathbf{r}\times\dot{\mathbf{r}} + \mathbf{L} \tag{3.61}$$

The left side of (3.60) would be the angular momentum of the cm.

$$\mathbf{L} = \sum_{i=1}^{n} m_i\,\boldsymbol{\rho}'_i\times\dot{\boldsymbol{\rho}}'_i \tag{3.62}$$

Similarly, we obtain the total kinetic energy as:

$$K = \frac{1}{2}\sum_{i=1}^{n} m_i\,\dot{\mathbf{r}}_i^2 \tag{3.63}$$

$$K = \frac{1}{2}M\dot{\mathbf{r}}^2 + \dot{\boldsymbol{\rho}}\cdot\sum_{i=1}^{n}\dot{m}_i\,\dot{\boldsymbol{\rho}}'_i + \frac{1}{2}\sum_{i=1}^{n} m_i\,\dot{\boldsymbol{\rho}}_i'^2 \tag{3.64}$$

If we move the body framework to the center of mass then:

$$K = \frac{1}{2}M\dot{\boldsymbol{\rho}}^2 + E_k \tag{3.65}$$

with:

$$E_k = \frac{1}{2}\sum_{i=1}^{n} m_i\,\dot{\boldsymbol{\rho}}_i'^2 \tag{3.66}$$

On the other hand, we know that:

$$\dot{\boldsymbol{\rho}}'_i = \dot{\boldsymbol{q}}\times\boldsymbol{\rho}'_i \tag{3.67}$$

where $\dot{\boldsymbol{q}}$ is the angular velocity vector.

Using equation (3.62) then:

$$\mathbf{L} = \sum_{i=1}^{n} m_i\,\boldsymbol{\rho}'_i\times(\dot{\boldsymbol{q}}\times\boldsymbol{\rho}'_i) = \sum_{i=1}^{n} m_i\,[\boldsymbol{\rho}_i'^2\,\dot{\boldsymbol{q}} - (\boldsymbol{\rho}'_i\cdot\dot{\boldsymbol{q}})\,\boldsymbol{\rho}'_i] \tag{3.68}$$

Now, identifying the right side of equation (3.68) as the component of the inertia tensor, equation (3.57) can be rewritten as:

$$\mathbf{L} = \mathbf{I}\,\dot{\mathbf{q}} \tag{3.69}$$

Combining equations (3.67) and (3.66):

$$E_k = \frac{1}{2}\sum_{i=1}^{n} m_i \left(\dot{\boldsymbol{q}} \times \boldsymbol{\rho}_i'\right) \cdot \left(\dot{\boldsymbol{q}} \times \boldsymbol{\rho}_i'\right) = \frac{1}{2}\dot{\boldsymbol{q}} \cdot \mathbf{L} = \frac{1}{2}\dot{\boldsymbol{q}}^T \mathbf{I}\, \dot{\boldsymbol{q}} \tag{3.70}$$

If we have the principal axis of inertia, therefore equation (3.70) becomes:

$$E_k = \frac{1}{2}(I_1\, \dot{q}_1^2 + I_2\, \dot{q}_2^2 + I_3\, \dot{q}_3^2) \tag{3.71}$$

where $\dot{q}_1, \dot{q}_2$ and $\dot{q}_3$ are the components of the angular velocity vector q.

Or, in general, in terms of angular momentum.

$$E_k = \frac{1}{2}\mathbf{L}^T\, \mathbf{I}^{-1}\, \mathbf{L} \tag{3.72}$$

3.3.6 Time derivative in rotating frames

If we have a cosine directors matrix A which is rotating with constant angular velocity $\dot{\mathbf{q}}$ the rate of change of A is:

$$\dot{\mathrm{A}} = \dot{\mathbf{q}} \times \mathrm{A} \tag{3.73}$$

On the other hand, if we can express the variation in time of a vector in a reference system $\mathbf{a}'$ along one coordinate axis in a different system, let's say in a new body frame, denoted by $\mathbf{a}$ then with a transformation matrix A we have:

$$\mathbf{a} = \mathrm{A}\, \mathbf{a}' \tag{3.74}$$

Now, if we want the time derivative of $\mathbf{a}$, therefore:

$$\dot{\mathbf{a}} = \mathbf{a}' + \mathrm{A}\, \dot{\mathbf{a}}' \tag{3.75}$$

Which from the rotational matrix properties (See C) we obtain:

$$\boxed{\dot{\mathbf{a}} = (\dot{\mathbf{a}}')_b + \dot{\mathbf{q}} \times \mathrm{A}} \tag{3.76}$$

Equation (3.76) is known as the transport theorem (Cline, 2020).

3.4 Proportional Control plus Velocity Feedback

The development of new technology has advanced the study of physical systems. Mathematical modeling allows us to understand the behavior of these systems. A controller $\boldsymbol{\tau_c}$ is a control loop mechanism feedback where a desired position q_d is introduced to the system (Figure 3.9). To validate our model, the physical parameters can be modified in the control law, if the simulation is different from what we are expected then the parameters of the model should be changed (Ljung, 2000). The main purpose of a controller is to reduce to zero the error $\tilde{\mathbf{q}}(\mathbf{t}) = \mathbf{q_d} - \mathbf{q}(\mathbf{t})$ of the system, that is:

$$\lim_{t \to \infty} \tilde{\mathbf{q}}(\mathbf{t}) = 0 \tag{3.77}$$

or:

$$\lim_{t\to\infty} \mathbf{q(t)} = \mathbf{q_d} \tag{3.78}$$

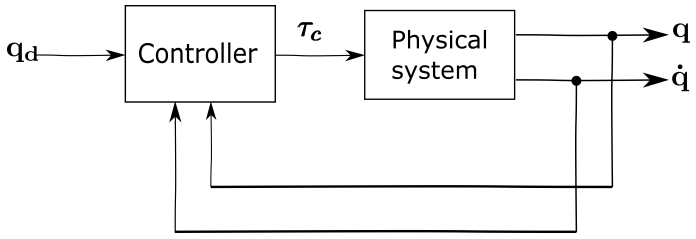


Figure 3.9: Position controller block diagram. Taken and adapted from Kelly (2006).

The final dynamic equation is obtained by adding the action of $\boldsymbol{\tau_c}$ to the dynamic equations of the system. In the state form, the closed-loop equation has the form:

$$\frac{d}{dt}\begin{bmatrix}\mathbf{q_d} - \mathbf{q} \\ \dot{\mathbf{q}}\end{bmatrix} = f(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{q_d}, \mathbf{M(q)}, \mathbf{C(q, \dot{q})}, \mathbf{g(q)}) \tag{3.79}$$

Depending on the application of our physical system, different controllers can be applied. For this thesis, a PD controller, specifically a proportional control plus velocity feedback was used.The control law is represented by:

$$\boldsymbol{\tau_c} = \mathbf{K_p}\,\tilde{\mathbf{q}} - \mathbf{K_v}\,\dot{\mathbf{q}} \tag{3.80}$$

where, for our case, $\mathbf{K_p} = diag[k_{p1}, k_{p2}, k_{p3}]$ and $\mathbf{K_v} = diag[k_{p1}k_{p2}, k_{p3}]$ with components $\in \mathbb{R}$ and are constants usually called position and velocity gains. Figure (3.10) represents the block-diagram of the control system.

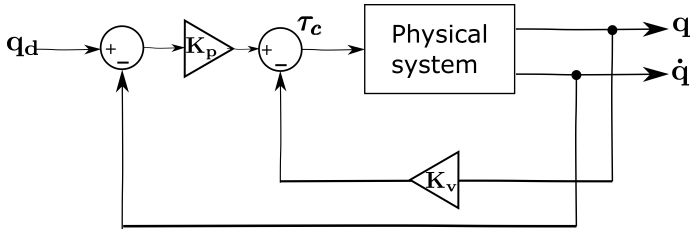


Figure 3.10: Proportional and velocity feedback controller. Taken and adapted from Kelly (2006).

Equation (3.80) was used to validate our dynamics equation in the next section. If the behavior of the physical parameters is not desired, then the mathematical model could be wrong or the input parameters should be modified. Nevertheless, for our case, the input data (the mass and dimensions of the testbed and the reaction wheels) are values previously registered. Therefore, if we have a different expected behavior of our system, we would need to compute a new equation of motion.

Chapter 4

Mathematical modeling

At the beginning of this research project, the mathematical model was computed by applying robotics theory because the position of the reaction wheel in the testbed was similar to a spherical wrist. However, the rigid body theory has a closer approach to our problem. The next section describes how the kinematic model was obtained with robotics and rigid body theory, once we obtain the kinematics we were able to plot the workspace. The dynamic model was obtained with rigid body dynamics from Newtonian and Lagrangian mechanics.

4.1 Kinematic model

Spherical wrist perspective

As the first approach, the kinematic model of the platform was estimated, assuming that it is similar to a spherical wrist of a robot manipulator. The testbed has two reaction wheels placed at two axes. Each reaction wheel represents a rotational joint from the point of view of a robotic arm manipulator. The testbed has only two reactions wheels, which means two degrees of freedom. Nevertheless, our general model has three rotational joints (Figure 4.1). We fixed one of the axes to obtain two degrees of freedom by keeping constant one of the rotational matrices in the D-H methodology. Finally, only one rotation was allowed in the model to compute the 1-dof model.

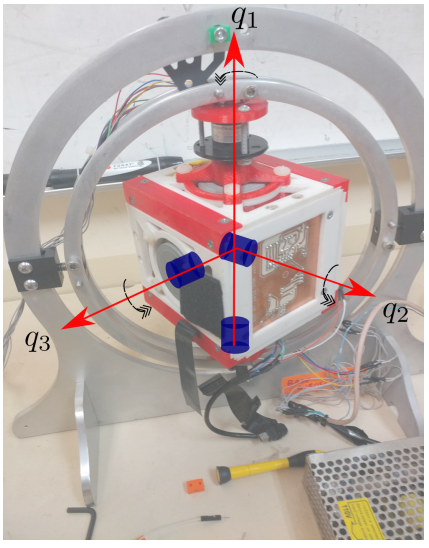


Figure 4.1: The blue cylinders represent the position of the reaction wheels and q_1, q_2 and q_3 are the different components of the angular velocity.

The kinematics model can be written as $x_i = f(q)$ for $i = 1, 2, 3$, where the function $f(q_i)$ is a homogeneous matrix that has information about the orientation and the position of the end effector and is represented by equation (4.1) and x_i are the components of x, y, z. We want to track the position and orientation of the end effector to control its pointing, so in the ground test system, the last frame of reference $O - x_3\, y_3\, z_3$ is represented by an "antenna". The method used to obtain the homogeneous transformation matrix was the D-H methodology presented in chapter 3. We chose the initial frame $O - x_0\, y_0\, z_0$ in the basis of one inertial wheel, the next framework $O - x_1\, y_1\, z_1$ was located in the next inertial wheel from a distance d_1 after the rotation $-\pi/2$ the angle α_1. The frame $O - x_2\, y_2\, z_2$ was estimated rotating α_2 by an angle of $\pi/2$ and having a link offset a_2. Finally, the last framework $O - x_3\, y_3\, z_3$ was obtained moving the last framework by a distance d_3. Figure 4.2 shows the position and orientation of the rotational joints used. The D-H parameters are presented in table (4.1). On the other hand, Figure 4.2, represents the spherical wrist and its respective frameworks according to the D-H convention.

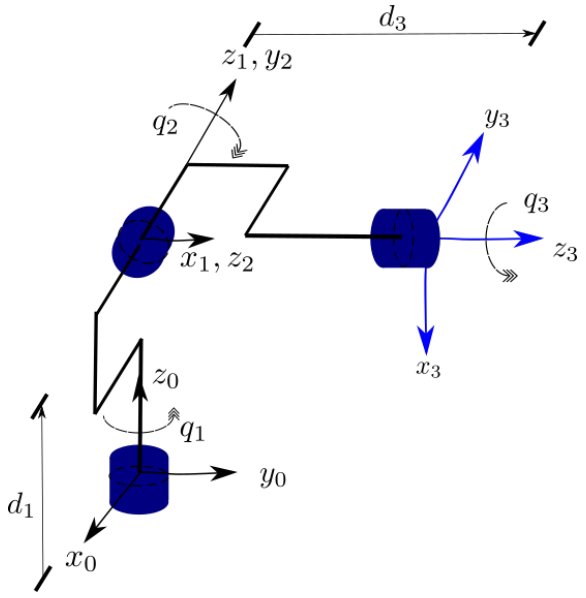


Figure 4.2: Orientation and position of frame of references in the spherical wrist.The blue cylinders represent the rotational joints while q_1, q_2 and q_3 are the different components of the angular velocity. The frameworks and the desplacementes follows de D-H methodology.

Link	q_i	d_i	a_i	α_i
A1	q_1	d_1	0	$\frac{-\pi}{2}$
A2	q_2	0	a_2	$\frac{\pi}{2}$
A3	q_3	d_3	0	0

Table 4.1: D-H parameters of the platform.

These parameters, were necessary to obtain the general homogeneous transformation matrix represented by equation (4.1). q_i are variable spins in the rotational joints. Therefore for the general case, the kinematic model is represented by the homogeneous matrix:

$$A = \begin{bmatrix} cq_1cq_2cq_3 - sq_1sq_3 & -cq_3sq_1 - cq_1cq_2sq_3 & cq_1sq_2 & a_2cq_1cq_2 + d_3cq_1sq_2 \\ cq_2cq_3sq_1 - cq_1sq_3 & -cq_1cq_3 - cq_2sq_1sq_3 & sq_1sq_2 & a_2cq_2sq_1 + d_3sq_1sq_2 \\ -cq_3sq_2 & sq_2sq_3 & -cq_2 & d_1 - d_3sq_2 - a_2cq_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

therefore the final position is given by:

$$\mathbf{r} = \begin{bmatrix} a_2 cq_1 cq_2 + d_3 cq_1 sq_2 \\ a_2 cq_2 sq_1 + d_3 sq_1 sq_2 \\ d_1 - d_3 sq_2 - a_2 cq_2 \end{bmatrix} \tag{4.2}$$

while the orientation of the final position is given by:

$$R = \begin{bmatrix} cq_1 cq_2 cq_3 - sq_1 sq_3 & -cq_3 sq_1 - cq_1 cq_2 sq_3 & cq_1 sq_2 \\ cq_2 cq_3 sq_1 - cq_1 sq_3 & -cq_1 cq_3 - cq_2 sq_1 sq_3 & sq_1 sq_2 \\ -cq_3 sq_2 & sq_2 sq_3 & -cq_2 \end{bmatrix}. \tag{4.3}$$

Note that for 2-dof and 1-dof one or two q were fixed it means that $q = 0$, and that for our case, d_1=a_2=d_3=5 cm.

Euler angles perspective

Another and the simplest way to obtain the kinematics model is using Euler Angles. Applying the theory shown in Chapter 3, we have a rotation matrix for each rotation in our testbed. Note that our testbed is seen as a rigid body. Now we define a vector $\mathbf{r}$ from the center of the body to the position of our final framework $O - x_3\, y_3\, z_3$ (Figure 4.3) and we multiply it by equation (3.25) obtaining the kinematic model (where to generalize $\psi = q_1, \theta = q_2$ and $\phi = q_3$).

$$\mathbf{r}' = \mathrm{R}\mathbf{r} \tag{4.4}$$

If we assume $\mathbf{r}(r_1, r_2, r_3)$ then:

$$\mathbf{r}' = \begin{bmatrix} r_1(cq_1 c_{q3} - sq_1 cq_2 sq_3) + r_2(cq_1 sq_3 + cq_2 cq_3 sq_1) + r_3(sq_1 sq_2) \\ r_3(cq_1 sq_2) - r_2(sq_1 sq_3 - cq_1 cq_2 cq_3) - r_1(cq_3 sq_1 + cq_1 cq_2 sq_3) \\ r_3 cq_2 - r_2 cq_3 sq_2 + r_1 sq_2 sq_3 \end{bmatrix} \tag{4.5}$$

where the components of the vector $\mathbf{r}$ depend on the position of the center of mass.

The main difference is that with this theory our fixed framework is placed in the center of the rotation of the body, in contrast with robotics theory, where the body-fixed framework is placed in one of the faces of the cube.

4.2 Dynamic model

The dynamic equations of motion were computed by applying the Newtonian and Lagrangian approaches to obtain Euler's equations of motion for rigid bodies. We applied these equations to our CubeSat testbed.

Newton's theory

From Newton's Law, if we have the principal axis of inertia or the inertia from the center of rotation (which can be diagonalized), then the time derivative of equation (3.69) is (Cline, 2020):

$$\frac{d\,\mathbf{L}}{dt} = \frac{d}{dt}(\mathrm{I} \cdot \dot{\mathbf{q}}) = \boldsymbol{\tau} - \dot{\mathbf{q}} \times \mathbf{L} \tag{4.6}$$

where I is the inertia tensor, $\dot{\mathbf{q}}$ is the angular velocity vector and $\boldsymbol{\tau}$ is the associated torque as:

$$\boldsymbol{\tau} = \sum_{i=1}^{n} \mathbf{r_i} \times \mathbf{F}_i \tag{4.7}$$

where $\mathbf{F}_i$ is a force total force. The torque due to the component of an external force $\mathbf{F}_i^e$ in the center of mass and all the internal forces due to the attraction of the particles $\mathbf{f}_{ij}$. Consequently:

$$\boldsymbol{\tau} = \sum_{i=1}^{n} \mathbf{r_i} \times \mathbf{F}_i^e \sum_{i=1}^{n} \sum_{i=1, j\neq 1}^{n} \mathbf{r_i} \times \mathbf{f}_i j \tag{4.8}$$

By Newton's third law, the terms of the right side of the previous equation are zero, therefore the torque only depends on the external forces. Applying third Newton's law to equation (4.6) we obtain:

$$\mathbf{I}\dot{\mathbf{q}} = \boldsymbol{\tau} - \dot{\mathbf{q}} \times (\mathbf{I}\dot{\mathbf{q}}) \tag{4.9}$$

or in an angular momentum form:

$$\dot{\mathbf{L}} = \boldsymbol{\tau} - (\mathbf{I}^{-1}\mathbf{L}) \times \mathbf{L} \tag{4.10}$$

Supposing the body is spinning around the principal axis of inertia, or a diagonalized tensor, equation (4.9) can be rewritten as:

$$\begin{bmatrix} I_1\ddot{q}_1 - (I_2 - I_3)\dot{q}_3\dot{q}_2 \\ I_2\ddot{q}_2 - (I_3 - I_1)\dot{q}_1\dot{q}_3 \\ I_3\ddot{q}_3 - (I_1 - I_2)\dot{q}_1\dot{q}_2 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} \tag{4.11}$$

or:

$$\begin{bmatrix} \dot{L}_1 - (\frac{1}{I_2} - \frac{1}{I_3})\, L_2\, L_3 \\ \dot{L}_2 - (\frac{1}{I_3} - \frac{1}{I_1})\, L_3\, L_1 \\ \dot{L}_3 - (\frac{1}{I_1} - \frac{1}{I_2})\, L_1\, L_2 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}. \tag{4.12}$$

Lagrange theory

Assuming the same inertia tensor such as the one used in Newton theory, from equation (3.70), we have:

$$K = \frac{1}{2} \sum_{i=1}^{n} \mathbf{I}_i \dot{\mathbf{q}}_\mathbf{i} \tag{4.13}$$

Now, if we take the Euler angles as the generalized coordinates we have, for the case of Ψ component:

$$\frac{d}{dt} \sum_{i=1}^{3} \frac{\partial K}{\partial \dot{q}_i} \frac{\partial \dot{q}_i}{\partial \dot{\Psi}} - \sum_{i=1}^{3} \frac{\partial K}{\partial q_i} \frac{\partial q_i}{\partial \Psi} = 0 \tag{4.14}$$

On one hand:

$$\frac{\partial K}{\partial q_i} = I_i \dot{q}_i \tag{4.15}$$

and, applying equations (3.26), (3.27) and (3.28):

$$\frac{\partial q_1}{\partial \Psi} = \dot{\Phi}\, s\theta\, c\Psi - \dot{\theta}\, s\Psi = \dot{q}_2 \tag{4.16}$$

$$\frac{\partial q_2}{\partial \Psi} = \dot{\Phi}\, s\theta\, s\Psi - \dot{\theta}\, c\Psi = -\dot{q}_1 \tag{4.17}$$

$$\frac{\partial q_3}{\partial \Psi} = 0 \tag{4.18}$$

On the other hand:

$$\frac{\partial \dot{q}_1}{\partial \dot{\Psi}} = \frac{\partial \dot{q}_2}{\partial \dot{\Psi}} = 0 \tag{4.19}$$

and

$$\frac{\partial \dot{q}_3}{\partial \dot{\Psi}} = 0 \tag{4.20}$$

Now, from the previous equations, we have the Lagrange equation for an Euler angle as:

$$I_3\, \ddot{q}_3 - I_1\, \dot{q}_1\, \dot{q}_2 + I_2\, \dot{q}_2\, (-\dot{q}_1) = 0 \tag{4.21}$$

or if we have any torque:

$$I_3\, \dot{q}_3 - (I_1 - I_2)\, \dot{q}_1\, \dot{q}_2 = \tau_3 \tag{4.22}$$

Similarly, we can do the same with the other two Euler's angles if we pick the unit vector $\hat{e}_3$ in the inertial frame randomly. With this, we obtain Euler's equations which are equation (4.11).

4.2.1 Mathematical model of the testbed

The dynamics model has been obtained by applying the rigid body theory discussed in chapter 3. In general, from Euler's equations, we have:

$$\mathrm{I}^{cr}\, \ddot{\mathbf{q}} + \dot{\mathbf{q}} \times \mathrm{I}^{cr} \dot{\mathbf{q}} = \boldsymbol{\tau} = \boldsymbol{\tau_g} + \boldsymbol{\tau_e} + \boldsymbol{\tau_c} \tag{4.23}$$

where the general torque $\boldsymbol{\tau}$ is conformed by all the external torques. For our case, we consider the torque due to gravity $\boldsymbol{\tau_g}$, the torque due to the friction of each electrical motor $\boldsymbol{\tau_e}$, and the torque produced due to the controller $\boldsymbol{\tau_c}$. Note that equation (4.23) has the form equation (3.2), which is the general dynamic model of a mechanical system. For our case, the matrix of masses is the inertia matrix and the Coriolis matrix depends only on $\dot{\mathbf{q}}$. Equation (4.23) is similar to the general model presented in equation (3.2) where the matrix of mass is represented by I.

Figure 4.3 represents the new rigid body framework, where the cm is slightly displaced in one of the axes. Now, let's imagine that we are not working with the principal axis of inertia. It means, we have all the components of the inertia tensor. Therefore, we can calculate the general equation of motion.

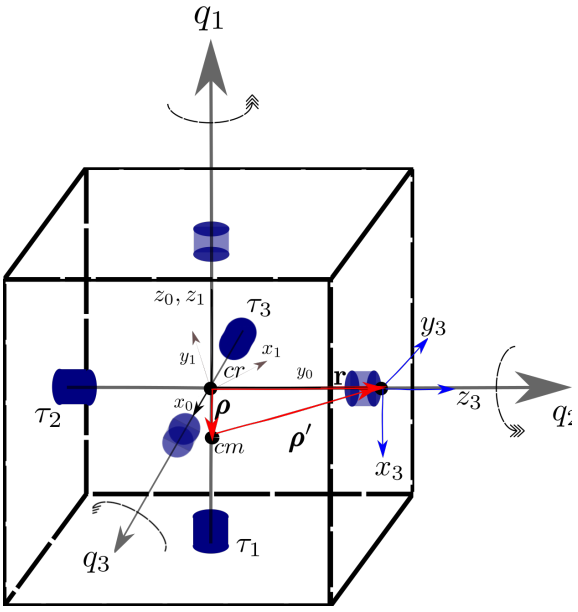


Figure 4.3: CubeSat testbed body diagram 3-dof. The vector $\mathbf{r}$ provides the position of the final framework (x_3, y_3, z_3) from the center of rotation.

$$\begin{bmatrix} I_{11} & I_{12} & I_{13} \\ I_{21} & I_{22} & I_{23} \\ I_{31} & I_{32} & I_{33} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} + \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} \times \begin{bmatrix} I_{11} & I_{12} & I_{13} \\ I_{21} & I_{22} & I_{23} \\ I_{31} & I_{32} & I_{33} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \boldsymbol{\tau} \tag{4.24}$$

The angular position q_1, q_2, q_3 depend on the torque of the reaction wheels(τ_1, τ_2, τ_3) respectively. In general, the torque due to gravity in any object is a cross product of the external force (gravity) and the position vector to the center o mass. As the testbed is rotated, the gravity vector changes its orientation, so it is multiplied by a rotation matrix. The latter means:

$$\boldsymbol{\tau_g} = m\,\boldsymbol{\rho} \times \mathrm{R}_0^3\,\mathbf{g} \tag{4.25}$$

where $\boldsymbol{\rho}$ is the vector to the center of mass that is displaced in the z axis, R_0^3 is the rotation matrix estimated with the Euler Angles in rotation ZXZ and $\mathbf{g}$ is the gravity vector which our case has a component in the z axis. This means:

$$\boldsymbol{\tau_g} = \begin{bmatrix} 0 \\ 0 \\ \rho \end{bmatrix} \times m \begin{bmatrix} cq_1cq_3 - sq_1cq_2sq_3 & cq_1sq_3 + sq_1cq_2cq_3 & sq_1sq_2 \\ -sq_1cq_3 - cq_1cq_2cq_3 & cq_1cq_2cq_3 - sq_1sq_3 & cq_1sq_2 \\ sq_1sq_3 & -sq_2cq_3 & cq_2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{4.26}$$

$$= m\rho g \begin{bmatrix} -cq_1\,sq_2 \\ sq_1\,sq_2 \\ 0 \end{bmatrix} \tag{4.27}$$

On the other hand,

$$\boldsymbol{\tau_e} = \mathbf{K_f}\dot{\mathbf{q}} = \begin{bmatrix} k_{f1}\dot{q}_1 \\ k_{f2}\dot{q}_2 \\ k_{f3}\dot{q}_3 \end{bmatrix} \tag{4.28}$$

where $\mathbf{K_f}$ is a diagonal matrix as the constant kf_i for $i = 1, 2, 3$ is the motor constant for each motor, and:

$$\boldsymbol{\tau_c} = \mathbf{K_p}\tilde{\mathbf{q}} - \mathbf{K_v}\dot{\mathbf{q}} = \begin{bmatrix} k_{p1}\,\tilde{q}_1 - k_{v1}\,\dot{q}_1 \\ k_{p2}\,\tilde{q}_2 - k_{v2}\,\dot{q}_2 \\ k_{p3}\,\tilde{q}_3 - k_{v3}\,\dot{q}_3 \end{bmatrix} \tag{4.29}$$

with k_{pi} and k_{vi} for $i = 1, 2, 3$ as the constants of proportionality for each reaction wheel controller. Then we obtain:

$$\begin{bmatrix} I_{11}\ddot{q}_1 + I_{12}\ddot{q}_2 + I_{13}\ddot{q}_3 + \dot{q}_2(I_{31}\dot{q}_1 + I_{32}\dot{q}_2 + I_{33}\dot{q}_3) - \dot{q}_3(I_{21}\dot{q}_1 + I_{22}\dot{q}_2 + I_{23}\dot{q}_3) \\ I_{21}\ddot{q}_1 + I_{22}\ddot{q}_2 + I_{23}\ddot{q}_3 + \dot{q}_1(I_{31}\dot{q}_1 + I_{32}\dot{q}_2 + I_{33}\dot{q}_3) - \dot{q}_3(I_{11}\dot{q}_1 + I_{12}\dot{q}_2 + I_{13}\dot{q}_3) \\ I_{31}\ddot{q}_1 + I_{32}\ddot{q}_2 + I_{33}\ddot{q}_3 + \dot{q}_1(I_{21}\dot{q}_1 + I_{22}\dot{q}_2 + I_{23}\dot{q}_3) - \dot{q}_3(I_{11}\dot{q}_1 + I_{12}\dot{q}_2 + I_{13}\dot{q}_3) \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} \tag{4.30}$$

with:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} k_{p1}\,\tilde{q}_1 - k_{v1}\,\dot{q}_1 - mg\rho cq_1\,sq_2 - k_{f1}\dot{q}_1 \\ k_{p2}\,\tilde{q}_1 - k_{v2}\,\dot{q}_1 + mg\rho sq_1\,sq_2 - k_{f2}\dot{q}_2 \\ k_{p3}\,\tilde{q}_1 - k_{v3}\,\dot{q}_1 - k_{f3}\dot{q}_3 \end{bmatrix} \tag{4.31}$$

If we diagonalize the inertia tensor or if we are working with the principal axis of inertia, we obtain the Euler equation of motion (equation 4.11) with the external torques.

$$\begin{bmatrix} I_1\ddot{q}_1 - (I_2 - I_3)\dot{q}_3\dot{q}_2 \\ I_2\ddot{q}_2 - (I_3 - I_1)\dot{q}_1\dot{q}_3 \\ I_3\ddot{q}_3 - (I_1 - I_2)\dot{q}_1\dot{q}_2 \end{bmatrix} = \begin{bmatrix} k_{p1}\,\tilde{q}_1 - k_{v1}\,\dot{q}_1 - mg\rho cq_1sq_2 - k_{f1}\dot{q}_1 \\ k_{p2}\,\tilde{q}_2 - k_{v2}\,\dot{q}_2 + mg\rho sq_1sq_2 - k_{f2}\dot{q}_2 \\ k_{p3}\,\tilde{q}_3 - k_{v3}\,\dot{q}_3 - k_{f3}\dot{q}_3 \end{bmatrix} \tag{4.32}$$

If we have only two inertial wheels, which is the real case (Figure 4.4), then the equation of motion is:

$$\begin{bmatrix} I_1\ddot{q}_1 - (I_2 - I_3)\dot{q}_3\dot{q}_2 \\ I_2\ddot{q}_2 - (I_3 - I_1)\dot{q}_1\dot{q}_3 \end{bmatrix} = \begin{bmatrix} k_{p1}\,\tilde{q}_1 - k_{v1}\,\dot{q}_1 - mg\rho cq_1sq_2 - k_{f1}\dot{q}_1 \\ k_{p2}\,\tilde{q}_2 - k_{v2}\,\dot{q}_2 + mg\rho sq_1sq_2 - k_{f2}\dot{q}_2 \end{bmatrix} \tag{4.33}$$

For equation (4.33) note that there is no torque for the third axis, which means that there is no acceleration $\ddot{q}_3 = 0$ and, as a consequence, $\dot{q}_3 = constant$.

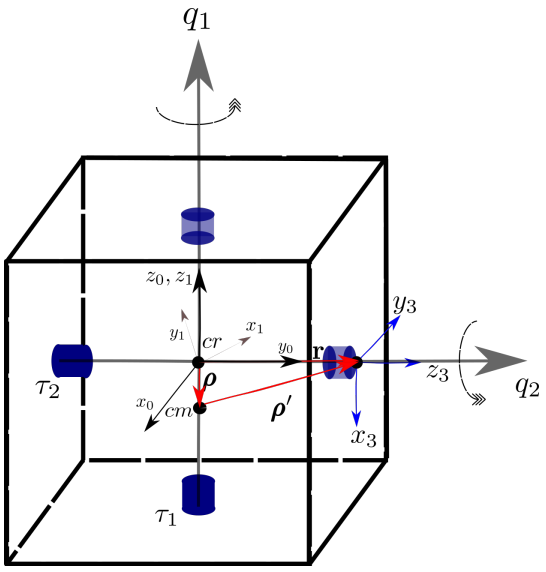


Figure 4.4: CubeSat testbed body diagram 2-dof. The vector $\mathbf{r}$ provides the position of the final framework (x_3, y_3, z_3) from the center of rotation. The angular position q_1, q_2 depend on the torque of the reaction wheels(τ_1, τ_2) respectively.

Now for 1-dof (Figure 4.5) we have:

$$\left[I_1\ddot{q}_1\right] = \left[k_{p1}\,\tilde{q}_1 - k_{v1}\,\dot{q}_1 - mg\rho cq_1 sq_2 - k_{f1}\dot{q}_1\right] \tag{4.34}$$

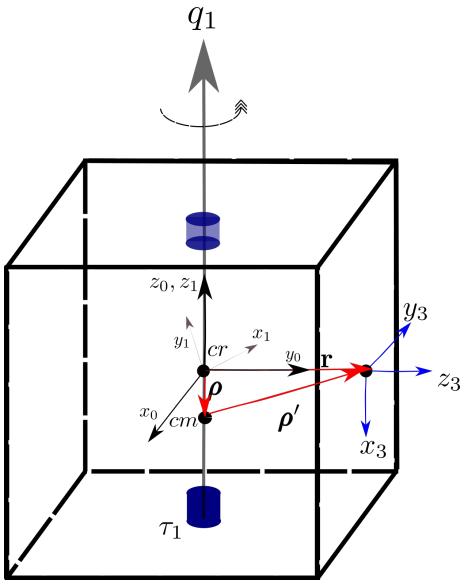


Figure 4.5: CubeSat testbed body diagram 1-dof. The vector **r** provides the position of the final framework (x_3, y_3, z_3) from the center of rotation. The angular position q_1 depend on the torque of the reaction wheels(τ_1) respectively.

Equations (4.32), (4.33) and (4.34) are our dynamical model for 3-dof, 2-dof and 1-dof respectively.

4.2.2 Simulink simulation

The simulation was made with the software Simulink. Firstly, a CAD drawing (computer-aided design) in Solidworks (Figure 4.6) was developed. Here, the measure specifications of CubeSats discussed in chapter 1 were applied. Besides, the design of the test-bed was simulated. The density of the material in the simulation was Aluminium 6061 (Al6061), which is a material used in the structure of CubeSats(Yost et al., 2020). This led the model to have a final mass of approximately 1.1 kg.

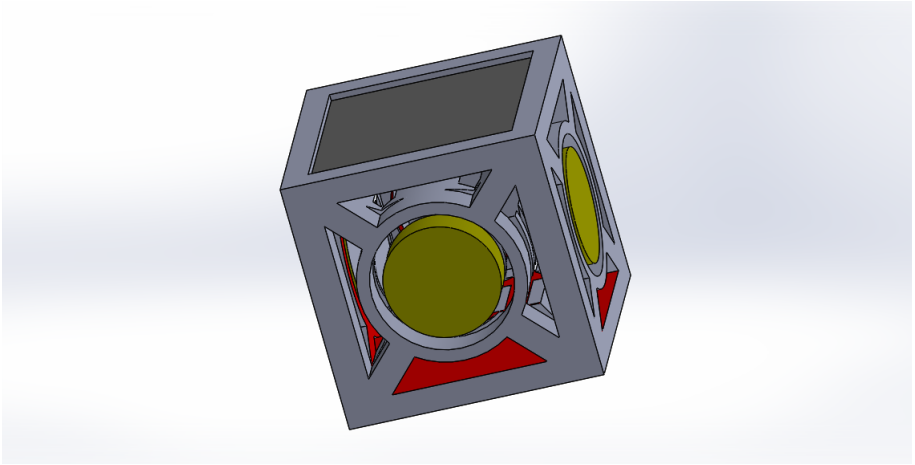


Figure 4.6: CubeSat CAD simulation created in Solidworks.

Once the CAD model was obtained, it was exported to Simulink Simscape Multibody (Figure 4.7) since the libraries of Simscape allow inserting reference frameworks, frameworks translations, and rotations. Depending on the number of degrees of freedom, different translations and rotations were added.

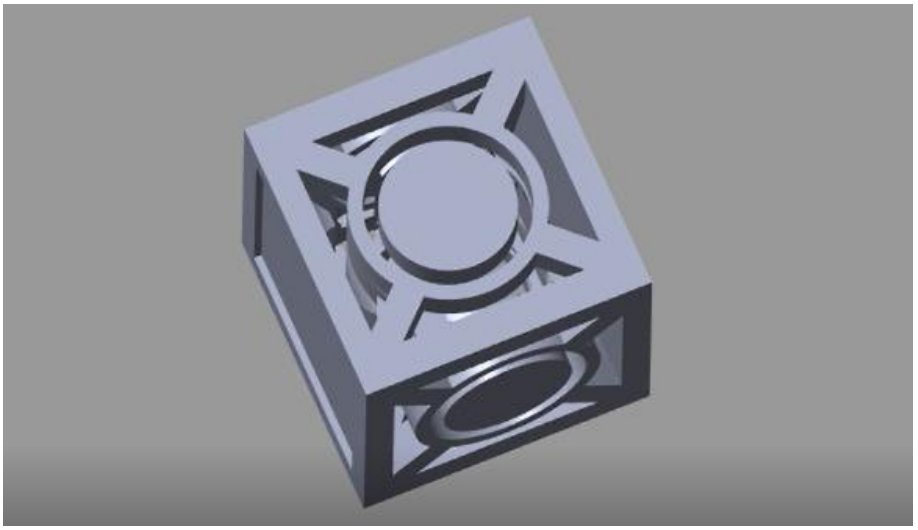


Figure 4.7: CubeSat Simulink simulation.

Finally, the data from the simulations were taken and exported in a script of MATLAB where the data was plotted. For the robotics simulations, the main difference was the position of the frameworks and the final rotational matrix obtained with the D-H convention. The code and block diagrams can be found in the appendix E.

Chapter 5

Simulations and validation of the dynamical modeling

This chapter shows the results of the simulations. First, it shows the results of 1-dof, the 2-dof, and finally 3-dof respectively. Each subsection has the simulations of the dynamic model, which means simulation of equations (4.34), (4.33) and (4.32) respectively, and Simulink simulations from a rigid body and robotics perspective. We used the library ode45 to solve the differentials equations, which utilize a Runge-Kutta numerical integration method (MATLAB, 2021). The final reference framework was placed at 5 cm from the center of rotation of the cube. Moreover, the constants of the controller were previously calibrated by varying their values until we found a good behavior of the system. As we wanted to maintain a final position and orientation, for all the simulations, the desired final velocity was zero rad/s. For all the cases, we used the proportional plus velocity feedback control having a close loop system with proportionality constants previously calibrated ($k_p = 1$ Nm and $k_v = 10$ Nms/rad). According to the CAD simulation, the principal axis of inertia are $I_1 = 0.002487$ kgm^2, $I_2 = 0.002487$ kgm^2 and $I_3 = 0.002518$ kgm^2. The constant of friction of the reaction wheel was taken from their data sheet, which is $k_f = 0.01315$ Nms/rad. Finally, the initial condition for were $q_i = 0$rad and $\dot{q}_i = 0$rad/s from $t = 0$ to $t = 50$s.

5.1 1 reaction wheel

Making the assumption that there is only one reaction wheel in the testbed, we simulated 1-dof. Figures 5.1, 5.2 and 5.3 are the dynamic model simulation, Figures 5.4, 5.5 and, 5.6 are from the simulation of the rigid body in simulink. Finally, Figures 5.7, 5.8 and, 5.9 are from the simulation from robotics in Simulink. These plots show the trajectory of the movement of the satellite from different perspective and the angular position and angular velocity respectively. For all the simulation, the desired position in the controller for this case is $q_1 = \pi$ rad. The code and block diagrams can be found in appendix E.

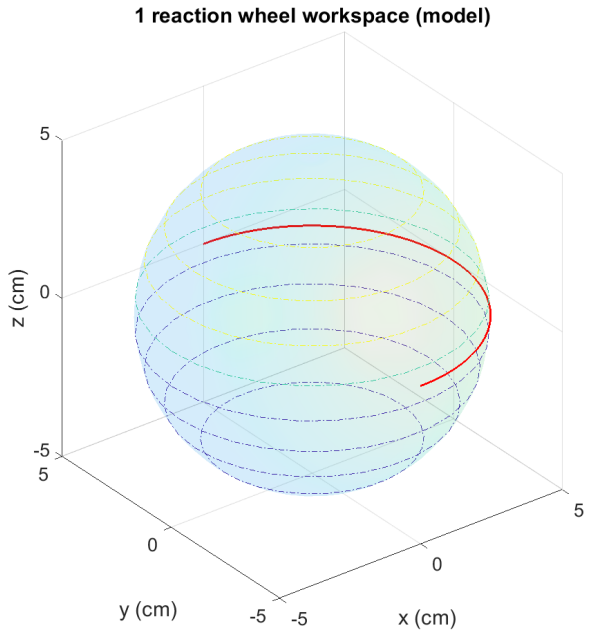


Figure 5.1: 1-dof theoretical model workspace (3D view). The body is initially at rest. The red line is the trajectory of the final framework from $t = 0$ to $t = 50$ s. The desired final position was π rad.

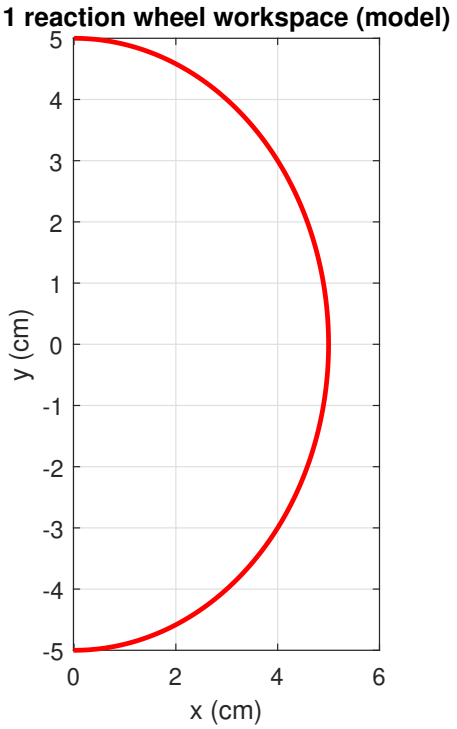


Figure 5.2: 1-dof theoretical model workspace from the top view. The body is initially at rest. The red line is the trajectory of the final framework from $t = 0$ to $t = 50$ s. The desired final position was π rad.

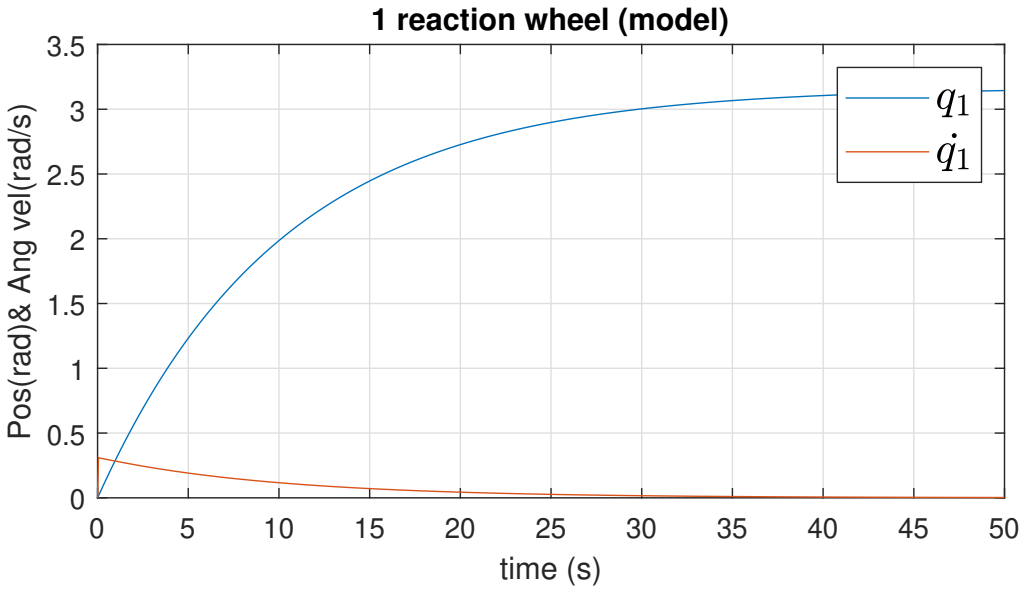


Figure 5.3: 1-dof theoretical model angular velocity and angular position. The body is initially at rest and the simulation runs from $t = 0$ to $t = 50$ s. The desired final position was π rad.

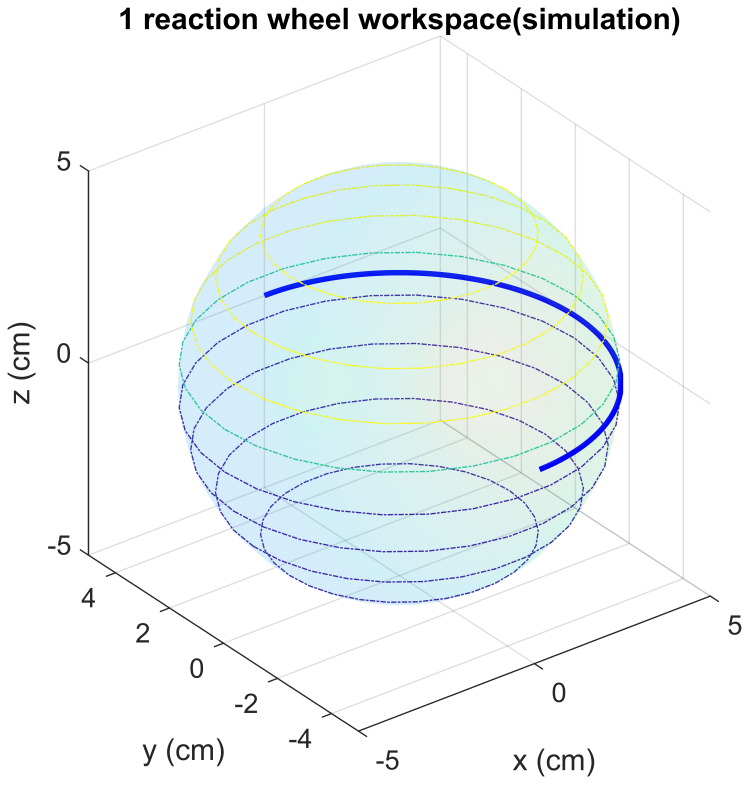


Figure 5.4: 1-dof simulation model workspace (3D view). The body is initially at rest. The blue line is the trajectory of the final framework from $t = 0$ to $t = 50$ s. The desired final position was π rad.The block diagram of this simulation is presented in Figure E.1.

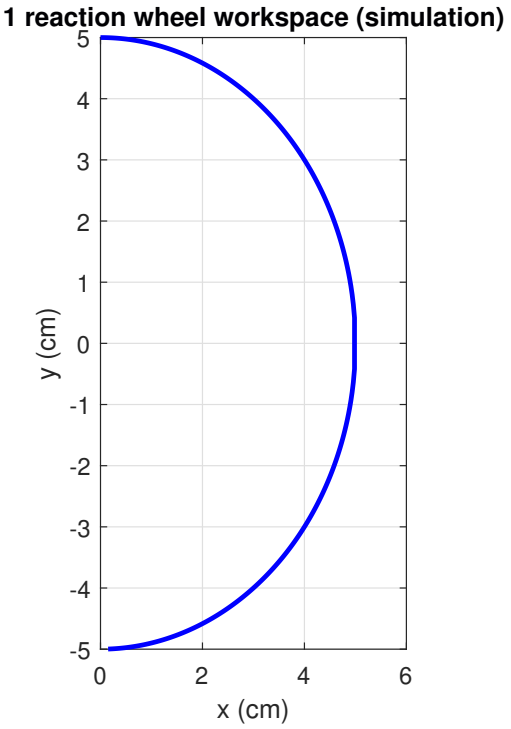


Figure 5.5: 1-dof simulation model workspace from the top view. The body is initially at rest. The blue line is the trajectory of the final framework from $t = 0$ to $t = 50$ s. The desired final position was πrad. The block diagram of this simulation is presented in Figure E.1.

Figure 5.6: 1-dof angular velocity and angular position. The body is initially at rest and the simulation runs from $t = 0$ to $t = 50$ s. The desired final position was π rad. The block diagram of this simulation is presented in Figure E.1.

Figure 5.7: 1-dof theoretical model workspace (3D view with robotics). The body is initially at rest. The green line is the trajectory of the final framework from $t = 0$ to $t = 50$ s. The desired final position was π rad. The block diagram of this simulation is presented in Figure E.4.

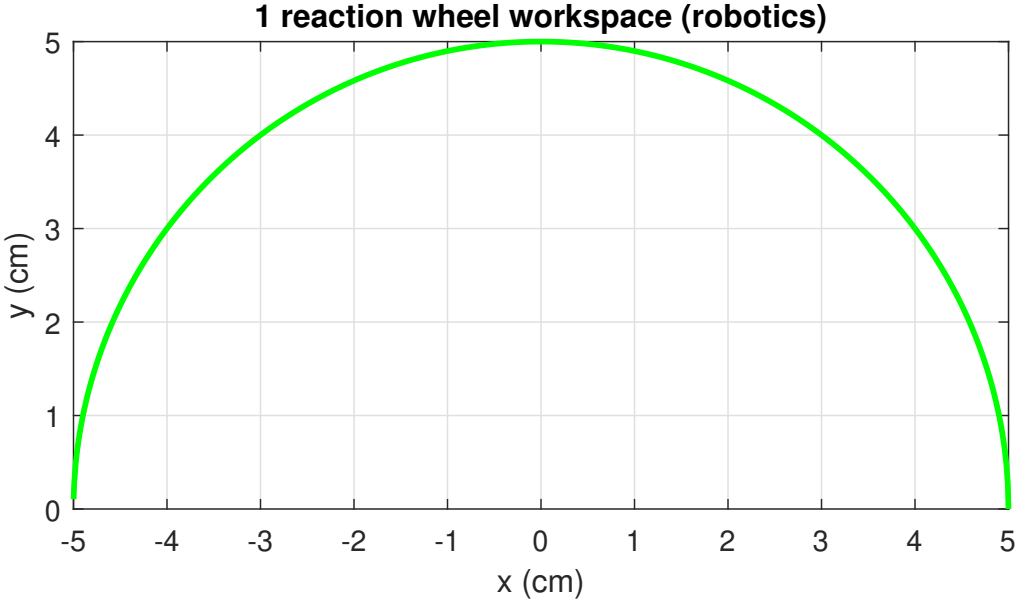


Figure 5.8: 1-dof theoretical model workspace (robotics) from the top view. The green line is the trajectory of the final framework from $t = 0$ to $t = 50$ s. The desired final position was π rad. The block diagram of this simulation is presented in Figure E.4.

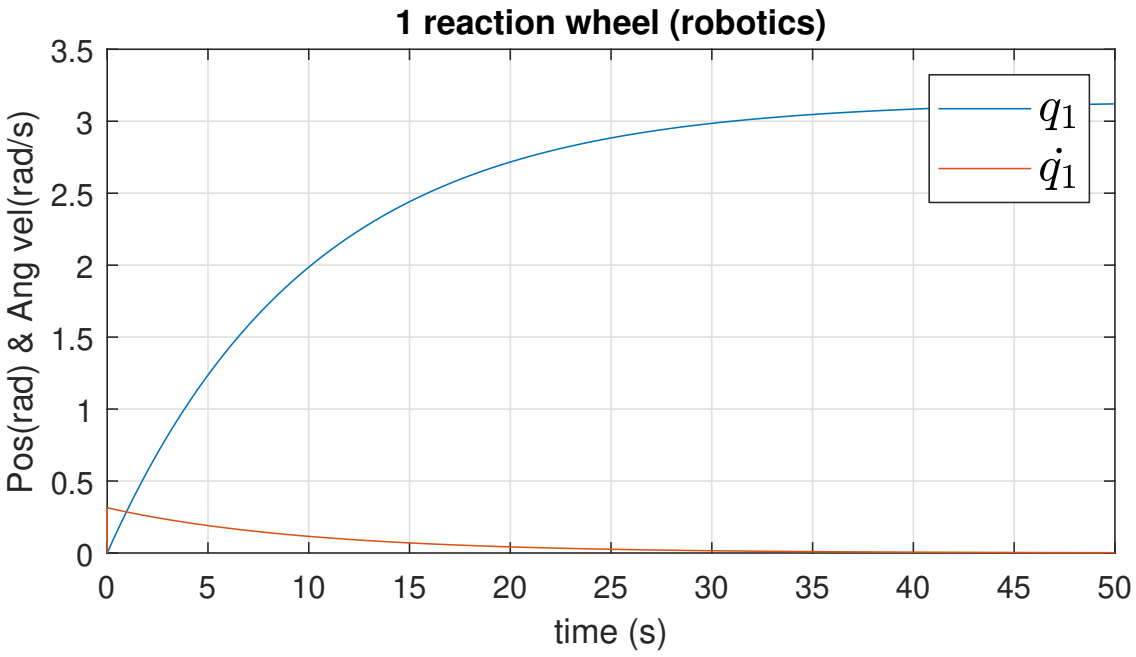


Figure 5.9: 1-dof theoretical model angular velocity and angular position (robotics). The body is initially at rest and the simulation runs from $t = 0$ to $t = 50$ s. The desired final position was π rad. The block diagram of this simulation is presented in Figure E.4.

Note that all trajectories in workspaces are half of a circumference (which was desired), which means the desired angular position was $q_1 = \pi$. Nevertheless, Figure 5.8 shows a rotated circumference which does not mean that simulation is wrong since this variation is due to the construction of the D-H convention. On the other hand, thanks to the controller, angular position and trajectories were obtained as they were desired for all the cases. With the input data, the model and the simulations follow the expected behavior.

5.2 2 reaction wheels

The following results were obtained for 2-dof or two reaction wheels. For these cases the desired positions were $q_1 = \pi$ and $q_2 = 2\pi$. Figures 5.10, 5.13 and, 5.16 represents the trajectory of the movement of the cube dynamic model, rigid body and robot simulation respectively. Figures 5.14 and, 5.17 shows the angular position for the three simulation. While Figures 5.12, 5.15 and, 5.18 show the angular velocity for the dynamic model. As it was before, the code and block diagrams can be found in appendix E.

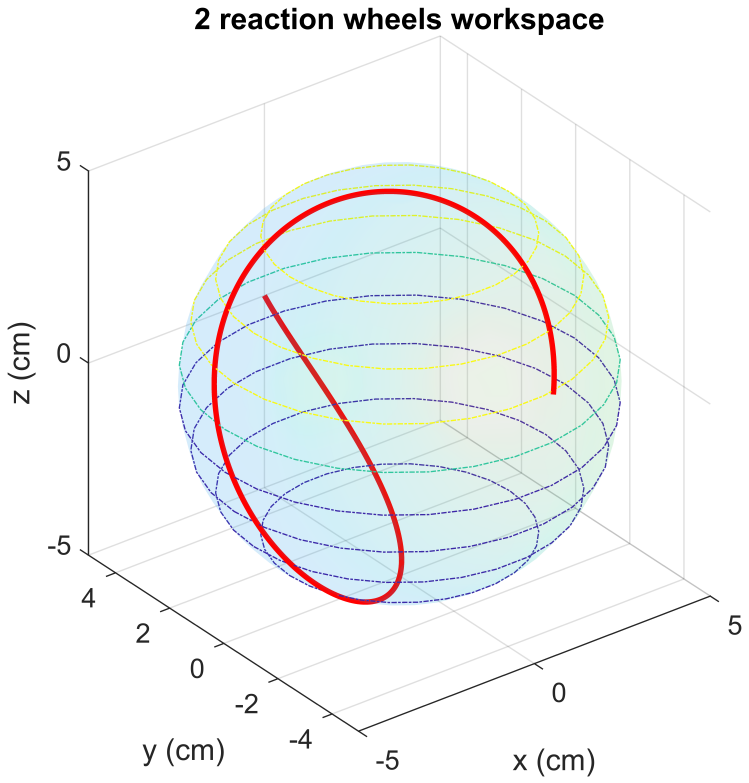


Figure 5.10: 2-dof theoretical model workspace. The body is initially at rest. The red line is the trajectory of the final framework from $t = 0$ to $t = 50$ s. The desired final position were $q_1 = \pi$ and $q_2 = 2\pi$ rad.

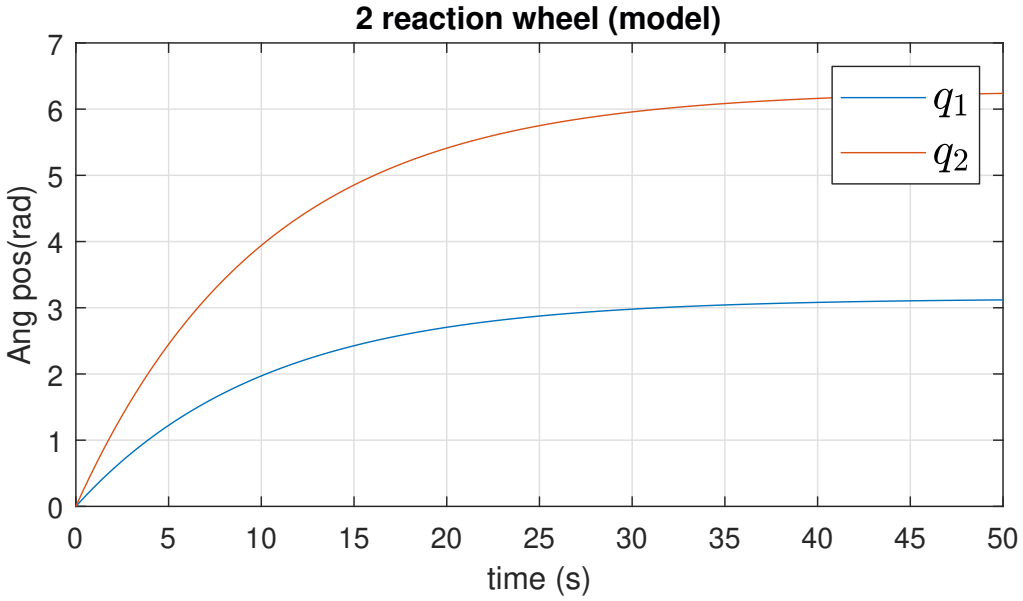


Figure 5.11: 2-dof theoretical model angular position. The body is initially at rest. and the simulation runs from $t = 0$ to $t = 50$ s. The desired final positions were $q_1 = \pi$ and $q_2 = 2\pi$ rad.

Figure 5.12: 2-dof theoretical model angular velocity. The body is initially at rest. and the simulation runs from $t = 0$ to $t = 50$ s.

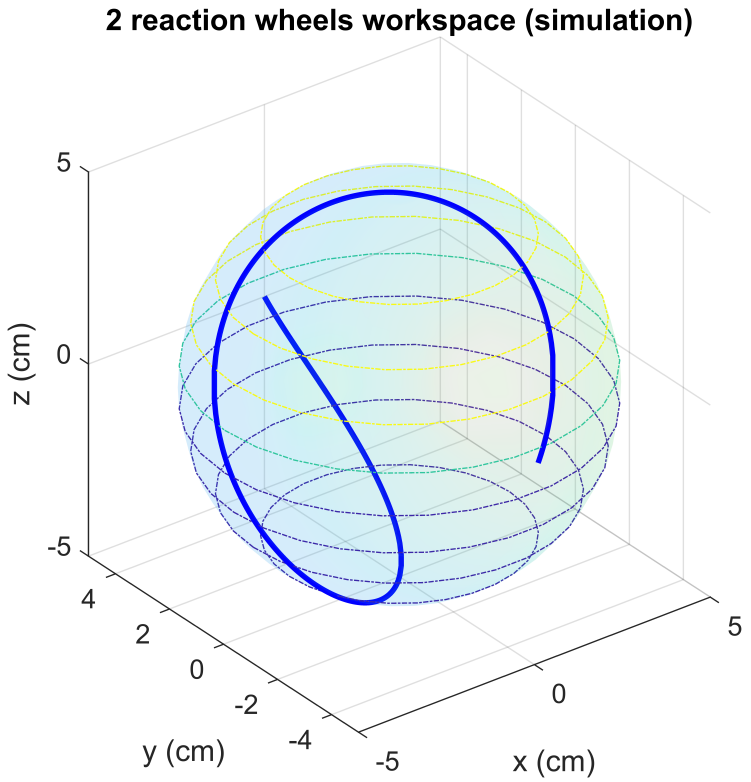


Figure 5.13: 2-dof simulation workspace. The body is initially at rest. The blue line is the trajectory of the final framework from $t = 0$ to $t = 50$ s. The desired final position were $q_1 = \pi$ and $q_2 = 2\pi$ rad. The block diagram of this simulation is presented in Figure E.2.

Figure 5.14: 2-dof simulation angular position. The body is initially at rest. and the simulation runs from $t = 0$ to $t = 50$ s. The desired final positions were $q_1 = \pi$ and $q_2 = 2\pi$ rad. The block diagram of this simulation is presented in Figure E.2.

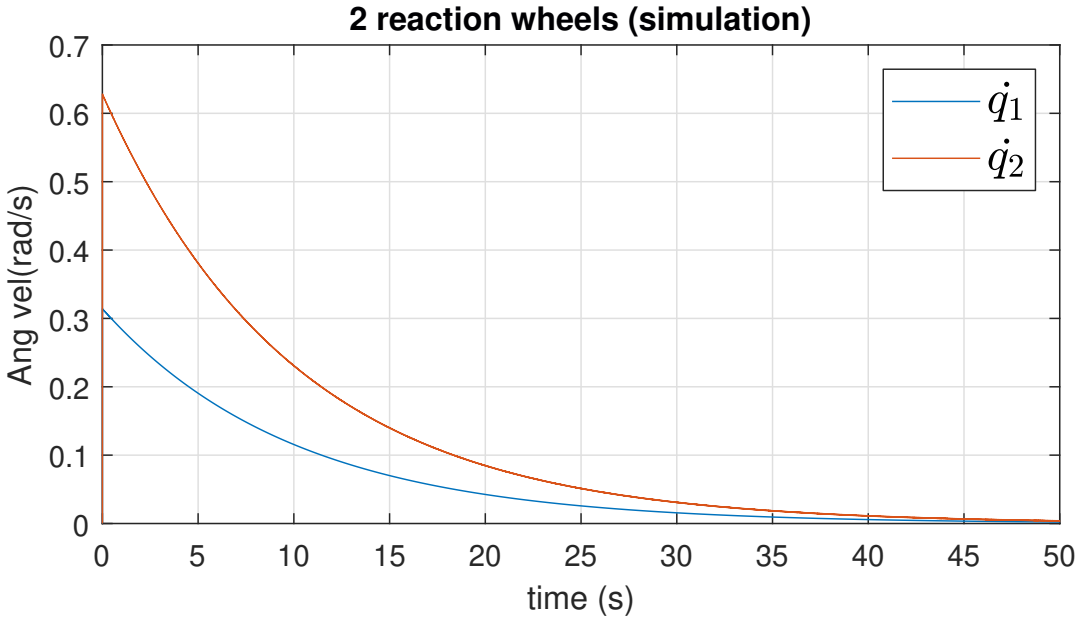


Figure 5.15: 2-dof simulation angular velocity. The body is initially at rest. and the simulation runs from $t = 0$ to $t = 50$ s. The desired final velocities were $\dot{q}_1 = \dot{q}_2 = 0$ rad. The block diagram of this simulation is presented in Figure E.2.

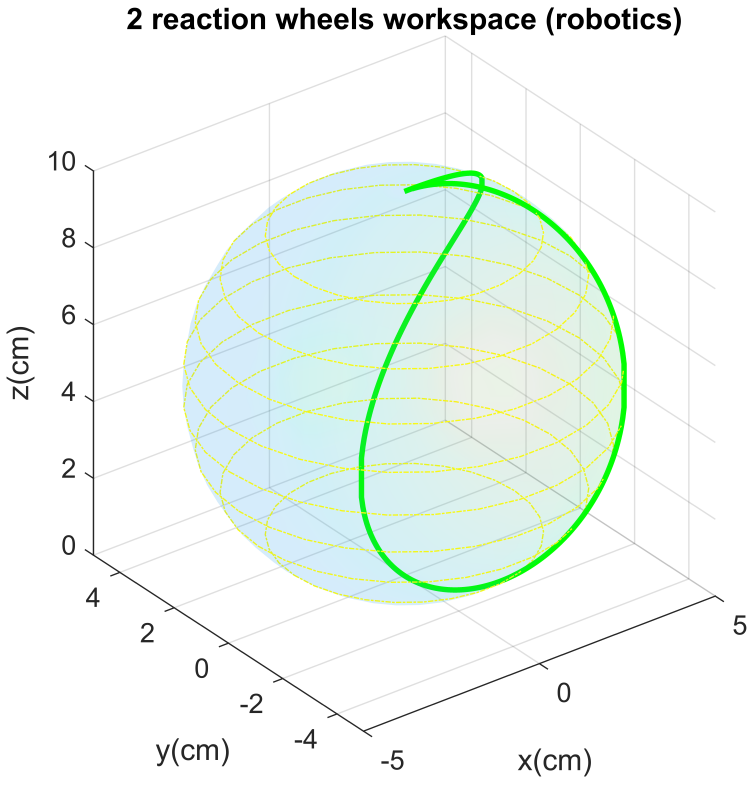


Figure 5.16: 2-dof theoretical model workspace (robotics). The body is initially at rest. The green line is the trajectory of the final framework from $t = 0$ to $t = 50$ s. The desired final position were $q_1 = \pi$ and $q_2 = 2\pi$ rad. The block diagram of this simulation is presented in Figure E.5.

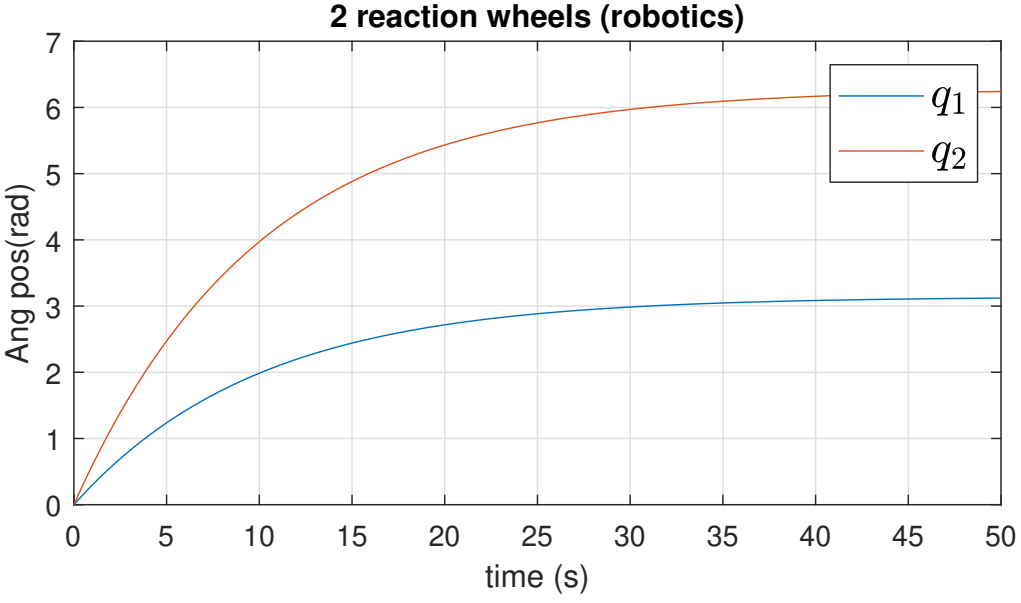


Figure 5.17: 2-dof theoretical model angular position (robotics). The body is initially at rest. and the simulation runs from $t = 0$ to $t = 50$ s. The desired final positions were $q_1 = \pi$ and $q_2 = 2\pi$ rad. The block diagram of this simulation is presented in Figure E.5.

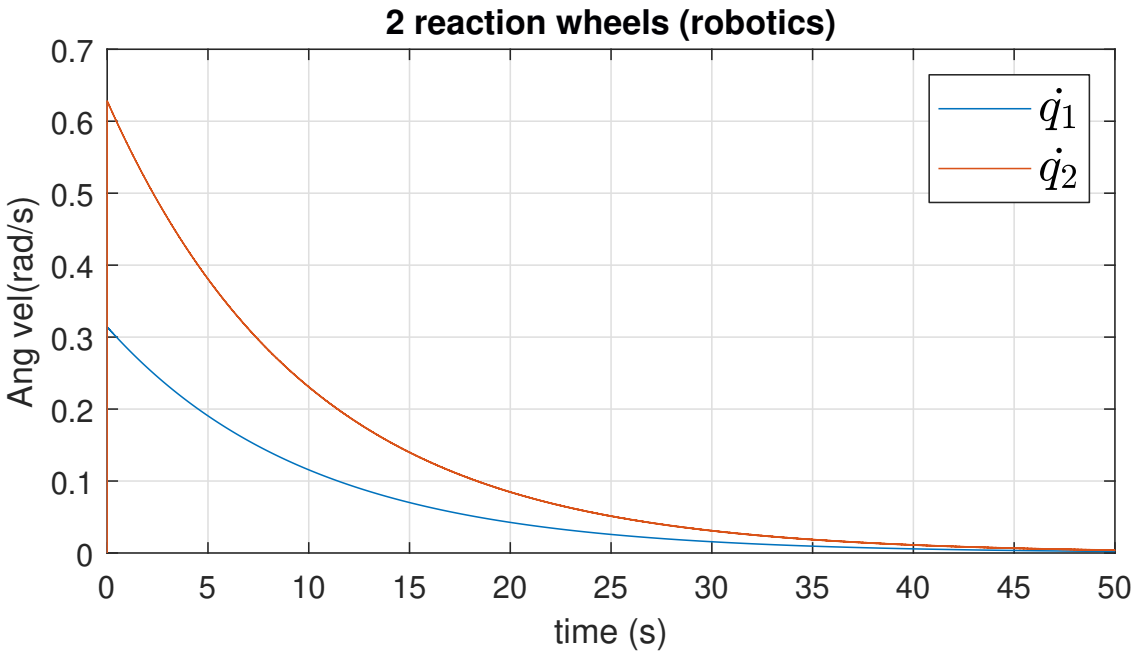


Figure 5.18: 2-dof theoretical model angular velocity (robotics). The body is initially at rest. and the simulation runs from $t = 0$ to $t = 50$ s. The block diagram of this simulation is presented in Figure E.5.

From the simulations, we can see that the trajectory in the figure made with robotics is different. Again, this does not mean that the simulations are wrong since this discrepancy is due to the mathematical construction of rigid body theory and D-H methodology. In contrast, the model and simulation of RBD are similar, which suggests our model looks correct. Additionally, the angular position and velocity for all the cases were those desired.

5.3 3 reaction wheels

Finally, the following results are from a 3-dof simulation. Figures 5.19, 5.20 and 5.21 present the results from the dynamic model simulation. Results of rigid body simulation are show in Figures 5.22, 5.23 and, 5.24. Figures 5.25, 5.26 and, 5.27 are from the robotics simulation. Note that the desired final positions for the three cases were $q_1 = 2\pi$ and $q_2 = \frac{3}{4}\pi$ and $q_3 = \pi$, while all the other testbed parameters remain the same such as the 1-dof and 2-dof simulations. Finally, the code and block diagrams can be found in appendix E.

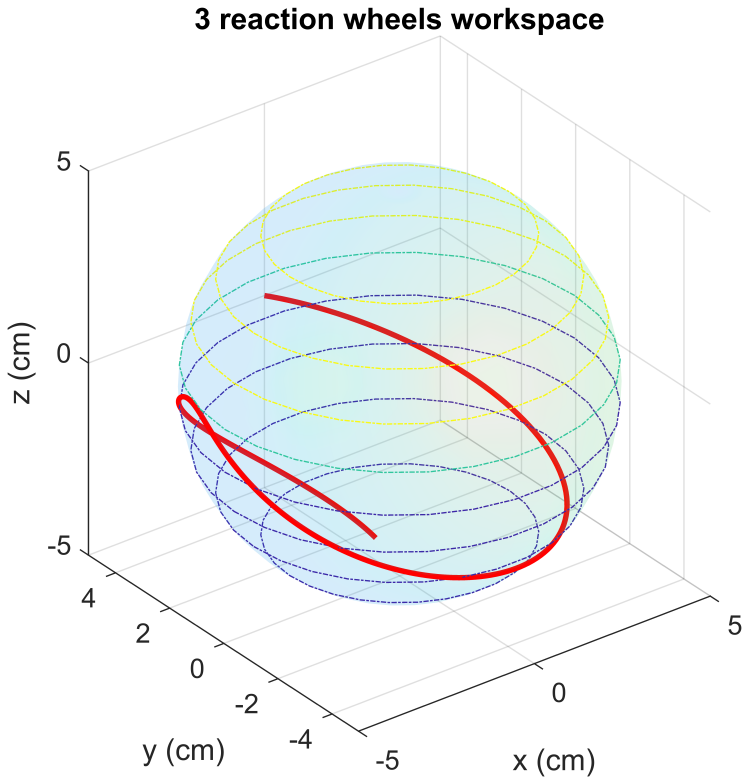


Figure 5.19: 3-dof theoretical model workspace. The body is initially at rest. The red line is the trajectory of the final framework from $t = 0$ to $t = 50$ s. The desired final position were $q_1 = 2\pi$ $q_2 = 3\pi/4$ and $q_3 = \pi$ rad.

Figure 5.20: 3-dof theoretical model angular position. The body is initially at rest. and the simulation runs from $t = 0$ to $t = 50$ s. The desired final positions were $q_1 = 2\pi$ $q_2 = 3\pi/4$ and $q_3 = \pi$ rad.

Figure 5.21: 3-dof theoretical model angular velocity. The body is initially at rest. and the simulation runs from $t = 0$ to $t = 50$ s. The desired final positions were $\dot{q}_1 = \dot{q}_2 = \dot{q}_3 = 0$ rad/s.

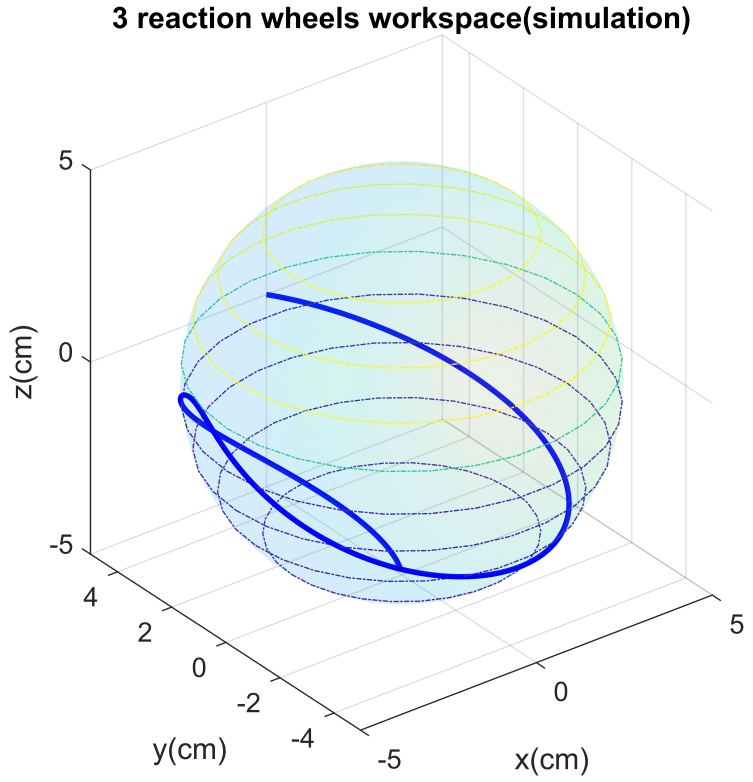


Figure 5.22: 3-dof simulation workspace. The body is initially at rest. The blue line is the trajectory of the final framework from $t = 0$ to $t = 50$ s. The desired final position were $q_1 = 2\pi$ $q_2 = 3\pi/4$ and $q_3 = \pi$ rad. The block diagram of this simulation is presented in Figure E.3.

Figure 5.23: 3-dof simulation angular position. The body is initially at rest. and the simulation runs from $t = 0$ to $t = 50$ s. The desired final positions were $q_1 = 2\pi$ $q_2 = 3\pi/4$ and $q_3 = \pi$ rad. The block diagram of this simulation is presented in Figure E.3.

Figure 5.24: 3-dof simulation angular velocity. The body is initially at rest. and the simulation runs from $t = 0$ to $t = 50$ s. The desire final positions were $\dot{q}_1 = \dot{q}_2 = \dot{q}_3 = 0$ rad/s. The block diagram of this simulation is presented in Figure E.3.

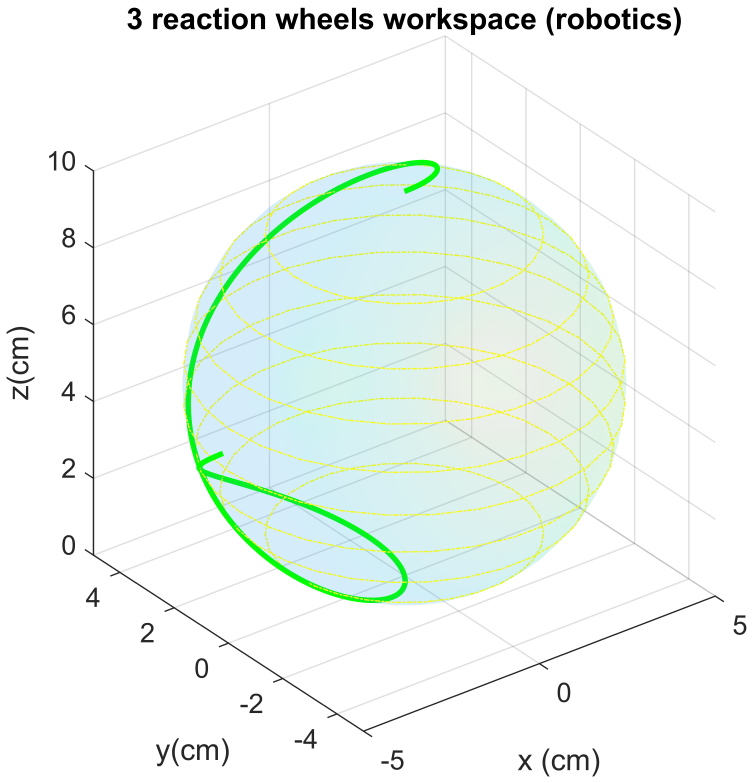


Figure 5.25: 3-dof theoretical model workspace (robotics). The body is initially at rest. The green line is the trajectory of the final framework from $t = 0$ to $t = 50$ s. The desired final position were $q_1 = 2\pi$ $q_2 = 3\pi/4$ and $q_3 = \pi$ rad. The block diagram of this simulation is presented in Figure E.6.

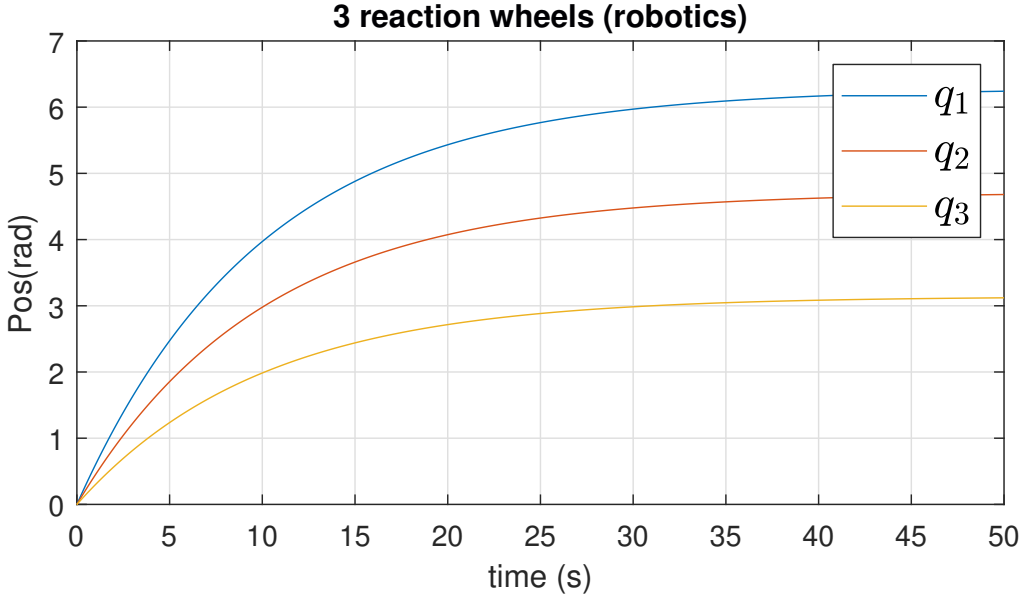


Figure 5.26: 3-dof theoretical model angular position (robotics) with the desired values $q_1 = 2\pi$ $q_2 = 3\pi/4$ and $q_3 = \pi$ rad. The block diagram of this simulation is presented in Figure E.6.

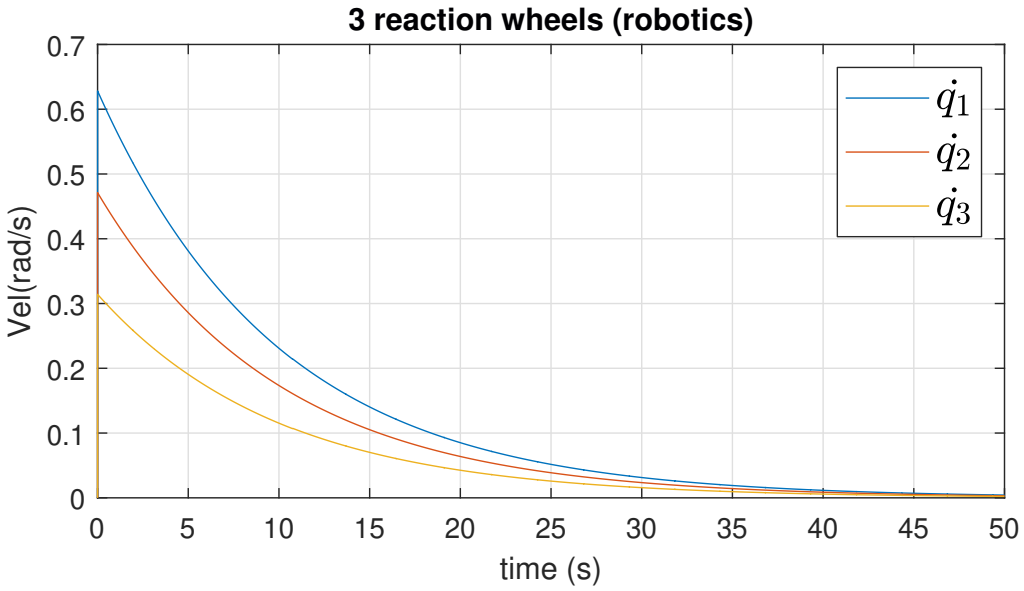


Figure 5.27: 3-dof theoretical model angular velocity (robotics).The body is initially at rest. and the simulation runs from $t = 0$ to $t = 50$ s. The desire final positions were $\dot{q}_1 = \dot{q}_2 = \dot{q}_3 = 0$ rad/s. The block diagram of this simulation is presented in Figure E.6.

Workspace trajectories in the simulations where we used RBD are the same and again, when we used D-H we observe a difference due to the theory. The desired angular position was obtained and the angular velocity was zero, which suggests our model is correct because the behavior is as desired.

Chapter 6

Conclusion and future work

This thesis project presented a dynamic model and simulation of a 2-dof CubeSat testbed. Due to the similarity of the position of reaction wheels in the testbed with a spherical wrist, first, we had initially decided to use robotics arm manipulators theory. Nevertheless, once we started to study more rigorously the physics of the satellite ground test system, we decided to apply rigid body theory. Robotics interprets the dynamics with joined rigid body, while rigid body theory studies the dynamics as a whole body. The latter is more related to the application that the testbed would have. Also, rigid body theory is typically used to study satellites. Therefore we applied Euler's equations of motion to our system and we opted to use generalized coordinates to get an easy visualization of the rotational frameworks. Once we obtained the general 3-dof equation, we decided to reduce the number of degrees of freedom to get the 2-dof and 1-dof models.

The CAD model was made with Solidworks because it permits to vary the physical properties of the system. Once we had the CAD model, we used Simulink due to its connectivity with Matlab, which allows us to use the CAD model for a better understating of the system.

To validate the model, we used a variation of a PD controller. The simulations from rigid body dynamics were quite similar to the results obtained from our model. In contrast, the robotics simulation showed another workspace. The latter is because the D-H convention does not represent the position and orientation of a single rigid body. If we could obtain the same path in the workspace, the other controller should be used. However, that was not part of the objectives of this thesis.

For the 1-dof case, we observed that with the initial condition and the desired position we obtain half of a circle because our desired condition was only π radians. Figures 5.1, 5.4 and 5.7 show the 3D spacework. Figures 5.2 , 5.5 and 5.8 present the spacework from the top view where we can appreciate that for the robotic, even with the same initial and desired conditions, the trajectory is different, this is because we used the D-H theory for the robotics case which has a different mathematical construction. We can observe in Figures 5.3, 5.6 and 5.9 that the desired position and velocity is reached with the proposed initial conditions.

The 2-dof case is the most important because it is the case of our testbed. Figures 5.10, 5.13 and 5.16 ilustrate the workspace for the model, simulation and robotics perspective with the same initial condition and desired position and velocity. Note again that the model and the simulation have the same spacework while the robotics workspace is different. Figures 5.11, 5.14 and 5.17 confirm that the controller and the model are correct because they reach the desired position. Additionally, Figures 5.12, 5.15 and 5.18, which are the angular velocities, also suggest that our model is correct.

Finally, the general case was the 3-dof, which can be applied to another testbed with one more reaction wheel. Figures 5.19, 5.22 and 5.25 depict the workspace for the 3-dof case. Again, the trajectory followed by the end effector in the robotics case is different than the others where we used rigid body dynamics. Figures 5.20, 5.23 and 5.26 present the angular position of the 3

reaction wheel for all the simulations. Figures 5.21, 5.24 and 5.27 show the velocities of the three simulations. One more time we got the same position and velocities for each reaction wheel with the proposed initial condition and the testbed parameters and the desired output.

With the final results, we can affirm that the general and specific objectives, were achieved. Still, we have some suggestions for future studies of the testbed:

1. Due to the lockdown, the experimental simulation could not be conducted, which is important if we want completely validate the model.
2. After conducting the physical experiments, we strongly recommend analyzing the possible errors to improve the mathematical model.
3. It is important to generate a closed CAD model with all the components that the CubeSat has. Besides, the identification of the center of mass will be required to get a better model.
4. We recommend applying other theories to the testbed such as quaternions or rigid body theory with Euler Angle to develop a better controller.
5. Depending on the control objective, other controllers could be applied to the system.

We strongly believe that this research work will help to understand the dynamics of satellites testbeds to future students or workers, who can apply this knowledge to develop new science and technology that can be beneficial for humanity.

Appendix A

Inertia matrix properties

A.1 Apendix A: Intertia matrix properties

The mass matrix or in our case, the inertia matrix has some mathematical properties.

1. M($\mathbf{q}$) is a positive definite matrix

2. Its determinant is positive

Now, to know if a model is right, we can obtain some conditions based on the last two properties. For the inertia matrix, we have:

$$\begin{bmatrix} I_{11} & I_{12} & I_{13} \\ I_{21} & I_{22} & I_{23} \\ I_{31} & I_{32} & I_{33} \end{bmatrix} \tag{A.1}$$

If we want to add an uncertain term ϵ we have:

$$\begin{bmatrix} I_{11}+\epsilon_{11} & I_{12}+\epsilon_{12} & I_{13}+\epsilon_{13} \\ I_{21}+\epsilon_{21} & I_{22}+\epsilon_{22} & I_{23}+\epsilon_{23} \\ I_{31}+\epsilon_{31} & I_{32}+\epsilon_{32} & I_{33}+\epsilon_{33} \end{bmatrix} \tag{A.2}$$

What we are going to do is to estimate the determinant to know the limit values of these uncertain terms to have the model in our limit. Then:

$$\begin{aligned} det[M(q)] = [(I_{22}+\epsilon_{22})(I_{33}+\epsilon_{33}) - (I_{23}+\epsilon_{23})(I_{32}+\epsilon_{32})](I_{11}+\epsilon_{11}) - \\ [(I_{21}+\epsilon_{21})(I_{33}+\epsilon_{33}) - (I_{23}+\epsilon_{23})(I_{13}+\epsilon_{13})](I_{12}+\epsilon_{12}) + \\ [(I_{21}+\epsilon_{21})(I_{32}+\epsilon_{32}) - (I_{22}+\epsilon_{22})(I_{31}+\epsilon_{31})](I_{13}+\epsilon_{13}) > 0 \end{aligned} \tag{A.3}$$

As there is symmetry in the matrix and therefore:

$$\begin{aligned} &(I_{11}+\epsilon_{11})(I_{22}+\epsilon_{22})(I_{33}+\epsilon_{33}) - (I_{11}+\epsilon_{11})(I_{23}+\epsilon_{23})^2 - (I_{22}+\epsilon_{22})(I_{13}+\epsilon_{13})^2 - \\ &(I_{33}+\epsilon_{33})(I_{12}+\epsilon_{12})^2 + 2(I_{12}+\epsilon_{12})(I_{23}+\epsilon_{23})(I_{13}+\epsilon_{13}) > 0 \end{aligned} \tag{A.4}$$

For our first analysis let us assume that we are working with the principal axis of inertia, implying that the last equation becomes:

$$(I_{11}+\epsilon_{11})(I_{22}+\epsilon_{22})(I_{33}+\epsilon_{33}) > 0 \tag{A.5}$$

We know that the elements of the diagonal must be positive, but the error could be positive or negative:

-$I_{11} < \epsilon_{11} < \infty$	-$I_{22} < \epsilon_{22} < \infty$	-$I_{33} < \epsilon_{33} < \infty$

For our second analysis, from the latter we know that the diagonal must be positive or zero, and the square terms must be positive or zero too. If we define $\alpha = (I_{11} + \epsilon_{11})(I_{22} + \epsilon_{22})(I_{33} + \epsilon_{33})$, $-\gamma = (I_{11}+\epsilon_{11})(I_{23}+\epsilon_{23})^2+(I_{22}+\epsilon_{22})(I_{13}+\epsilon_{13})^2+(I_{33}+\epsilon_{33})(I_{12}+\epsilon_{12})^2$ and$\varphi = 2(I_{12}+\epsilon_{12})(I_{23}+\epsilon_{23})(I_{13}+\epsilon_{13})$, then:

$$\alpha + \varphi - \gamma > 0 \tag{A.6}$$

Then $\varphi < 0$

Case 1. The three terms are negative

$(-)$	$(-)$	$(-)$
if $0 \leq I_{12}$	if $0 \leq I_{23}$	if $0 \leq I_{13}$
$\Rightarrow \epsilon_{12} \leq -I_{12}$	$\Rightarrow \epsilon_{23} \leq -I_{23}$	$\Rightarrow \epsilon_{13} \leq -I_{13}$
if $I_{12} < 0$	if $I_{23} < 0$	if $I_{13} < 0$
$\Rightarrow -\infty < \epsilon_{12} \leq 0$	$\Rightarrow -\infty < \epsilon_{23} \leq 0$	$\Rightarrow -\infty < \epsilon_{13} \leq 0$

Case 2. Two terms are positive one is negative

$(+)$	$(+)$	$(-)$
if $0 \leq I_{12}$	if $0 \leq I_{23}$	if $0 \leq I_{13}$
$\Rightarrow -I_{12}\epsilon_{12} \leq \infty$	$\Rightarrow I_{23}\epsilon_{23} \leq \infty$	$\Rightarrow \epsilon_{13} \leq -I_{13}$
if $I_{12} < 0$	if $I_{23} < 0$	if $I_{13} < 0$
$\Rightarrow I_{12} \leq \epsilon_{12}$	$\Rightarrow I_{23} \leq \epsilon_{23}$	$\Rightarrow -\infty < \epsilon_{13} \leq 0$

(Similarly for the other cases where one term is negative).

In other words, if ϵ is enough small to get the cases presented, we can have a valid dynamical model, which for our case we achieve with our simulations and their respective inputs.

Appendix B

Skew symmetric matrices

B.1 Skew-symmetric matrix

A skew-symmetric matrix is defined as a $n \times n$ matrix S if and only if (Siciliano et al., 2008):

$$\mathrm{S}^{\mathrm{T}} + \mathrm{S} = 0 \tag{B.1}$$

We denote the set of all 3×3 skew symmetric matrices as $so(3)$. If $\mathrm{S} \in so(3)$ with components $s_{ij}, i, j = 1, 2, 3$ then from equation (B.1) we can see that $s_{ii} = 0$; that is that the diagonal terms of S are zero and the off diagonal terms $s_{ij}, i \neq j$ satisfy $s_{ij} = -s_{ji}$. Therefore S contains only three independent entries and every 3×3 skew symmetric matrix has the form of:

$$\mathrm{S} = \begin{bmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{bmatrix} \tag{B.2}$$

Now, if we have a 3-vector $\mathbf{a} = (a_x, a_y, a_z)^T$, we can define the skew symmetric matrix S(a) as:

$$\mathrm{S} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{B.3}$$

Skew symmetic matrix properties

1. The operator S is linear, which means:

$$\mathrm{S}(\alpha\,\mathbf{a} + \beta\,\mathbf{b}) = \alpha\,\mathrm{S}(\mathbf{a}) + \mathrm{S}(\mathbf{b}); \tag{B.4}$$

   With $a, b \in \mathbb{R}^3$ and $\alpha, \beta \in \mathbb{R}$

2. For any vectors $\mathbf{a}$ and $\mathbf{p} \in \mathbb{R}$, then:

$$\mathrm{S}(\mathbf{a})\,\mathbf{p} = \mathbf{a} \times \mathbf{p} \tag{B.5}$$

   any rotation matrix $\mathrm{R} \in SO(3)$ and $\mathbf{a} \in \mathbb{R}^3$:

$$\mathrm{R}\,\mathrm{S}(\mathbf{a})\mathrm{R}^{\mathrm{T}} = \mathrm{S}(\mathrm{R}\mathbf{a}) \tag{B.6}$$

3. For $\mathrm{S} \in so(n)$ and $\mathbf{a} \in \mathbb{R}^n$, then:

$$\mathbf{a}^{\mathbf{T}}\,\mathrm{S}\,\mathbf{a} = 0 \tag{B.7}$$

Appendix C

Rotation matrix derivation and properties

C.1 Derivation of a Rotation Matrix

If a rotation matrix R is a function of a single variable q Hence R=R(q) $\in SO(3)$ for every q. Since R is orthogonal for all q then:

$$\mathrm{R}(q)\,\mathrm{R}(q)^{\mathrm{T}} = \mathrm{I} \tag{C.1}$$

If we differentiate both sides of the last equation we obtain:

$$\frac{d\mathrm{R}}{dq}\mathrm{R}(q)^{T} + \mathrm{R}(q)\frac{d\mathrm{R}}{dq} = 0 \tag{C.2}$$

Now we define:

$$\mathrm{S} \equiv \frac{d\mathrm{R}}{dq}\mathrm{R}(q)^{\mathrm{T}} \tag{C.3}$$

Therefore:

$$\mathrm{S}^{\mathrm{T}} = \mathrm{R}(q)\frac{d\mathrm{R}}{dq} \tag{C.4}$$

which is represented in equation (B.1).

In different words, the matrix defined in equation (C.3) is a skew-symmetric matrix. If we multiply both sides of equation (C.3) on the right by R and the property of equation (C.1) we obtain:

$$\frac{d\mathrm{R}}{dq} = \mathrm{S}\,\mathrm{R}(q) \tag{C.5}$$

As an example, let us assume that R=$\mathrm{R}_{x,q}$. It means the rotation by an angle q about the axis x, which:

$$\mathrm{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cq & sq \\ 0 & sq & -cq \end{bmatrix} = \tag{C.6}$$

$$\frac{d\mathrm{R}}{dq} = \mathrm{S}\,\mathrm{R}(q) = \begin{bmatrix} 0 & -a_z\,cq + a_y\,sq & a_z\,sq + a_y\,cq \\ a_z & -a_x\,sq & -a_x\,cq \\ -a_y & a_x\,cq & -a_x\,sq \end{bmatrix} \therefore a_z = a_y = 0 \tag{C.7}$$

or if $a_x = 1$

$$\frac{d}{dq}\mathrm{R} = \mathrm{S}([1\,0\,0]^T)\,R(q); \tag{C.8}$$

Now, using equation (C.5) and differentiating R in terms of t:

$$\dot{\mathrm{R}}(t) = \frac{d\mathrm{R}}{dt} = \frac{d\mathrm{R}}{dq}\frac{dq}{dt} = \mathrm{S}([1\,0\,0]^{\mathrm{T}})\,\mathrm{R}(q)\,\dot{q} = \mathrm{S}([\dot{\mathbf{q}}\,0\,0]^{\mathrm{T}})\,\mathrm{R}(q) = \mathrm{S}(\dot{q}(t))\,\mathrm{R}(q(t)) \tag{C.9}$$

Finally:

$$\dot{\mathrm{R}}(t) = \dot{\mathbf{q}}\,\mathrm{R}(t) \tag{C.10}$$

or if we multiply by the right by $\mathrm{R}^{\mathrm{T}}(t)$:

$$\dot{\mathbf{q}} = \dot{\mathrm{R}}(t)\,\mathrm{R}^{\mathrm{T}}(t) \tag{C.11}$$

C.2 Rotation matrix properties

Rotational matrix has several special properties, the most important are:

1. The determinant of R is one:

$$det(\mathrm{R}) = 1 \tag{C.12}$$

2. The inverse of R is its transpose:

$$\mathrm{R}^{-1} = \mathrm{R}^T \tag{C.13}$$

3. The dot product of any row with itself is always one.
4. The dot product of any row with any other row is always zero.
5. The dot product of any column with any other column is always zero.

Appendix D

Torque free motion solution

One curious case is that o torque-free motion. Let's imagine we are working with the principal axis of inertia and that two of the inertia are equal. It means $I = I_1 = I_2 \neq I_3$. If we have no torques, from the Euler's equation of motion, we have:

$$I_1\ddot{q}_1 - (I_3)\dot{q}_3\dot{q}_2 = 0 \tag{D.1}$$

$$I_2\ddot{q}_2 - (I_3 - I)\dot{q}_1\dot{q}_3 = 0 \tag{D.2}$$

$$I_3\ddot{q}_3 = 0 \tag{D.3}$$

From equation (D.3), we note that $\dot{q}_3 = \Omega$, which is a constant.

Now, we define a new variable for equations (D.1) and (D.2):

$$\lambda = \frac{I_3 - I}{I}\,\dot{q}_3 \tag{D.4}$$

As a consequence:

$$I_1\ddot{q}_1 + \lambda\dot{q}_2 = 0 \tag{D.5}$$

$$I_2\ddot{q}_2 - \lambda\dot{q}_1 = 0 \tag{D.6}$$

If we time derived the last two-equation then:

$$I_1\dddot{q}_1 + \lambda\ddot{q}_2 = 0 \tag{D.7}$$

$$I_2\dddot{q}_2 - \lambda\ddot{q}_1 = 0 \tag{D.8}$$

Now, substituting equation D.6 in D.7 we obtain:

$$I_1\dddot{q}_1 + \lambda^2\dot{q}_1 = 0 \tag{D.9}$$

which as the form of the well known simple harmonic motion differential equation with solution:

$$\dot{q}_1 = Acos(\lambda t) \tag{D.10}$$

Similarly,

$$\dot{q}_2 = Asin(\lambda t) \tag{D.11}$$

The last solutions are present in many other fields in physics showing that the theory applied in different fields can be useful to solve other problems.

Appendix E

Code and Block Diagram

Presents the code used to obtain the results presented in chapter 5. For more information about the code used please contact the author.

E.1 Dynamic model code

1dof dynamic model code (*dynamic_model_simulation_1dof.m*)

```
% clc
  clear all
% close all

%variables are defined
syms qd1 qd2 qd3 qdd1 qdd2 qdd3 I11 I22 I33 m

g=[0;0;-9.81];   %gravity vector
r=[0;5;0]; %Bodyframe vector
rho=[0;0;0.05]; %Center of mass vector
m=1.13; %satellite's mass

q_ic = [0, 0]; % initial conditions: q(t=0)=0.5; qd(t=0)=0.
tspan = [0:0.05:50];
[t, q] = ode45(@odeFun, tspan, q_ic);

%We plot the angular position and velocity for q1
figure
plot(t, q);
yline(pi)
legend({'$q_{1}$', '$\dot{q_{1}}$'},'Location', 'best','Interpreter', ...
    'latex','FontSize', 16)
title('1 reaction wheel (model)')
xlabel('time (s)')
ylabel('Pos(rad), Ang vel(rad/s) & Singal')
grid on
i=gca;
set(i,'PlotBoxAspectRatio',[1,1/2,1]) %aayuda a escalar los ejes
export_fig i -pdf -transparent model_1dofvelocity_position.pdf %nombre de la figura

q1=q(:,1);

%We calculate de rotation matrix and external torques due to gravity
for i=1:1001

EA=  [ cos(q1(i)),sin(q1(i)),   0;
       -sin(q1(i)), cos(q1(i)), 0;
             0,         0   ,   1];

rp=EA*r;
```

```
40
41  rhokk1=rho;
42  rhogkk=m*EA*g;
43
44  taug=cross(rhokk1,rhogkk);
45
46  x2(i)=rp(1);
47  y2(i)=rp(2);
48  z2(i)=rp(3);
49
50  end
51
52  %We plot the workspace
53  figure
54  plot3(x2,y2,z2,'-r','LineWidth',2)
55  title('1 reaction wheel workspace (model)')
56  xlabel('x (cm)')
57  ylabel('y (cm)')
58  zlabel('z (cm)')
59
60  hold on
61  %We plot a sphere surface
62  [X Y Z] = sphere(30);
63  X2 = X * r(2);
64  Y2 = Y * r(2);
65  Z2 = Z * r(2);
66  contour3(X2,Y2,Z2,8,'-.')
67  hold on
68  h = surfl(X2, Y2, Z2);
69  set(h, 'FaceAlpha', 0.1)
70  shading interp
71  grid on
72  set(gca,'PlotBoxAspectRatio',[1,1,1]) %scale the axis
73  export_fig gca -pdf model_1dof_spacework2.pdf %Figure name -transparent
74
75  figure
76  h=plot(x2,y2,'r','LineWidth',2)
77  title('1 reaction wheel workspace (model)')
78  xlabel('x (cm)')
79  ylabel('y (cm)')
80  grid on
81  h=gca;
82  set(h,'PlotBoxAspectRatio',[1/2,1,1]) %scale the axis
83  export_fig h -pdf -transparent model_1dof_spacework.pdf %Figure name
84
85  % figure
86  % plot3(taug(1),taug(2),taug(3))
87  %     title('gravity')
88  %     xlabel('x (cm)')
89  %     ylabel('y (cm)')
90  %     zlabel('z (cm)')
91  %     grid on
92
93  %This funtion solve the differential equation
94  function dq = odeFun(t, q)
95
96
97    % I33=0.0000164108;    %reaction wheel inertia kgm2
98      I1=0.02487; %Inertia (kgm2)
99      rho=[0;0;0.05]; %CM vector (m)
100     ms=1.13; %satellite's mass (kg)
101     mw=0.0052;  %masa de la rueda inercial
102     m=ms+4*mw; %total mass
103     g=[0;0;-9.81];   %gravity vector (m/s2)
104
105     kp=1; %controler constant kp
106     kv=10; %controler constant kv
107     qD3=pi; %desired position for q1
108     qe=qD3-q(1); %estimated error for q1
109     b=0.01315; %friction constant
```

```
110     %m=0.0052;  %inertial wheel mass
111     tau=kp*qe-kv*q(2);%+sin(t); %controller
112
113     % q(1) = q, q(2) = dq
114
115     q(1);
116     %the system is solved
117     dq = zeros(2, 1);
118
119     %states vector
120     dq(1) = q(2);
121
122     %rotation matrix Z
123     EA=  [ cos(q(1)),sin(q(1)),   0;
124            -sin(q(1)), cos(q(1)), 0;
125                  0,         0   ,   1];
126     %torque due to gravity
127     rhokk1=rho;
128     rhogkk=m*EA*g;
129     taug=cross(rhokk1,rhogkk);
130
131     %final differential equation
132     dq(2) = tau/I1+taug(3)-b*qe/I1+erf(t);%+(m*d*g)/I1;%; %descomentar el 2do ...
            termino del lado izquierdo de la igualdad para agregar gravedad
133
134 end
```

2dof dynamic model code (*dynamic_model_simulation_2dof.m*)

```
%Modelo 2sdof
% clc
 clear all
% close all

%variables are defined
syms qd1 qd2 qd3 qdd1 qdd2 qdd3 I11 I22 I33 m

I1=0.002487; %The inertia of one axis
I2=I1; %two axis with the same inertia value
I3=0.0025188; %inertia of the third axis
g=[0;0;-9.81];   %gravity vector
r=[0;5;0]; %Bodyframe vector
rho=[0;0;0.05]; %Center of mass vector
m=1.13; %satellite's mass

%We define a variable to simplify
Ij=(I2-I3)/I1;
Ik=(I3-I1)/I2;

q_i=[0,0,0,0]; %initial conditions
tspan=[0:0.05:50]; %time
[t,q]=ode45(@odeEuler,tspan,q_i); %the equations are solved

%We plot the angular position
figure
plot(t, q(:,1),t, q(:,3))
legend({ '${q_1}$','${q_2}$'},'Location', 'best','Interpreter', ...
    'latex','FontSize', 16)
title('2 reaction wheel (model)')
xlabel('time (s)')
ylabel('Ang pos(rad)')
grid on
k=gca;
set(k,'PlotBoxAspectRatio',[1,1/2,1]); %scale the axis
export_fig k -pdf -transparent model_2dofposition.pdf %name of the figure

%We plot the angular velocity
figure
plot(t, q(:,2),t, q(:,4))
legend({ '$\dot{q_1}$','$\dot{q_2}$'},'Location', 'best','Interpreter', ...
    'latex','FontSize', 16)
title('2 reaction wheel (model)')
xlabel('time (s)')
ylabel('Ang vel(rad/s)')
grid on
l=gca;
set(l,'PlotBoxAspectRatio',[1,1/2,1]); %scale the axis
export_fig l -pdf -transparent model_2dofvelocity.pdf %name of the figure


figure
plot(t, q(:,5),t, q(:,6))
legend({ '${q_3}$','$\dot{q_3}$'},'Location', 'best','Interpreter', ...
    'latex','FontSize', 16)
title('3 reaction wheel (model)')
xlabel('time (s)')
ylabel('Pos(rad), Ang vel(rad/s)')
grid on

q1=q(:,1);
q2=q(:,3);

%We calculate de rotation matrix and the spacework
for i=1:601

EA=  [cos(q1(i)), cos(q2(i)).*sin(q1(i)), sin(q1(i)).*sin(q2(i));
```

```
 -sin(q1(i)), cos(q1(i)).*cos(q2(i)), cos(q1(i)).*sin(q2(i));
        0,        -sin(q2(i)),         cos(q2(i))];  %ZX

rp=EA*r;

x2(i)=rp(1);
y2(i)=rp(2);
z2(i)=rp(3);

end

%We plot the workspace
figure
plot3(x2,y2,z2,'-r','LineWidth',2)
title('2 reaction wheels workspace')
xlabel('x (cm)')
ylabel('y (cm)')
zlabel('z (cm)')
%We plot a sphere surface
hold on
[X Y Z] = sphere(30);
X2 = X * r(2);
Y2 = Y * r(2);
Z2 = Z * r(2);
contour3(X2,Y2,Z2,8,'-.')
%hold on
h = surfl(X2, Y2, Z2);
set(h, 'FaceAlpha', 0.1)
shading interp
grid on
j=gca;
set(j,'PlotBoxAspectRatio',[1,1,1]); %scale the axis
export_fig j -pdf  model_2dof_spacework2.pdf %name of the figure


%This funtion solve the differential equation
function dq = odeEuler(t, q)

    I1=0.002487; %Inertia(kgm2)
    I2=0.002487;
    I3=0.0025188;
    rho=[0;0;0.05]; %CM vector (m)
    ms=1.13; %satellite's mass (kg)
    mw=0.0052;  %masa de la rueda inercial
    m=ms+4*mw; %total mass
    g=[0;0;-9.81];   %gravity vector (m/s2)
    b=0.01315; %friction constant

    %We do a susbtition
    Ij=(I2-I3)/I1;
    Ik=(I3-I1)/I2;
    Il=(I1-I2)/I3;

    kp=1; %controler constant kp
    kv=10; %controler constant kv
    qD1=pi; %desired position for q1
    qD2=2*pi;  %desired position for q2
    qe1=qD1-q(1); %estimated error for q1
    qe2=qD2-q(3); %estimated error for q2
    qe3=pi;

    tau1=kp*qe1-kv*q(2);%+90*sin(t); %controller for q1
    tau2=kp*qe2-kv*q(4); %+ 90*cos(t); %controller for q2

    dq = zeros(4, 1);

    %rotation matrix ZX
    EA=  [cos(qe1), cos(qe3).*sin(qe1), sin(qe1).*sin(qe3);
          -sin(qe1), cos(qe1).*cos(qe3), cos(qe1).*sin(qe3);
                    0,        -sin(qe3),         cos(qe3)];
```

```
135
136 %        EA=  [cos(q(1)), cos(q(3)).*sin(q(1)), sin(q(1)).*sin(q(3));
137 %             -sin(q(1)), cos(q(1)).*cos(q(3)), cos(q(1)).*sin(q(3));
138 %             0,        -sin(q(3)),         cos(q(3))];
139
140     %Torque due to gravity
141     rhokk1m=rho; %vector to the CM
142     rhogkkm=m*EA*g; %estimation the gravity term
143     taugm=cross(rhokk1m,rhogkkm)  ; %estimation of the torque
144
145     % q(1) = q, q(2) = dq
146     %states vector for q1
147     dq(1) = q(2);
148     %final differential equation for q1
149     dq(2) = tau1/I1+Ij*qe2*qe3 +taugm(3)-b*qe1/I1;
150
151     %states vector for q2
152     dq(3) = q(4);
153     %final differential equation for q2
154     dq(4) = tau2/I2+Ik*qe1*qe3-taugm(2)-b*qe2/I2;
155 end
```

3dof dynamic model code (*dynamic_model_simulation_3dof.m*)

```
%Modelo 3dof
% clc
 clear all
% close all

%variables are defineds
syms qd1 qd2 qd3 qdd1 qdd2 qdd3 I11 I22 I33 m

I1=0.002487; %The inertia of one axis
I2=I1; %two axis with the same inertia value
I3=0.0025188; %inertia of the third axis
g=[0;0;-9.81];   %gravity vector
r=[0;5;0]; %Bodyframe vector (cm)
rho=[0;0;0.05]; %Center of mass vector
m=1.13; %satellite's mass

%We define a variable to simplify
Ij=(I2-I3)/I1;
Ik=(I3-I1)/I2;

q_i=[0,0,0,0,0,0]; %initial conditions
tspan=[0:0.05:50]; %time
[t,q]=ode45(@odeEuler,tspan,q_i); %the equations are solved

%We plot the angular position
figure
plot(t, q(:,1),t, q(:,3),t, q(:,5))
legend({ '${q_1}$','${q_2}$','${q_3}$'},'Location', 'best','Interpreter', ...
    'latex','FontSize', 16)
title('3 reaction wheel (model)')
xlabel('time (s)')
ylabel('Ang pos(rad)')
grid on
k=gca;
set(k,'PlotBoxAspectRatio',[1,1/2,1]); %ayuda a escalar los ejes
export_fig k -pdf -transparent model_3dofposition.pdf %figure name

%We plot the angular velocity
figure
plot(t, q(:,2),t, q(:,4),t, q(:,6))
legend({ '$\dot{q_1}$','$\dot{q_2}$','$\dot{q_3}$'},'Location', ...
    'best','Interpreter', 'latex','FontSize', 16)
title('3 reaction wheel (model)')
xlabel('time (s)')
ylabel('Ang vel(rad/s)')
grid on
l=gca;
set(l,'PlotBoxAspectRatio',[1,1/2,1]); %ayuda a escalar los ejes
export_fig l -pdf -transparent model_3dofvelocity.pdf %nombre de la figura


q1=q(:,1);
q2=q(:,3);
q3=q(:,5);

%We calculate de rotation matrix and the spacework
for i=1:601

EA=[   cos(q1(i)).*cos(q3(i)) - cos(q2(i)).*sin(q1(i)).*sin(q3(i)), ...
    cos(q1(i)).*sin(q3(i)) + cos(q2(i)).*cos(q3(i)).*sin(q1(i)), ...
    sin(q1(i)).*sin(q2(i));
 - cos(q3(i)).*sin(q1(i)) - cos(q1(i)).*cos(q2(i)).*sin(q3(i)), ...
     cos(q1(i)).*cos(q2(i)).*cos(q3(i)) - sin(q1(i)).*sin(q3(i)), ...
     cos(q1(i)).*sin(q2(i));
                            sin(q2(i)).*sin(q3(i)),                          ...
                                -cos(q3(i)).*sin(q2(i)),         cos(q2(i))] ;  %ZXZ

```

```
61  rhokk1=rho;
62  rhogkk=m*EA*g;
63
64  taug=cross(rhokk1,rhogkk);
65  taugx(i)=taug(1);
66  taugy(i)=taug(2);
67  taugz(i)=taug(3);
68
69  rp=EA*r;
70
71  x2(i)=rp(1);
72  y2(i)=rp(2);
73  z2(i)=rp(3);
74
75  end
76
77  %We plot the workspace
78  figure
79  plot3(x2,y2,z2,'-r','LineWidth',2)
80  title('3 reaction wheels workspace')
81  xlabel('x (cm)')
82  ylabel('y (cm)')
83  zlabel('z (cm)')
84  grid on
85
86  %We plot a sphere surface
87  hold on
88  [X Y Z] = sphere(30);
89  X2 = X * r(2);
90  Y2 = Y * r(2);
91  Z2 = Z * r(2);
92  contour3(X2,Y2,Z2,8,'-.')
93  hold on
94  h = surfl(X2, Y2, Z2);
95  set(h, 'FaceAlpha', 0.1)
96  shading interp
97  grid on
98  j=gca;
99  set(j,'PlotBoxAspectRatio',[1,1,1]); %ayuda a escalar los ejes
100 export_fig j -pdf model_3dof_spacework.pdf %nombre de la figura
101
102 %This funtion solve the differential equation
103 function dq = odeEuler(t, q)
104
105     I1=0.002487; %Inertia (kgm2)
106     I2=0.002487; %Inertia (kgm2)
107     I3=0.0025188; %Inertia (kgm2)
108     rho=[0;0;0.05]; %CM vector (m)
109     ms=1.13; %satellite's mass (kg)
110     mw=0.0052;  %masa de la rueda inercial
111     m=ms+4*mw; %total mass
112     g=[0;0;-9.81];   %gravity vector (m/s2)
113     b=0.01315; %friction constant
114
115     kp=1; %controler constant kp
116     kv=10; %controler constant kv
117     qD1=2*pi; %desired position for q1
118     qD2=1.5*pi; %desired position for q2
119     qD3=pi; %desired position for q3
120     qe1=qD1-q(1); %estimated error for q1
121     qe2=qD2-q(3); %estimated error for q2
122     qe3=qD3-q(5); %estimated error for q3
123
124     %We do a susbtition
125     Ij=(I2-I3)/I1;
126     Ik=(I3-I1)/I2;
127     Il=(I1-I2)/I3;
128
129     tau1=kp*qe1-kv*q(2);%sin(t); %controller for q1
130     tau2=kp*qe2-kv*q(4)  ;%+sin(t); %controller for q2
```

```
131     tau3=kp*qe3-kv*q(6);%+sin(t);%controller for q3
132
133     dq = zeros(6, 1);
134     %se resuelve el sistema para q3
135
136     %rotation matrix ZXZ
137     EA=[   cos(q(1)).*cos(q(3)) - cos(q(2)).*sin(q(1)).*sin(q(3)), ...
            cos(q(1)).*sin(q(3)) + cos(q(2)).*cos(q(3)).*sin(q(1)), sin(q(1)).*sin(q(2));
138         - cos(q(3)).*sin(q(1)) - cos(q(1)).*cos(q(2)).*sin(q(3)), ...
                cos(q(1)).*cos(q(2)).*cos(q(3)) - sin(q(1)).*sin(q(3)), ...
                cos(q(1)).*sin(q(2));
139                             sin(q(2)).*sin(q(3)),                          ...
                                    -cos(q(3)).*sin(q(2)),         cos(q(2))] ;  %ZXZ
140
141     %Torque due to gravity
142     rhokk1m=rho;
143     rhogkkm=m*EA*g;
144     taugm=cross(rhokk1m,rhogkkm);
145
146     % q(1) = q, q(2) = dq
147     %states vector for q1
148     dq(1) = q(2);
149     %final differential equation for q1
150     dq(2) = tau1/I1+Ij*qe2*qe3+taugm(3)-b*qe1/I1;; %descomentar el 2do termino ...
            del lado izquierdo de la igualdad para agregar gravedad
151
152     %states vector for q2
153     dq(3) = q(4);
154     %final differential equation for q2
155     dq(4) = tau2/I2+Ik*qe1*qe3+taugm(2)-b*qe2/I2; %descomentar el 2do termino del ...
            lado izquierdo de la igualdad para agregar gravedad
156
157     %states vector for q3
158     dq(5) = q(6);
159     %final differential equation for q3
160     dq(6) = tau3/I3+Il*qe1*qe2+taugm(1)-b*qe3/I3; %descomentar el 2do termino del ...
            lado izquierdo de la igualdad para agregar gravedad
161 end
```

E.2 Simulink simulations rigid body

1dof Simulink block diagram rigid body dynamics (*RB_simulink_simulation_1dof.slx*)

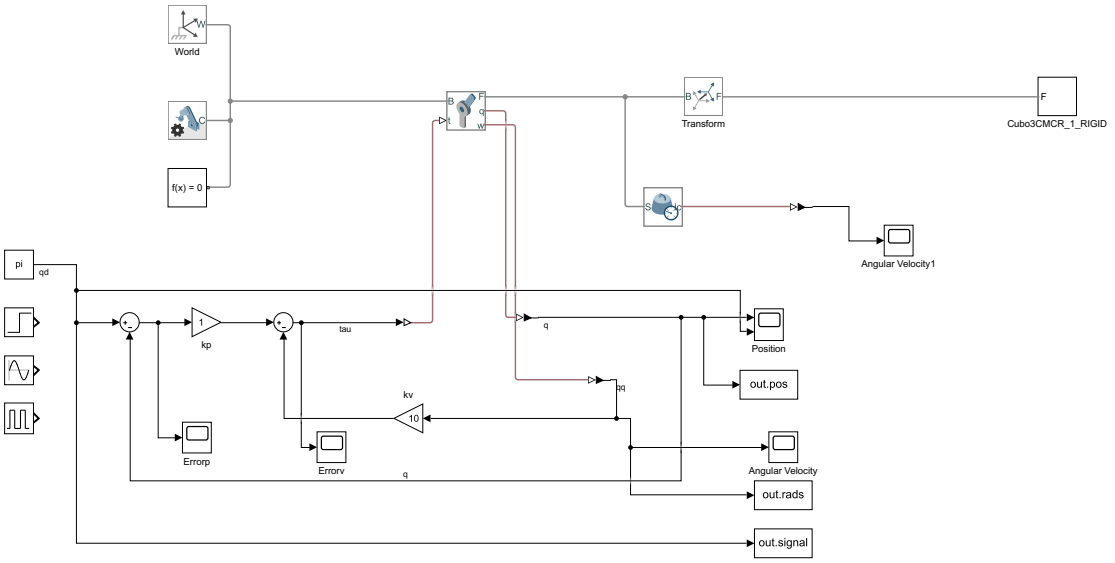


Figure E.1: Block diagram simulation of 1-dof CubeSat (rigid body).

1dof Simulink simulation rigid body dynamics code (*RB_simulink_simulation_1dof.m*)

```
% code for RBD 1-dof
% we define the variables taken from Simulink
q1=out.pos;
r=[0;5;0];

% we begin a cicle where the rotational matrix will be estimated
for i=1:65085

EA=  [ cos(q1(i)),sin(q1(i)),   0;
       -sin(q1(i)), cos(q1(i)), 0;
             0,         0   ,   1];

rhop=EA*r;

x2(i)=rhop(1);
y2(i)=rhop(2);
z2(i)=rhop(3);

end

% we plot the angular postion and velocity from simulink
plot(out.tout,out.pos, out.tout,out.rads)%,out.tout,out.signal)
yline(out.signal)
legend({'$q_{1}$', '$\dot{q_{1}}$'},'Location', 'best','Interpreter', ...
    'latex','FontSize', 16)
title('1 reaction wheel (simulation)')
xlabel('time (s)')
ylabel('Pos(rad), Ang vel(rad/s) & Signal')
grid on
i=gca;
set(i,'PlotBoxAspectRatio',[1,1/2,1])
export_fig i -pdf -transparent Cubo3CMRvelocity_position.pdf %we export the figure

% we plot the angular the workspace from simulink
figure
plot3(x2,y2,z2,'-b','LineWidth',2)
title('1 reaction wheel workspace(simulation)')
xlabel('x (cm)')
ylabel('y (cm)')
zlabel('z (cm)')
grid on

%We plot a sphere surface
hold on
[X Y Z] = sphere(30);
X2 = X * r(2);
Y2 = Y * r(2);
Z2 = Z * r(2);
contour3(X2,Y2,Z2,8,'-.')
hold on
h = surfl(X2, Y2, Z2);
set(h, 'FaceAlpha', 0.1)
shading interp
grid on
set(gca,'PlotBoxAspectRatio',[1,1,1])
export_fig gca -pdf Cubo3CMRSpace_work.pdf %we export the figure -transparent

figure
h=plot(x2,y2,'-b','LineWidth',2)
title('1 reaction wheel workspace (simulation)')
xlabel('x (cm)')
ylabel('y (cm)')
grid on
h=gca;
set(h,'PlotBoxAspectRatio',[1/2,1,1]) % scale the axis
export_fig h -pdf -transparent Cubo3CMRSpace_work2.pdf %name of the figure
```

2dof Simulink block diagram rigid body dynamics (*RB_simulink_simulation_2dof.slx*)

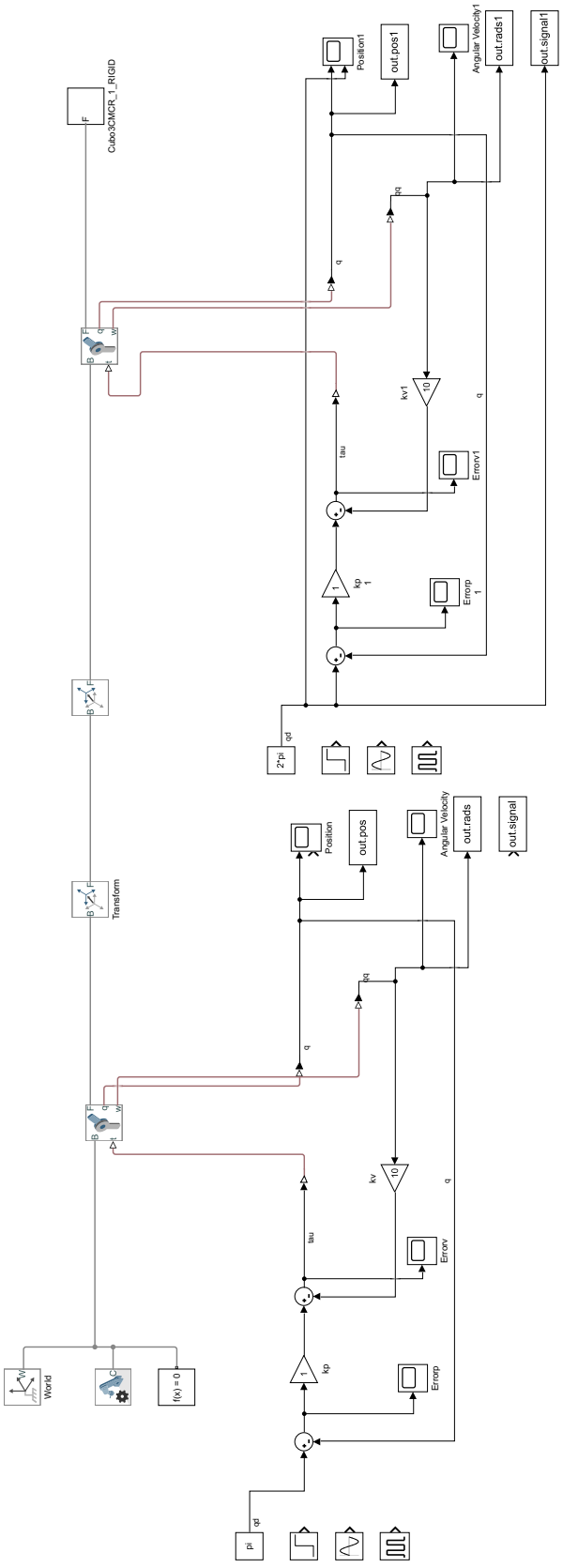


Figure E.2: Block diagram simulation of 2-dof CubeSat (rigid body).

2dof Simulink simulation rigid body dynamics code (*RB_simulink_simulation_2dof.m*)

```
% code for RBD 2-dof
% we define the variables taken from Simulink
q1=out.pos;
q3=out.pos1;
r=[0;5;0]; %BodyFrame vector (cm)

%We calculate de rotation matrix and the spacework
for i=1:60574
EA=[ cos(q1(i)),  sin(q1(i)).*cos(q3(i)) , sin(q1(i)).*sin(q3(i));
       -sin(q1(i)), cos(q1(i)).*cos(q3(i)), cos(q1(i)).*sin(q3(i));
       0, -sin(q3(i)), cos(q3(i))];   %ZX
rhop=EA*r;
x2(i)=rhop(1);
y2(i)=rhop(2);
z2(i)=rhop(3);

end

%We plot the angular position for the reaction wheels
figure
plot(out.tout,out.pos,out.tout,out.pos1)
legend({'$q_{1}$','$q_{2}$'},'Location', 'best','Interpreter', ...
    'latex','FontSize', 16)
title('2 reaction wheels (simulation)')
xlabel('time (s)')
ylabel('Ang pos(rad)')
grid on
k=gca;
set(k,'PlotBoxAspectRatio',[1,1/2,1]);
export_fig k -pdf -transparent Cubo3CMR2dofposition.pdf

%We plot the angular velocity for the reaction wheels
figure
plot(out.tout,out.rads,out.tout,out.rads1)
legend({'$\dot{q_{1}}$','$\dot{q_{2}}$'},'Location', 'best','Interpreter', ...
    'latex','FontSize', 16)
title('2 reaction wheels (simulation)')
xlabel('time (s)')
ylabel('Ang vel(rad/s)')
grid on
l=gca;
set(l,'PlotBoxAspectRatio',[1,1/2,1]);
export_fig l -pdf -transparent Cubo3CMR2dofvelocity.pdf

%We plot the worspace
figure
plot3(x2,y2,z2,'-b','LineWidth',2)
title('2 reaction wheels workspace (simulation)')
xlabel('x (cm)')
ylabel('y (cm)')
zlabel('z (cm)')

%We plot a sphere surface
hold on
[X Y Z] = sphere(30);
X2 = X * r(2);
Y2 = Y * r(2);
Z2 = Z * r(2);
contour3(X2,Y2,Z2,8,'-.')
hold on
h = surfl(X2, Y2, Z2);
set(h, 'FaceAlpha', 0.1)
shading interp
grid on
j=gca;
set(j,'PlotBoxAspectRatio',[1,1,1]);
export_fig j -pdf  Cubo3CMR2dofSpace_work3.pdf
```

3dof Simulink block diagram rigid body dynamics (*RB_simulink_simulation_3dof.slx*)

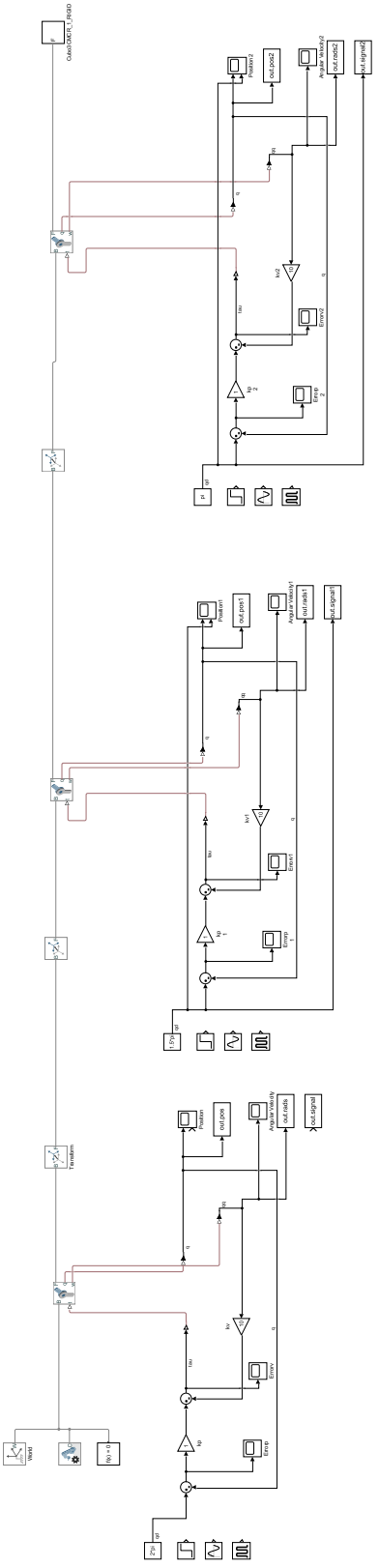


Figure E.3: Block diagram simulation of 3-dof CubeSat (rigid body).

3dof Simulink simulation rigid body dynamics code (*RB_simulink_simulation_3dof.m*)

```
% we define the variables taken from Simulink
q1=out.pos;
q2=out.pos1;
q3=out.pos2;
d1=5;
d2=0;
d3=5;
KK0=q1*0;

% we begin a cicle where the homogenous matrix will be estimated
for i=1:3076

   A03= [ sin(q1(i)).*sin(q3(i)) + cos(q1(i)).*cos(q2(i)).*cos(q3(i)), ...
        cos(q3(i)).*sin(q1(i)) - cos(q1(i)).*cos(q2(i)).*sin(q3(i)), ...
        cos(q1(i)).*sin(q2(i)), d3.*cos(q1(i)).*sin(q2(i))-d2.*sin(q1(i)); ...
        cos(q2(i)).*cos(q3(i)).*sin(q1(i)) - cos(q1(i)).*sin(q3(i)), - ...
        cos(q1(i)).*cos(q3(i)) - cos(q2(i)).*sin(q1(i)).*sin(q3(i)), ...
        sin(q1(i)).*sin(q2(i)), d2.*cos(q1(i)) + d3.*sin(q1(i)).*sin(q2(i)); ...
        -cos(q3(i)).*sin(q2(i)), sin(q2(i)).*sin(q3(i)), cos(q2(i)), d1 + ...
        d3.*cos(q2(i)); 0,   0,   0,   1];

x2(i)=A03(1,4);
y2(i)=A03(2,4);
z2(i)=A03(3,4);
end

% we plot the angular postion from simulink
figure
plot(out.tout,out.pos,out.tout,out.pos1, out.tout,out.pos2)
legend({'$q_{1}$','$q_{2}$','$q_{3}$'},'Location', 'best','Interpreter', ...
    'latex','FontSize', 16)
title('3 reaction wheels (robotics)')
xlabel('time (s)')
ylabel('Pos(rad)')
grid on
k=gca;
set(k,'PlotBoxAspectRatio',[1,1/2,1]);
export_fig k -pdf -transparent robot3dof_position.pdf

% we plot the angular velocity from simulink
figure
plot(out.tout,out.rads,out.tout,out.rads1,out.tout,out.rads2)
legend({'$\dot{q_{1}}$','$\dot{q_{2}}$','$\dot{q_{3}}$'},'Location', ...
    'best','Interpreter', 'latex','FontSize', 16)
title('3 reaction wheels (robotics)')
xlabel('time (s)')
ylabel('Vel(rad/s)')
grid on
l=gca;
set(l,'PlotBoxAspectRatio',[1,1/2,1]);
export_fig l -pdf -transparent robot3dof_velocity.pdf

% we plot the workspace estimated with the homogenous matrix
figure
plot3(x2,y2,z2,'-g','LineWidth',2)
title('3 reaction wheels workspace (robotics)')
xlabel('x (cm)')
ylabel('y(cm)')
zlabel('z(cm)')
%We plot a sphere surface
[X Y Z] = sphere(30);
X2 = X * d3;
Y2 = Y * d3;
Z2 = Z * d3+5;
hold on
contour3(X2,Y2,Z2,8,'-.')
hold on
```

```
59 h = surfl(X2, Y2, Z2);
60 set(h, 'FaceAlpha', 0.1)
61 shading interp
62 grid on
63 j=gca;
64 set(j,'PlotBoxAspectRatio',[1,1,1]);
65 export_fig j -pdf  robot3dof_workspace.pdf
```

E.3 Simulink Simulations robotics

1dof Simulink block diagram robotics (*robotics_simulink_simulation_1dof.slx*)

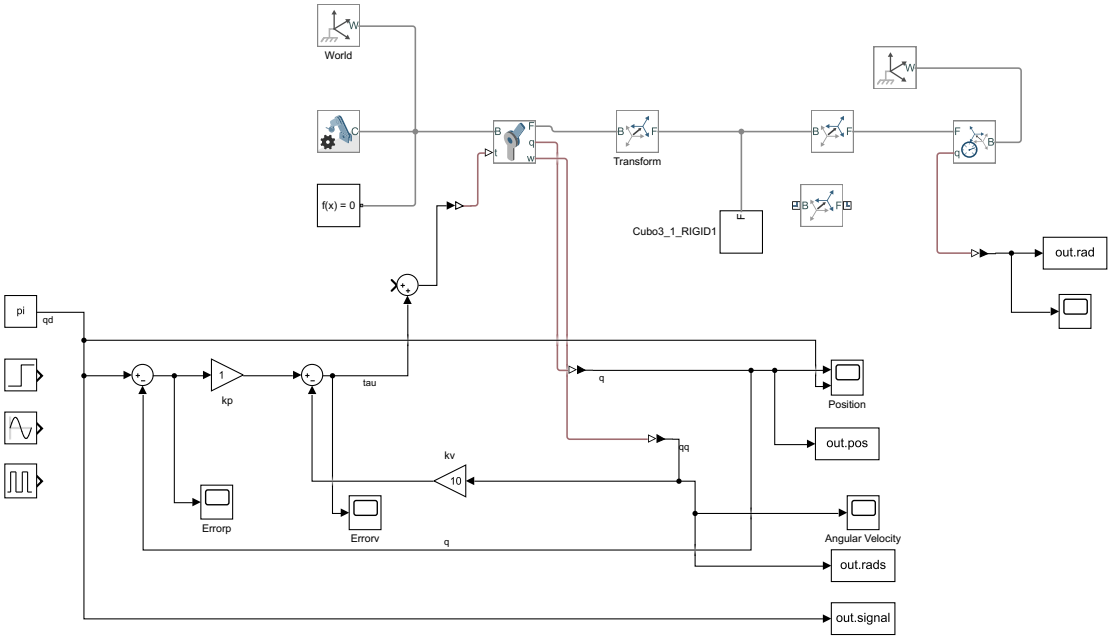


Figure E.4: Block diagram simulation of 1-dof CubeSat (robotics).

1dof Simulink simulation code robotics (*robotics_simulink_simulation_1dof.m*)

```
% code for robotics 1-dof
% we define the variables taken from Simulink
q1=out.pos;
r=[0;5;0]; %BodyFrame vector (cm)
%We calculate de rotation matrix and the spacework
for i=1:50035
EA=  [ cos(q1(i)), sin(q1(i)),   0;
       -sin(q1(i)), cos(q1(i)), 0;
             0,         0   ,   1]; %Z
rhop=EA*r;
x2(i)=rhop(1);
y2(i)=rhop(2);
z2(i)=rhop(3);

end
%we plot the position and velocity
figure
plot(out.tout,out.pos, out.tout,out.rads)%,out.tout,out.signal)
yline(out.signal)
legend({'$q_{1}$', '$\dot{q_{1}}$'},'Location', 'best','Interpreter', ...
    'latex','FontSize', 16)
title('1 reaction wheel (robotics)')
xlabel('time (s)')
ylabel('Pos(rad), Ang vel(rad/s) & Signal')
grid on
i=gca;
set(i,'PlotBoxAspectRatio',[1,1/2,1])
export_fig i -pdf -transparent robot1dof_workspace2.pdf %we export the figure

%we plot the workspace
figure
plot3(x2,y2,z2,'-g','LineWidth',2)
%legend({'$q$'},'Interpreter', 'latex','FontSize', 16)
title('1 reaction wheel workspace (robotics)')
xlabel('x (cm)')
ylabel('y (cm)')
zlabel('z (cm)')

%We plot a sphere surface
hold on
[X Y Z] = sphere(30);
X2 = X * r(2);
Y2 = Y * r(2);
Z2 = Z * r(2)+5;
contour3(X2,Y2,Z2,8,'-.')
hold on
h = surfl(X2, Y2, Z2);
set(h, 'FaceAlpha', 0.1)
shading interp
grid on
grid on
set(gca,'PlotBoxAspectRatio',[1,1,1])
export_fig gca -pdf robot1dof_position_velocity.pdf %we export the figure ...
    -transparent

figure
h=plot(x2,y2,'-g','LineWidth',2)
title('1 reaction wheel workspace (robotics)')
xlabel('x (cm)')
ylabel('y (cm)')
grid on
h=gca;
set(h,'PlotBoxAspectRatio',[1,1/2,1]) %scale the axis
export_fig h -pdf -transparent robot1dof_workspace.pdf %figure name
```

2dof Simulink block diagram robotics (*robotics_simulink_simulation_2dof.slx*)

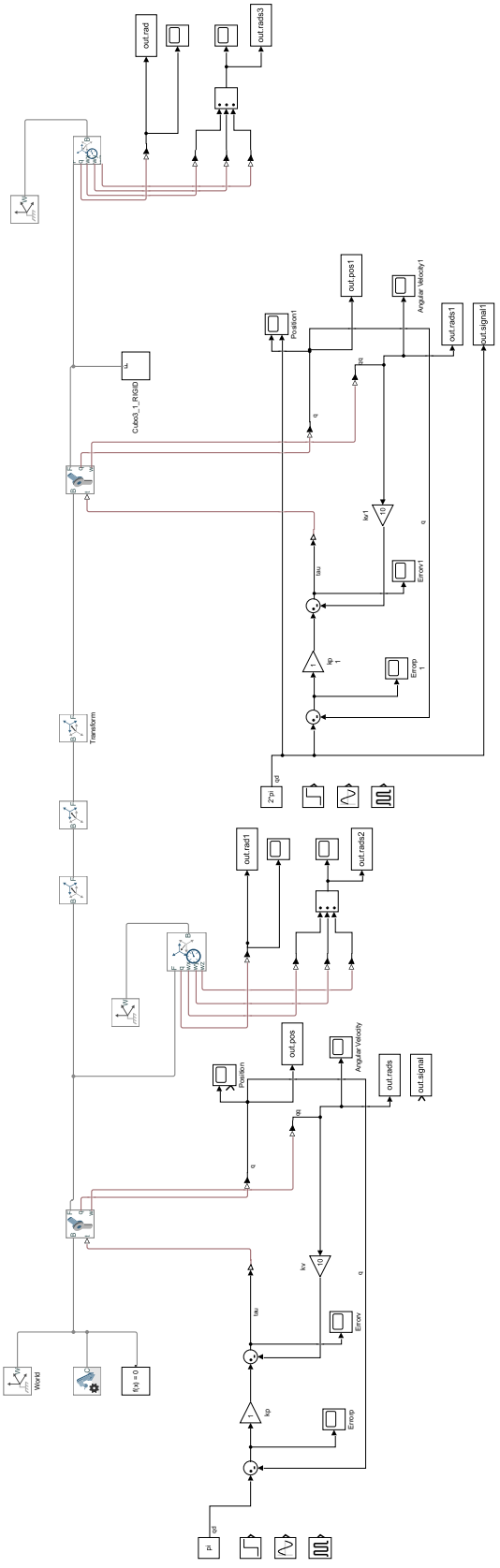


Figure E.5: Block diagram simulation of 2-dof CubeSat (robotics).

2dof Simulink simulation code robotics (*robotics_simulink_simulation_2dof.m*)

```
% code for robotics 2-dof
% we define the variables taken from Simulink
q1=out.pos;
q3=out.pos1;
d1=5;
d2=0;
d3=5;
KK0=q1*0;

% we begin a cicle where the homogenous matrix will be estimated
for i=1:163531

    A03=[ -sin(q1(i)).*sin(q3(i)),  - cos(q3(i)).*sin(q1(i)) , 0, ...
        d3.*cos(q1(i)).*sin(q3(i)) - d2.*sin(q1(i));
       cos(q1(i)).*sin(q3(i)), cos(q1(i)).*cos(q3(i)), sin(q1(i)), d2.*cos(q1(i)) ...
           + d3.*sin(q1(i)).*sin(q3(i));
       cos(q3(i)), sin(q3(i)), 0,                d1 + d3.*cos(q3(i));
       0,               0,       0,                        1];

x2(i)=A03(1,4);
y2(i)=A03(2,4);
z2(i)=A03(3,4);
end

% we plot the angular postion from simulink
figure
plot(out.tout,out.pos,out.tout,out.pos1)
legend({'$q_{1}$','$q_{2}$'},'Location', 'best','Interpreter', ...
    'latex','FontSize', 16)
title('2 reaction wheels (robotics)')
xlabel('time (s)')
ylabel('Ang pos(rad)')
grid on
k=gca;
set(k,'PlotBoxAspectRatio',[1,1/2,1]);
export_fig k -pdf -transparent robot2dof_position.pdf

% we plot the angular velocity from simulink
figure
plot(out.tout,out.rads,out.tout,out.rads1)
legend({'$\dot{q_{1}}$','$\dot{q_{2}}$'},'Location', 'best','Interpreter', ...
    'latex','FontSize', 16)
title('2 reaction wheels (robotics)')
xlabel('time (s)')
ylabel('Ang vel(rad/s)')
grid on
l=gca;
set(l,'PlotBoxAspectRatio',[1,1/2,1]);
export_fig l -pdf -transparent robot2dof_velocity.pdf

% we plot the workspace estimated with the homogenous matrix
figure
plot3(x2,y2,z2,'-g','LineWidth',2)
title('2 reaction wheels workspace (robotics)')
xlabel('x(cm)')
ylabel('y(cm)')
zlabel('z(cm)')

%We plot a sphere surface
[X Y Z] = sphere(30);
X2 = X * d3;
Y2 = Y * d3;
Z2 = Z * d3+5;
hold on
contour3(X2,Y2,Z2,8,'-.')
hold on
h = surfl(X2, Y2, Z2);
```

```
64 set(h, 'FaceAlpha', 0.1)
65 shading interp
66 grid on
67 j=gca;
68 set(j,'PlotBoxAspectRatio',[1,1,1]);
69 export_fig j -pdf  robot2dof_workspace.pdf
```

3dof Simulink block diagram robotics (*robotics_simulink_simulation_3dof.slx*)

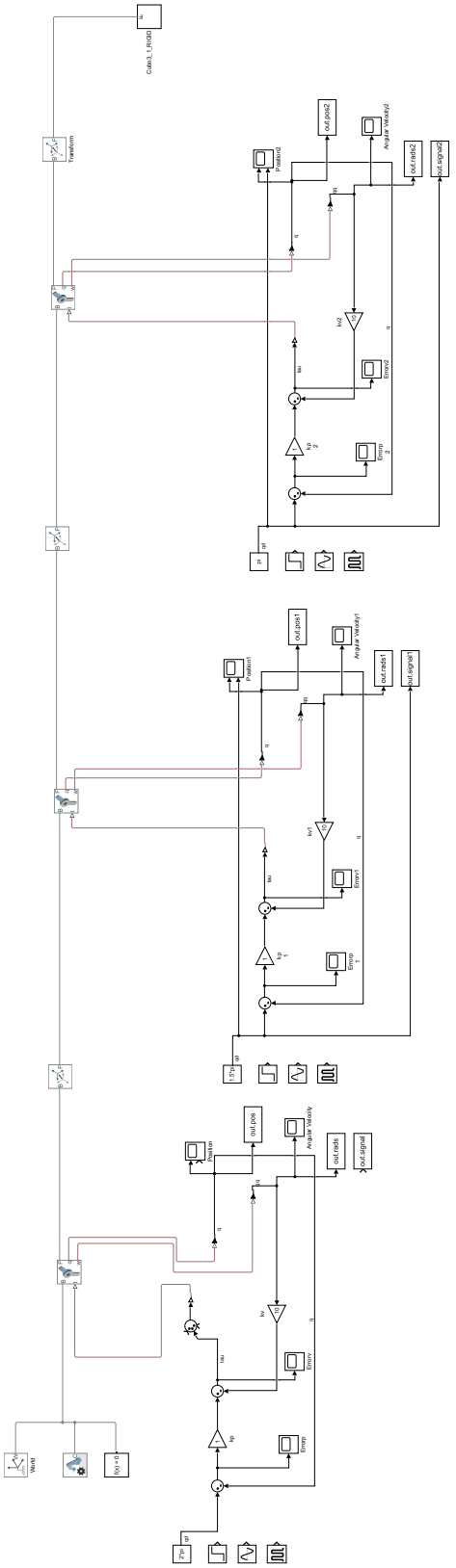


Figure E.6: Block diagram simulation of 3-dof CubeSat (robotics).

3dof Simulink simulation code robotics (*robotics_simulink_simulation_3dof.m*)

```
% we define the variables taken from Simulink
q1=out.pos;
q2=out.pos1;
q3=out.pos2;
d1=5;
d2=0;
d3=5;
KK0=q1*0;


% we begin a cicle where the homogenous matrix will be estimated
for i=1:3076


   A03= [ sin(q1(i)).*sin(q3(i)) + cos(q1(i)).*cos(q2(i)).*cos(q3(i)),   ...
       cos(q3(i)).*sin(q1(i)) - cos(q1(i)).*cos(q2(i)).*sin(q3(i)), ...
       cos(q1(i)).*sin(q2(i)), d3.*cos(q1(i)).*sin(q2(i))-d2.*sin(q1(i));
          cos(q2(i)).*cos(q3(i)).*sin(q1(i)) - cos(q1(i)).*sin(q3(i)), - ...
              cos(q1(i)).*cos(q3(i)) - cos(q2(i)).*sin(q1(i)).*sin(q3(i)), ...
              sin(q1(i)).*sin(q2(i)), d2.*cos(q1(i)) + d3.*sin(q1(i)).*sin(q2(i));
                        -cos(q3(i)).*sin(q2(i)),                                ...
                            sin(q2(i)).*sin(q3(i)),         cos(q2(i)), ...
                                                  d1 + d3.*cos(q2(i));
                          0,       0,           0,           1];

x2(i)=A03(1,4);
y2(i)=A03(2,4);
z2(i)=A03(3,4);
end

% we plot the angular postion from simulink
figure
plot(out.tout,out.pos,out.tout,out.pos1, out.tout,out.pos2)
legend({'$q_{1}$','$q_{2}$','$q_{3}$'},'Location', 'best','Interpreter', ...
    'latex','FontSize', 16)
title('3 reaction wheels (robotics)')
xlabel('time (s)')
ylabel('Pos(rad)')
grid on
k=gca;
set(k,'PlotBoxAspectRatio',[1,1/2,1]);
export_fig k -pdf -transparent robot3dof_position.pdf

% we plot the angular velocity from simulink
figure
plot(out.tout,out.rads,out.tout,out.rads1,out.tout,out.rads2)
legend({'$\dot{q_{1}}$','$\dot{q_{2}}$','$\dot{q_{3}}$'},'Location', ...
    'best','Interpreter', 'latex','FontSize', 16)
title('3 reaction wheels (robotics)')
xlabel('time (s)')
ylabel('Vel(rad/s)')
grid on
l=gca;
set(l,'PlotBoxAspectRatio',[1,1/2,1]);
export_fig l -pdf -transparent robot3dof_velocity.pdf

% we plot the workspace estimated with the homogenous matrix
figure
plot3(x2,y2,z2,'-g','LineWidth',2)
title('3 reaction wheels workspace (robotics)')
xlabel('x (cm)')
ylabel('y(cm)')
zlabel('z(cm)')
%We plot a sphere surface
[X Y Z] = sphere(30);
X2 = X * d3;
Y2 = Y * d3;
```

```
60 Z2 = Z * d3+5;
61 hold on
62 contour3(X2,Y2,Z2,8,'-.')
63 hold on
64 h = surfl(X2, Y2, Z2);
65 set(h, 'FaceAlpha', 0.1)
66 shading interp
67 grid on
68 j=gca;
69 set(j,'PlotBoxAspectRatio',[1,1,1]);
70 export_fig j -pdf  robot3dof_workspace.pdf
```

Bibliography

Agrawal, B. N. and Rasmussen, R. E. (2001). Air-bearing-based satellite attitude dynamics simulator for control software research and development. In Jr., R. L. M., editor, *Technologies for Synthetic Environments: Hardware-in-the-Loop Testing VI*, volume 4366, pages 204 – 214. International Society for Optics and Photonics, SPIE.

Bahu, A. and Modenini, D. (2020). Automatic mass balancing system for a dynamic CubeSat attitude simulator: development and experimental validation. *CEAS Space Journal*, 12(4):597–611.

Beebi, L. (2016). Robust reaction wheel attitude control of satellites. *International Journal of Scientific & Engineering Research*, 7(4):5099–608.

Borjas, J. (2014). Texto: Modelado matemático de sistemas físicos. Technical report, Universidad Nacional de Callao.

Bosselar, M. (2020). Frequency domain system identification and control of a cubesat experimental setup. Technical report, Eindhoven University of Technology.

Bouwmeester, J. and Guo, J. (2010). Survey of worldwide pico- and nanosatellite missions, distributions and subsystem technology. *Acta Astronautica*, 67(7-8):854–862.

CalPoly (2015). CubeSat Design Specification Rev. 13. Technical report, California Polytechnic State University.

Casaday, W. (1966). *Reaction Wheel with Brushless DC Motor Drive*. NASA contractor report. National Aeronautics and Space Administration.

Census (2021). The cubesat phenomenon. Taken: 2022-01-30 from: `https://census.psl.eu/spip.php?rubrique21&lang=en`.

Chesi, S. and Romano, M. (2020). Angularly unbounded three-axis spacecraft simulator. US Patent 10,621,883.

Cline, D. (2020). *Variational Principles in Classical Mechanics*. LibreTexts.

da Silva, R. C., Guimarães, F. C., de Loiola, J. V. L., Borges, R. A., Battistini, S., and Cappelletti, C. (2019). Tabletop testbed for attitude determination and control of nanosatellites. *Journal of Aerospace Engineering*, 32(1):04018122.

de la Cerda, J. (2019). Painani-i: primer nanosatélite mexicano puesto en el espacio. Taken: 2019-10-10 from: `http://todos.cicese.mx/sitio/noticia.php?n=1368#.YgBucOrMLIU`.

Denavit, J; Hartenberg, R. (1955). A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*.

Escobedo Lugo, L. (2012). Simulador para pruebas de control de orientación para nanosatélites. Master's thesis, Facultad de Ingeniería. Universidad Nacional Autónoma de México.

Familton, J. C. (2015). *Quaternions: a history of complex noncommutative rotation groups in theoretical physics*. PhD thesis, Columbia University.

Flores Gómez, S. I. (2012). Simulación y pruebas de control de estabilización para micro-satélites. Master's thesis, Universidad Nacional Autónoma de México.

Fortier, P. and Michel, H. (2003). *Computer Systems Performance Evaluation and Prediction*. Digital Press.

Gavrilovich, I. (2016). *Development of a robotic system for CubeSat Attitude Determination and Control System ground tests*. phdthesis, University of Montpellier.

Goldstein, H., Poole, C., and Safko, J. (2002). *Classical Mechanics*. Addison Wesley.

Gutiérrez Medina, A. (2015). Desarrollo de un entrenador experimental con dos ruedas de reacción para un cubesat educativo. Master's thesis, Centro de Investigación Científica y de Educación Superior de Ensenada.

Haeussermann, W. and Kennel, H. (1960). A satellite motion simulator. *Astronautics*, 5.

Halliday, D., Resnick, R., and Walker, J. (2018). *Fundamentals of Physics*. 11th edition.

IPN (2021). Ipn y unam realizan con éxito lanzamiento del nanosatélite en cabo cañaveral. Taken: 2021-02-21 from: www.ipn.mx/imageninstitucional/comunicados/ver-comunicado.html?y=2021&n=118.

Jain, P. and Nkoma, J. (2019). *Introduction to Classical Mechanics: Kinematics, Newtonian and Lagrangian*. Mkuki Na Nyota Publishers.

JLTZ (2020). Estudiantes mexicanos participarán en misión de space jltz. Taken: 2021-07-11 from:http://jltz.space/2020/11/04/formaran-estudiantes-de-mexico-con-mision-satelital-internacionald2-atlacom-1/.

Kelly, R. ; Santibañez V.;; Loaría, A. (2006). *Control of robot manipulators in joint space*. Springer.

Kiesbye, J., Messmann, D., Preisinger, M., Reina, G., Nagy, D., Schummer, F., Mostad, M., Kale, T., and Langer, M. (2019). Hardware-in-the-loop and software-in-the-loop testing of the move-ii cubesat. *Aerospace*, 6(12).

Kluever, C. (2018). *Space Flight Dynamics*. Aerospace Series. Wiley.

Larson, W. J. and Wertz, J. R. (1992). Space mission analysis and design. Technical report, Torrance, CA (United States); Microcosm, Inc.

Ljung, L. (2000). Model error modeling and control design. *IFAC Proceedings Volumes*, 33(15):31–36. 12th IFAC Symposium on System Identification (SYSID 2000), Santa Barbara, CA, USA, 21-23 June 2000.

Marion, J. (1965). Classical dynamics of particles and systems. *Science*, 150(3699):1018–1019.

MATLAB (2021). ode45. Taken: 2022-01-30 from: www.mathworks.com/help/matlab/ref/ode45.html.

Meissner, D. M. (2009). A three degrees of freedom test-bed for nanosatellite and cubesat attitude dynamics, determination, and control. Master's thesis, Naval Posgraduate School.

Mittelsteadt, C. O. and Mehiel, E. A. (2007). The cal poly spacecraft attitude dynamics simulator - cp/sads. In *AIAA Guidance, Navigation and Control Conference and Exhibit*.

Modenini, D., Bahu, A., Curzi, G., and Togni, A. (2020). A dynamic testbed for nanosatellites attitude verification. *Aerospace*, 7(3).

NanoSats (2021). Nanosats data base. Taken: 2021-08-20 from: www.nanosats.eu/.

Pelton, J. (2020). *Handbook of Small Satellites Technology, Design, Manufacture, Applications, Economics and Regulation: Technology, Design, Manufacture, Applications, Economics and Regulation*. Spirnger International Publishing.

Prado, J., Bisiacchi, G., Reyes, L., Vicente, E., Contreras, F., and Mesinas, M. (2005). Three-axis air-bearing based platform for small satellite attitude determination and control simulation. *Journal of Applied Research and Technology*, 3.

Rimrott, F. (1989). *Introductory Attitude Dynamics*. Mechanical Engineering Series. Springer New York.

Schwartz, J., Peck, M., and Hall, C. (2003). Historical review of air-bearing spacecraft simulators. *Journal of Guidance, Control, and Dynamics*, 26.

Selva, D. and Krejci, D. (2012). A survey and assessment of the capabilities of Cubesats for Earth observation. *Acta Astronautica*, 74:50–68.

Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2008). *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 1st edition.

Spong, M. W., Hutchinson, S., Vidyasagar, M., et al. (2006). *Robot modeling and control*. John Wiley Sons, INC, 1st edtion edition.

Thomson, W. (1986). *Introduction to Space Dynamics*. Dover Books on Aeronautical Engineering. Dover Publications.

Tibbs, M. L. (2015). Design and test of an attitude determination and control system for a 6u cubesat using afit's cubesat testbed. Master's thesis, Air Force Institute of Technology.

Toorian, A., Diaz, K., and Lee, S. (2008). The cubesat approach to space access. In *2008 IEEE Aerospace Conference*, pages 1–14.

Trégouët, J.-F., Arzelier, D., Peaucelle, D., Pittet, C., and Zaccarian, L. (2015). Reaction wheels desaturation using magnetorquers and static input allocation. *IEEE Transactions on Control Systems Technology*, 23(2):525–539.

UNAM (2021). Refrenda la unam su potencial científico con lanzamiento de nanosatélite al espacio. Taken: 2021-03-11 from:www.dgcs.unam.mx/boletin/bdboletin/2021_176.html.

UPAEP (2019). Aztechsat-1. Taken: 2019-08-11 from: https://upaep.mx/aztechsat.

Wertz, J. R. (1978). *Spacraft Attitude Determination and Control*. Kluwer Academic Publishers.

Woo, H., Rico, O., Chesi, S., and Romano, M. (2011). Cubesat three axis simulator(cubetas). In *AIAA Modeling and Simulation Technologies Conference*.

Yost, B., A., E., Roland, B., Roberto, C., G. D. P. A. G. A. K. B. R. A. S. J. S. R. T. J., and S., W. (2020). Nasa state of the art small spacecraft technology. Technical report, Small Spacecraft Systems Virtual Institute Ames Research Center.

Zavala, S. (2019). Modelado matemático para la ciencia e ingeniería aeroespacial. Technical report, Tecnológico Nacional de México.