
Politecnico di Torino

Master of Science in Mechatronics Engineering

Master's Degree Thesis

Design of an attitude control system for
microsatellites and for LICIACube mission



Supervisor:

Prof. Elisa Capello

Co-supervisor:

Dr. Alessandro Balossino

Student:

Pasquale Centore

ID. 268081

Academic Year
2020/2021

Abstract

CubeSats, as a compact and miniaturized version of traditional Satellites, are nowadays a trend thanks to the numerous goals reached by the space agencies missions. Among the various companies involved in the space field, Argotec is participating to NASA Double Asteroid Redirection Test (DART) mission with its CubeSat Light Italian CubeSat for Imaging of Asteroids (LICIACube). Since the goal of the mission is to acquire many pictures during a high-speed fly-by of an asteroid, the Attitude Determination Control System (ADCS) is a critical part of LICIACube.

The design of an attitude control system for small satellites is a challenging task, especially when high velocity - high precision is asked and to include the adaptability to different satellites. Different approaches are possible, each with its own pros/cons, strictly depending on situations.

The work done in this thesis explores the possibilities of realizing an ADCS for small satellites as modular and flexible as possible, using LICIACube as a reference model. The goal is to implement and test such solution on the next platform generation.

To successfully achieve mission requirements, strict performances for the control system are to be considered; moreover, the spacecraft shall have different operating modes, all of them using Reaction Wheels as main actuators and thrusters as support.

After considering the worst possible set of initial conditions to work with, the model of LICIACube has been developed on Simulink. Different operative modes are considered. Firstly, Fully Operational Mode, where LICIACube works in nominal conditions, is shown. A nonlinear robust controller based on the Sliding Mode theory is applied for attitude tracking problem. The simulated sensors noise is handled by an Extended Kalman Filter. The second operating mode is the Detumbling Mode. A set of Proportional Integrative Derivative (PID) controllers is used to reduce the microsatellite residual angular velocity caused by the detachment from DART.

The last operating mode is the Desaturation Mode. The Reaction Wheels are discharged of their accumulated momentum by directly sending commands to the actuators through a custom logic. The resulting high angular velocity of the satellite is then reduced with thrusters, whose modelling involves the implementation of a switching management logic for fuel consumption minimization.

To simulate the correct behavior of the ADCS, a finite-state structure on Stateflow makes possible to let all control systems communicate, while guaranteeing a certain level of abstraction and a good degree of flexibility, allowing the customization of all control parameters at will. The structure allows the ADCS, through the analysis of the sensors data, to switch independently in the correct operating mode.

The results show that the control systems are compatible, and the performances are in line with the mission requirements.

In conclusion, although DART mission has strict mission requirements, the developed ADCS seems adequate for the task; moreover, the Stateflow structure provides an abstraction layer that makes viable the concept of adaptability for similar kind of CubeSats

Contents

Introduction	9
Thesis environment.....	Errore. Il segnalibro non è definito.
Thesis focus	Errore. Il segnalibro non è definito.
Thesis outline.....	22
Control techniques description.....	23
1.1 Control systems.....	23
1.2 PID control.....	25
1.3 Sliding Mode Control.....	29
Mathematical Model.....	35
2.1 Dynamics.....	35
2.2 Kinematics.....	37
2.3 Actuation Systems.....	37
2.4 Fully Operational Mode Controller.....	40
2.5 Detumbling Mode Controller.....	43
2.6 Filtering.....	45
Matlab and Simulink simulations.....	49
3.1 Simulink solver.....	49
3.2 Dynamics, kinematics and main actuator.....	50
3.3 Fully operational mode.....	51
3.3.1 Reference Generation block.....	51
3.3.2 Extended Kalman Filter.....	53
3.3.3 Simulation parameters 1.....	53
3.3.4 Simulation results 1.....	55
3.4 Detumbling mode.....	60
3.4.1 Simulation parameters 2.....	60
3.4.2 Simulation results 2.....	62

Desaturation mode	66
4.1 Discharge step	69
4.2 Simulation Parameters 3	70
4.3 Simulation results 3	71
4.4 Detumble step	74
4.5 Simulation parameters 4	77
4.6 Simulation results 4	78
Structural testing.....	81
5.1 Structure implementation and user-friendly interface.....	84
5.2 Final Simulation Parameters.....	86
5.3 Final Simulation.....	87
Conclusions.....	90
Bibliography.....	93

Introduction

The concept of space exploration, born in the 1950s during the space race between United States and Soviet Union, aroused an ever-growing interest during the years. The possibility to reach the untouchable, to extend beyond the human limits has been the moving factor that raised the interest of people towards scientific careers and studies, leading to new discoveries on the Solar System and the Universe in general. Governments and private parties encourage still today the investment in space projects, for many motivations. Space exploration is a reflection of societal and cultural aspects. Technologies made up for space missions have often been transformed to suit life on Earth, improving many living aspects. An example may be the Global Positioning System (GPS), born as a military project of the U.S. Department of Defense and now accessible to almost all electronic smart devices, or the Memory foam, a material made up of polyurethane and other chemicals developed by NASA to improve the safety of the aircraft cushions, now used in mattresses.

In recent years NASA has warmly encouraged the employment of autonomous systems, marking Robotics and Autonomous Control as high-priority fields of study to further improve the human influence and knowledge on the Solar System. This line of thought has been welcomed with great emphasis by all the world space agencies, that began to adapt old technologies and develop new ones, all involving autonomous systems and navigation, making the presence of a human in orbit often unnecessary. Satellite and microsatellite development have been a natural course of action, given the premises. Nowadays there has been great advance in the space sector thanks to the great number of microsatellites launched. This trend helped not only to boost the process of space exploration, but also the development of new technologies. New spacecraft architectures have been studied and implemented successfully on many different types of CubeSats.[15]

CubeSats are a normalized class of nanosatellites at first imagined as instructive apparatuses or innovation demonstrators, cheap, versatile and fast to realize.

These qualities led to their massive employment for space projects. Interplanetary missions may profit by their adaptability, low cost - high rewards and modular architecture to obtain detailed scientific data.

They are also known as “U-class spacecraft” since they are made up by cubic modules (10x10x10 cm), indicated with U, that represent the basic modular units. Each module can

Introduction

have a maximum mass of 1.3 kg and can host all the functional elements needed to the task to accomplish, as scientific instruments or more general payloads. The most common builds are 6U or 12U CubeSats, but it is possible to find many other combinations.

Modules determine indirectly also CubeSats classifications, since they are divided by weight:

- Minisatellite (100-500kg)
- Microsatellite (10-100kg)
- Nanosatellite (1-10kg)
- Picosatellite (0.1-1kg)

CubeSats follow a standard defined by the California Polytechnic State University (Cal Poly). The standard provides guidelines on necessary modules, possible protrusions, secondary payloads, dimensions. Indications are also given for the electronics and the structure, making possible to use commercial-off-the-shelf (COTS) components: attention must be placed on the radiation toleration, that is critical for deep space applications, sublimation and outgassing effects that can make the mission fail.

CubeSats are employed to accomplish many tasks, with focus on projects or tests with a below average success rate that cannot justify the cost of a large satellite. The use of these spacecrafts grants numerous advantages, many of which derives from the small dimensions.

One of the most important advantage is the reduction of the deployment cost. By following the standard designed by Cal Poly, all CubeSats have fixed forms and characteristics, making them suitable to be launched in multiples, or to fit in the remaining space of an almost full launch vehicle. Moreover, the smaller, the lighter.

Smaller dimensions often mean also lower risks during the launch phases: the CubeSats design specially minimizes the risk for the launch vehicles and the other payloads.

Lastly, while normally a satellite is encapsulated with its launcher, resulting in a good amount of work for the proper interface between the two, the interface between CubeSats and shuttle is encapsulated and is standard. It saves time and money while also allowing the exchanges of payloads on shuttles with short notice.

Today, CubeSats are sent in orbit using a common deployment system called Poly-Picosatellite Orbital Deployer as secondary payloads, with a successful deployment of more than 1200 of them.

Being dimensions the only difference with their cousins, CubeSats are internally organized as real satellites. They have to handle many different tasks at the same time, as power management and primary control task. All these situations can't be handled by a single main computer for many reasons: temperature, memory, fault prevention.

Introduction

There are generally numerous small computers in every spacecraft, each of which deals with a single task and communicates with the main one through specific protocol communications. Moreover, the necessity for payloads to communicate with the main computer leads to the implementation of other small computing units that deal with interfacing the two parts; even if this may be extremely redundant, it can be necessary also to prevent the overloading of the primary computer or to guarantee the operational continuity of other, higher priority, needs. It is a priority-based communication protocol, where the main computer, when in standby situation, may be lent to payloads for related tasks.

The attitude and determination control double task requires a specific small computer too. It is present and necessary in nearly every spacecraft, to avoid solar damage to components, to charge batteries through solar panels or simply for orbital maneuvers.

- Attitude determination is “the process of computing the orientation of the spacecraft relative to either an inertial reference or some object of interest, such as the Earth”. [9]
- Attitude control is “the process of orienting the spacecraft in a specified predetermined direction”. It is divided in two areas, where attitude stabilization deals with maintaining the orientation, while attitude maneuver allows to control the reorientation of the spacecraft. [9]

The Attitude Determination and Control System (ADCS) is a crucial subsystem of the spacecraft that handles this double task. Its goal is to provide pointing accuracy and stability of the entire body, or just of its antennas, payloads and every critical part of the operations. Hardware-like, Attitude Control technology for CubeSats is essentially the miniaturized version of the standard satellites, without loss of performance. Moreover, it is the most crucial aspect of a spacecraft to achieve autonomous control and navigation, without the help of human touch, as warmly recommended by NASA.

For Attitude Determination, a series of sensors may be equipped:

- An Inertial Measurement Unit (IMU) is the backbone for almost every navigation and control application. It is an electronic device typically used for aircraft and

Introduction

spacecraft maneuvers that works by detecting linear accelerations through accelerometers and angular velocity through gyroscopes, generally one per axes.

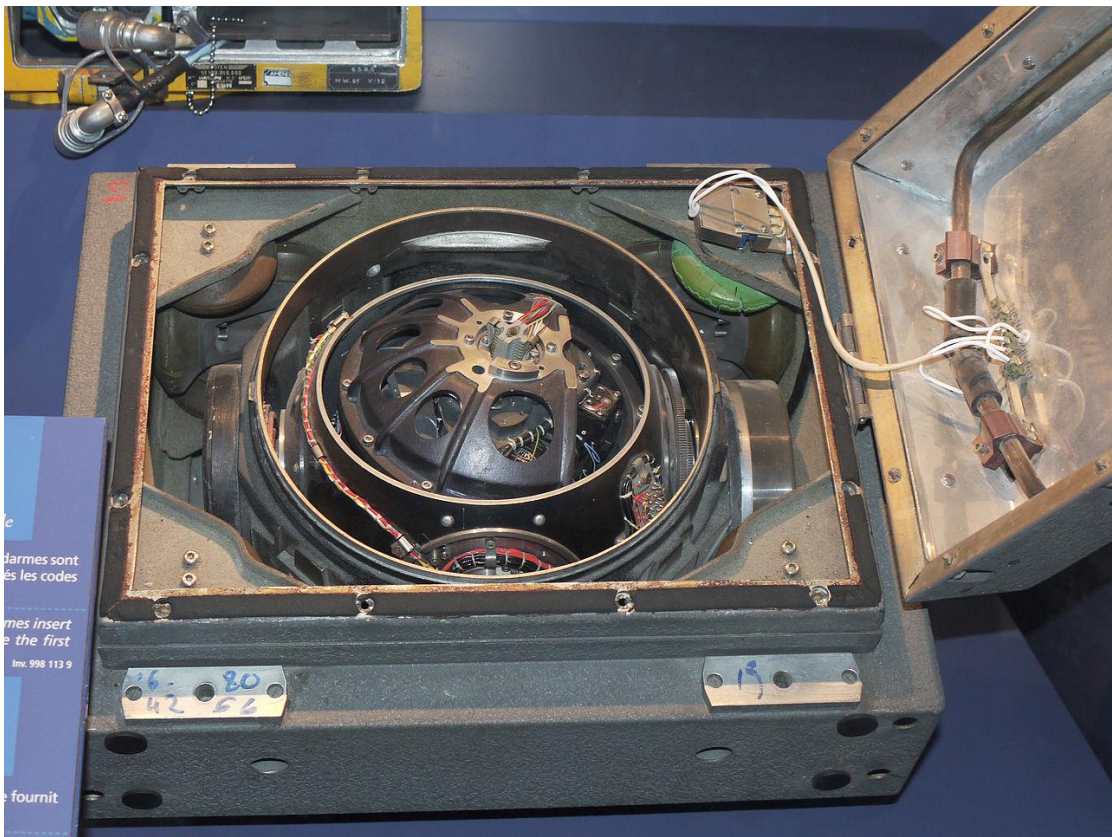


Figure 1: Inertial Measurement Unit (IMU)

- For Low Earth Orbit (LEO) CubeSats, the IMU is often accompanied by a series of Magnetometers, in order to detect and exploit the Earth magnetic field.
- In the end, Star Trackers generally complete the sensor equipment. They are made by an optical device, which takes images of the star in vision, and an electronic device, that compares them to the star catalogue present in memory. In this way it is possible to measure the position of the stars and, by mathematical consideration, the actual attitude of the body.

Coupled with sensors, the actuators are the effective means through attitude control is obtained. Active attitude control requires actuators that can impart a torque to the entire body. Two different classes are available:

- Reaction-type actuators can generate a torque on the spacecraft that can be considered as external. Like Thrusters that exercise momentum through mass expulsion, or Magnetic Torquers that exploit the Earth magnetic field to make the

Introduction

system rotate, they have the capability to change the spacecraft angular momentum.

- Momentum exchange-type can only generate torque internal to the spacecraft. This way, the entire system can rotate, but the overall angular momentum does not change. Examples are Reaction Wheels and Momentum Wheels.[10]

Generally, Momentum exchange-type may be considered as more reliable, not being dependent by magnetic fields or fuel mass consumption. However, they are often accompanied by reaction-types as secondary actuators, to resolve saturation situations. Reaction types instead are powerful and often faster, but they have a limited use, both for environment and effective usage time.

Reaction wheels (RW) are one of the most used Momentum exchange-type actuators. They are composed by two elements: a high inertia spinning wheel and an electric motor. The motor, a brushless DC, controlled by an Onboard Computer (OBC), makes the wheel spin, producing an internal torque. Every Reaction wheel can produce torque only on a single axis, so, for complete maneuvering, 3 of them placed on perpendicular axes are necessary. Being Momentum-exchange devices, they can produce internal momentum, but can't change the total angular momentum on the CubeSat.

However, angular momentum can be redistributed between wheels and satellite, thus changing the orientation.

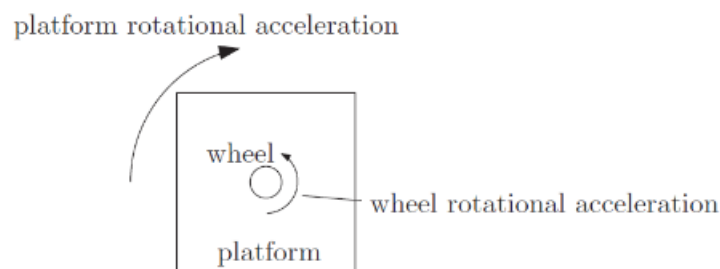


Figure 2: RWs Momentum exchange [10]

Reaction Wheels provide high pointing accuracy and are very useful for small rotations. However, they may be not enough to deal with attitude control in a spacecraft. The presence of disturbances, as solar wind, gravity gradients ecc, may affect the total angular momentum of the system. This external effect will manifest as stored angular momentum in the RWs, thus constantly rotating with a certain speed.

Introduction

Since internal torque is given by accelerating the wheels, if the current angular velocity of the wheel is near the limit, the torque expressed may be not enough since it will be impossible to accelerate further. The Reaction Wheel will be in saturation state.

Saturation can be handled only with the presence of a secondary, reaction-type, actuator that will unload the built momentum with an external torque.

Among all the possible Reaction-type actuators to desaturate Reaction Wheels, thrusters are probably the most diffused choice. They are propulsive devices used for space navigation, station keeping, trajectory and attitude control, being them able to generate both forces and torques.[10]

Their large diffusion is due to their reliability, not being dependent on ambient magnetic or gravitational field. They can be used at will, by turning them on through the OBC commands. However, their usage is not unlimited. The reason why they are used as secondary actuators lies in the expendable propellant they require. This has two side effects: limited functioning time, that limits the possible mission time window, and possible changes in the spacecraft inertia matrix, with consequences in the control system. By limiting their usage as secondary actuators, all these problems are bounded.

Thrusters work by following the conservation energy law. By expelling mass at high velocity, the spacecraft host receives an impulsive acceleration. In space, the absence of friction and other opposing forces allows to maintain the momentum and navigate till the destination without further fuel consumption. Being this the base concept, thrusters exist in many different form and variations, depending on the fuel and technology.

CubeSats generally implement "Cold Gas Thrusters". They are the cheapest, simplest and most reliable propulsion systems available for attitude control. In fact, they have been appointed as the "simplest manifestation of a rocket engine" because their design consists only of a fuel tank, a valve, a propelling nozzle and some pipes. Their usage has been predominant since CubeSats have strict regulations against pyrotechnics and hazardous materials, while Cold Gas Thrusters do not house any combustion and they do not contaminate the environment.

The functioning is simple: the gas, that is stored at high pressure inside the propellant tank, is ejected with a high velocity (kinetic energy) in the direction opposite to the satellite desired direction. Depending on the situation, it is so possible to reduce or increase the kinetic energy of the spacecraft. The nozzle plays an important part in this. It is generally a convergent-divergent nozzle, shaped such that the high pressure, low-velocity gas that enters the nozzle is expanded when it goes towards the throat. The throat

Introduction

is the narrowest part of the thruster, and there the gas velocity arrives near the speed of sound. The gas used is Freon or Hydrazine.

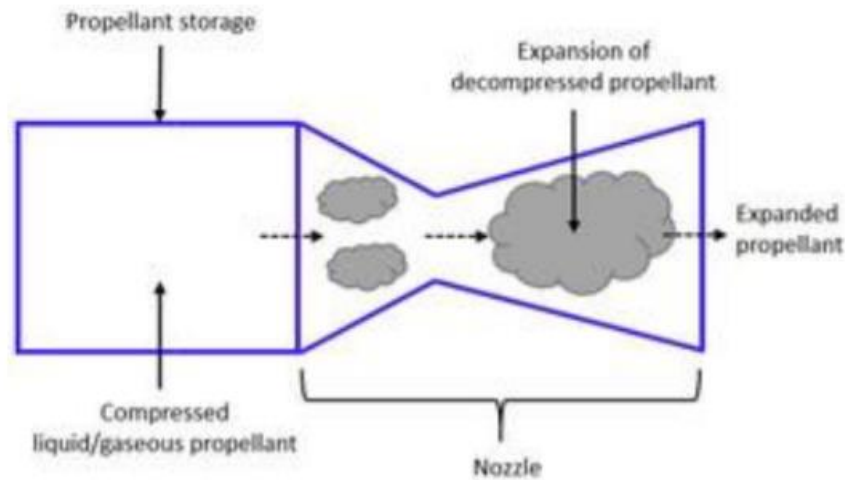


Figure 3: Cold Gas Thruster [10]

In [19] thrusters are presented as the best method to rapidly unload the angular momentum on Reaction Wheels. In the paper, their performance is confronted with the one obtained with magnetorquers. Magnetorquers offer a slow and continuous help in the wheels discharging, but they are limited on Earth orbits applications. Thrusters can instead offer a rapid but imprecise desaturation of the Reaction Wheels, at the cost of fuel consumption, side effect that can reduce the time window of the space mission. By comparing the two performances, thrusters are the ones most used for their versatility. Moreover, their functioning is based on PWM, so they do not need a proper control architecture, differently to magnetorquers that need a continuous variable control law, generated in the paper by an Adaptive controller. In the end of the paper, however, the author shows how the unload of the angular momentum can be done even better when the thrusters and magnetorquers are used in synchro.

The presence of three actuators in a single spacecraft is, however, a very rare situation. Nowadays space missions are becoming more complex. There are always new different and additional tasks that require different satellites structures, as antennas, cameras, manipulators, thrusters etc.

This complexity has led to a more complex modelization of satellites, which often include nonlinear terms.

Introduction

To better model and understand the dynamics, kinematics and overall performances of the ADCS, while also maintaining a certain degree of accuracy and robustness, a nonlinear control is required. Linear control is still a viable, not optimal solution. [4]

During the last years, nonlinear control systems have advanced a lot in two aspects. There have been great advances in theoretical approaches, also thanks to a more mathematical point of view, and in application-driven developments.

Many different control techniques are the fruit of these studies. Some of them gained high popularity due to the ability to suit the case in many different scenarios. The extensive application of theoretical concept at their base made them able to work in most of all real-world problems.

In this thesis work, the main control architecture chosen is based on the Sliding Mode theory. This choice has been influenced by the bibliography studied. In [20] attitude control and docking maneuvers of a spacecraft, equipped with a set of 12 thrusters, are dealt with. The solution proposed involves the implementation of two controllers, both of them exploiting the Sliding Mode theory. For the docking maneuvers, which is based on position tracking of the spacecraft, a first order Sliding Mode controller with a discontinuous control law is implemented. However, having to deal with on/off thrusters, some feasibility problems arise: the controller alone cannot regulate the firing of the rockets, guarantee minimum fuel consumption and minimize the position error in the trajectory assumed. For this reason, it is helped by a switching algorithm that regulates the thrusters activation. The control is split in two phases, depending on the distance from the target, guaranteeing maximum precision in the closest part of the maneuver.

The Attitude control is instead resolved with a second order Sliding-Mode algorithm. The difference with the previous controller is in the continuous control law, which steers to zero in finite time both the sliding output and its first time derivative, but it will be further analyzed in the next chapter. The necessity to implement a second order controller is due to the presence of stronger disturbances respect to the first situation, as the chattering effect.

The simulation analysis show that it is possible to have good performances of the controller even when thrusters are switched at low frequency.

Another Sliding Mode control application is discussed in [3], where attitude and translation motion for a 6 degree of freedom spacecraft is resolved through a second order Sliding Mode control. Many different control architectures are studied at the beginning of the paper. However, the author is of the opinion that, among the controllers that offer a

Introduction

zero steady state tracking error in finite time, Sliding Mode is the best in term of robustness. Other architectures that do not reach zero tracking error in finite time are not considered in the application. In the end, along with simulations that prove how the system responds well when affected by disturbances and inertia uncertainties, the stability of the control technique is proven by studying a candidate Lyapunov function. The study proves how Sliding Mode control is suitable and well performing for spacecraft applications, both in Earth orbit and in deep space.

Background of the thesis

Double Asteroid Redirection Test (DART) is a space mission launched by NASA for the necessity to study a first strategy for planetary defense.

It is the first planetary defense test ever projected. Led by Johns Hopkins Applied Physics Laboratory (APL) and managed under NASA's Solar System Exploration Program at Marshall Space Flight Center, its aim is to test some technologies to redirect the trajectory of every hazardous asteroid possibly impacting the Earth. [2]

The asteroid chosen for this task is Didymos. It is a binary asteroid, with the first body of about 780m diameter and a natural satellite of 160m, called Dimorphos, that revolves around it. It belongs to the Apollo asteroids class.

To accomplish the goal, a kinetic impactor technique will be used: the DART satellite will be accelerated until 6,6 km/s and, then, it will crash on the asteroid, changing its motion by a fraction of 1%. That is a very small change, but enough to be seen and studied with telescopes on Earth, since the orbital period of the objective will be modified by several minutes. DART will have to collide with Dimorphos, since orbit variations will be easier to evaluate.

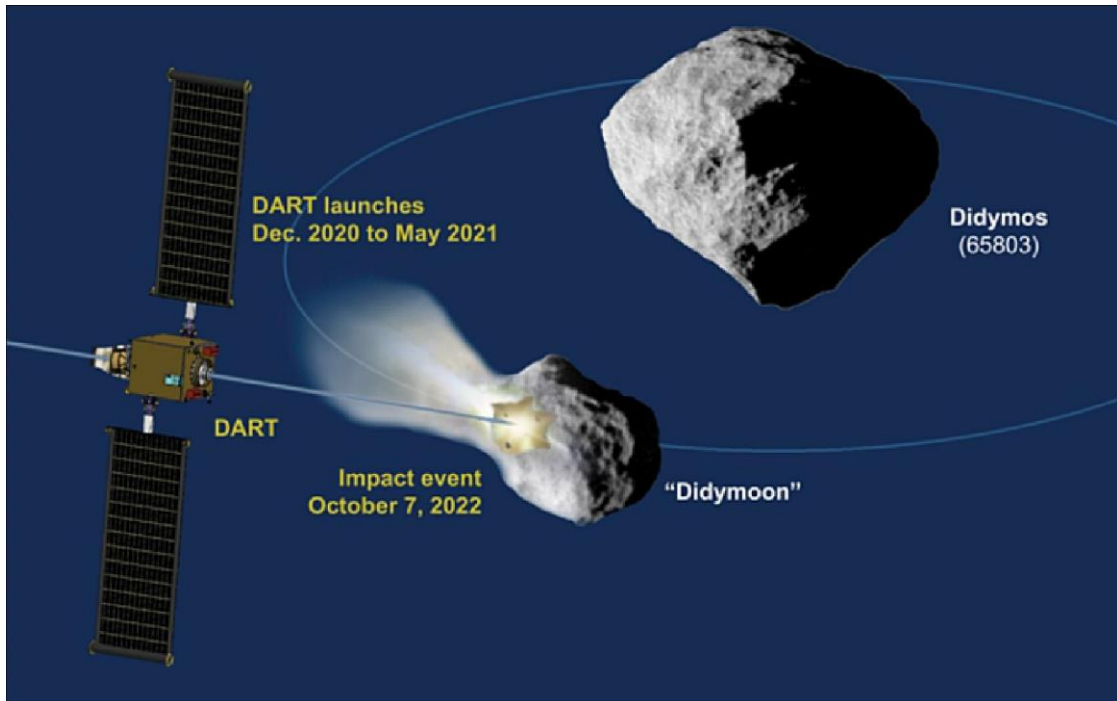


Figure 4: DART mission [2]

The launch window for the mission to begin starts in July 2021. The DART launch will be done aboard a SpaceX Falcon 9 rocket. The spacecraft will then detach from the main satellite and will navigate for over a year, in autonomous navigation, until intercepting Didymos' moonlet in September 2022, at a distance of 11 million kilometers from Earth.

In 2019, one year later the born of the mission, Argotec won the tender issued by NASA as the only Italian company to participate and began the developing of LICIACube.

Argotec is an italian aerospace engineering company, founded in 2008 in Turin by David Avino. At the beginning, the company offered a training service to astronauts, carried out at European Astronaut Center in Colonia. It moved then to the developing of payloads to be sent in orbit, focusing on space food.

The steppingstone for a greater success was the launch of the first coffee machine able to work in orbit, the ISSPRESSO. The machine was sent to the ISS in 2015.

Today, its activities mainly concern the production of microsattellites for deep space and the research and development of innovative solutions for supporting and improving space explorers' work.

The company works under a politic of "all-in-house concept", trying to cover every aspect of microsattellites production on its own, from components development to design and other aspects.

Introduction

LICIACube (Light Italian CubeSat for Imaging of Asteroids), as the objective of the tender won, is one of the most important projects in Argotec. It is a piggyback, deep-space oriented, CubeSat, 6U dimension, weight of approximately 12kg, that will detach from the DART satellite. It is an example of the evolution of nanosatellite technology, that is now ready for deep space applications.

While staying in heliocentric orbit, its primary purpose is not only to document the impact of the spacecraft, but also to take multiple images of the plume that will be ejected after, all from different phase angles and for a certain span of time.



Figure 5: LICIACube [12]

The images will be useful also to have additional information on the asteroid spectral class. The detach from DART will happen 120 hours before the impact. In this time, the CubeSat will travel separately from the spacecraft to the optimal observation point. In the meanwhile, it will be possible to communicate and command the microsatellite from Earth. [13]

After approximately 200 seconds from the release from DART and potential corrective maneuvers, LICIACube will be in an optimal trajectory in order to take data from the impact region. The Closest Approach (C/A) to Didymos B will be at 55km from the target area and the spacecraft will need a high angular velocity, about 7 deg/s, to follow correctly the passing asteroids. This is the first difficult aspect of the mission.[11]

Moreover, it is necessary to consider the Field of View (FoV) of the optical payload (i.e. 4.10°). The impact zone should always be inside the camera objective, so pointing error must be maintained below 2°. This aspect is, probably, the most challenging one.

Introduction

The last critical aspect to consider is the high round trip time of the communications. Satellite and Earth will be so far from each other that communications will need about 30 seconds to be exchanged. This delay is a great difficulty for the fly-by phase due the high velocity of the asteroid: this means that the critical part of the mission must be autonomously performed.

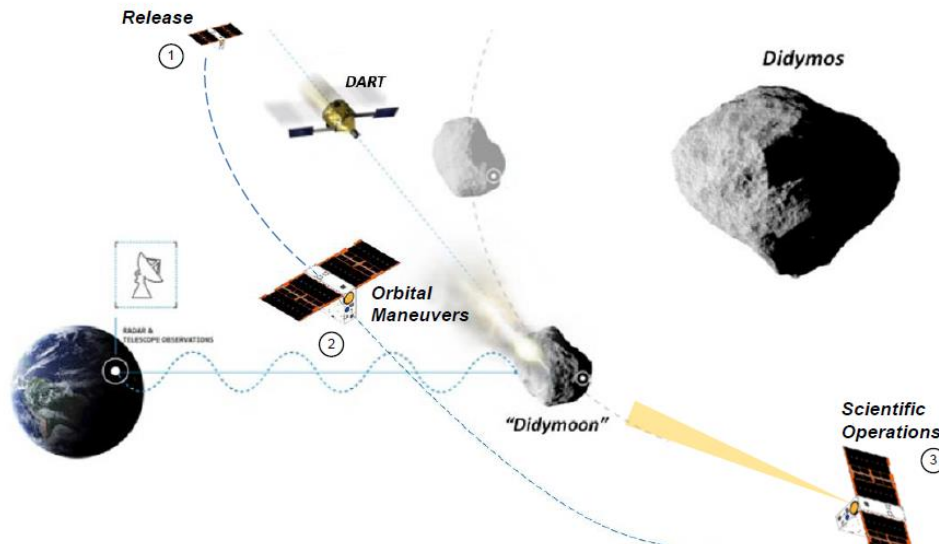


Figure 6: LICIAcube fly-by [12]

For this reason, great importance is given to the Attitude Determination and Control Subsystem (ADCS), that has to provide great maneuverability while guaranteeing high precision.

The navigation system of LICIAcube considers all these conditions. It is composed by three different modules:

1. The Imaging System (IS) processes the images taken from the optical payload and makes decision accordingly.
2. The Trajectory Recognition module generates the trajectory of the CubeSat to effectively follow Didymos B starting from the data of the optical payload, i.e. it generates the reference signals.
3. The Attitude Control Module task computes the control torque vector to be fed to the ADCS for the success of the fly-by. It checks if the torque vector generated is compatible with the trajectory generated in the previous step.

The output of the navigation system is the definition of the desired attitude for the satellite. This means that the reference signals, both for quaternions and angular velocity of the satellite, are generated.

Introduction

These signals are then used as starting point to build a closed-loop control, where the ADCS can operate. It will be responsible to compute the errors in the CubeSat current attitude and to make them converge to a value small enough for the correct completion of the mission.

The LICIACube ADCS is supported by an IMU, two sun sensors and a star tracker, that provide the inputs for the control system. As actuators, Reaction Wheels are the primary means of orientation, while a Propulsion System (PS) serves as support. The Propulsion System is approximately 2U in volume and uses six 25mN cold gas thrusters to develop 500N – sec of total impulse. The velocity provided is so 37 m/s of ΔV for 14 kg of CubeSat. Each of these six thrusters can operate independently from the others through an integrated microprocessor controller, making it possible to perform ACS maneuvers [17]. Desaturation of the RWs is handled by the PS, but only when the ADCS sensors send a warning for safety limits exceeded.

The most critical part of the mission for the ADCS, the fly-by moment, will last for 389 seconds. In this period, LICIACube will perform a heavy braking maneuver that will result in a high velocity variation. That is the most difficult point, since it is easy to lose the objective from the camera.

Objectives and motivation

The work done and described in this thesis is focused on the design and developing of a suitable ADCS, able to switch between different functioning modes in an autonomous manner, while maintaining certain requirements, using the exact mission initial conditions and data, for Argotec LICIACube. Necessary operative modes comprehend Detumbling Mode for the detach, Fully Operative Mode for the fly-by phase and Desaturation Mode, to restore the nominal working conditions of the Reaction Wheels. To each mode corresponds a singular, completely dedicated, control system. The ADCS developed must begin its functioning after reaching the observation point, that will happen in about 120 hours, when the proximity operations will begin. The mission requirements come from the moment of Closest Approach (C/A): given the LICIACube limited FoV and the great precision required for the images by the scientific aspect of the mission, the tracking of Didymos will require high angular velocities, that must be expressed by the CubeSat Reaction Wheels, while making sure that thrusters can desaturate them when it is needed.

Introduction

The goal is to set up and validate a realistic system model on Simulink and, then, develop a set of robust controllers through different simulations. These controllers should regulate the behavior of the satellite with many different initial conditions.

A state-finite machine structure is then implemented on Stateflow; this allows to simulate the entire mission taking into consideration also the initial deployment phase and the possibility to have a situation with a saturated actuator

Moreover, it must be easy to customize: the ADCS is developed and tested with real mission data on a hardware that simulates the real satellite, but it will be not mounted for real on the CubeSat. It will be used as official ADCS in future satellites.

Thesis outline

- In Chapter 1 the basic principles of control theory are presented. Sliding Mode Control and PID control are explained, being them used in two operative modes.
- In Chapter 2 the mathematical model of LICIACube is presented. Reaction Wheels exchange momentum mechanism and Extended Kalman Filter theory is also dealt with.
- Chapter 3 deals with the Matlab and Simulink simulation environment. Model implementation and simulation settings are discussed. Then, Detumbling and Fully Operational Mode are showed: in the first mode, focus is placed on the Sliding Mode Controller realization, the generation of the dataset necessary for the simulation and the setting of the filter; in the second mode, attention is directed towards the PID controller and the other, simpler, filter.
In the end, the simulation of both modes is shown through Matlab graphs.
- Chapter 4 is entirely dedicated to the Desaturation mode. It is split in two control systems and needs an entire chapter to better explain the technical choices made during the developing of the project. Simulations are exposed at the end of the chapter.
- Chapter 5 is dedicated to interface experiments. The Stateflow structure, that imitates a finite-state-machine in order to simulate the behavior of the ADCS, is analyzed. It regulates the switches between operating modes by reading sensors data. Complete simulation results are discussed at the end.
- Chapter 6 is devoted to the thesis conclusions, possible improvements and what next.

Chapter 1

Control techniques description

1.1 Control systems

A system can initially be seen as a black box, without knowing what is inside. It is possible to see that it accepts inputs and returns outputs, following some unknown process, which is often the object of study. In the control branch of engineering, the inputs and outputs are both signals, while the system is generally a plant. They are represented in easy to read graphs called block diagrams.

In aerospace applications, the system control is a matter of great importance. The success of a mission heavily relies on the possibility to obtain desired outputs in situations where not everything has been foreseen, like external disturbances, by commanding, often autonomously, some inputs. Here lies the importance to be able to handle at will plants, that is possible through some class of systems, called control systems.

The role of the control system is to transform the given input into some other signal that the plant will, as known, transform in a wanted output: it manipulates the real input to feed the plant with a precisely calculated signal. In order to do that, a block that precedes the plant is needed, called controller. There exist many different types of controllers, depending on their logic and architecture. Their choice depends on the case study. However, all controllers behave better when the control system is in closed loop.

Closed loop control systems are configurations where sensors are used to calculate the values of the output of the plant. Thanks to this new data, it is possible to change accordingly the input to give to the plant through the controller in the next cycle. This mechanism is called feedback. It is clear that this configuration helps to reduce the effects of possible disturbances that enter directly in the plant, allowing a recalibration of the input.

Open loop configuration, instead, do not allow the use of sensors and, so, do not use the feedback. In almost all applications open loop configuration do not find much space, and when they do, they offer poor performances.

A single loop in a closed loop control starts with the reference signal that simulates the wanted behavior of the plant. This signal is fed to the controller along with the measured output of previous cycles from the sensors. It will be responsible for the input of the plant,

which will be affected from disturbances too. The aim of the control system is to drive the error between effective output and reference signal to zero.

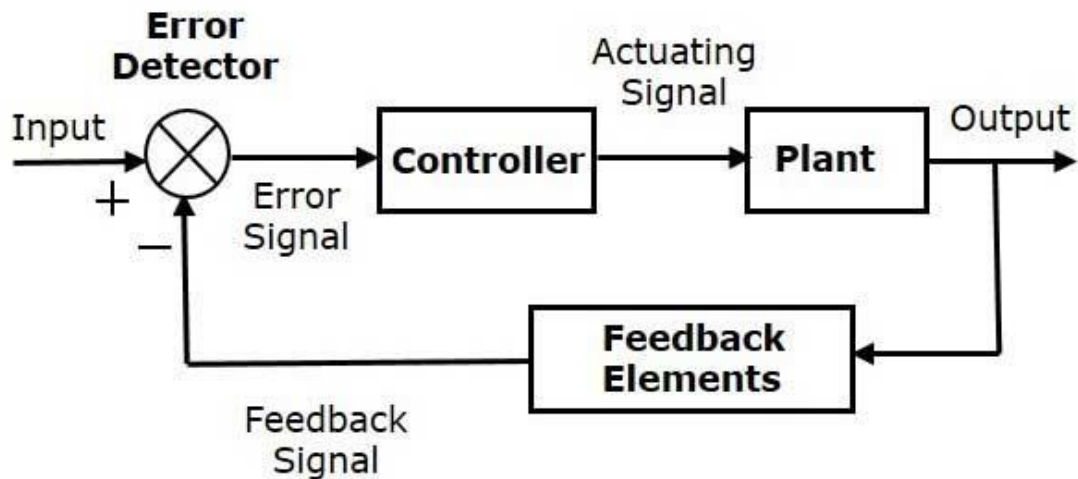


Figure 7: Closed loop system

Having specified the concept of closed control loop, the focus is posed on controllers.

Control theory is a branch of engineering that deals with all types of systems that accept inputs. Its purpose is to design controllers which, along with the feedback mechanism, allow to properly control the system. The concept of the feedback is to make the system aware of the difference between the reference signal and the output and make him react accordingly to bring it closer to the desired value.

Control theory is divided in two branches.

- Linear control theory is for those systems for which the superposition principle holds. Of these, the biggest part is represented by the linear time invariant (LTI) systems, for which parameters don't change over time. They are based on linear differential equations.

The approach to LTI systems is made by the "basic" control methods: they are techniques of great generality that sets the fundamentals of more complex control methods. However, LTI systems are not so frequent and mostly belongs to academic examples.

- Nonlinear control theory deals with systems for which the principle of superposition does not hold. It covers a much higher number of real-world systems: they are majorly nonlinear in nature when they are considered over a wide operating range.

Chapter 1: Control techniques description

In some cases, when it is needed to control the system only in certain condition, a nonlinear system can be linearized around an equilibrium point. This allows to have, then, a linear approach to the problem.

Generally, nonlinear control can be applied also to LTI systems. However, the results are not always optimal, even if it is possible to obtain some interesting specifics or effects.

The possibility to apply nonlinear control to LTI systems is one of the primary reasons behind the growing interest of this matter. Often, in LTI systems, many forces and hard nonlinearities are designed with too much simplicity or approximation, causing a trade-off between performance and effective control. Also, uncertainties are always present. Nonlinear strategies allow to improve the overall performances by dealing correctly with uncertainties or forces that vary differently from linear ones. So, while linear control laws put constraints on physical quantities to achieve a certain accuracy, nonlinear control allows to compensate errors to achieve unmatched performances. They introduce robustness or adaptability to parameters sudden change. Moreover, linear systems that are not observable or controllable in the linear sense can be observable/controllable in the nonlinear sense.

1.2 PID control

PID controllers are nowadays the most common industrial form of feedback: more than 95% of control loops in process control are of PID type. They are an important ingredient of a distributed control system. Being them often combined with other function blocks as selectors, simple functions or sequential, they can be the protagonist for the building of a complicated autonomous system. PIDs are involved also in more complex control architectures, organized hierarchically. An example may be the model predictive control, where PIDs are at the foundation, in the lowest level [16].

Although technology improved a lot during the years, PID controllers have survived the many changes that came along, probably thanks to the advent of microprocessors. Nowadays, in fact, they are always implemented software-like through the use of embedded microprocessors. This allowed to provide additional features to these controllers, as adaptation, anti-windup, automatic tuning.

PID controller exercise a triple control action. Their algorithm is generally indicated through the formula

$$u(t) = K(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt})$$

Where $u(t) \rightarrow$ Control signal

$e(t) \rightarrow$ Control error

From the equation it can be seen that the control signal is the sum of three components. The first one is the Proportional term (P), that is proportional to the error signal. In the equation it is indicated with K and can be considered as a control action “based on the present” [16]. Macroscopically, by increasing the term, the system will respond more aggressively to the presence of error between output and reference signal, and so the error will decrease, but will not tend asymptotically to zero. Moreover, higher gain values increase also the tendency of the system to oscillate.

The second term is the Integrative term (I). It is proportional to the integral of the error and the corresponding parameter in the equation is T_i , called integral time. It can be considered as a control action “based on the past” of the system. By decreasing the integral value T_i , it is possible to enhance the integral action: this brings to a further reduction of the error signal, allowing to arrive to a null steady state error. As for proportional action, also here by decreasing too much T_i oscillations appear.

The last term is the Derivative term(D). Proportional to the derivative of the error, it appears in the equation as T_d as derivative time. This is the control action “based on the future”. By increasing the parameter, it is possible to add a damping effect on the system. However, the value of T_d should not be too high, or the oscillatory effect begins again. This phenomenon can be understood easily considering the derivative time as the horizon of

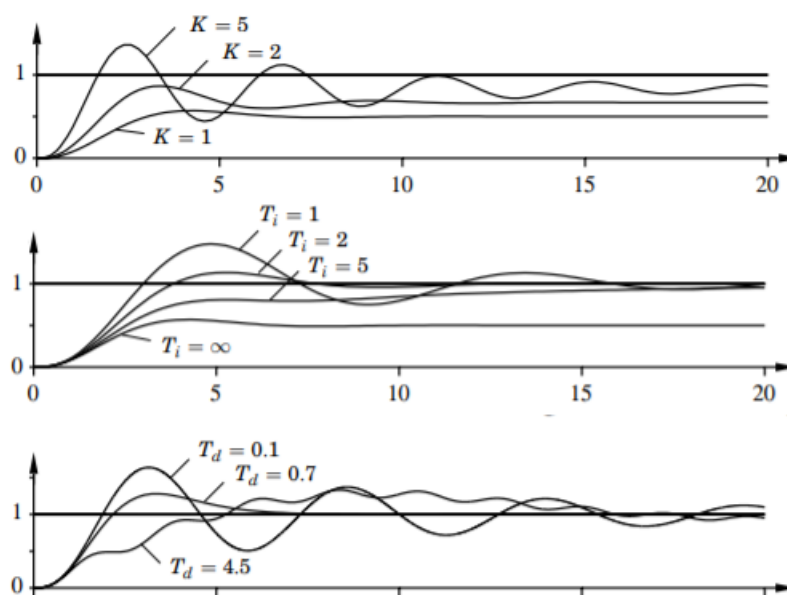


Figure 8: Proportional, Proportional-integrative, PID effects [16]

Chapter 1: Control techniques description

prediction of the “future”, which the derivative term is based on. If it is too high, the future becomes too much uncertain and the effect is lost.

Before using a PID, however, the correct implementation of the algorithm is not the only thing to pay attention to. Other effects should be considered, like the necessity of filtering the noise or the windup effect.

The derivative action of a PID can, in fact, bring some noise problems. A differentiation is generally sensitive to noise: in the case of a sinusoidal disturbance with a frequency ω , the derivative action of the controller will bring in the control signal a component that is linearly dependent by noise $\omega * \cos(\omega t)$. Depending on the frequency value, the effect on the system can be disastrous.

To avoid this effect, it is necessary to change the control algorithm of the PID. In particular, the derivative action must be implemented as

$$D = -\frac{sKT_d}{1 + sT_d/N}Y$$

Instead of $D = sT_dY$, where s is the Laplace variable. It can be considered as the ideal derivative term sT_d , but filtered by a first-order system with a time constant of T_d/N . In this way it is possible to filter high-frequency noise, while the gain is at maximum $K * N$. The total transfer function of a PID controller becomes, so:

$$H(s) = -K\left(1 + \frac{1}{sT_i} + \frac{sT_d}{1 + \frac{sT_d}{N}}\right)$$

The windup effect is, instead, a nonlinear phenomenon that affects not only PID controllers, but all control architectures that use integrative terms, being it caused by the integral action along with the saturation effects of the actuator.

In normal working conditions, it is possible that the control variable reaches the limits of the actuator. In these situations, the control loop is no more linear, and the system runs as an open loop, because no matter what the output is, the actuator will remain at its limit. The real problem starts when the controller uses an integrative term, as the PID. In case of saturation, the integral term may become very large, or “winds up”, since the error is continually integrated and cannot be lowered much due to actuators limits. In order to return to normal, the error must assume opposite sign for a certain period of time, with large transients and possible oscillations as consequences.

Chapter 1: Control techniques description

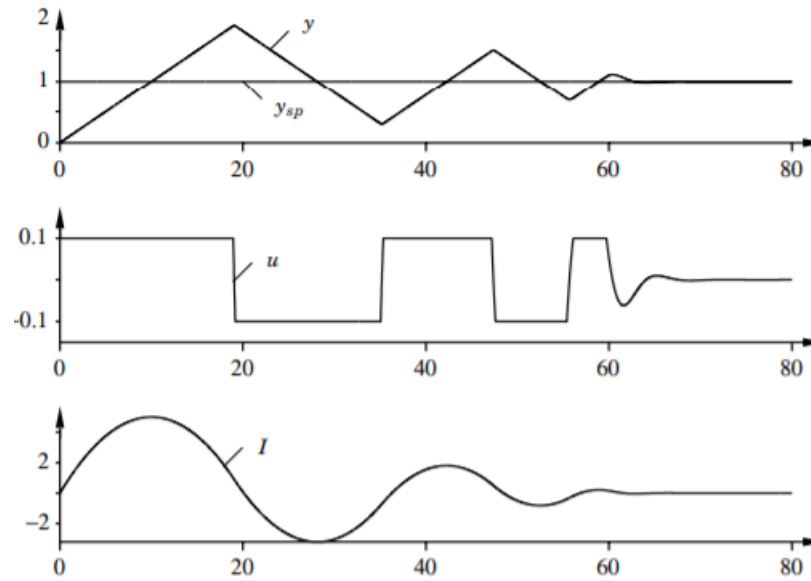


Figure 9: Windup effect example [16]

This phenomenon is well known and, with the advent of digital controllers, many ways to avoid it were ideated. One attempt, that however leads to poor performances, is to introduce limiters so that the control signal never reaches the actuator limits.

1.3 Sliding Mode Control

Sliding mode control (SMC) is a nonlinear technique close to feedback linearization. The target systems are nonlinear systems in companion form.

A system is said to be in companion form if it is described by the equation

$$\dot{x}^{(n)} = f(x) + b(x) * u$$

With u = scalar control input

x = scalar output

$x = [x, \dot{x}, \dots, \overset{(n-1)}{\ddot{x}}]^T$ = state vector

$f(x)$ and $b(x) \rightarrow$ nonlinear functions of the states, with $b(x)$ invertible.

It is easy to note that, even if there are derivatives of x , there is no derivative of the input u . In state-space representation the equation can be written as

$$\frac{d}{dt} \begin{bmatrix} x_1 \\ \dots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} x_2 \\ \dots \\ x_n \\ f(x)+b(x)*u \end{bmatrix}$$

Differently from feedback linearization, however, these models may have some imprecisions of various nature, that can be classified in two different kinds

1. Structured uncertainties: they are errors and bad approximations of the terms done during the modeling of the system.
2. Unstructured uncertainties: also called unmodeled dynamics, they are due to underestimation of the system order or neglectation of terms.

Generally, inaccuracies and uncertainties can lead to bad effects on nonlinear control systems. In chaotic systems even a small difference in initial condition can result in huge variations of the results.

Differently from Adaptive Control, which approaches these imprecisions by regulating the controller each cycle, Sliding Mode Control deals with them by counting on robustness. It allows to obtain stability and good performances at the cost of an extreme high control activity [6]. Another feature to consider is the reduced-order compensated dynamics and the finite-time convergence [3].

The main idea of the SMC is to define a subset of plant states combination, denoted as "sliding surface". It is considered as the exclusive set of states that the plant may assume,

Chapter 1: Control techniques description

in order to have good performances and linear behavior. Then, the controller activity is split in two parts. In the first phase, the reaching phase, it has to drive the nonlinear dynamics of the system on the desired surface; then, in the second one, these dynamics must be maintained on the surface, avoiding all nonlinear irregularities. The split of these two phases is underlined in the control law, which is formed by a nominal feedback control term, to linearize and reach the surface, and a discontinuous term that allows to cancel the uncertainties and make the error converge to zero.

As first step to a Sliding Mode controller design, it is necessary to analytically define and calculate a sliding surface equation.

Taking as example a SISO system described by the equations

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

With f , g and h smooth functions on R^{n_x} , g invertible function. The system is “affine” in u .

To apply sliding mode control, it is necessary to have a system expressed in the general form, and the state must be measurable. If it is not the case, an observer should be employed.

The general form of the system may be expressed as

$$\begin{aligned}\dot{\mu} &= \begin{bmatrix} \mu_2 \\ \dots \\ \ddot{\mu}_\gamma \\ a(x) + b(x) * u \end{bmatrix} \\ \dot{\varphi} &= \omega(\mu, \varphi) \\ y &= \mu_1\end{aligned}$$

Where: γ =relative degree of the system

$\mu = (\mu_1, \dots, \mu_\gamma) = (y, \dot{y}, \dots, y^{(\gamma-1)})$ as external dynamics

$\varphi \in R^{n-\gamma}$ as internal dynamics, assumed locally asymptotically stable

(μ, φ) as the state of the system in general form.

After defining the tracking error, in order to have a finite convergence time, as

$$\tilde{y} = r - y$$

With r as reference signal, it is possible to define the sliding surface: it is the set of states described by

Chapter 1: Control techniques description

$$S(t) \triangleq \{x \in R^n: s(x, t) = 0\}$$

Where $s: R^{n+1} \rightarrow R$ is the function

$$s(x, t) \triangleq \tilde{y}^{(\gamma-1)} + k_\gamma \tilde{y}^{(\gamma-2)} + \dots + k_2 \tilde{y}$$

And k_i are parameters chosen so that the roots of the polynomial

$$P(\lambda) = \lambda^{\gamma-1} + k_\gamma \lambda^{\gamma-2} + \dots + k_2$$

Have negative real part. These parameters influence deeply the behavior of the sliding surface. By placing the roots of the polynomial P on the complex plane, it is possible to have a stronger convergence, a minor settling time etc. The proof can be obtained by Laplace transforming the equation $s(x, t)$

The sliding function, alone, does not guarantee a proper robust controller, if implemented. It still lacks two important properties. It must be invariant and attractive.

In order to obtain these properties, a control law u is designed. It is formed by two parts, each of them deals with one of the two characteristics.

The sliding surface is invariant if, given a plant trajectory that is already on the sliding function S , it remains on it:

$$x(t) \in S(t) \rightarrow x(\tau) \in S(\tau) \quad \forall \tau \geq t$$

To guarantee this condition, it is necessary to impose that $\dot{s} = 0$. This means that

$$\tilde{y}^{(\gamma)} + k_\gamma \tilde{y}^{(\gamma-1)} + \dots + k_2 \dot{\tilde{y}} = 0$$

By substituting $\tilde{y}^{(\gamma)} = r^{(\gamma)} - y^{(\gamma)} = r^{(\gamma)} - a(x) - b(x) * u$

And solving for u , it is possible to obtain the control law

$$u_s = \frac{1}{b(x)} (r^{(\gamma)} - a(x) + k_\gamma \tilde{y}^{(\gamma-1)} + \dots + k_2 \dot{\tilde{y}})$$

This part of the control law allows to make $S(t)$ an invariant set.[7]

The other part deals with the attractiveness. Assuming that, in a certain time t , the trajectory of the plant is not on S , to make the surface attractive it is necessary to add a discontinuous term to the control law. This term allows to continuously reduce the

Chapter 1: Control techniques description

tracking error. As stated in [6] and similarly to the adaptive control, the term to be introduced is

$$k_1 \text{sign}(s(x, t))$$

With $k_1 > 0$.

In this way, the equation of the sliding function derivative is

$$\dot{s}(x, t) = r^{(\gamma)} - a(x) - b(x) * u + k_\gamma \tilde{y}^{(\gamma-1)} + \dots + k_2 \dot{\tilde{y}} = -k_1 \text{sign}(s(x, t))$$

It can be seen that, from the choice of the signum function, not only attractiveness is obtained. Convergence in a finite time of the sliding function is also proved. In fact

$$s(x, t)\dot{s}(x, t) < 0 \quad \forall x, t$$

Implies that $S(t)$ is attractive. Moreover if

$$s(x, t) > 0 \rightarrow \dot{s} = -k_1$$

Or if

$$s(x, t) < 0 \rightarrow \dot{s} = k_1$$

In both cases, by Laplace transforming, it is possible to see

$$s(t) \rightarrow 0$$

$$x(t) \rightarrow S(t)$$

That means that convergence is obtained in finite time.[7]

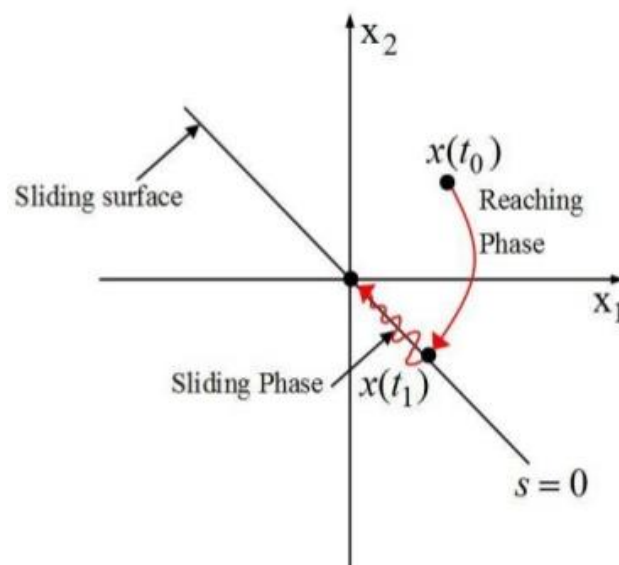


Figure 10: Sliding surface (credit to Mathworks)

Chapter 1: Control techniques description

The choice of the signum function brings, however, also a side effect. The introduction of a discontinuous term in the control law implies a very high control activity that may lead to a phenomenon called “chattering” that manifests as high frequency oscillations around the sliding surface. Generally, there are two main causes of chattering appearance

- Unmodeled dynamics of the system. Many times, small time constants dynamics are neglected during the modeling of the plant. They manifest in unexpected and quick turns in the control law behavior.
- Bad relations between control activity and sampling frequencies. Digital controllers operate at a fixed operational frequency, i.e. finite sampling rate. If the control switching frequency requested is greater than the sampling rate, the controller can not keep up with the request, leading in “discretization chatter”. Switching frequency can't exceed sampling frequency.[8]

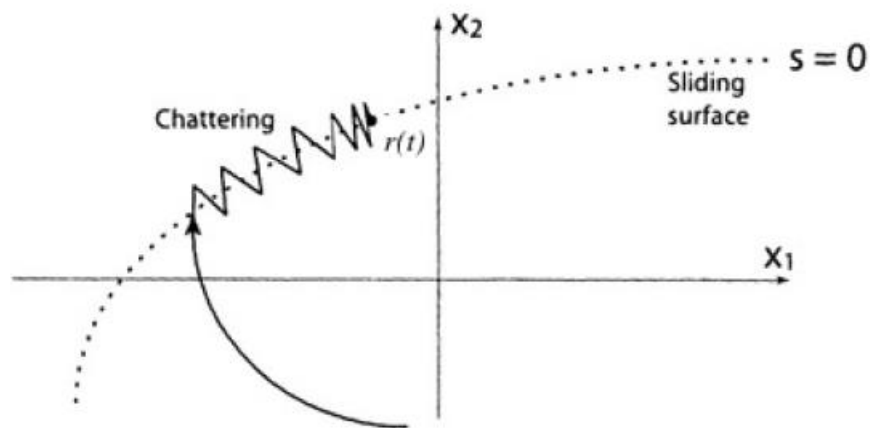


Figure 11: Chattering effect [7]

The disturbance effect obtained by introducing the discontinuous term in the controller belongs to the second category. There are, however, different ways to reduce it.

More complex methods like High Order Sliding Mode (HOSM) algorithms maintain the robustness of the controller and reduce the chattering by extending the study of the sliding function also to their derivatives. By imposing the finite convergence directly to the derivatives of the sliding function is, in fact, possible to obtain a smoother signal free of disturbances.

Simpler methods, that can be complementary, involve some observation in the control law. Integral Sliding Mode Control (ISMC) sticks with the study of the sliding function and its first derivative but changes the control signal in the sum of two terms. A first, linear

Chapter 1: Control techniques description

term stabilizes and linearizes the system, as in classical SMC theory. The second one is an integral term that acts as a filter. The resulting control signal is so guaranteed to be continuous, thus leading to low excitation of the unmodeled dynamics.

Lastly, it is possible to reduce chattering by choosing another function that resembles the signum, but smoother, called sigmoid function $\sigma(\eta s)$, where η is a parameter that can be designed to determine the slope. A typical example is $\sigma(\eta s) = \tanh(\eta s)$.

The final control law of a Sliding Mode Controller can be written. Writing the tracking error as

$$\tilde{\mu} = (\tilde{\mu}_1, \dots, \tilde{\mu}_\gamma) = (\tilde{y}, \dot{\tilde{y}}, \dots, \tilde{y}^{(\gamma-1)})$$

The equation of sliding mode control becomes

$$u = \frac{1}{b(x)} (r^{(\gamma)} - a(x) + k_\gamma \tilde{\mu}_\gamma + \dots + k_2 \tilde{\mu}_2 + k_1 \sigma(\eta s))$$

Robustness is the base concept of a Sliding Mode Control. After defining the general control law, it is possible to prove mathematically how uncertainties and approximations do not lead to effective changes in the controller behavior.

Supposing that a system to be described is not exactly known, but there are some uncertainties. The system equation is

$$y^{(\gamma)} = a(x) + \Delta a(x) + b(x) * u$$

Where Δa is an unknown smooth function in R^{n_x} bounded

$$||\Delta a(x)|| \leq \bar{\Delta} \quad \forall x \in R^n$$

By considering as sliding surface and control law equations the same presented before, the property of robustness derives from the study of the time derivative of the sliding surface

$$\begin{aligned} \dot{s}(x, t) &= r^{(\gamma)} - a(x) - \Delta a(x) - b(x) * u + k_\gamma \tilde{y}^{(\gamma-1)} + \dots + k_2 \dot{\tilde{y}} \\ &= -\Delta a(x) - k_1 \text{sign}(s(x, t)) \end{aligned}$$

It means that, if $k_1 > \bar{\Delta}$ the finite convergence property still holds, and the controller is robust to uncertainties until a certain error.[7]

Chapter 2

Mathematical Model

Prior to show the effective realization of the ADCS for LICIA Cube, along with all its control system, it is necessary to make considerations and discuss about some general aspects of CubeSats cinematics and dynamics, with digressions on the fundamental physics.

2.1 Dynamics

In spacecrafts, dynamic and kinematics equations that govern the body motion can be considered as the sequence of two nonlinear subsystems.

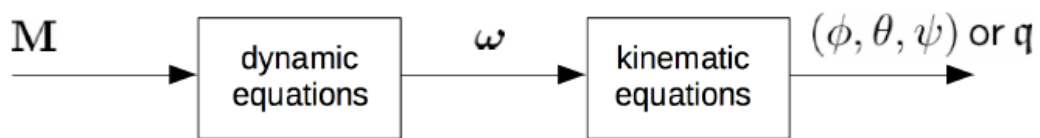


Figure 12: Dynamics and cinematics [14]

The dynamic block deals with all the parts of the system through it is possible to pass from the torque expressed on the body M to the angular velocity assumed ω .

For rotating body like spacecrafts, the dynamics are well described by the Euler's equations for rigid bodies. They are vectorial quasilinear first-order differential equation that describe rotations of a rigid body with reference to a rotating reference frame, fixed within the body and parallel to its principal axes of inertia.

The general, matricial form of the equation is:

$$I\dot{\omega} + \omega \times (I\omega) = M$$

Where I = Inertia matrix of the rigid body

ω = Angular velocity of the body about the principal axes

M = Applied torque.

While, if the Inertia matrix of the body is diagonal, the equation may be written in the three-dimensional principal orthogonal coordinates:

Chapter 2: Mathematical model

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

Where M_i = Applied torques components.

I_i = Principal moments of inertia

ω_i = Angular velocity components.

For LICIACube, however, these equations must be further extended.

As stated before, the dynamic block takes as input the torque M expressed on the body. It is, so, very much linked to the type of actuator that the spacecraft hosts, in this case, Reaction Wheels. With RWs, it is necessary to make a separation between LICIACube body inertia and Reaction Wheels inertia, since there is a continuous transfer of angular momentum between the two elements.

By indicating as I_{sat} the inertia matrix of the satellite and I_{RW} the inertia matrix of the actuators, the general form dynamic equation becomes:

$$I_{sat}\dot{\omega} = M - (\omega_{sat} \times I_{sat}\omega_{sat}) - (\omega_{sat} \times I_{RW}\omega_{RW})$$

The above equation underlines the double effect of Reaction Wheels on the system.

The torque M is the torque applied on the body of the satellite by the actuators. It is important to note that this is not anymore an external torque, but an internal one, since it is dealing with a Momentum Exchange actuator. Opposite to this contribution, while the second term considers the resistance effect of the satellite inertia, as normally present in standard Euler's equation, is the last term.

$(\omega_{sat} \times I_{RW}\omega_{RW})$ considers the further resistance to motion of the entire system, this time given by the angular momentum of the Reaction Wheels. It is also referred as "coupled term"

The use of this type of Momentum-exchange actuator introduces, so, a double change in the Euler's equation. This equation, alone, describes the dynamics of LICIACube.

A different situation stands for the Desaturation phase. The presence of two actuator systems, that have to communicate and cooperate, introduces another term in the dynamics of the spacecraft. It will be analyzed in the dedicated chapter later.

2.2 Kinematics

A spacecraft can be considered as a rigid body that moves w.r.t. an inertial frame. In the ADCS developing only the rotation of the body is studied, being it possible to divide translation and rotation independently.

Kinematics, as the second block of the nonlinear block series, take as input the angular velocity of LICIAcube and give its effective attitude in space, in the form of Euler angles or, more frequently, quaternions. This output is the last element of the control loop.

In the case study, the chosen representation is in quaternions, since it must be coherent with the reference signal. The passage from angular velocity to quaternion is made easily possible thanks to quaternion properties. In particular

$$\dot{q} = \frac{1}{2} Q \omega$$

Where $\omega \rightarrow$ Angular velocity

$\dot{q} \rightarrow$ Derivative of attitude quaternion

And Q is the following transformation matrix:

$$Q = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix}$$

By integrating the result, it is possible to obtain the actual quaternion q . It is important to note that, after integration, the quaternion must be normalized, since the norm must always be equal to 1.

In this thesis work, the real part of the quaternion is indicated as q_0 .

2.3 Actuation Systems

LICIAcube hosts two actuation systems, one primary and the other one serves as support and, so, is used only in specific conditions. The main one is represented by the Reaction Wheels. Based on the angular momentum conservation law as stated in previous chapters, their functioning starts by commanding, through a signal from the controller, the electric motor of each wheel.

Chapter 2: Mathematical model

It is important to understand the physic principle at the base of Reaction Wheels induced rotation on the spacecraft. Activating through the electric motor the actuators, they produce a torque given by

$$T_{RW} = \dot{h}_{RW} = I_{RW} * \dot{\omega}_{RW}$$

Where I_{RW} → Inertia Matrix of the RWs

And $\dot{\omega}_{RW}$ → Angular acceleration of the RWs

The angular momentum of a Reaction Wheel can be expressed as

$$h_{RW} = I_{RW} * \omega_{RW}$$

By considering the total angular momentum of the entire system, given by spacecraft and Reaction Wheels, as

$$h_{tot} = h_{sat} + h_{RW}$$

It is possible to understand, for the angular momentum conservation law, that the spacecraft will undergo a torque in the opposite direction equal to

$$T_{sat} = I_{sat} * \dot{\omega}_{sat}$$

Where I_{sat} → Inertia matrix of the satellite

And $\dot{\omega}_{sat}$ → Angular acceleration of the satellite.

Being this the functional concept of the Reaction Wheels, the thesis focuses on their possible modelization. In particular, attention is placed on how the blocks react to the signal sent from the controller. Commanded from the controller through an electrical signal that indicates the required torque to drive the error to 0, it is not assured that the torque request is always fulfilled. As every actuator, there is an upper and lower saturation limit, that makes this block nonlinear. Moreover, wheels cannot rotate at an angular velocity higher than a certain value: this must be considered when, at each cycle, the torque expressed is calculated, since they cannot be accelerated further, resulting in a null expressed torque. It is necessary to find a characteristic that links the requested torque and the expressed torque with respect to the wheels angular velocities. It is not a simple equation as reported above, since this relation depends on the actuator past values and accumulated angular momentum.

Chapter 2: Mathematical model

Argotec offered help in this matter. Assisting the work of a previous student in internship, they provided a script that simulates the actuator characteristic. Numerical values were, calculated by continuous testing on the hardware, commanding a set torque and measuring the effective rotation of the wheels. They were then interpolated, thus obtaining a discontinuous characteristic.

Cold Gas thrusters are used as secondary actuation system. As stated before, they work by ejecting mass to create a thrust. Depending on the fact that generated thrust vector can pass or not through the spacecraft center of mass, they can create forces or torques. Not always the force generated is controllable, since they work in a ON/OFF fashion. However, it is possible to change their installation position in order to reduce or increase the moment arm r .

Due to their nature, thrusters can only provide force or torque in one direction, opposite to the mass ejected. For this reason, in order to produce both negative and positive torques, thrusters must work in couples. Generally, for each movement in attitude control, no more than 2 thrusters are used at time.

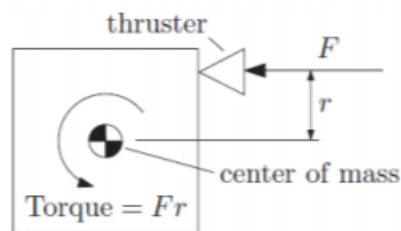


Figure 13: Thruster functioning [10]

Their functioning follows the Tsiolkovsky rocket equation, or ideal rocket equation. It relates the maximum change of velocity of the rocket when no other external forces act, called Δv , with the exhaust velocity (velocity that the expelled mass assumes) and the initial and final mass of the rocket, due to fuel consumption. [10]

$$\Delta v = v_e \ln \frac{m_0}{m_f}$$

Where

v_e → Exhaust velocity $v_e = g_0 I_{sp}$

m_0, m_f → Rocket initial and final mass

Δv → Maximum change of velocity

In particular, the acceleration given, if there are no other forces except for thrusters, is

$$\dot{m} = \frac{m_0 dV/dt}{v_e} = \frac{F}{g_0 I_{sp}}$$

However, the implementation of a thruster attitude control does not imply equations only. Normally, to achieve a certain grade of precision in attitude maneuvers, thrusters are activated in pulses, through the use of a Pulse Width Modulation mechanism. The concept will be further studied in the desaturation chapter.

2.4 Fully Operational Mode Controller

The Fully Operational Mode stands for the normal functioning of LICIACube, where it has to rotate following the reference signal sent by the optical payload. In particular, during the fly-by, the CubeSat shall operate accordingly to this control system: this means that the control system has to provide maximum stability and accuracy, along with a certain grade of robustness. It is possible, in fact, that there can be changes in the inertia matrix of the satellite for possible fuel consumption due to desaturation phases.

The core part of this control system is the controller section. Its aim is to provide the correct torque request, to be sent to the actuator, to properly control the spacecraft. It is done in two steps:

- Sliding Surface calculation: takes as input the error quaternion and the error angular velocity of the spacecraft and calculates the sliding surface which has to be brought to 0 to achieve good performances.
- Control law implementation: uses every information that sensors can give to implement a control law for requesting the correct torque to the actuators.

The nonlinear control technique chosen is the Sliding Mode Control. This was not the only possible choice, since there were valid alternatives.

Adaptive control and Sliding Mode are among the best possible solutions given the mission requirements and, thus, were the two options considered.

Adaptive control offers a dynamic solution to possible changes in the satellite inertia and not well estimated plant parameters, allowing a continue recalibration of the control system. Accuracy and settling time are also guaranteed, probably achieving a result that cannot be improved by other techniques. However, there were some aspects that led to the choice of Sliding Mode Control:

1. Adaptive control can slow down the control activity. The updating of the variables of the controller, which is not deterministic, may lead to longer cycles. Longer cycles mean longer major steps, and so loss of performances in many fast systems.
2. Robustness is not guaranteed. Adaptive control is a technique that, differently from Sliding Mode Control, does not linearize the system. Linearizing the system is useful since small changes in the initial conditions lead to small changes in the output. For nonlinear systems, this does not hold. This means that, with disturbances or changes in initial conditions higher than a certain value, control will not be achieved.

Essentially, the robustness propriety as showed in chapter 1.3 does not hold. There are, obviously, means to cover this lack, with higher order or more complex control structures, but they are beyond the scope of the thesis.

For LICIACube, while a good adaptive controller is still viable, a possible slight loss in performance in exchange for robustness is preferred.

3. Adaptive control can be of harder implementation. The main problem is to design a correct adaptation mechanism and to analyze the tracking convergence. In particular, for the latter, it is necessary to study a candidate Lyapunov function, that is not always easy to find. This procedure may take some time and efforts.

Sliding Mode Control is, so, considered the best choice for LICIACube. It is vastly used in many of its variant in many aerospace applications. Moreover, its robustness suits perfectly the concept of modularity: with little to no changes, the controller should work with different CubeSats belonging to the same family.

The first step to implement the controller is the design of the sliding function.

Defining as state vector of the system $x = [q_0 \ q_1 \ q_2 \ q_3 \ \omega_1 \ \omega_2 \ \omega_3]^T \in R^7$ and $u = [u_1 \ u_2 \ u_3] \in R^3$ and considering as base equations of the system the relations

$$\dot{q} = \frac{1}{2} Q \omega$$

$$\dot{\omega} = -I_{sat}^{-1} \omega \times I_{sat} \omega + I_{sat}^{-1} * u$$

$$y = q$$

With relative degree $\gamma = 2$

It is possible to calculate the sliding function with the same method described before.

Normally, it would be

$$s(q, \omega, t) = \tilde{\omega} + k_2 \tilde{q}$$

With $\tilde{\omega} = \omega_r - \omega$ as error angular velocity

$\tilde{q} = q_r - q$ as quaternion error

$k_2 =$ setting parameter.

The sliding surface function considers both attitude and angular velocity for the correct maneuvering of the spacecraft. However, since LICIACube requires strict performances, some small changes help to improve the control action. The function

$$s(q, \omega) = \tilde{\omega} + k_2 \text{sign}(\tilde{q}_0) \tilde{q}$$

Guarantees the shortest orientation path [14] and was, thus, chosen for the mission, considering the possibility of rotations greater than π radians or on more than one axes. The second step of the subsection is the control law implementation. By deriving the sliding function and putting

$$\dot{s} = 0$$

So that the sliding surface is invariant, inverting with reference to the control input u it is possible to calculate the first part of the control law.

$$u_s = I_{sat} \left(\dot{\omega}_r + \frac{k_2}{2} (|\tilde{q}_0| \tilde{\omega} + \text{sign}(\tilde{q}_0) \tilde{q} \times (\omega_r + \omega)) \right) + \omega \times I_{sat} \omega$$

Where:

$\dot{\omega}_r \rightarrow$ Reference angular acceleration of the spacecraft

$\omega_r \rightarrow$ Reference angular velocity of the spacecraft

$\tilde{q}_0 \rightarrow$ Real part of the error quaternion [13]

It is now necessary to add the term for the attractiveness of the sliding surface. Normally, the signum function would be implemented. However, chattering, as already stated in the first chapter, is a disturbing phenomenon that cannot be overseen in this application.

Among the possible solutions, the signum is substituted by a sigmoid function that depends on a modifiable parameter and on the sliding function. The final control law becomes

$$u = u_s + k_1 I_{sat} \tanh(\eta s)$$

Where:

k_1 → Setting parameter

η → Setting parameter that represents the slope. It must be as high as possible, to resemble the signum function.

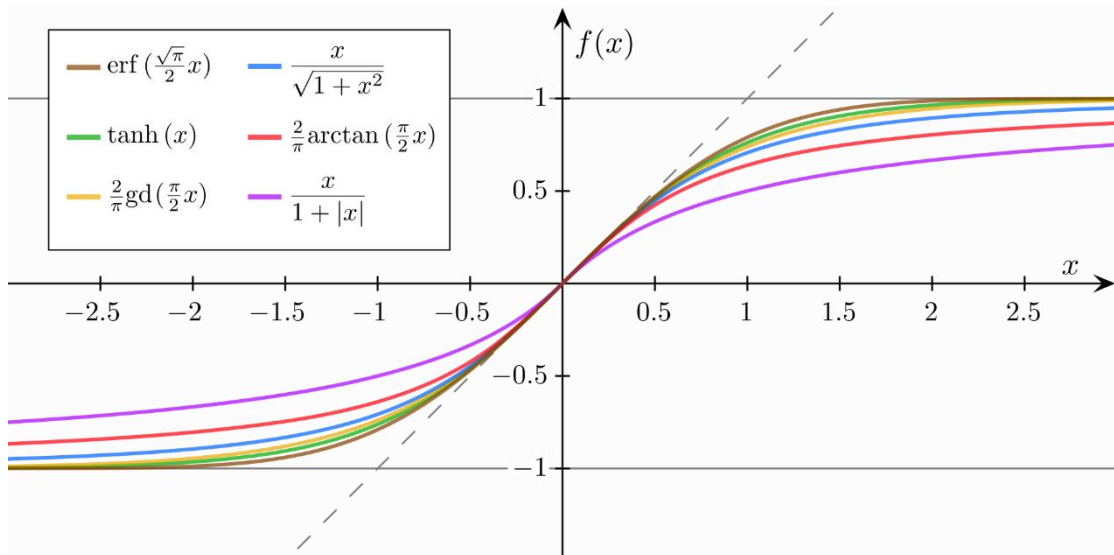


Figure 14: Sigmoid functions

In the end, the three parameters that regulate the control action are chosen. There is no particular rule to follow. Firstly, η must be set to a high enough value to reduce chattering. Then, a good strategy to apply is to fix k_1 while changing k_2 in a trial and error procedure. It is worth to remember that the coefficients k_n can be designed considering that the closed-loop dynamics near the sliding surface is determined by the roots of the polynomial $P(\lambda)$.

2.5 Detumbling Mode Controller

The Detumbling mode is a one-time-only situation, when the LICIAcube separates from DART. In fact, by mission data, the detach can happen with a worst-case scenario where the CubeSat assumes an angular velocity of 7° on a single axis. Aim Detumbling mode is to slow down the spacecraft, until the angular velocity goes below a safety limit where fully operative mode can work.

No particular requirements are asked, since precision and null error are asked during the nominal working of LICIAcube, in particular during the fly-by phase.

For this reason, a PID controller seems more than enough to deal with the phase. It is, in fact, easy to implement, in particular with the advent of digital controllers, and does not require much resources, leaving space on the OBC for more important matters.

As stated before, the equation of the PID is

$$u(t) = K(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt})$$

Being it the sum of the three terms: proportional, integrative and derivative.

However, as every control architecture, also PIDs have to be configured correctly. Depending on the plant properties, some may need only 2 control actions: Almost 90% of industrial applications requires only a PI controller, while few situations a PD or a PID.

The configuration of a PID is generally done through trial and error tests, still following some guidelines. There are different tuning guides, while the most known are developed by Ziegler and Nichols: they are based on characterization of process dynamics by a few parameters and simple equations for the controller parameters.

The step response method is the first tuning method that is generally tested. It is based on process information in the form of the open-loop step response obtained from a bump test [16]. Very simple process models are used. From the test, it is possible to obtain two parameters, α and L , from which the PID parameters are determined.

By testing, it is determined firstly the point where the slope of the step response has its maximum. Then a tangent at this point is drawn. The two parameters are obtained by intersecting the tangent with the coordinate axes.

<i>Controller</i>	<i>K</i>	<i>T_i</i>	<i>T_d</i>	<i>T_p</i>
P	1/a			4L
PI	0.9/a	3L		5.7L
PID	1.2/a	2L	L/2	3.4L

Figure 15: Ziegler-Nichols parameters [16]

Being this a very simple approach to a PID configuration, its results do not assure good performances, in particular in this case where the spacecraft model is nonlinear. There are many other methods that can, possibly, give values much more suited to the case. In the application described in this thesis work, the PID tuning started by following the Ziegler method, just to have an initial configuration and a general idea on the dimensions of the values to be used, but then diverted to trial and error procedure to find the best

fitting parameters. The final controller is effectively a set of 3 PIs, one per axis, involving so only two control actions.

2.6 Filtering

As stated in previous chapters, in real systems the feedback process is achieved through the use of sensors that read the effective states of the plant and return these data to the control loop, of which they are part. In most of real-world situations however, not all state variables are effectively measured. There may be different causes for this, like limited budget, since sensors can be expensive, or limited space in the CubeSat.

Even if sensors are expensive and reduce the already limited space in spacecrafts, in many situations they can be employed redundantly in number. It is a common practice vastly used for safety and fault tolerance reasons. Not so rarely, a series of fails can happen, which make the OBC turn off one or more components. The possibility to have a failed sensor leads to the necessity to implement spare ones or to introduce fault tolerance algorithms.

Redundancies are also necessary to guarantee noise free data. Sensors are never completely accurate, but every measurement is subject to an estimation error defined as

$$e = x - \tilde{x}$$

Where \tilde{x} is the estimation of the component and x is the real value.

The error values over time can be generally modeled by a Gaussian distribution and are subject of study for every type of estimation filter. The use of one or more redundant sensors allows to avoid the use of an algorithm by comparing two measurements and so mitigating the effects of noise by exploiting mathematical methods as Mean Squared Error or simply by calculating the average value. This process is called data fusion.

LICIACube, however, is not equipped with redundancies in the sensors field. As stated before, it hosts an IMU and a star tracker, along with two sun sensors that are generally always present at least in a pair, given their nature. This absence of further sensoristic components is a common practice in CubeSats, where the limited budget and the relatively small risk of the missions to be accomplished justify the possibility of a failure. It is a trade-off between risk and costs. This means that LICIACube can only rely on an estimation algorithm or a software application that can actively reduce the noise in the measurements.

Observers, filters and estimators are the most used software components, which primary purpose is to reduce noise, since redundancy is not achievable by software.

The developing of nonlinear control theory brought to the appearance of many different observers, of which the Kalman Filter is the most used.

In this section, LICIACube ADCS noise reduction mechanism, that relies on an Extended Kalman Filter (EKF), is discussed.

In statistics and control theory, the Extended Kalman Filter, which is the nonlinear version of the Kalman filter, is an algorithm that produces estimations of unknown variables by using a series of measurements with random noise and disturbances, taken over time. The particularity of these estimations is that they tend to be more accurate of the real measurements by estimating a joint probability distribution over the variables for all the time steps. It is a recursive, two step estimation process:

- In the **prediction step** the filter estimates the current state variables, along with their uncertainties. This is done thanks to the copy of the system model, often simplified, placed inside the filter.
- During the **correction step** the algorithm takes the outcome of the next measurement and updates the estimation of the states through a weighted average. The more an estimation is certain, the more weight it has in the average.

The formulation of the problem is done in discrete time, suiting perfectly the case study.

Given a discrete-time nonlinear system in the form

$$\begin{aligned}x_{k+1} &= f(x_k, u_k) + d_k \\ y_k &= h(x_k) + d_k^y\end{aligned}$$

Where $x_k \rightarrow$ State of the system at instant k

$u_k \rightarrow$ System input at instant k

$y_k \rightarrow$ System output at instant k

$d_k, d_k^y \rightarrow$ Disturbance and measurement noise on the output.

The aim of the filter is to make an estimation of the state of the plant \hat{x}_k , when the state x_k along with d_k, d_k^y is not measured, by starting from the current and past measurement of y_k and u_k

The process begins with the prediction step. The filter computes a preliminary prediction x_k^p of the state x_k following the equation

$$x_k^p = f(\hat{x}_{k-1}, u_{k-1})$$

This prediction is then corrected and improved during the correction step through the formula

$$\hat{x}_k = x_k^p + K_k \Delta y_k$$

Where $\Delta y_k = y_k - h(x_k^p)$ and K_k is a weight matrix that multiplies the error on the output. In the second step, in fact, the output measurement is compared with the output calculated by the filter starting from the prediction of the state. The higher the error, the more the prediction is adjusted. This second step is the crucial one, increasing the estimation accuracy and enhancing the filter stability properties.[14]

K_k derives from two other matrices, F_k and H_k , that are obtained by linearizing the system along certain trajectories. In particular

$$F_k = \frac{\partial f}{\partial x}(x_k, u_k)$$

Is the Jacobian of the function f computed in (x_k, u_k) , and

$$H_k = \frac{\partial h}{\partial x}(x_k)$$

Is the Jacobian of h computed in x_k .

An Extended Kalman Filter can be designed to suit different scenarios, where disturbances can vary. The design is performed through the setting of two matrices, Q^d and R^d . They are diagonal matrices which values are linked to the variance of the disturbance and measurement noise.

$$Q^d = E[d_k d_k^T]$$

$$R^d = E[d_k^y (d_k^y)^T]$$

However, since it can be difficult to calculate the effective variance of these disturbances, the two matrices are tuned by a trial and error method.

The last step is the definition of the matrix P_k . It is defined as

$$P_k = E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T]$$

It is linked with the covariance of $x_k - \hat{x}_k$. As stated before, the EKF takes the actual and previous measurements to make an estimation of the state, that gets better the more are

the time steps it runs. The possibility to improve the estimation of the next steps is given by the matrix P_k , that gets more accurate at each iteration. It is generally initialized as an identity matrix, since there is no info on the covariance of $x_k - \hat{x}_k$, while in the successive steps is calculated with the formula

$$P_k^p = F_{k-1}P_{k-1}F_{k-1}^T + Q^d$$

Along with matrix P_k^p , during the initialization phase it is necessary to set also the initial estimated state \hat{x}_0 , generally as 0.

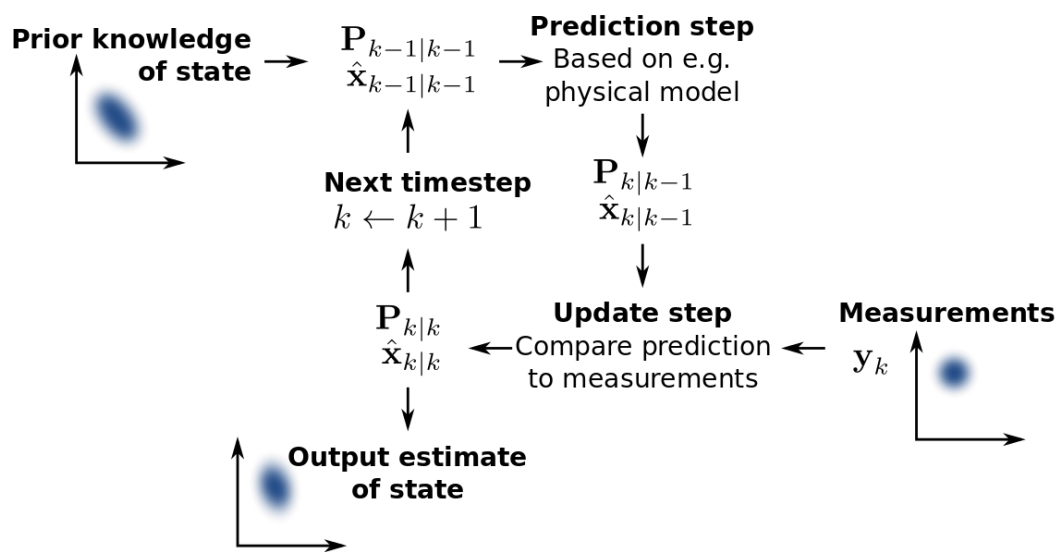


Figure 16: Extended Kalman Filter concepts (credit to Wikipedia)

Although the EKF grants a relatively simple, well working estimator for nonlinear systems, differently from its linear version it is not optimal. Extended Kalman Filter does not guarantee that the variance of the estimation error norm is minimized. Moreover, not even stability is theoretically guaranteed, since its performance strictly depends on uncertainties and disturbances on the initial condition. However, many different works host an EKF with great results, proving that for aerospace systems it is generally effective.

Chapter 3

Matlab and Simulink simulations

In this chapter the Fully Operational mode, or Regime mode, and Detumbling mode will be thoroughly analyzed.

Regime mode has to provide all the performances necessary for the success of the mission, while Detumbling mode needs to be simple enough to not occupy too many resources on the limited space of the OBC.

Before showing the choices for their developing, it is necessary to take a look at Simulink solver parameters, that are presented at the beginning of the chapter.

At the end, instead, some simulations with real mission data are shown.

3.1 Simulink solver

As almost all control systems for real world applications to be run on hardware, also in this case the control system, which is digital, must be discrete time. This means that the simulation is not continuous, but the system evolution is simulated and studied at fixed time steps.

The primary problem is to find the correct duration of the time steps of the simulation, which is bound to the concept of minor and major cycle.

Normally, in software applications with no regulation, interval between each simulation step, or step size, is not always known a-priori. It depends on the calculations made on each cycle. If an upper bound is not fixed, each time step may have different duration. This is due to hardware computational velocity not infinite, strictly depending on the situation. Moreover, hardware offered performances are not always at the peak of technology, especially for satellites, where on board computers have limited memory. This results in time steps that can be very variable, with loss of accuracy in real-time applications.

All this must be considered during the developing of any of the control modes for LICIAcube: the solver must be set as fixed time-step.

To decide which value for the time step is adequate, it is necessary to know all component operating frequency, by looking at the data sheet of the CubeSat. The fastest components are sensors, which operate at 250hz, the slowest are the Reaction Wheels, 5hz.

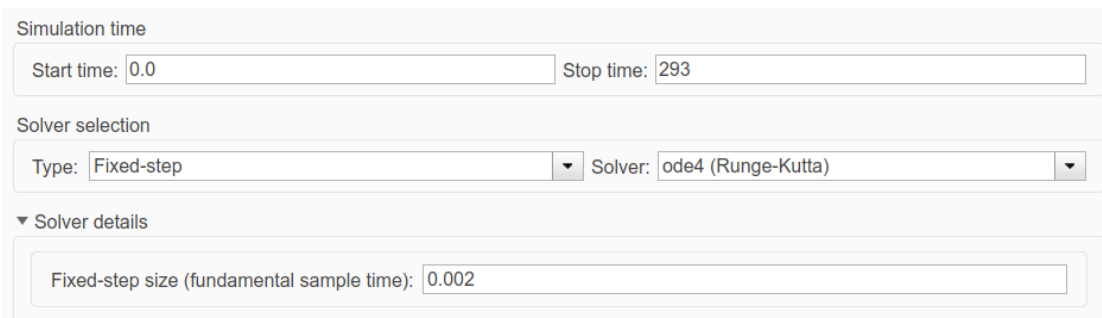
For this reason, the major cycle corresponds to 0.2 seconds, while the minor cycle to 0.004 seconds.

However, for safety reason, Simulink time step is set to 0.002 seconds. This means that every two simulation steps, one does not involve an effective evolution of the system.

The different operating frequencies of the blocks can be implemented by defining the blocks as “atomic units”. Simulink recognizes them as singular entities that can operate and return an output only after the respective sample time.

To not be underestimated is the communication between two blocks that work at different frequency. If the operating frequencies are multiple, Simulink can handle alone the situation by holding the last value obtained as output from the slowest block.

In any other cases, an error will occur and explicit blocks that regulate the exchange of data will be needed. They will be used later.



The image shows the Solver settings dialog box in Simulink. It is divided into three sections: 'Simulation time', 'Solver selection', and 'Solver details'. In the 'Simulation time' section, the 'Start time' is set to 0.0 and the 'Stop time' is set to 293. In the 'Solver selection' section, the 'Type' is set to 'Fixed-step' and the 'Solver' is set to 'ode4 (Runge-Kutta)'. In the 'Solver details' section, the 'Fixed-step size (fundamental sample time)' is set to 0.002.

Figure 17: Solver settings

3.2 Dynamics, kinematics and main actuator

The dynamics, kinematics and the main actuator block have been already discussed in chapter 2 in their general form. In this section some interfacing and secondary aspects are analyzed. They hold for all operational modes.

Right after the Controller block, the communication with the Actuator block is regulated by a forced delay. The same is done at the end of the control loop, when the actual quaternion is read by the sensors and returned to the Reference Generation block.

They are not necessary for the software simulation. However, tests on hardware revealed an inevitable delay of one major cycle in the first case, and 4 major cycles in the second. While the first one has been validated, the delay on the sensors reading is still under validation. Seeming this odd, also there only a unitary delay is introduced.

Reaction Wheels have been implemented as atomic unit blocks of 5hz sampling rate. Dynamics and Kinematics block have been set as the fastest blocks along with the sensors,

at 250hz. Being the last two blocks that imitates physical effects, there would have been no sense in making them slower.

3.3 Fully operational mode

In the following, the first operative mode of the developed ADCS is discussed. Fully Operational mode is the nominal working condition of LICIACube. It must provide the highest control precision possible to the success of the mission. Before showing the simulations, the exclusive blocks of this mode are presented.

3.3.1 Reference Generation block

As shown in a previous chapter, the control law implemented requires as inputs many variables that depends not only on the attitude of the spacecraft, but also on its speed and acceleration. These variables, however, are not immediate to find and feed to the SMC.

LICIACube optical payload returns a series of quaternion as reference signal for the control system to follow. No indications are given to its angular speed or acceleration. It is so necessary to derive these informations starting from the info given by sensors.

To derive this data is the purpose of the Reference Generation block, the first of the control loop. It is essentially composed by three elements, where the second is the crucial one.

The first part deals with the recovery of the reference quaternions. In the control system, the optical payload is not simulated, thus the quaternions cannot be received in real-time. For the ADCS simulation purposes, an already generated possible fly-by trajectory is used as reference signal: it is a set of quaternions on an Excel file that are taken through a Simulink functionality with a frequency of 5hz. The trajectory is, so, not known a priori from the system, simulating realistically the optical payload.

The generated reference signal is so ready to be handled in order to obtain all the data needed by the controller, that happens in the second block.

Firstly, each quaternion taken is converted in the respective Euler angles, in order to have a more understandable idea of the effective rotation of the spacecraft. The conversion passes initially through the Direct Cosine Matrix

$$T = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

And then from DCM to 321 Euler Angles with the formulas

$$\begin{aligned}\phi &= \text{atan2}(T_{32}, T_{33}) \\ \theta &= \text{atan2}(-T_{31}, \sin\phi T_{32} + \cos\phi T_{33}) \\ \psi &= \text{atan2}(-\cos\phi T_{12} + \sin\phi T_{13}, \cos\phi T_{22} - \sin\phi T_{23})\end{aligned}$$

With

$\phi \rightarrow$ Roll

$\theta \rightarrow$ Pitch

$\psi \rightarrow$ Yaw

The use of these formulas introduces an issue. When a rotation of π radians is done, due to Matlab displaying convention, the angles described pass from π to $-\pi$. This sudden change of sign is not a problem alone, but it becomes one when the angles are derived in order to obtain the angular velocity. The discontinuity generated, in fact, drives the derivative to huge, incorrect, values.

To resolve the issue, a counter is added. Every time the spacecraft rotates on any axis more than π , the counter goes up (or down) and the corresponding angle is added of "counter" times π .

This observation allows to safely obtain the time-derivative of the angles. It is then possible to obtain the angular velocity of the spacecraft through the Tait-Bryan 321 kinematics equations. In particular

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} * \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

Being calculated by starting from the reference quaternions, this angular velocity so obtained can be considered as the reference angular velocity ω_r , needed for the Sliding Mode controller. In the end, by performing its time-derivative, it is possible to obtain also the reference angular acceleration $\dot{\omega}_r$.

In the final part of the subsystem the errors signals are calculated. The reference signals are compared with the data from sensors, thus completing the data set for the controller. Since the optical payload returns an attitude quaternion at a pace of 5 samples per second, the Reference Generation block has been set as an atomic unit on Simulink with a sampling rate of 0.2 seconds.

3.3.2 Extended Kalman Filter

A filtering action in the Fully Operational mode is particularly important, since it is necessary to reduce the possible noise coming from the sensors lecture. The presence of disturbances can, in fact, stack up and drive the spacecraft off its trajectory, or can make the Reaction Wheel accumulate momentum, inducing a premature saturation condition. The Extended Kalman Filter implemented to eliminate this possibility follows the theory presented in chapter 2.6. The equations are implemented in a subsystem of the model, where the estimated state variable vector is $\hat{x} = [\hat{q}_0 \ \hat{q}_1 \ \hat{q}_2 \ \hat{q}_3 \ \hat{\omega}_1 \ \hat{\omega}_2 \ \hat{\omega}_3]$. Moreover, the estimation algorithm requires a copy of the plant model, that can be simplified respect to the original. However, the more it is simplified, the less the accuracy of the estimation.

In this case the model is not changed, instead also the effect of the Reaction Wheels has been modeled and considered in the filter.

The information that are fed to the EKF come from both the Star Tracker and the IMU. This is a risky but worthy choice. Normally, in order to estimate the states of the plant, the EKF would need only the readings of a single sensor. Through the use of dynamic and kinematic equations, in fact, it would have been possible to derive the angular velocity from quaternions or viceversa, estimating them in an indirect way, using as information only one of the two readings. The use of a double sensor allows to add robustness to the filter, since it is statistically almost impossible to have critical errors on both the Star Tracker and the IMU at the same time. However, a sensor come always with random errors in the measurement. By reading two of them, two disturbances come into play: the signals are more uncertain, and the filter needs to be designed more finely in order to not lose precision.

The readings of two different feedback signals is so, a sort of bet. More noise is added, but at the same time the system is more robust, since error on a single sensor can be overlooked. A poor tuning of the EKF makes this design choice not convenient for every possible situation.

Since in Simulink no sensors are effectively simulated, the random noise on each reading is implemented through specifics block that add random white noise on the signals fed to the filter.

The EKF block has been set as an atomic unit with a sample rate of 0.004s.

3.3.3 Simulation parameters 1

Prior to show the images of the simulation results, initial condition and tuning values are here reported, with some considerations.

Simulink block	Parameter	Value	Sample time
Sliding mode controller	k_1	0.02	0.004 seconds (250 Hz)
	k_2	2	
	η	100	
Extended Kalman Filter	Q_d	$0.01^* \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix}$	0.004 seconds (250 Hz)
	R_d	$200^* \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix}$	

The parameter η was set at 100 since it needed to resemble enough the signum function, being it a sigmoid function as specified in chapter 2.4. K_1, K_2, Q_d, R_d were, instead, chosen by following a trial and error procedure. While there is no particular physical consideration about K_1 and K_2 , the last two terms are similar to weight matrices for considering the variance of the noises. Simply put, in these simulations the noise on the state is considered almost null, while the noise on the sensors reading is taken as the real problem. As reference signals for the system, the Excel file provides every 0.2 seconds a new quaternion of the fly-by phase

Lastly, as initial condition for the testing, the following parameters have been chosen

Initial quaternion	$[-0.487859; -0.793255; -0.059954; -0.359368]$
Initial satellite angular velocity	$[0;0;0]$
Initial Reaction Wheel angular velocity	$[0;0;0]$
Initial value of EKF covariance matrix P	$0.01^* \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix}$

The initial quaternion is the first quaternion of the fly-by series. However, also results with initial attitude different from the reference ones are displayed. The initial value of the matrix P indicates that the first estimation of the states given by the sensors is pretty accurate.

3.3.4 Simulation results 1

The first image shown is about the attitude of the spacecraft, described in quaternion. Both reference and assumed quaternions are displayed: however, since only four lines are visible, it is easy to understand that the respective components coincide macroscopically.

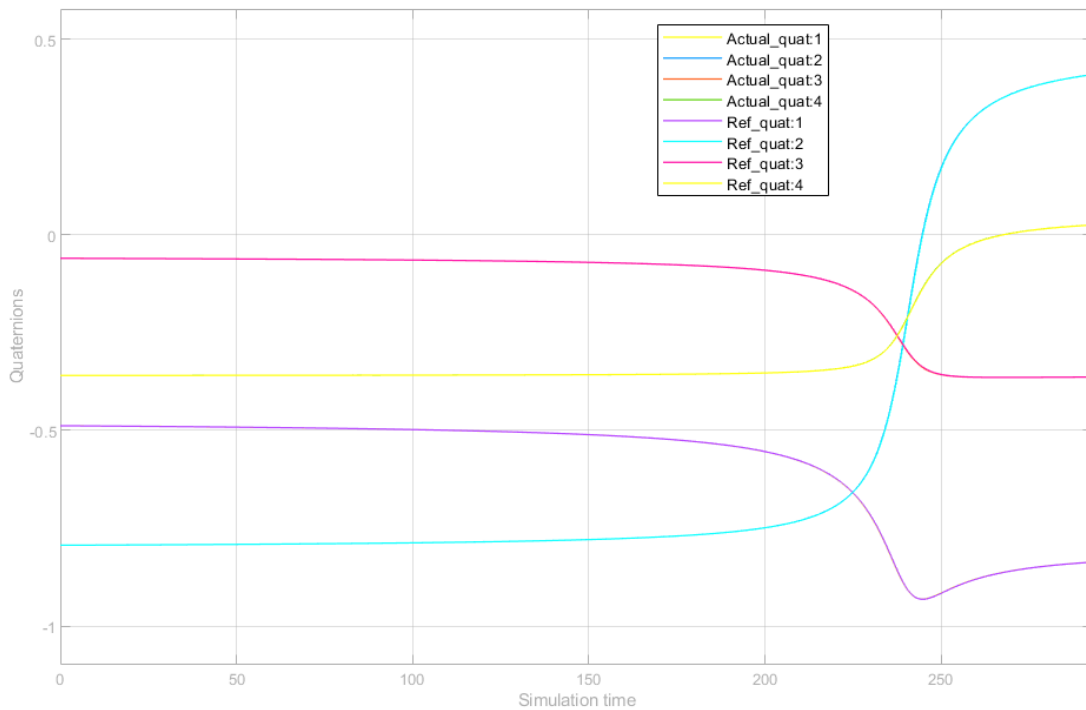


Figure 18: Spacecraft attitude in quaternions

The second image is to have a better understanding of the precision grade of the control. It is a zoom of the spacecraft assumed attitude at the moment of the fly-by. The real component of the two quaternions differ for a maximum of less than 1/1000.

Chapter 3: Matlab and Simulink simulations

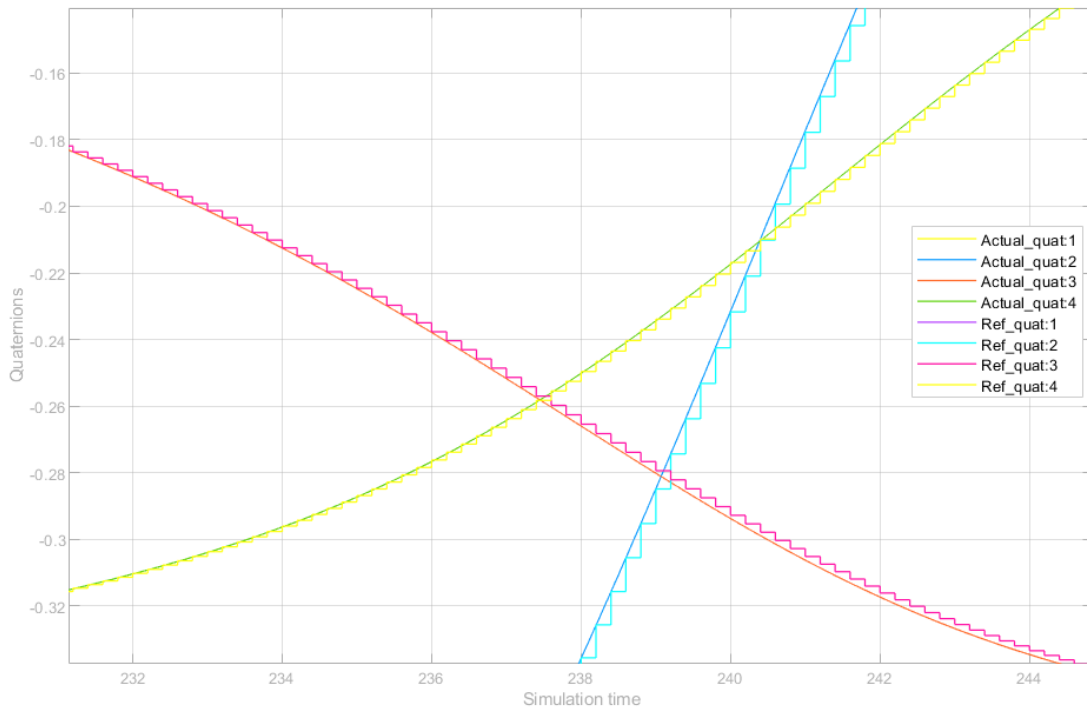


Figure 19: Zoom on spacecraft attitude

The same is done with angular velocity.

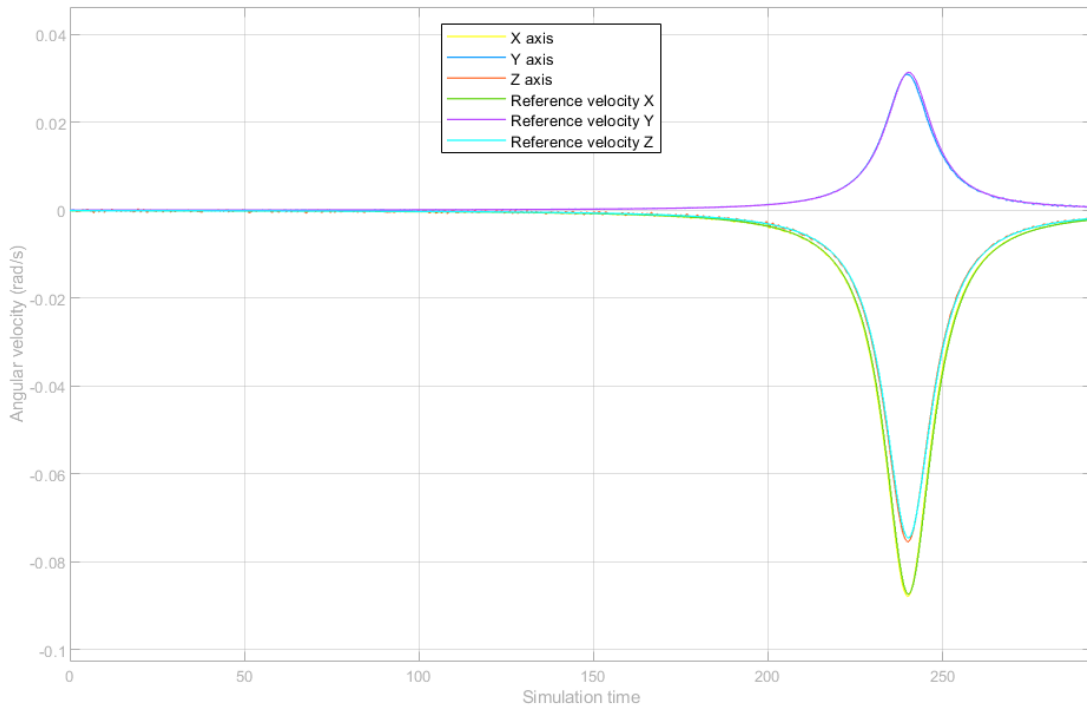


Figure 20: Spacecraft angular velocity

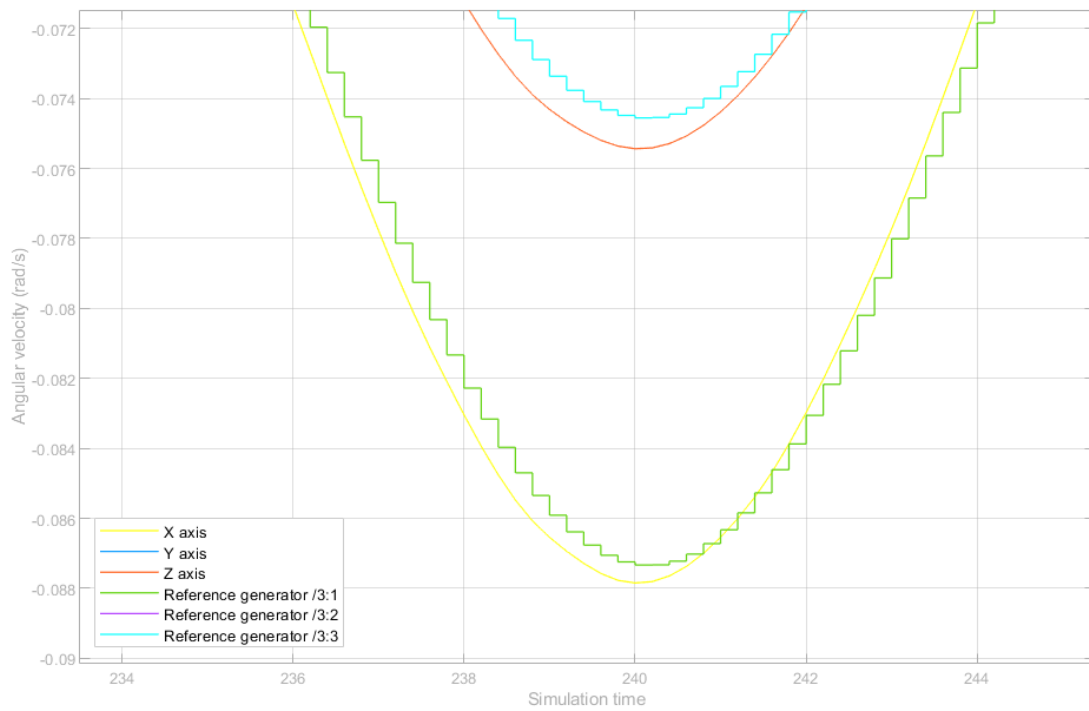


Figure 21: Zoom on spacecraft angular velocity

The difference in angular velocity respect to the reference signal can be translated as a pointing error of less than 1° . Given the FoV of the optical payload, Dydimos is never lost. Here is reported the filtering effect of the EKF. Noise is almost entirely rejected in quaternion estimation. A worse result is obtained in the angular velocity estimation.

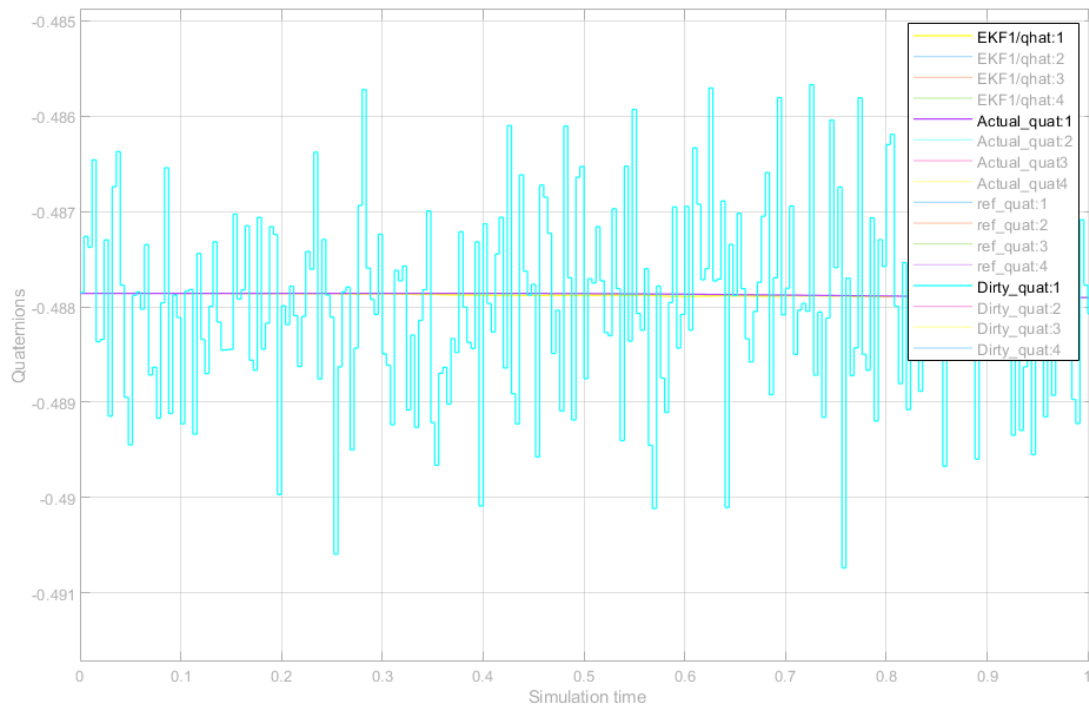


Figure 22: Extended Kalman filter on quaternions

Chapter 3: Matlab and Simulink simulations

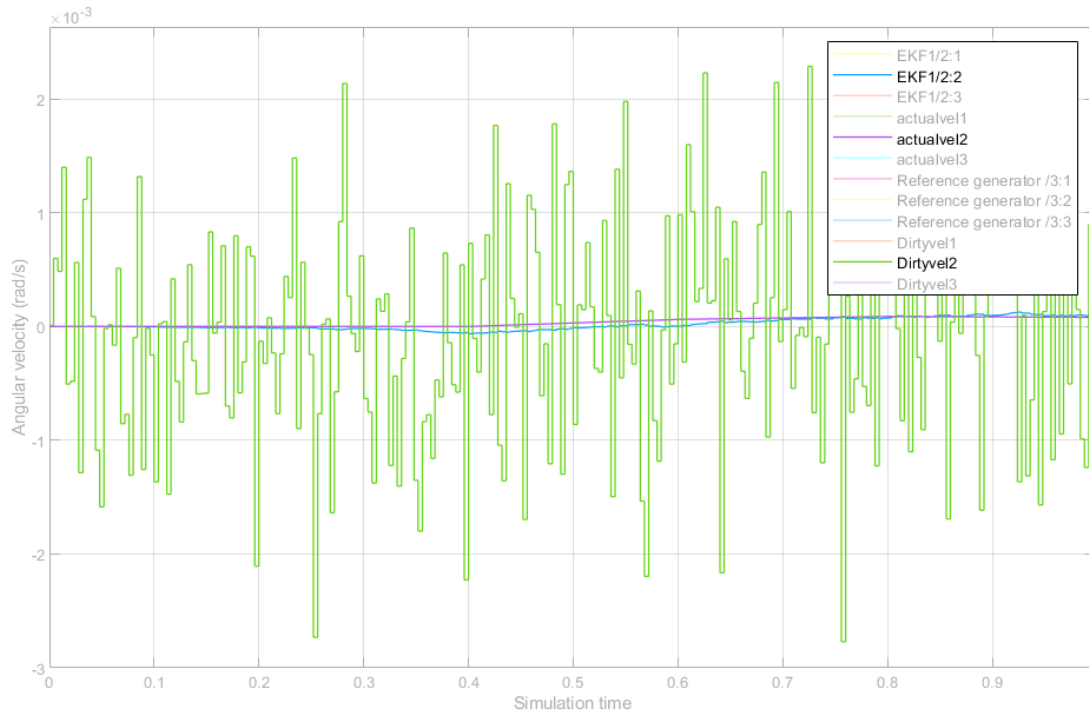


Figure 23: Extended Kalman filter on angular velocity

The torque expressed by the Reaction Wheel is displayed in the image below. Saturation conditions are very far, since the maximum possible torque is four times the actual one. Worth to note that the Y axis has been normalized for privacy purpose.

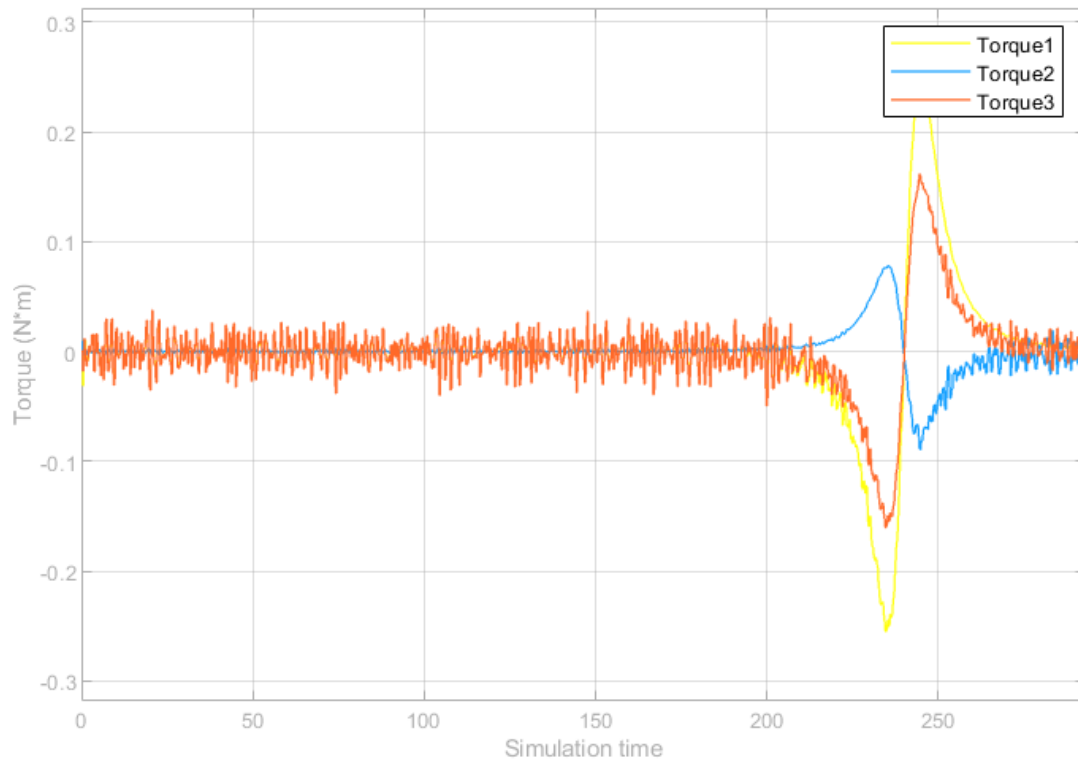


Figure 24: Reaction Wheels expressed torque

Chapter 3: Matlab and Simulink simulations

In the end, another simulation is done with initial quaternion different from the reference one. This situation should never happen if the optical payload and sensors work with no errors.

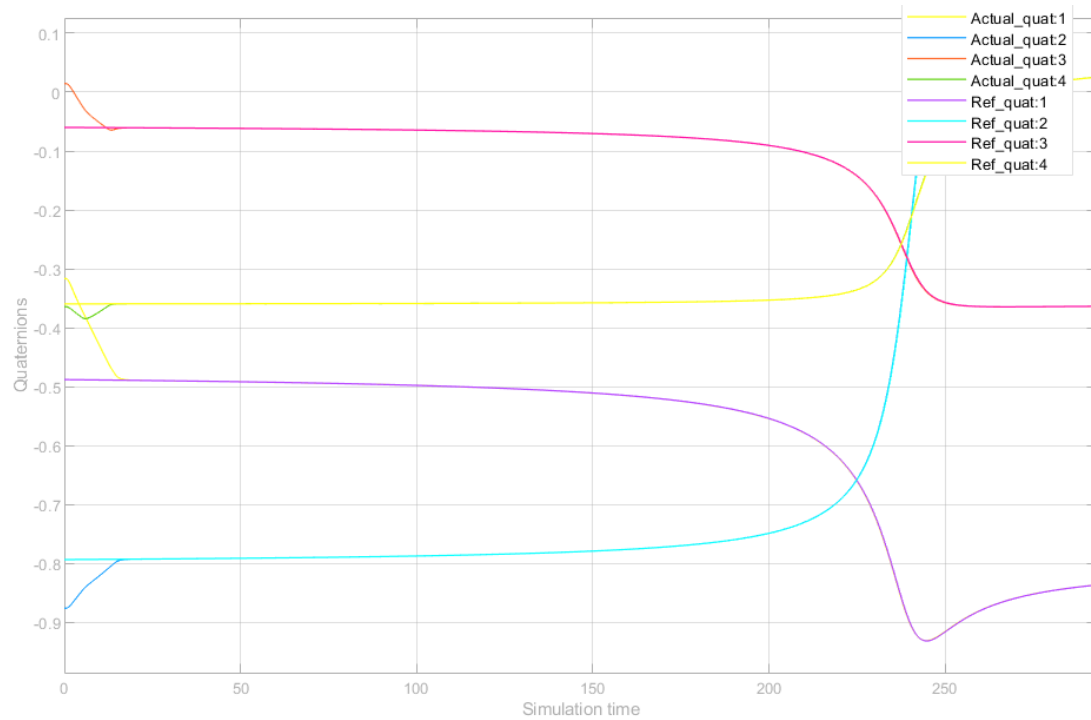


Figure 25: Spacecraft attitude with initial error

It can be easily seen that, even with a starting condition different from the normal one, the Sliding Mode controller can reach the reference signal in few seconds. The robustness provided can, in fact, deal with this error.

3.4 Detumbling mode

Detumbling mode appears as a much-simplified Regime mode. Kinematic block is no more needed, since there is no control on quaternions. Only the angular velocity of the spacecraft matters, so all useless subsystems have been discarded. Actuator and Dynamic blocks are, instead, identical to the previous control system.

The reference signal is a constant value that indicates the desired residual angular velocity that the spacecraft must assume in order to consider the detumbling phase as ended. Arrived at the desired value, the ADCS should switch operating mode, never returning to this one.

The controller, which is a set of 3 PIDs, one per axis, is implemented through PID Simulink blocks. They receive as input the estimated angular velocity of the satellite and return as output a controller torque request, sent to the actuator block. Their tuning started by making consideration by looking at the order of the typical values accepted as torque request and the order of the angular velocity of the satellites. Then, it continued by using Ziegler-Nichols method as guide and ended with some trial and error to determine the best values. Attention was placed in particular on the necessity to avoid the request of a torque higher than the saturation limit of the Reaction Wheels.

The Extended Kalman Filter is a simplified version of the previous one: since the state vector of the plant is now only $x = [\omega_1 \ \omega_2 \ \omega_3]$, all the matrices and equations of the filter are smaller. Inside the block the model implemented takes into account the Reaction Wheels effect. Moreover, input signals arrive only from the IMU, since the data fusion with the Star Tracker does not provide any useful information; instead it introduces only more noise on the measurements. For simulation purposes only, noise is simulated through a specific block in order to make the output measurements “dirty”.

Almost all blocks are set as atomic unit with a sample rate of 0.004 seconds. The only exception is still the Reaction Wheels, which are at 0.2 seconds as always.

3.4.1 Simulation parameters 2

As for the Fully Operational mode, before showing the results of the simulation, also here the tuning values and initial conditions are showed.

Simulink block	Parameters	Values	Sample time
Reference signal		[0.01;0.01;0.01] rad/s	Continuous time
PID X axis	P-I	[0.1;0.001]	0.004s (250hz)
PID Y axis	P-I	[0.013;0.00025]	0.004s (250hz)
PID Z axis	P-I	[0.01;0.001]	0.004s (250hz)
Extended Kalman Filter	Q_d	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	0.004s (250hz)
	R_d	$2000 * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	

The derivative action in the controller was not needed. The nonlinearity of the plant made the use of a derivative function critical, since not controllable oscillations were inducted. The parameters of the filter are chosen by trial and error.

For the initial condition, two cases were considered. Firstly, the conditions of the worst-case scenario were considered. Data mission reports a maximum of 0.1 rad/s for a single axis after the detach from DART. For simulations purposes, even if it is not possible, the maximum angular velocity has been extended on every axes.

In the second simulation, however, this scenario was further worsened to assure the good working of the control system, increasing the velocity to 0.24 rad/s.

Initial spacecraft angular velocity	[0.1;0.1;0.1] rad/s
	[0.24;0.24;0.24] rad/s
Initial Reaction Wheel angular velocity	[0;0;0]

Initial value of EKF covariance matrix P	$0.1^* \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
---	---

3.4.2 Simulation results 2

Firstly, the Angular Velocity of the spacecraft is shown, with a 0.1 rad/s initial condition and then with 0.24 rad/s.

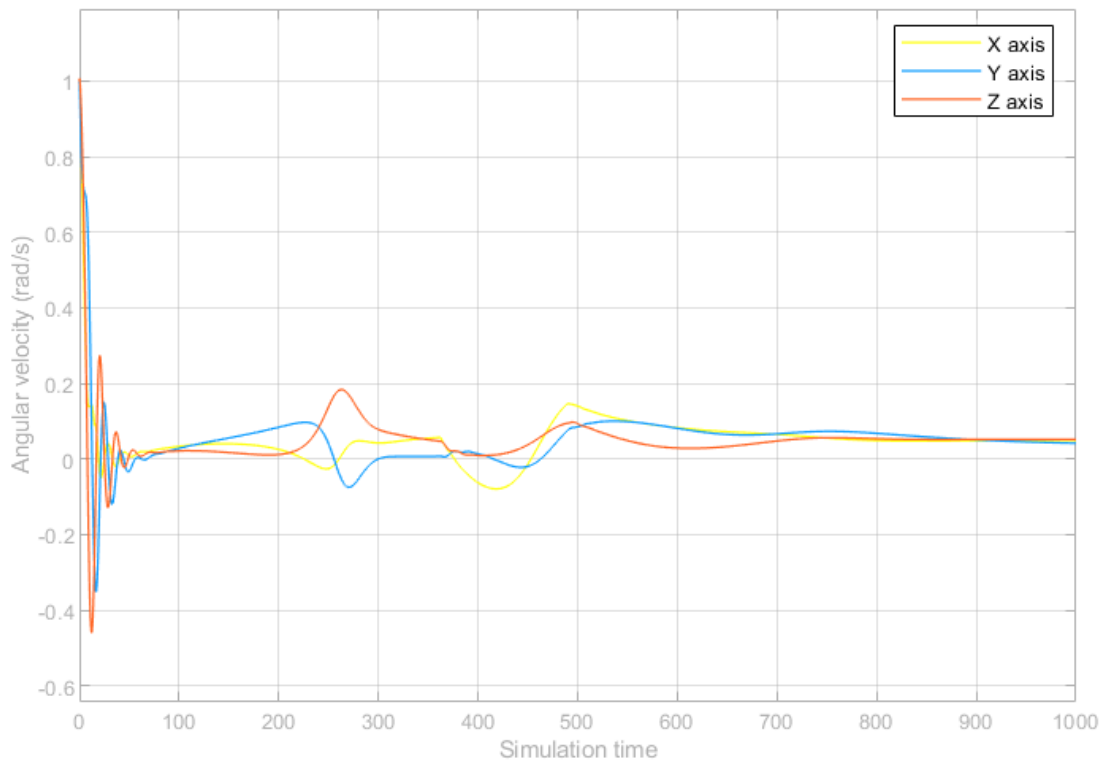


Figure 26: Spacecraft angular velocity at 0.24 rad/s

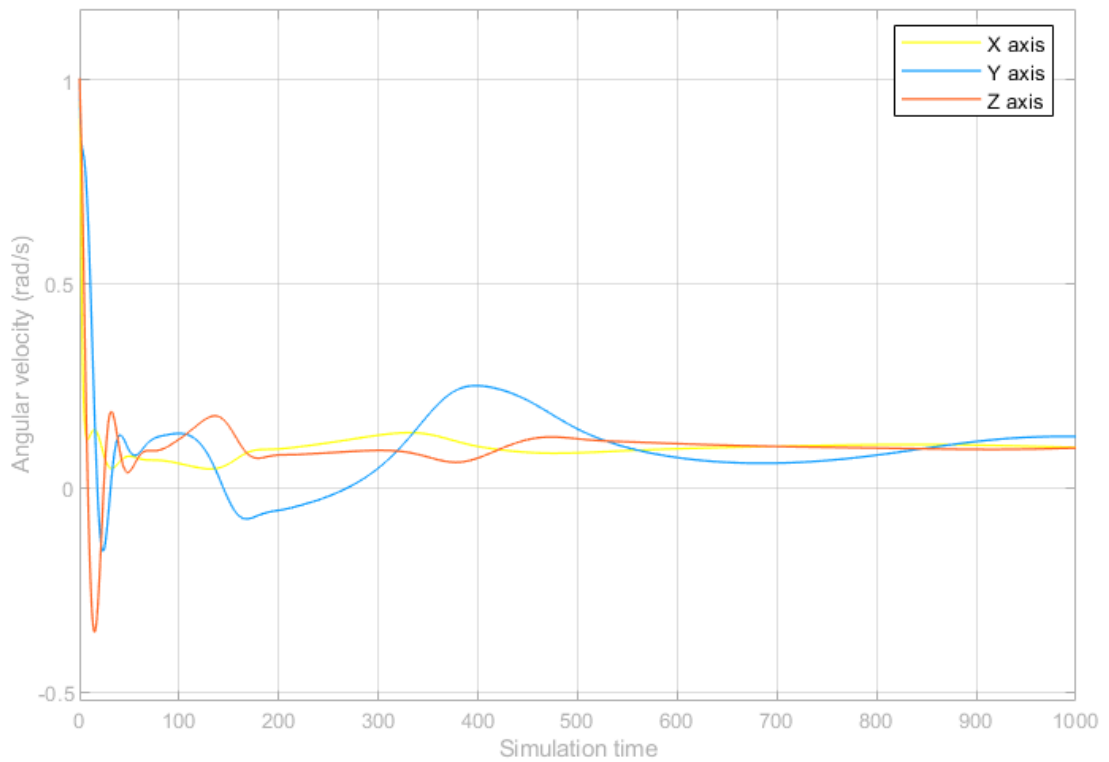


Figure 27: Spacecraft angular velocity at 0.1 rad/s

In both cases, the spacecraft can be considered as safely slowed down to the desired value after 10 minutes, even if after 100 seconds it assumes a good enough behavior.

In the end, it is possible to see the state of the Reaction Wheels during the detumbling. Here they are surely more strained than in the previous operational mode, but saturation conditions are not reached. In the images are shown the torques expressed at 0.24 rad/s and 0.1 rad/s initial conditions

Chapter 3: Matlab and Simulink simulations

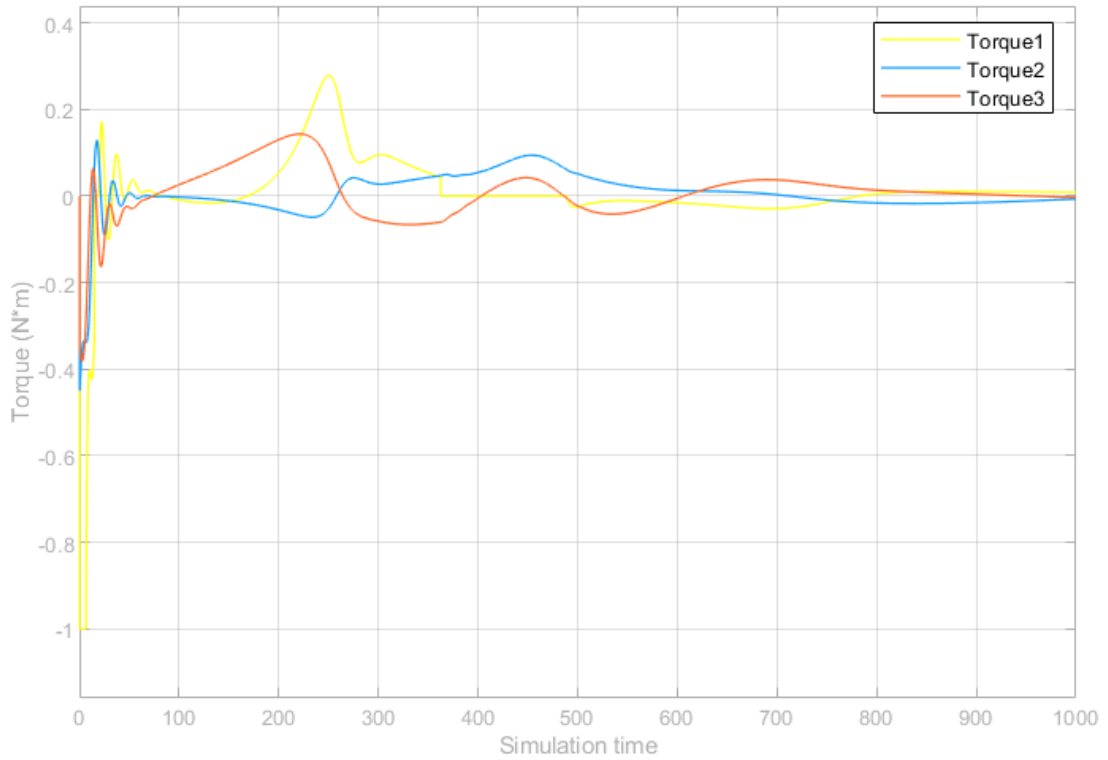


Figure 29: Normalized Reaction Wheel expressed torque at 0.24 rad/s

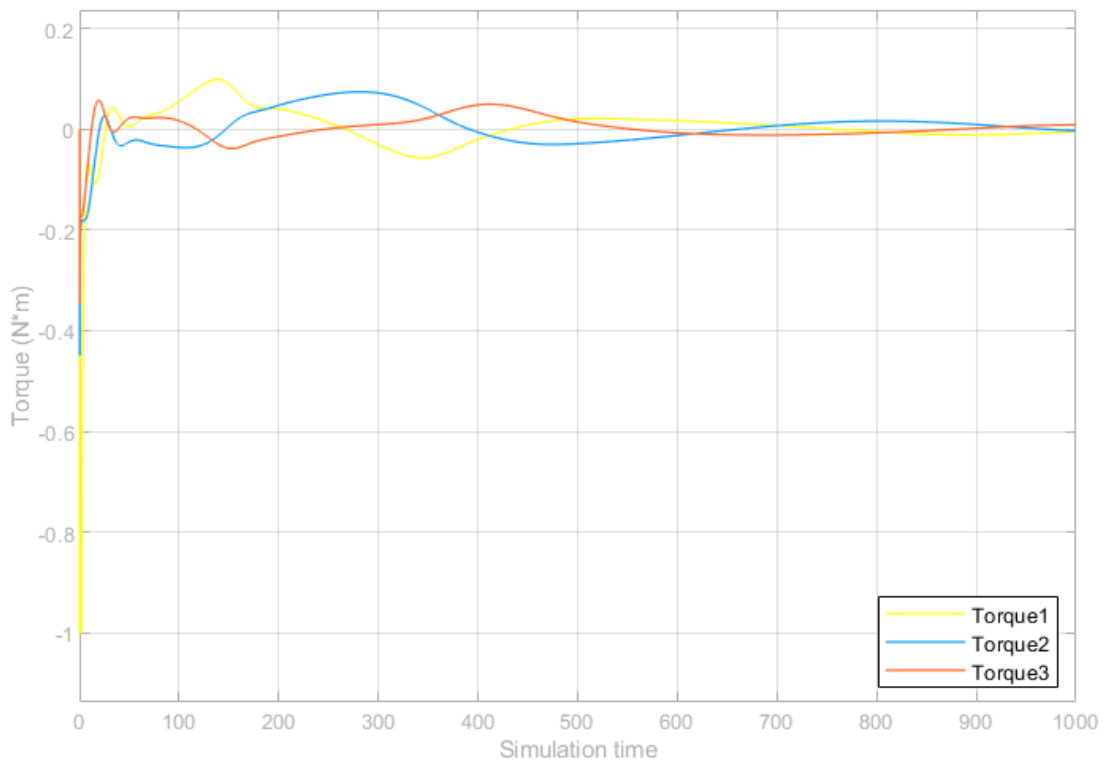


Figure 28: Normalized Reaction Wheel expressed torque at 0.1 rad/s

Chapter 3: Matlab and Simulink simulations

It is possible to note, however, that during the first seconds of simulations, the Reaction Wheels have to produce the highest torque possible. This is normal, since they have to respond as fast as possible to such a high angular velocity.

Chapter 4

Desaturation mode

The main problem of a Momentum-exchange actuator such as Reaction Wheels is the fact that the total angular momentum on the entire spacecraft does not change. It can be redistributed almost at will between actuator and body, but there is no total variation from external causes. Normally, this would not be a problem: in a complete isolated case, where no disturbances and no forces act on the CubeSat, the only side effect of this choice could be the impossibility to reach certain attitudes if the Reaction Wheel are chosen poorly, which does never happen in a studied space mission, where hardware units are tested over and over.

However, open space is not free of external forces. CubeSats can be affected by gravity gradients, solar wind and other secondary disturbances that tend to change the spacecraft attitude. These effects are sometimes considered when developing an ADCS, sometimes not. In fact, they are so little that a robust controller can simply ignore them, by correcting the orientation of the satellite when the error stacks up a little.

No matter how they are dealt with, these disturbances are to be considered external forces and, so, affect the overall angular momentum of the body, in particular if they can act for a lot of time without being taken into account.

The necessity to maintain always a certain attitude even if external forces exercise a little torque on the body makes the Reaction Wheels accelerate. The goal is, in fact, to nullify any possible unwanted acceleration.

When the disturb effect is nullified, however, the wheels, that have accelerated in order to produce torque, can not be slowed down. Slowing down would mean exercise a torque in the opposite sense respect to the previous, changing again the orientation of the CubeSat. They have, so, accumulated momentum: too much can limit the actuators, since they cannot accelerate anymore due to their limit on angular speed, and thus cannot produce torque.

For mission purposes it is necessary to maintain the satellite always in an almost nominal working condition. It is possible to restore the Reaction Wheel initial condition through the use of another, Reaction-type, actuator. This phase is called desaturation phase.

Chapter 4: Desaturation mode

As stated in the introduction, LICIACube hosts two actuators. The main one is represented by a set of three Reaction Wheels. The secondary one is a set of thrusters, that belongs to the Propulsion System (PS). The purpose of the thruster is to make the navigation of the CubeSat possible and to desaturate the main actuator, when it is needed. The latter is possible only by making the two actuators work in a synchronous mode: the wheels have to return to their initial condition, moving all the accumulated momentum on the body of the spacecraft and so inducing a torque in a certain verse; the thrusters need to be switched on and produce a torque equal and opposite to the one induced by the wheels. The result, in case of perfect torque matching, is that the LICIACube does not change its attitude, while all the stored momentum is discharged. In reality, a perfect desaturation without loss of pointing is really difficult to reach. LICIACube Thrusters and Reaction Wheels are not perfectly matched.

The difficulty to obtain such a result lies in the functioning mode of the thrusters. They work in a ON/OFF fashion, so it is not possible to regulate proportionally their thrust. A single thrust of few milliseconds can generate a torque much higher than the one produced by Reaction Wheels. By using both the actuators at the same time, the resultant torque acting on the spacecraft would make it rotate in favor of the thrusters.

This problem can be resolved through Pulse Width Modulation control. "By commanding thrusters torque pulses with an appropriate width, whereas width is intended a certain duty cycle T_t , the average torque exerted throughout the sample period is equal to the average commanded control torque" [10]. However, also PWM has its own limitation. It can approximate a continuously variable control torque, but there can be problem when the control torque becomes very small. This can happen when a small correction in the attitude is required. Turning ON and OFF a thruster requires time. They are not capable of creating pulses shorter than a certain limit, called Minimum Torque Impulse Bit (MTIB). This value indicates the minimum amount of time that a thruster can be turned on for. To avoid this situation, every control system that deals with this type of actuator takes into account the MTIB and incorporates a dead zone: it is a value that determines the level of control error from where thrusters can be turned on. That is, if there is an attitude error, until it becomes bigger than the dead zone value, thrusters are not used, because otherwise they would have been activated for a time minor of MTIB. It is obvious to note that MTIB and dead zone are directly related.

Chapter 4: Desaturation mode

The presence of dead zone makes clear that an attitude control using thrusters as main actuators cannot be very precise. LICIA Cube is the perfect example, since even if it hosts six thrusters, they are used almost only for the navigation, while attitude control is their secondary task, starting only when the OBC sends an alarm signal for saturation beginning.

Of these six thrusters, two of them are axial, used exclusively for navigation purposes. The other four are used for attitude control. Their displacement makes very easy the management for attitude control, allowing a complete control although standard configurations require six fully operational thrusters.

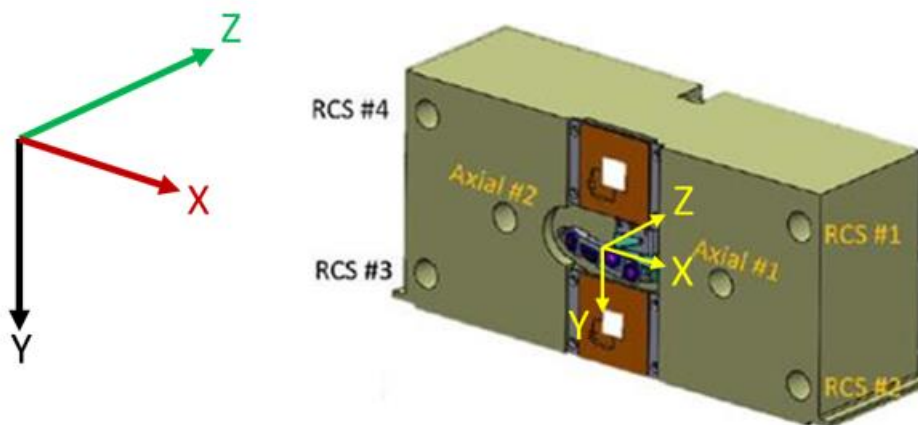


Figure 30: Thruster displacement

The four attitude control thrusters are positioned at the corner of the rear surface. Each of them is inclined of 30° respect to the XY surface and of 45° - 135° - 225° - 315° respect to YZ surface. This allows to generate torque on the spacecraft on every axis by turning on only two thrusters at time. However, the inclination on the XY surface brings an unavoidable translation along the Z axis. This is, however, not a problem, since it is not expected to have a desaturating phase during the fly-by.

In this thesis, however, desaturation has been a great problem to handle. PWM is not easy to implement and the results were not optimal. The communication between Reaction Wheels and thrusters did not give good results. The major problem was the difference in reference systems between the actuators: having received the RW block from Argotec without a real data sheet, guessing the correct reference system took a lot of time. For this reason, the approach to the problem has followed a different route.

Being the discharging of the accumulated angular momentum the major problem, with loss of pointing as side effect that can be easily resolved by taking continuously track of the attitude and velocity assumed by the spacecraft, the control system to deal with desaturation has been divided in two parts.

In the first part, the **discharge step**, Reaction Wheels are directly commanded in order to slow down till a zero angular velocity.

In the second part, the **detumble step**, thrusters are turned on in order to reduce the angular velocity of LICIA Cube.

In the following, each step is analyzed, and simulations are presented.

4.1 Discharge step

The concept of the first step of the Desaturation Mode is very simple: having Reaction Wheels incapacitated to produce Torque in a direction because they are already spinning at an angular velocity near the saturation limit, it is necessary to slow them down. Ideally, their angular velocity should return to zero. This is possible by commanding directly the actuator, sending as torque request a value that would make the RWs spin to the direction opposite to the actual one.

The control system works on this guideline. It is an angular velocity control loop, where the initial condition is given by the effective velocity state of the Reaction Wheels, that is obtainable through the sensors already mounted in the actuator. There is still no presence of the thrusters, that will be turned on in the next step. At each iteration, depending on the wheels velocities, a torque is commanded directly to the actuator to slow them down. Obviously, slowing down, the actuators induce a torque on the spacecraft body, making it rotate. At the end of this desaturation step, the RWs are still, while the CubeSat is rotating at an angular velocity that is continuously calculated through the usual Dynamic and Kinematic blocks. It will be the subject of the second step.

The control loop starts from the lecture of the sensors inside the Reaction Wheels, then a sort of controller or logic must command the correct torque to the actuator block. However, there is no immediate mathematical relation between wheels angular velocity and torque required to return them to zero. The only possibility would be to take the RWs angular velocity, to derive them and then obtaining the respective torque by the formula

$$T_{RW} = I_{RW} * \dot{\omega}_{RW}$$

The problem is that to derive a signal is almost never a good idea. In case of quick variation, in fact, there could be high spikes in its derivative, leading to a numeric error.

To resolve the matter, a custom logic has been implemented as controller inside the loop. By taking inspiration from the actuator characteristic given by the company, the controller is a piecewise function that links reaction wheels velocities and requested torque. Taking as input the actual wheels angular speed, the logic block returns one of four possible torque requests, depending on the situation: the higher the momentum on the actuators, the higher is the torque requested, so that it is possible to reach a complete discharge faster. Near the zero velocity, the torque request is minimum. This allows to obtain a more precise result, since control on the Reaction Wheels is not perfect.

When the angular velocity is below a safe limit, typically 1 rad/s, the torque request is stopped, and the phase is ended. The parameters that define the piecewise function, that allows to better define the torque values ranges, can be modified at will depending on the quality of the control desired. The satellite is now constantly rotating, with no further acceleration.

All the other blocks of the control system are equal to the ones seen in the Regime Mode.

4.2 Simulation Parameters 3

The simulation of this control system is focused on monitoring the Reaction Wheels state by exercising the necessary torque to drive them to zero. The discharging of the angular momentum of all three wheels is done simultaneously, since they are independent from each other. The piecewise function can be tuned with three parameters that define the torque ranges. In the following simulation, the initial conditions chosen describe a situation where the satellite is in an ideal situation, with zero angular velocity and an attitude described by the identity quaternion. The piecewise function parameters are reported in the table.

	Angular speed range	Requested torque
Upper limit	≥ 300 rad/s	0.00549 N*m
Medium limit	$\geq 20 \cup \leq 300$ rad/s	0.0027 N*m

Lower limit	$\geq 1 \cup \leq 20$ rad/s	0.0005 N*m
Wheel discharged	≤ 1 rad/s	0 N*m

The simulations show a worst-case scenario, where all Reaction Wheels are in saturation. Their initial angular velocity is 800 rad/s each. Their behavior is, however, identical. The angular velocity assumed by the spacecraft follows.

4.3 Simulation results 3

In the first image, the Angular Velocity of the Reaction Wheels is shown. They are discharged all simultaneously and with the same torque commands. Since hardware-like they are equal, the image correctly shows that all three lines coincide. It is important to notice how the discharge goes slower the closer is the zero value due to the custom piecewise function.

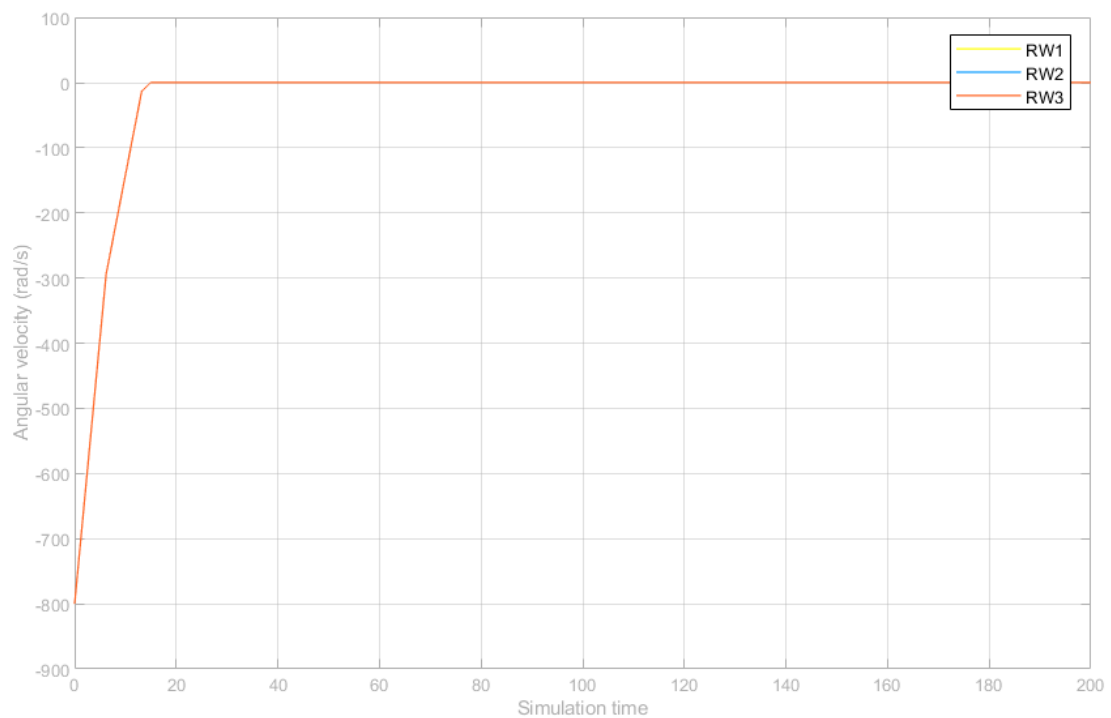


Figure 31: Reaction Wheel angular velocity in discharge phase

The effect of the discharge function can be seen in a much easier way by looking at the torque expressed. As stated before, four level of torque are possible. They are displayed in the image below. For privacy purposes, the Y axis is normalized.

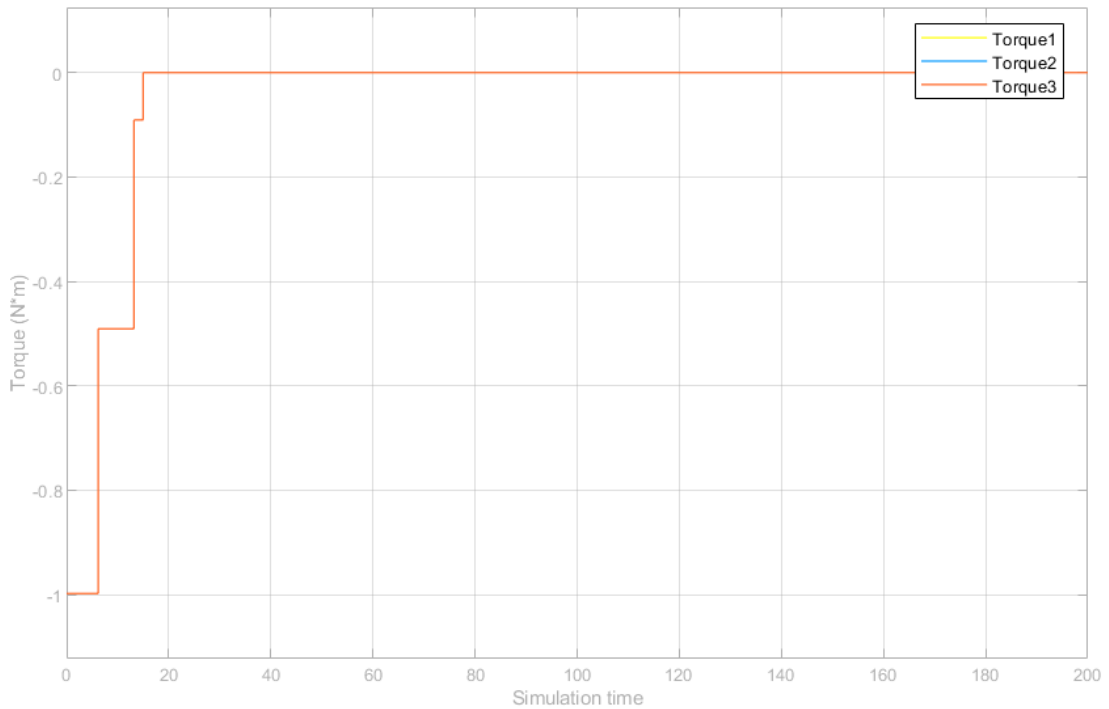


Figure 32: Reaction Wheels expressed torque in discharge phase

The discharge makes the satellite rotate at high angular velocity. This is visible in the next image. Since all inertia moments of the spacecraft are different, the respective angular velocities induced are different from each other.

When the Reaction Wheels expressed torque reaches zero, the RWs are considered as completely discharged and the phase ends. It is the turn of the Detumbling phase.

Chapter 4: Desaturation mode

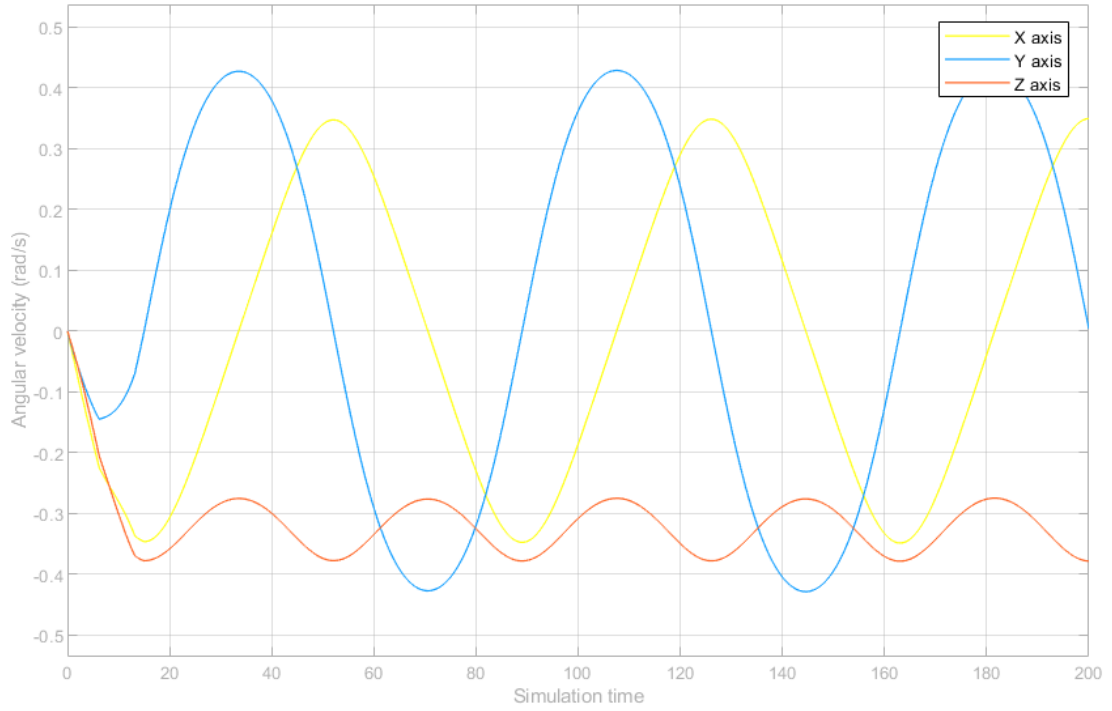


Figure 33: Spacecraft angular velocity in discharge phase

4.4 Detumble step

The second step of the Desaturation Mode is essentially a detumbling performed through thrusters. At the end of the previous step, in fact, the Reaction Wheels have been discharged, but as side effect the satellite has begun to spin at high angular velocities. Aim of this phase is to slow down the spacecraft by using thrusters. Reaction Wheels are no more considered, thus discarding completely the initial idea of the two LICIAcube actuators working simultaneously

Even though the dynamics equations remain more or less the same of the Detumbling mode, with attention on some aspects, the real problem is given by thrusters modeling. Working in an ON-OFF fashion, standard control techniques do not apply here. Except for an NMPC, which is not the case to be implemented in order to deal with a situational working condition, controlling thrusters for detumbling cannot rely on proportional control. A PID, for example, would not be aware of the presence of thruster, and would try to obtain attitude control through a continuously variable control signal, that does not work. The results would consist in never turning on thrusters or, in worse cases, the complete loss of control.

The detumble, moreover, often cannot be done simultaneously on every axis. This would mean using more than two thrusters each time. The more are used, the bigger the probability that, due to their geometric displacement, which is often well-studied, they could go in contrast.

LICIAcube thrusters placement is, on this matter, critical. In the case of a three axes detumbling for desaturation, a standard controller would tend to turn on all thrusters: LICIAcube is in fact equipped with four rockets with six possible couple combination, since they are always used in couple. Based on the CubeSat geometry, by using all of them at the same time, no effect on the attitude is obtained due to the thrusters nullifying each other contribute.

In space applications thruster management is dealt with Minimum Switching Thruster control. This technology substitutes the use of classical control architectures in spacecraft attitude control through rockets and aims to obtain a minimization in fuel consumption while respecting the Minimum Torque Impulse Bit, by controlling in an efficient way the rockets functioning. The first of the two actions allows to extend the time window of the mission, while minimizing the switching frequency of the actuator allows to activate less the thruster valves, reducing the electrical power consumption of the spacecraft. In [18] this technology is used to develop a high precision three-axis attitude control scheme for spacecrafts using ON-OFF Thrusters. As first step, the problem of the minimum fuel

consumption- switching control is formulated, taking into account all possible constraints on attitude and angular velocity error. In fact, it is not possible to have an infinite precision when handling thrusters, thus bringing the error to zero, but only a finite and determined precision. This is due to the finite switching frequency of the rockets. Then, the solution is presented: a first control law deals with the thruster switching management, by exploiting a time-varying hysteresis. A switching curve is defined, similarly to what happens in the Sliding Mode theory. Whenever this curve assumes values different from zero for more than a safety margin, the thrusters are commanded to fire. Which combination or how many rockets to activate depends obviously on the spacecraft geometry and must be studied beforehand. Then, the control law is further extended by using an Adaptive controller, which enhances the system control precision and allows to reject the time-varying disturbances that can affect the spacecraft.

In this thesis, the concept of minimum switching thruster control has been the starting point to the resolution of the Desaturation Mode. The solution found is a custom logic that activates thrusters in couple depending on which axis must be desaturated. This solution stands only for spacecrafts where thrusters are implemented similarly to LICIAcube, being it heavily dependent on the satellite geometry and rockets placement.

The control system is an angular velocity control loop, where the dynamic equations are essentially the same, but Reaction Wheels torque is substituted by Thrusters torque.

$$I_{sat}\dot{\omega}_{sat} = T_{thr} - \omega_{sat} \times I_{sat}\omega_{sat} - \omega_{sat} \times I_{RW}\omega_{RW}$$

However, the disturbance term of the Reaction Wheels is still present. This is a necessary consideration: even if Reaction Wheels are not involved in this control loop, it is possible that in the previous step of the Desaturation Mode not all the angular momentum has been discharged. In fact, as seen in the previous chapter, by customizing the discharging logic it is possible to consider the first step as concluded when the actuator is still at an angular velocity different from 0. This means that the coupled term, which is proportional to the angular velocity, still modifies the spacecraft behavior and must be taken into account.

The reference signal of the loop is simply the desired final angular velocity of the CubeSat. Ideally, it is 0 on every axis. However, being this a control system without a proper control architecture in the control engineering sense, a zero steady state error is almost never obtained.

Chapter 4: Desaturation mode

Thrusters modeling and control is entirely handled in a subsystem, which is divided in different blocks based on their purpose. First of all, there is the Switching Management block. Based on the angular velocity of the spacecraft, it decides which combination of thrusters shall be used. The combinations are obviously studied and chosen a priori, since they depend on rocket displacement and configuration. It sends as output Boolean variables that indicate the status of the thrusters. When the satellite has slowed down enough, it sends a signal to turn off the actuators. When they are turned off, the detumble step can be considered as ended. Thrusters activation follows some precedence rules. Since the detumbling cannot be done simultaneously on all axes due to the LICIAcube geometry, as stated before, precedence is given to slowing down the rotations on the X axis first, then Y and Z. A hysteresis process prevents the continuous switching of the rockets due to possible fluctuations in the angular velocity: thrusters are in function until spacecraft angular velocity is inside a safe zone. This safe zone is determined by the custom logic and can be chosen almost at will.

The Boolean values that determines the state of the actuators enter the proper Actuator block. Here the rocket equation, already presented in chapter 2.3 is implemented, obtaining effectively the thrust values expressed by the rockets, necessary to detumble the spacecraft. The block considers also the Minimum Torque Impulse Bit. A thruster cannot be in function for less than the MTIB: this is made possible by setting the block as an atomic unit with a sample rate of 0.02 seconds. The communication with the previous switching logic, which is set at 250 hz, is made possible through Rate Transition blocks. By considering the sample rate of the block, it is possible also to take track of the fuel consumption, which is taken out as output.

$$m_r = m_a - \dot{m} * t_{on} * y$$

Where

m_r → Remaining fuel mass of the spacecraft.

m_a → Available fuel mass.

\dot{m} → Fuel mass consumption per second, taken from LICIAcube Propulsion System data sheet.

t_{on} → Functioning time, multiple of MTIB.

y → Boolean variable sent by the switching logic.

Fuel tank in LICIAcube is common to every rocket. In the case of empty tank, a security check takes action and no force is obtained as output.

Chapter 4: Desaturation mode

In the end, the simulated forces generated by the thrusters are transformed in torques depending on the satellite geometry. The torque enters in the Dynamics block, and from there the new satellite angular velocity is obtained. The control loop has ended.

Before looking at the simulation results, some considerations on the Desaturation Mode are necessary.

While for Fully Operational mode and Detumbling Mode modularity has been easy to achieve by bringing out in the outer layer of the control system all possible customizable parameters, for the Desaturation Mode this is harder.

Attention must be paid to the two actuators block. Reaction Wheels actuator block has been made easy to replace. In the case of a different set of RWs to be mounted on a satellite, by changing the characteristic implemented and the inertia matrix, the control system is ready to go.

The same cannot be said about the Thrusters block. Thrusters implementation may vary for many different causes: different thrust value, propulsion system made by different number of rockets, different displacement on the satellite and so on. If thrusters change, the entire second step of the Desaturation mode must be modified.

4.5 Simulation parameters 4

The second step of the Desaturation Mode inherits as initial condition all the results of the previous phase. The control loop starts from the angular velocity assumed by the spacecraft and, by consuming fuel, tends to slow it down. The simulations follow the case study of the previous step: for this reason, as initial condition, the spacecraft is rotating on all three axes.

As always, the settings of the control loop are first shown. Then, it is the time for the initial conditions.

Hysteresis upper limit	0.05 rad/s
Hysteresis lower limit	0.01 rad/s
MBIT	0.02 s

Whenever the satellite angular velocities are above the upper limit of the hysteresis, the thrusters are turned on, thus slowing down the spacecraft, following the precedence rule stated before, till its velocity goes below the lower limit. However, if, after the detumble on a single axis is complete, for some reason its angular velocity increases again, the rockets are regulated by the hysteresis: while the spacecraft velocity is between the two limits, the detumble does not happen again.

Satellite angular velocities	$[-0.5; 0.2; -0.6]$ rad/s
Reaction wheel angular velocities	$[0; 0; 0;]$ rad/s
Fuel initial mass	1.275 kg

4.6 Simulation results 4

In the first image, it is possible to notice how the spacecraft slows down, settling himself to an angular velocity near the zero. The control obtained cannot guarantee a perfect precision since thrusters are difficult to handle but allows to make the CubeSat controllable by the Fully Operational mode.

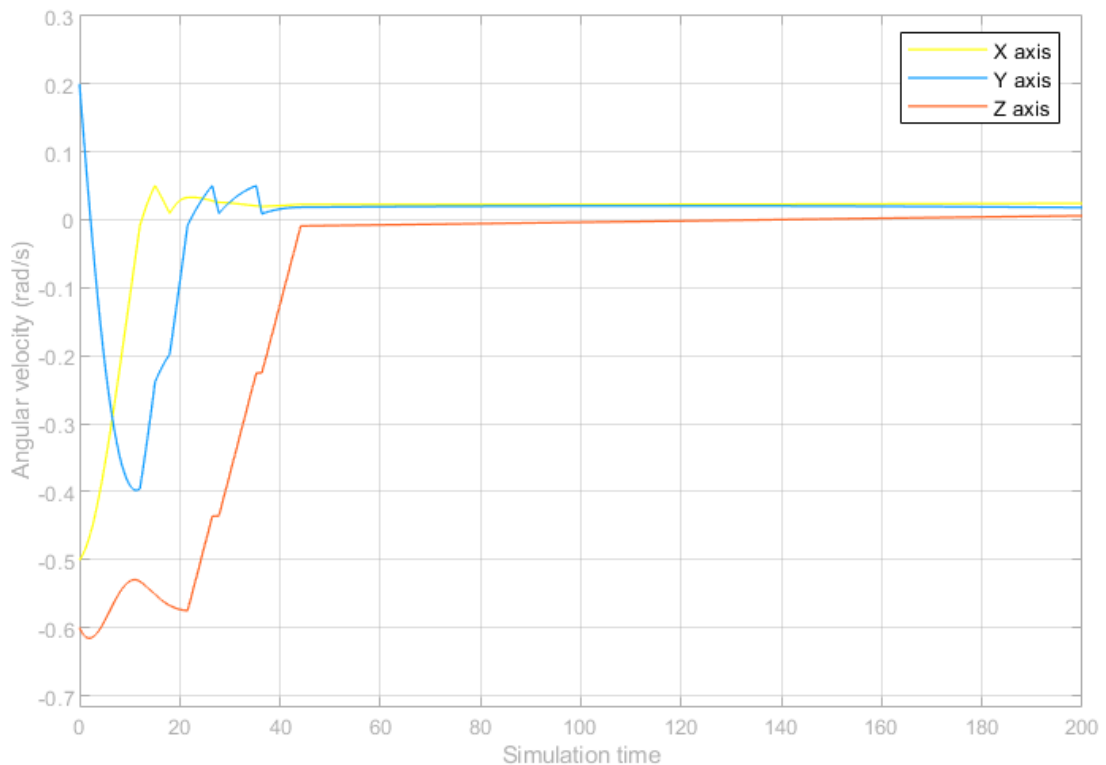


Figure 34: Spacecraft angular velocity in thruster detumbling phase

Chapter 4: Desaturation mode

It is also nice to notice how the precedence rule is respected. The X axis is the first to be slowed down, then follows Y and Z. Whenever X or Y return to a value higher than the hysteresis limit, the slowing down of the Z axis is interrupted.

The torques provided by the thrusters during this detumbling phase are the next subjects. The Y axis has been normalized for privacy purposes.

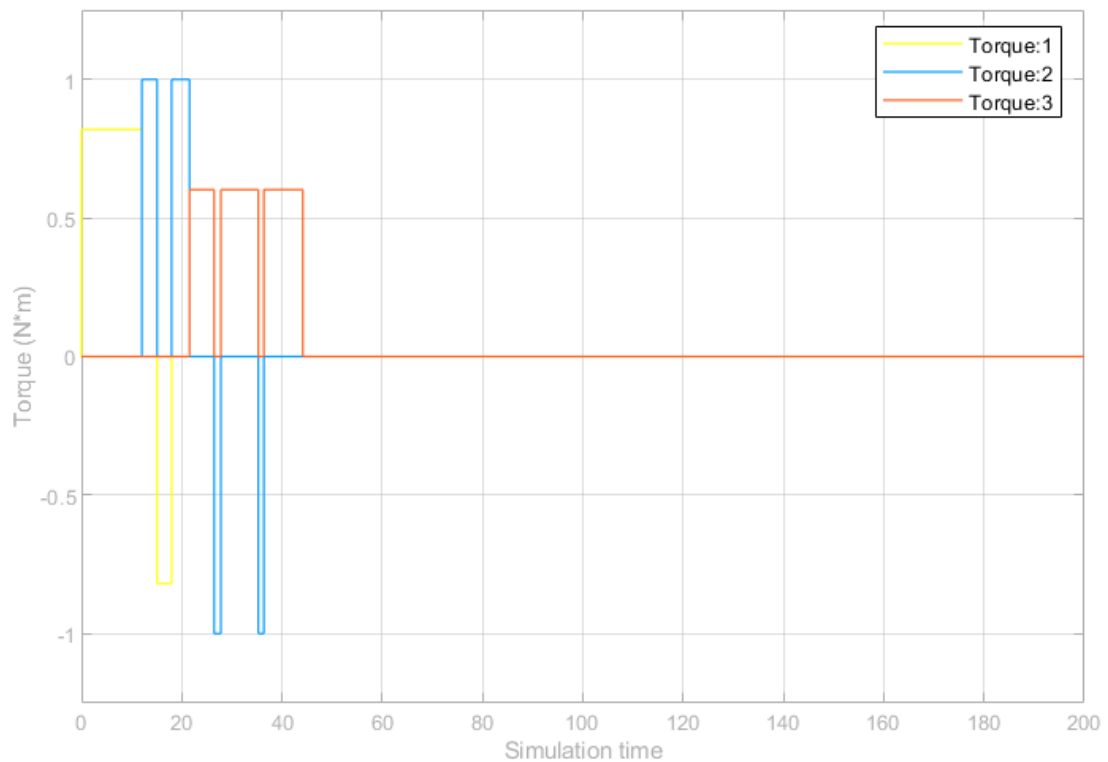


Figure 35: Thrusters expressed torque

In the end, it is worth to notice the time in which thrusters were turned on, to see how they respected the MBIT and how the combination works. Thrusters are always activated in couple, and each combination returns a positive or negative torque on one axis.

Chapter 4: Desaturation mode

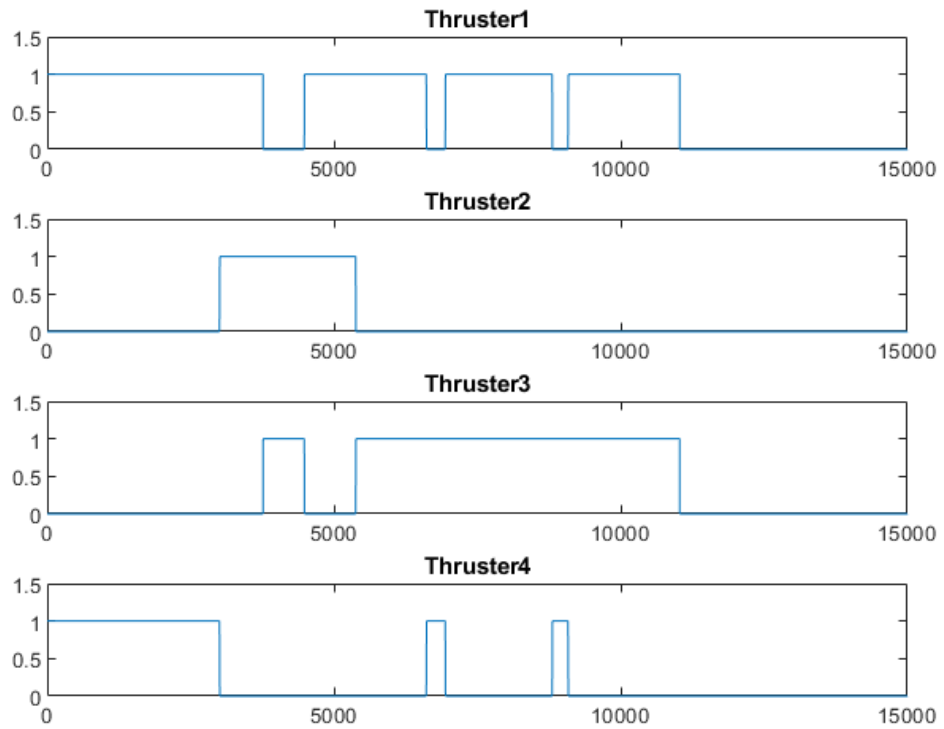


Figure 36: Thrusters combinations

As last thing to pay attention to, in the last image the X axis, that represents the Simulation time, has been reduced in order to better visualize the thruster activations. It is worth to note that, even if the simulation has been run for 200 seconds, the detumbling step lasts only less than 50 seconds. 15000 in the image stands for simulation time constants.

Chapter 5

Integration testing

All the simulations done and presented till now show how the spacecraft responds when controlled

- In Attitude, with the Fully Operative mode. The outputs to be studied here are the quaternions assumed by the spacecraft which have to coincide with the reference sent by the optical payload. The control system succeeds in performing the fly-by of Dydimos B with an irrisory error, guaranteeing the success of the mission.
- In Angular velocity, with the Detumbling mode. With focus on the spacecraft angular velocity, the rotations of the LICIAcube are slowed down with a steady pace, performing a trade-off between resources on the OBC used and control precision.
- In Torque, firstly with the Reaction Wheels and after with Thrusters, during the Desaturation Mode. Even if it lacks proper control laws, the restoration of nominal conditions of the actuators can be done with a good grade of precision, at the cost of fuel consumption.

All the operational modes proved to be effective and adequate to the case study, obtaining good results in the mission environment. This stands for when they are tested on their own, with fixed, known a-priori, initial conditions.

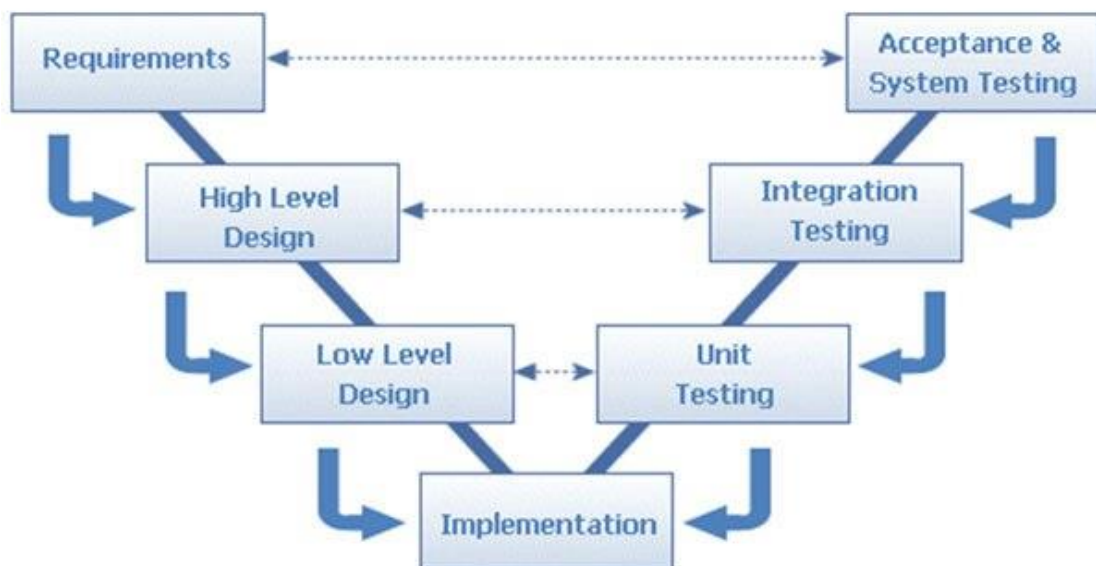


Figure 37: V-shaped software development

What has been done till now can be considered as the Implementation and Unit Testing of the V-shaped software development method. In the following, the next step of the cycle is studied: all control modes are integrated and tested in their entirety, simulating a complete ADCS.

Until now, each operative mode has worked with its own data. Starting from some initial conditions, they have generated, as output, quaternions, angular velocities of the spacecraft and states of the actuators. These data were not shared: they were all different, not linked, case studies.

In the case of a real ADCS, however, operating modes share the same dataset. Whenever a new operating mode is put in function, the initial conditions are directly read from the sensors, so that there is a certain continuity in the operations. Moreover, operating modes are switched not autonomously, but by sending signals through the control board. In space applications, when it is not necessary to have a completely autonomous spacecraft, it is the mission control center, on Earth, that deals with its remote control.

In order to define a complete ADCS to be mounted on future CubeSats, it is necessary to implement also these features.

For LICIACube, and in particular for this case study, two problems arise:

- The navigation from the detach to the optimal trajectory and the fly-by phase must be done autonomously. There is no remote control from the mission control center. This means that the switching among the different operating modes must be performed automatically, by reading sensors data or by some internal conditions to be set a-priori. This problem will be referred to as **switching problem**.
- For software validation it is necessary to simulate the entirety of the ADCS. However, for simulation purpose only, the lack of real sensors makes impossible to read directly the initial conditions of the new operating modes. To let all modes work together, it is necessary to make them communicate: essentially, the most important thing is to design some way to share data so that initial conditions of an operating mode are inherited from the output of another one. Quaternions, angular velocities and state of the actuators must be expressed as some sort of global memory, where every operational mode can read or write, one at a time. This problem will be referred to as **communication problem**.

Major focus has been set on the switching problem. Even before finding a method to implement the autonomous switching between control systems, it is necessary to find the

conditions from where the switch should start. As first idea, a good criterion to change mode would be based on the readings of the sensors. For example, whenever the angular velocity of the Reaction Wheels increases over a certain threshold, Desaturation Mode should be launched. However, this condition is not enough.

For certain spacecraft rotations, in fact, it could be possible that the RWs angular velocity may momentarily exceed the threshold, and then return to the safe zone in a limited span of time, when the attitude has been reached. This situation is not problematic, but the Desaturation would be launched anyway, with consequent, useless, fuel consumption.

To make this criterion a solution, it should be accompanied by a temporal condition. In order to make the switch happen, the RWs angular velocity should exceed the threshold and should maintain that value for a good span of time, which role is to make a distinction between random, momentarily fluctuation and real saturation situations.

By using sensors reading and introducing a safety time windows, it is so possible to have autonomous switches. Time is, however, not easy to implement.

Although Simulink alone grants many simulation methods and tools, it lacks easy ways to implement the concept of time in the control loops. It offers time arrays, and the command “linspace” in Matlab scripts can help in some situations, but none of these resolve the matter.

In this thesis, the resolution to these two problems has been developed by using another tool offered by Mathworks, that is Stateflow. It is a Simulink accessory tool that allows to simulate a finite-state machine like structure, where the possible states can be represented by Simulink and Matlab files or internal scripts. The passage from one state to the other is regulated by **transitions**. They are arrows linking two states that define the conditions under which the switch takes place. Conditions can be elementary and expressed directly on the arrows, or more complex and defined by custom Matlab functions. The only limitation is that all transition conditions must return Boolean variables.

Another point of transitions is that they offer switching conditions based on time. Stateflow works, as Simulink, in iteration steps: each constant of time, decided a priori, the system is evaluated again, giving a new result based also on the previous state of the system. However, differently from Simulink, Stateflow also supports in a user-friendly interface the concept of time. Even if it works on a fixed pace, it takes into account the time passed and allows its use as a transition variable.

Whenever one condition is true, the software passes from the current state to the next one. If no transition takes place, the software continues in running the current state.

The use of Stateflow resolves completely the switching problem. The different control systems can be implemented in the Stateflow-Simulink file as states of the finite-state machine, with transitions that allow to pass autonomously from one operative mode to the other. The conditions for which the switch should happen can be implemented through a double statement. One is made by the custom Matlab function that takes into account the sensor readings, the second exploits the time variable of the tool.

The communication problem is, instead, easier to deal with. Simulink alone grants the user the possibility to implement shared data through the Data Store Memory block. It is a block that allocates some memory for a global variable, accessible from every layer of the control loop. In this way it is possible to let all control system write their results (quaternions, angular velocities) in the respective global variable, one at time, and let them retrieve the initial conditions by reading the memory.

Being a Simulink tool, every layer introduced with Stateflow is recognized by the global memory, making its use compatible with the solution found for the switching problem.

5.1 Structure implementation and user-friendly interface

In order to simulate the entire ADCS functioning on software, all the Simulink control loops have been implemented in the Stateflow file as states of the structure.

However, to do so, some changes to the loops were necessary. All the outputs of the operative modes are now saved in the respective global memory blocks. All initial conditions, that were at the beginning set by hand depending on what scenario to simulate, are now inherited through the global memory. Fully Operative mode reference signals have been changed: they are no more retrieved by an Excel file, but another Stateflow layer deals with sending a series of reference quaternions, similar to the fly-by sequence, that change over time. This change is due to a conflict between Excel and the switches that happen in Stateflow. Whenever the Fully Operative mode is running, if a switch happens, the quaternions on the Excel file, which are taken every 0.2 seconds following a match between Simulation time on Simulink and time column on Excel, are not taken anymore. When the Regime mode resumes, after another switch, the Simulation time and the time column on Excel will not be matched anymore. The quaternions will not be taken anymore and there will be an error.

Chapter 5: Integration testing

After the modifications, a total of four states are present in the structure. The simulations begin with the Detumbling Mode. From this state, a transition goes to the Fully Operational mode and no transition enters this state: Detumbling Mode is never reached again. Then, Regime mode communicates with the two steps of the Desaturation mode with a series of transitions, forming a ring.

Transitions have been implemented through temporal conditions and Matlab internal scripts, as introduced before. Here the sensors readings, stored in the global memories, are compared with thresholds chosen accordingly, depending on the mission to accomplish.

Stateflow usage brings advantages also in other aspects. Transitions threshold can be designed as parameters passed down to the Stateflow layer as variables passed to a function, through Constant Parameter blocks. The same stands for all the parameters that define the control loops. By bringing all these parameter blocks on the upper layer of the software, it is possible to provide a user-friendly interface, where every new simulation with different values can be launched without navigating in the software. In synthesis, every parameter that defines a control loop or a transition threshold is taken out of the respective software layer and brought to the upper one. This provides a good level of abstraction, since any use of the software does not require to study in deep the states of the Stateflow (the operative mode) but can be done by simply adjusting the variables in full view on the upper layer.

Stateflow grants a new aspect of modularity to the ADCS. New CubeSats can be deployed with the software developed in this thesis by, in the best case, modifying only the inertia matrix. Only the detumble step of the Desaturation Phase may be an exception. However, if an Operative Mode needs to be customized, the only change to be done is in one of the states of the file. Changes are always maintained at minimum.

Before showing the final simulation of the entire ADCS, it is necessary to make some statements. Firstly, to test every possible switch and operational mode, it is necessary to use parameters that normally would never describe a CubeSat or a space mission. The reason for this choice lies in the Desaturation Mode: this mode is, technically, an emergency situation where the actuators are working far from the nominal condition. To make this state reachable during the simulation, which cannot last forever, the conditions for its activation have been made much less strict.

Another statement to do is that, in order to have a clear vision of the results, the control loops that did not calculate the quaternions assumed by the spacecraft had to be modified, adding the usual Kinematics block. In this way, whenever during the simulation the operating modes are switched, it is always possible to take track of the attitude of the satellite, even when it is not important. The results, that can be displayed through the Read Data Store Memory block on the upper layer of the software, are continuous, with no sign of interruption due to change of operative mode. However, with some attention is possible to notice the moment of the switches.

5.2 Final Simulation Parameters

In the following, the transition thresholds are reported in the table.

Desired residue velocity	Spacecraft residual angular velocity to switch from Detumbling to Regime	[0.01;0.01;0.01] rad/s
Reaction Wheels angular velocity superior limit	Maximum angular velocity that Reaction Wheels can withstand before switching from Regime to Desaturation	[500;500;500] rad/s
Reaction Wheels angular velocity inferior limit	Angular velocity of Reaction Wheels where discharge is complete. Switch to thruster management.	[1.5;1.5;1.5] rad/s

The first transition is true when the spacecraft has slowed down till the “Desired residue velocity” and at least 300 seconds have passed. This is to avoid undershoots in the control that generally happen in the first 50 seconds, making the satellite unstable.

The second transition, from Regime Mode to the Discharge phase, happens when the RWs rotate at an angular velocity greater than 500 rad/s for more than 10 seconds. In this situation, it is assumed that the RWs have accumulated too much momentum and the desaturation is needed.

The third transition is between the two steps of the Desaturation Mode. It happens when the RWs have slowed down below the “Reaction Wheels angular velocity inferior limit”.

The last transition, that goes from the Detumbling phase of Desaturation Mode to the Fully Operative Mode, does not depend on a particular parameter, but returns the Boolean true when the thrusters are turned off for a certain span of time.

The only simulation parameters belonging to the control loops modified for the complete simulation is the “Hysteresis upper limit”, which is set at 0.03 rad/s.

5.3 Final Simulation

The final simulation allows to monitor the behavior of the spacecraft during the entire simulation, passing through every operating mode. To test every mode, as stated before, initial conditions used are very strict. For this reason, Desaturation mode is launched two times and can be easily spotted in the graphs.

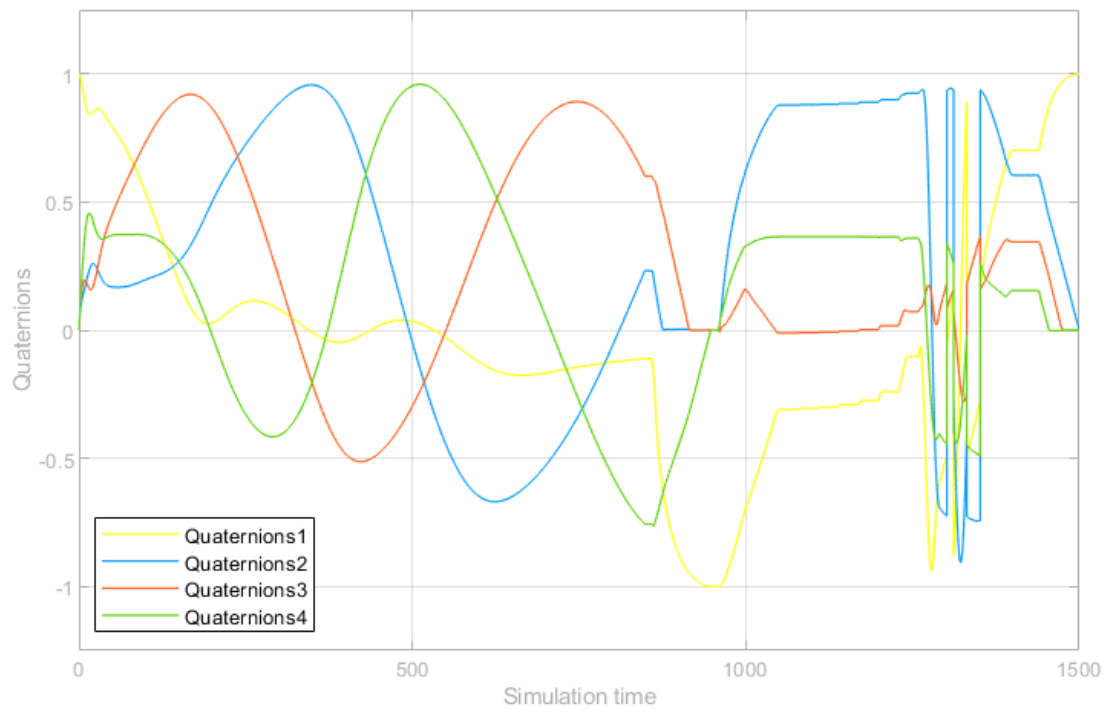


Figure 38: Spacecraft attitude in quaternions final simulation

The quaternions assumed by the spacecraft can easily allow the guess of what mode is running and when it is launched. During the first part of the simulation, the detumbling action makes the CubeSat slow down. The slowing down can be seen by the fact that the curves of the quaternions tend to flat. Since rotations are slower, so is the change in attitude in quaternions.

Then, the first switch happens at about 800 seconds. The reference signal sent by Stateflow makes the spacecraft rotate to the identity quaternion. Right after, a sequence

of signals is sent. It can be seen that the spacecraft acts as stated in the Regime mode, following the reference quaternion, which is changed every 30 seconds.

In the spacecraft angular velocity figure below, it is possible to notice the change in the reference signal. Between 1000 and 1250 seconds little increases in the angular velocity display how the CubeSat tends to rotate a little in order to assume the new commanded attitude. The same can be seen in the angular velocities of the Reaction Wheels, that are commanded to exercise the required torque.

The next switch happens at 1250 seconds, starting the Desaturation Mode. The new quaternion to be assumed by the Regime mode requires too much effort for the Reaction Wheels, which have already accumulated some momentum due to the detumbling phase. This launches the Discharge step, which makes the CubeSat rotate at high velocity. Quaternions are no more of interest, but Reaction Wheels angular velocity is brought to zero. Right after, the Detumbling step starts. The Reaction Wheels angular velocity remains unaffected, but the spacecraft slows down till a near zero value.

Regime mode is, then, started again, requiring another attitude asset that requires much effort to the Reaction Wheels. This time, the Desaturation phase is not called because the angular momentum accumulated is too much, but just because the actuator has to rotate at a value higher than the threshold for 30 seconds, when the time limit in the Stateflow structure has been set to 10.

The process, so starts over.

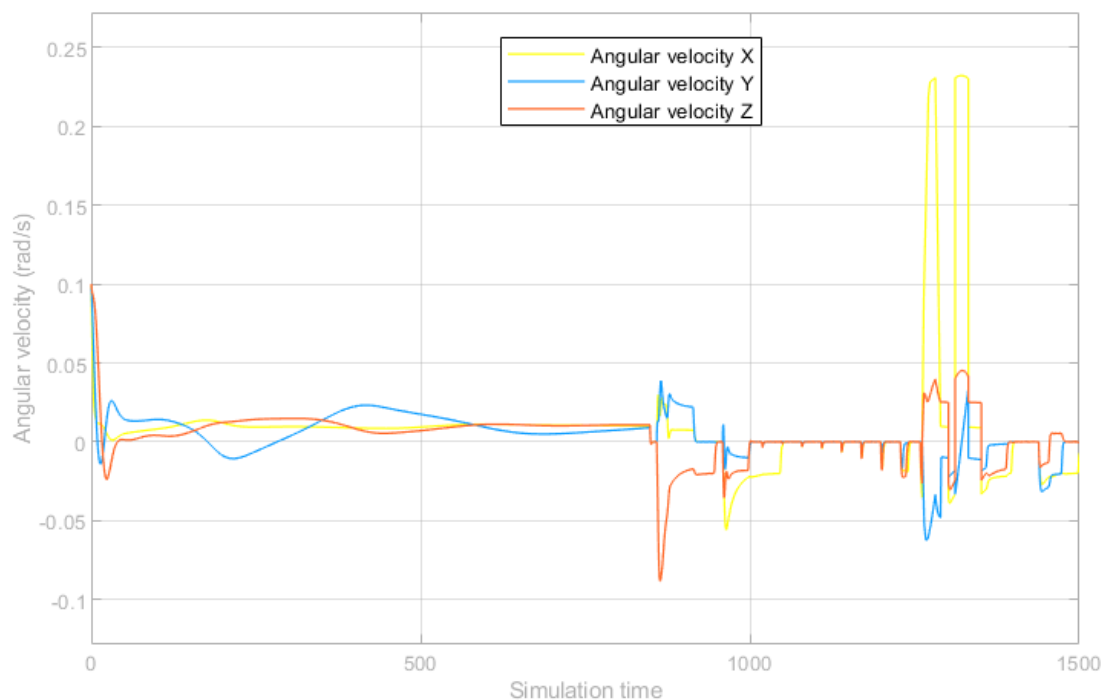


Figure 39: Spacecraft angular velocity final simulation

Chapter 5: Integration testing

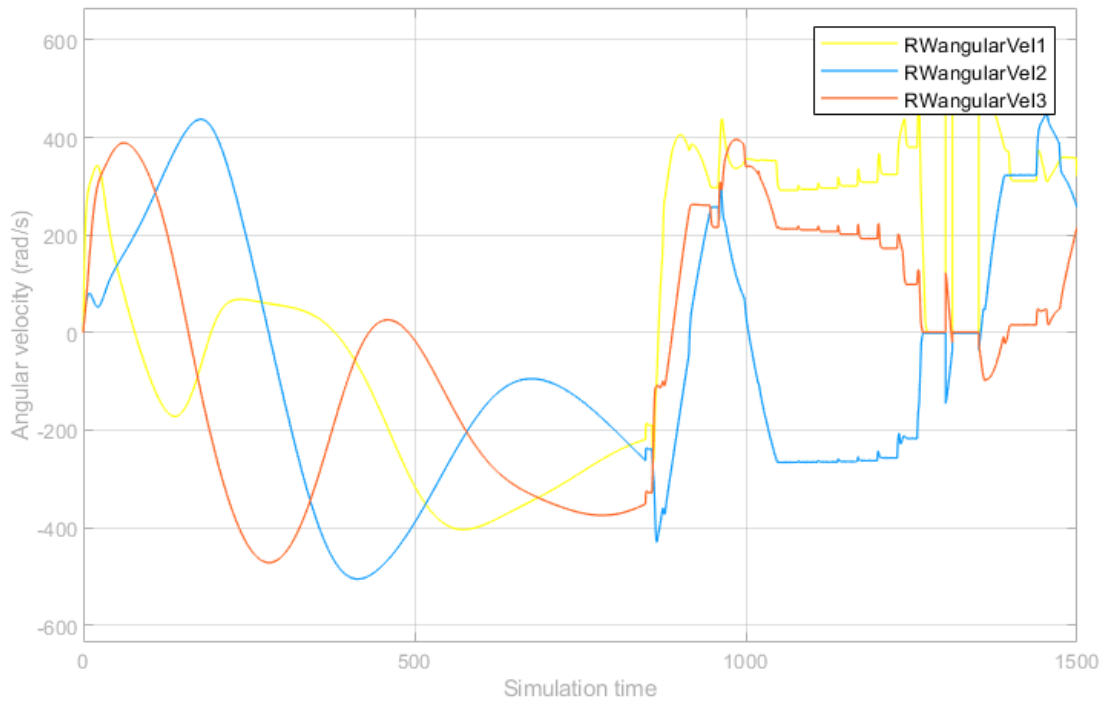


Figure 40: Reaction Wheels angular velocity final simulation

Chapter 6

Conclusions

This thesis work has dealt with the development of an attitude determination and control system for microsattellites. The spacecraft used as subject for this work has been LICIAcube, an Italian CubeSat involved in the DART mission. Starting from real data mission, the goal is to design and validate a control system able to work not only on the LICIAcube, assuring that the mission required performances are met, but on any other CubeSat belonging to the same family, possibly involved in future space missions. The developing of different operative modes has been made by always considering the necessity to achieve a certain grade of modularity.

Starting from an introduction on CubeSats in general and space missions, in the first part of the thesis a brief excursus on the ADCS and its primary hardware components is done. In Chapter 1 some concept of the control theory state-of-the-art were introduced. In particular, Sliding Mode Control and PID control were discussed, since they are the control methods chosen for the thesis application.

In Chapter 2 the mathematical model of LICIAcube has been introduced. Dynamics of the system comes from Euler's rigid body equations, while Kinematics derive from quaternions manipulation. Reaction Wheels work by exchanging angular momentum between spacecraft and themselves, while thrusters produce thrust by ejecting mass. The Extended Kalman filter theory concludes the model.

In Chapter 3 the first simulations have been showed. The focus has been set on the first two operative modes, the Fully Operative and Detumbling mode, with the second being a much simpler version of the first. After defining initial conditions and Simulink settings, the result of the simulation showed that both the control systems perform well and are suited for the study case. The error in pointing in the Regime mode is low enough to guarantee a fly-by sequence with no problem, even if the camera FoV is very low. The filtering effect is a nice perk of the system. The Detumbling mode, instead, even if it is not very precise, allows to slow down the spacecraft without requiring too many resources on the OBC.

In Chapter 4 the Desaturation Mode is explained. It has been divided in two control sub-systems, where the first deals with the discharge of angular momentum on the Reaction

Chapter 6: Conclusions

Wheels, the other deals with the Thrusters activation and successive detumbling of the spacecraft. This choice is preferred respect to making the two actuators work in synchronous in a single control system. The absence of a controller, substituted by two custom control logics, made the performances of the operative mode less precise, in particular in the second step, where thrusters force can make the CubeSat rotate with very low effort.

In Chapter 5, all the operational modes have been merged in a single file that simulates a real ADCS behavior, able to switch on his own depending on the sensors readings. Through Stateflow it has been possible to resolve all problems regarding the switches between modes, making the ADCS completely autonomous, suiting the LICIACube mission. The final simulation showed that the final control system behaves correctly even when the starting condition are strict. The transitions are well handled, and every movement is registered.

The performed work shows good performances and it can be considered as a prototype for real ADCS to be implemented in the future for real space missions. The software can be considered as tested and functioning, ready to the next step of the software developing. The modularity concept, asked to allow the implementation of the software on CubeSats of the same family of LICIACube, has reached a good point, since the recalibration of new simulations requires few steps. Hardware-in-the-loop tests are planned in order to further advance the developing, but logistic problems delayed the designed date and are now on hold.

However, there are many aspects of the work that can deserve attention and that can be improved. The first aspect is the control law of the Fully Operational mode. As stated in chapter 2, the controller used is a first order Sliding Mode controller, with the aid of a sigmoid function to avoid chattering. The selection of this controller can be considered a good compromise, even if some improvements should be possible. A second order Sliding Mode controller, with a particular propension towards the Super-Twisting algorithm, can resolve the chattering problem and achieve even better performances.

The second aspect is the Desaturation mode. Although the functioning is real and tested, it is not an optimal solution. A ready to be implemented, commercial ADCS would require a PWM command of the thrusters. Moreover, Reaction Wheels and rockets should work in synchronous, thus optimizing the time spent and the fuel consumed. It is worth to note, in fact, that fuel consumption is not minimized in this application, but only evaluated. The control, even if there is no really need to finely tune a rocket, is pretty imprecise and there

Chapter 6: Conclusions

is room to improve it. In the end, a real Minimum Switching Thruster Management would enhance the overall performances.

Some little bugfixes can be done on the Stateflow layer too. The contrast between Excel and Stateflow can surely be resolved without relying on a second Stateflow layer inside the control loop. This requires a better knowledge on the simulation time in Simulink, aspect that was considered as secondary in this work. Moreover, it is possible to bring the inertia matrices of spacecraft and Reaction Wheels outside their control loop in order to make them modifiable directly from the upper layer of the software.

One last aspect are the disturbances in the deep space, that in this work are not contemplated. Solar wind, eventual drag, gravity gradients are all secondary effects that have not an immediate feedback on the system but can stack up error with the passing of time. By modeling and taking them into consideration, it is possible to improve the performance of the system by developing a more precise, less robust controller.

A nice improvement for the thesis would be the implementation of a secondary Fully Operational control loop, based on other control architectures. The most promising technique surely is the Adaptive control, which can be considered complementary to the Sliding Mode. A confrontation between the two simulations may bring advices for better performances or a major modularity.

Next steps are to perform simulations on hardware, to test if the developed software is ready to real world applications.

Bibliography

- [1] "Attitude control for the LUMIO CubeSat in deep space" Alvaro Romero-Calvo, James Biggs, Francesco Topputo - October 2019.
- [2] <https://www.nasa.gov/planetarydefense/dart>
- [3] "Output Feedback Second Order Sliding Mode Control for Spacecraft Attitude and Translation Motion" – Chutipon Pukdeboon
- [4] "Satellite attitude control system design with nonlinear dynamics and kinematics of quaternion using reaction wheels" - Breno Braga Galvao, Maria Cristina Mendes Faustino, Luiz Carlos Gadelha de Souza - 2016.
- [5] Jamshed Iqbal*, Mukhtar Ullah, Said Ghani Khan, Baizid Khelifa, and Saša Ćuković – "Nonlinear control systems – A brief overview of historical and recent advances"- 2017.
- [6] "Applied Non-linear Control" – Jean-Jacques E. Slotine, Weiping Li – 1990.
- [7] "L08_sliding_mode" – Carlo Novara – Nonlinear control and aerospace application – Politecnico di Torino – 2020
- [8] "Chattering problem in sliding mode control systems" – Vadim Utkin, Hoon Lee – 2006
- [9] "Spacecraft Attitude Determination and Control" – J.R.Wertz – Springer Book Archive -1978.
- [10] "Actuators for position and attitude control" – Elisa Capello – Politecnico di Torino - 2019
- [11] "Autonomous navigation strategy on the LICIACube Micro-Satellite for Close Proximity Fly-By of the Didymoon Asteroid"- E.Fazzoletto, V. Marchese, P. Tricarico, V. Di Tana, F. Miglioretti, S. Simonetti, V. Della Corte, E. Dotto, M. Amoroso, G. Impresario, S.Pirrota, A. Zinzi.- 2020
- [12] "Technical Solutions to Monitor an Asteroid Space Impact" - M. Amoroso, S. Pizzurro, G. Impresario, V. Di Tana, S. Simonetti, F. Miglioretti, B. Cotugno, V. Marchese- 2019

Bibliography

- [13] "LS05_attitude_control" – Carlo Novara – Nonlinear control and aerospace application – Politecnico di Torino – 2020
- [14] "L11_ekf" – Carlo Novara – Nonlinear control and aerospace application – Politecnico di Torino – 2020
- [15] "MPC Design for CMG based Testbed" – Matteo Facchino – Politecnico di Torino - 2019-2020
- [16] "Control System Design" – Karl Johan Astrom – 2002
- [17] "LICIACube XACT ICD Addendum" – Argotec srl – 2020
- [18] "Minimum switching thruster control for spacecraft position pointing" – Mirko Leomanni, Andrea Garulli, Antonio Giannitrapani, Francesco Farina – 2017
- [19] "Optimal Combined Reaction-Wheel Momentum Management for Earth-Pointing Satellites" – Xiaojiang Chen, Y.Hashida, Stephen Hodgart, Willem H. Steyn – 1999
- [20] "Sliding_mode Control Strategies for Rendezvous and Docking Maneuvers" – Elisa Capello, Elisabetta Punta, Fabrizio Dabbene, Giorgio Guglieri, Roberto Tempo – 2017