FABIO BOBROW

THE CUBLI: MODELING AND NONLINEAR ATTITUDE CONTROL UTILIZING QUATERNIONS

São Paulo
2022

FABIO BOBROW

THE CUBLI: MODELING AND NONLINEAR ATTITUDE CONTROL UTILIZING QUATERNIONS

Revised Version

Thesis presented to Escola Politécnica da Universidade de São Paulo to obtain the degree of Doctor of Science

São Paulo
2022

FABIO BOBROW

THE CUBLI: MODELING AND NONLINEAR ATTITUDE CONTROL UTILIZING QUATERNIONS

Revised Version

Thesis presented to Escola Politécnica da Universidade de São Paulo to obtain the degree of Doctor of Science

Concentration area:

3139 - Systems Engineering

Advisor:

Prof. Dr. Bruno Augusto Angelico

Co-advisor:

Prof. Dr. Paulo Pereira da Silva

São Paulo
2022

Catalogação-na-publicação

Bobrow, Fabio
The Cubli: modeling and nonlinear attitude control utilizing quaternions / F. Bobrow -- versão corr. -- São Paulo, 2022.
180 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Energia e Automação Elétricas.

1.CONTROLE (TEORIA DE SISTEMAS E CONTROLE) 2.SISTEMAS DE CONTROLE 3.QUATERNIOS I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Energia e Automação Elétricas II.t.

ACKNOWLEDGMENTS

I would like to thank my advisor, Bruno A. Angelico, for his resilience and success in creating an applied control laboratory in a country that suffers so much adversity and lack of government support in research and education.

I would also like to thank my laboratory colleagues, Fabio Y. Toriumi, Gabriel P. Neves and Mateus M. Brugnolli, for all their support throughout the development of this work.

Finally, I would like to thank CAPES and FAPESP for its finantial support.

ABSTRACT

This thesis consists of the modeling and nonlinear attitude control of the Cubli, a cube with three reaction wheels mounted on orthogonal faces that turns into a reaction wheel based 3D inverted pendulum when positioned in one of its vertices. Although this plant is not new, the novelty of this work is the use of quaternions instead of Euler angles as feedback control states. A good advantage of using quaternions, besides avoiding singularities and trigonometric functions, is that it allows working out quite complex dynamic equations completely by hand utilizing vector notation. The entire modeling and nonlinear control law is derived without the need of any mathematical symbolic software in a very didactic and self-contained way, being a contribution to control education. In addition, the original plant utilizes six IMUs spread throughout the entire structure in previously known positions. The Cubli of this work utilizes an accelerometer together with a rate gyroscope in a quaternion based complementary filter, which requires only one IMU placed anywhere and yields equally satisfactory results. Modeling is performed utilizing Lagrange equations and it is validated through computer simulations and Poinsot trajectories analysis. The derived nonlinear control law is based on feedback linearization technique, thus being time-invariant and equivalent to a linear one dynamically linearized at the given reference. Moreover, it is characterized by only three straightforward tuning parameters. Computer simulations and experimental results are presented for validation, as well as all the mechanical, electronical and algorithm development of the system.

Keywords – Attitude control, Inverted pendulum, Feedback linearization, Lagrange equations, Modeling, Nonlinear control systems, Poinsot trajectories, Reaction wheels, Quaternions.

LIST OF FIGURES

1 Parabola 5

2 Parabola with imaginary axis 6

3 Broom Bridge Plaque 8

4 CubeSat[1] 11

5 Reaction wheel[2] 12

6 Inverted pendulum types 15

7 The Cubli 16

8 Cubli body parts 22

9 Euler angles (z-y-z sequence) 23

10 Same vector described in different coordinate frames 24

11 Euler angles and angular velocity 26

12 Euler angles (z-y-x sequence) 28

13 Quaternions 30

14 Rodrigues' rotation geometry 30

15 Same vector described in different coordinate frames 35

16 Quaternions and angular velocity 37

17 Cubli angular velocities 42

18 Structure parameters 44

19 Reaction wheel 1 parameters 45

20 Friction torque data and curve fitting 57

21 Cubli dynamics 58

22 Simulation 1a - Invariant analysis (quaternion) 60

23 Simulation 1a - Invariant analysis (energy) 60

24 Simulation 2 - Invariant analysis (angular momentum) 61

25 Cubli static equilibrium positions 62

26 Cubli static equilibrium positions rotation quaternions 62

27 Simulation 3 - Static equilibrium stable (quaternion) 64

28 Simulation 4 - Static equilibrium unstable (quaternion) 64

29 Cubli dynamic equilibrium motions 65

30 Simulation 5 - Dynamic equilibrium spin (Euler angles) 65

31 Cubli non-equilibrium position rotation quaternion 66

32 Simulation 6 - Dynamic equilibrium precession, nutation and spin (Euler angles) 66

33 Simulation 6 - Dynamic equilibrium (center of mass) 67

34 Cubli principal moment of inertia 68

35 Simulation 8 - Dynamic equilibrium (Euler angles) 69

36 Simulations 9 and 10 - Poinsot trajectories 70

37 Open loop poles (without reaction wheels) 73

38 Open loop poles (with reaction wheels angular velocity) 75

39 Open loop poles (with reaction wheels angular velocity and displacement) . 77

40 The Cubli with feedback linearization 80

41 The Cubli with state regulator and feedback linearization 85

42 The Cubli with state regulator and feedback linearization (with wheels) . . 88

43 Closed loop poles 91

44 The Cubli with state regulator and feedback linearization (with wheels) . . 92

45 Closed loop poles 95

46 Reaction wheels dynamics 97

47 Reaction wheels low-pass filter 98

48 Discrete low-pass filter 99

49 Cut-off frequency ω_c influence in a low-pass filter 100

50 Reaction wheels state observer . . . 101

51 Reaction wheels state observer rearranged . . . 101

52 Reaction wheels state observer considering internal dynamics . . . 102

53 Cubli orientation kinematics . . . 103

54 Cubli state observer . . . 105

55 Cubli state observer considering angular velocity . . . 106

56 Rate gyroscope . . . 106

57 Three-axis rate gyroscope . . . 107

58 Cubli state observer considering angular velocity with rate gyroscope . . . 108

59 Accelerometer . . . 108

60 Three-axis accelerometer . . . 108

61 Cubli state observer considering angular velocity with rate gyroscope and accelerometer . . . 110

62 MATLAB/Simulink model . . . 113

63 Simulation 1 - Without reaction wheels feedback . . . 114

64 Simulation 2 - With reaction wheels angular velocities feedback . . . 115

65 Simulation 3 - With reaction wheels angular velocities and displacements feedback . . . 116

66 Experiment 1 - Disturbance rejection . . . 118

67 Experiment 1 - Wheel velocity and angle feedback comparison . . . 119

68 Experiment 3 - Sinusoidal tracking reference . . . 120

69 Experiment 4 - Linear and nonlinear comparison . . . 121

70 The Cubli . . . 129

71 STM NUCLEO-L432KC[3] . . . 130

72 SparkFun 9DoF Sensor Stick[4] . . . 131

73 Maxon EC 45 Flat 50W[5] . . . 131

74 Maxon ESCON Module 50/5[6] . . . 132

75 Turnigy Graphene Panther 1000mAh 6S LiPo[7] 132
76 Electronic circuit block diagram 133
77 PCB board 133
78 PCB schematics 134
79 PCB renders 134
80 PCB pictures 135
81 Face 136
82 Reaction wheel 136
83 Motor support 136
84 Vertex 137
85 Firmware block diagram 139

CONTENTS

Part I: INTRODUCTION **1**

1 Outline **3**

2 Historical background **5**

2.1 Complex numbers[1] 5

2.2 Quaternions[2] 8

2.3 Satellites[3] 10

2.4 Attitude control 12

3 Literature review **15**

3.1 Inverted pendulums 15

3.2 The Cubli 16

3.3 Quaternion attitude control 17

Part II: THEORY **19**

4 Modeling **21**

4.1 System definition 21

4.2 Kinematics 22

4.2.1 Euler angles 23

4.2.1.1 Rotation matrix 24

4.2.1.2 Kinematic equation 25

4.2.1.3 Singularities 27

4.2.2 Quaternions 29

4.2.2.1 Spacial rotation 30

4.2.2.2 Quaternion fundamentals 31
4.2.2.3 Quaternion notation 32
4.2.2.4 Quaternion product 33
4.2.2.5 Rotation quaternion 34
4.2.2.6 Kinematic equation 37
4.3 Kinetics 41
4.3.1 Kinetic Energy 41
4.3.1.1 Angular velocities 42
4.3.1.2 Inertia matrices 43
4.3.1.3 Kinetic Energy 46
4.3.2 Potential Energy 48
4.3.2.1 Gravity 48
4.3.2.2 Potential Energy 48
4.3.3 Lagrange equations 49
4.3.3.1 Generalized coordinate q 50
4.3.3.2 Reaction wheels angular displacement 54
4.3.4 Kinetic equations 56
4.3.5 Friction forces 56
4.4 Dynamics 57
4.4.1 Dynamic equations 58
4.5 Validation 59
4.5.1 Invariant analysis 59
4.5.2 Singular motions 61
4.5.2.1 Static equilibrium 61
4.5.2.2 Dynamic equilibrium 64
4.5.3 Poinsot trajectories 69

5 Analysis **71**

5.1 Linearized dynamics without reaction wheels 71

5.2 Linearized dynamics with reaction wheels angular velocity 74

5.3 Linearized dynamics with reaction wheels angular velocity and displacement 75

6 Control **79**

6.1 Control strategy 79

6.2 Attitude Controller 80

6.2.1 Feedback linearization 80

6.2.2 State regulator 81

6.2.3 Controller gains 86

6.3 Attitude and Wheel Velocity Controller 87

6.3.1 State regulator 87

6.3.2 Controller gains 88

6.4 Attitude and Wheel Angle Controller 91

6.4.1 State regulator 91

6.4.2 Controller gains 92

7 Estimation **97**

7.1 Reaction wheels angular velocity and displacement 97

7.1.1 Low-pass filter 98

7.1.2 State observer 100

7.2 Cubli angular velocity and orientation 102

7.2.1 State observer 103

7.2.1.1 Rate gyroscope 106

7.2.1.2 Accelerometer 108

Part III: RESULTS **111**

8 Simulations **113**

8.1 Simulation without reaction wheels feedback 114

8.2 Simulation with reaction wheels angular velocities feedback 115

8.3 Simulation with reaction wheels angular velocities and displacements feedback 116

9 Experiments **117**

9.1 Disturbance rejection 117

9.2 Wheel velocity and angle feedback comparison 118

9.3 Sinusoidal tracking reference 120

9.4 Linear and nonlinear comparison 121

10 Conclusions **123**

References **125**

Appendix A – Construction **129**

A.1 Electrical and Electronics 129

A.1.1 Off-the-shelf components 130

A.1.2 Printed circuit board 132

A.2 Mechanics 135

A.2.1 Laser cut aluminum 135

A.2.2 3D printed ABS 137

A.2.3 Screws and standoffs 137

Appendix B – Firmware **139**

B.1 Modules 140

B.1.1 Attitude and reaction wheel controller 140

B.1.2 Reaction wheel estimator 143

B.1.3 Attitude estimator 144

B.2 Drivers 147

B.2.1 Motor 147

B.2.2 Hall sensor 148

B.2.3 Inertial measurement unit 148

B.3 Utilities 154

B.3.1 Parameters 154

B.3.2 Pin names 156

B.4 Main program 156

PART I

INTRODUCTION

1 OUTLINE

This thesis is organized in three main parts:

Part I: INTRODUCTION - In the first part, an historical background of the main topics and a literature review are presented.

Part II: THEORY - In the second part, the system model is introduced and its mathematical representation is obtained and validated through computer simulations. Then, its dynamic properties are analyzed and a nonlinear control strategy is synthetized. The developed state observers are also covered here.

Part III: RESULTS - In the third part, the actual physical implementation of the system is presented. Conclusions and possible future developments are draw from experimental results.

The whole prototype construction and programming process, from mechanics manufacturing, hardware selection and software implementation are detailed in appendix sections.

2 HISTORICAL BACKGROUND

A brief historical background of the main topics will be presented for unfamiliar readers. This part of the thesis has no impact on the development or contribution of the work itself. If the reader is already familiar with the subject, he can skip right to chapter 3 without loss of content.

The first topic is about the history of numerical systems, from the earliest days until the emergence of complex numbers. This is the basis to introduce an even more sophisticated numerical set: the ultracomplex numbers, also called quaternions. Then, the history and application of space satellite will be drawn, focusing on satellite attitude control system.

2.1 Complex numbers[1]

Consider the polynomial function $f(x)$ given by

$$f(x) = x^2 + 1\,. \tag{2.1}$$

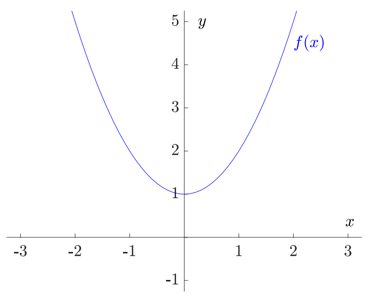


Figure 1: Parabola

To figure out where this equation equals to zero graphically, i.e., the roots of the function, all it needs to be done is to check where the function crosses the x-axis. However,

[1]Text adapted from Welch Labs YouTube Channel (https://youtu.be/T647CGsu0VU) and authorized for reproduction by the author (Stephen Welch)

as can be seen (Fig. 1), this parabola never crosses the x-axis, so, according to this plot, there are no solutions to the equation $x^2 + 1 = 0$.

But there is a little problem with this statement. A little over 200 years ago, a famous German mathematician by the name of Carl Friedrich Gauss (1777-1855) proved that every polynomial equation of degree n has exactly n roots. The polynomial from Eq. 2.1 has a highest power, or degree, of 2, so it should have 2 roots. Furthermore, Gauss's discovery is not just some random rule, today it is called the Fundamental Theorem of Algebra. If this plot disagrees with something so important called the Fundamental Theorem of Algebra, this might be a problem. What Gauss is telling is that there are two perfectly good values of x that could be plugged into $f(x)$ and get 0 out. Where could these two numbers be?

The short answer here is that this plot does not have enough numbers. Numbers are typically thought of existing on a one-dimensional continuum: the number line. Most common numbers live there: zero, one, negative numbers, fractions, even irrational numbers like π or $\sqrt{2}$. But this system is incomplete, and the missing numbers are not just further left or right, they live in a whole new dimension. Algebraically, this new dimension has everything to do with a problem that was mathematically considered impossible for over two thousand years: the square root of negative one.

When this missing dimension is included in the analysis, the function $f(x)$ ends up being something much more interesting (Fig. 2).

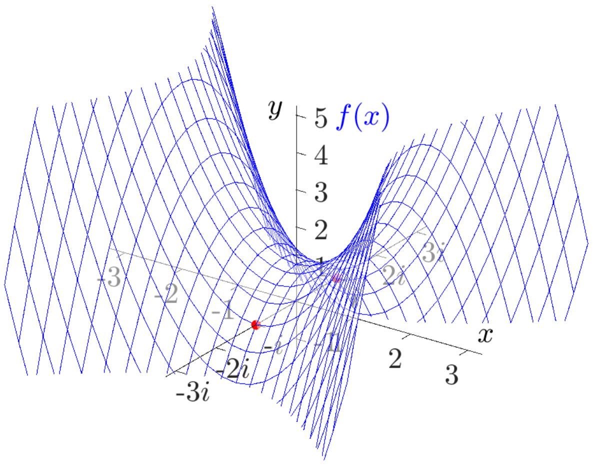


Figure 2: Parabola with imaginary axis

Now that the input number are in their full two-dimensional form, it is possible to see how the function $f(x)$ really behaves. And as it can be seen, the function does cross the x-plane (red dots). So why is this extra dimension that numbers possess not common knowledge? Part of this reason is that is has been given a terrible name, a name that suggests that these numbers are not even “real”. In fact, Gauss himself proposed these numbers should be given the name “lateral” instead of “imaginary”. To get a better handle on imaginary numbers and really understand what is going on in the last plot, it is nice to go back a bit in the history of numbers.

Early humans really only had use for the natural numbers, which makes sense because of how numbers were used: as a tool for counting things. So, to early humans, the number line would have just been a series of dots. As civilizations advanced, people needed answers to more sophisticated math question, like when to plant seeds, how to divide land, and how to keep track of financial transactions. The natural numbers just were not cutting it anymore, so the Egyptians innovated and developed a new, high-tech solution: fractions. Fractions filled in the gaps in our number line and were basically cutting-edge technology for a couple thousand years.

The next big innovations to hit the number line were the number zero and negative numbers, but it took a lot of time to get everyone on board. Since it is not obvious what these numbers mean or how they fit into the real world, zero and negative numbers were met with skepticism, and largely avoided or ignored. Some cultures were more suspicious than others, depending largely on how people viewed the connection between mathematics and reality. A great example here is Greek civilization: despite making huge strides in geometry, the Greeks generally did not accept negative numbers or zeros, after all, how could nothing be something?

What is even wilder is that this is not all ancient history, just a few centuries ago, mathematicians would intentionally move terms around to avoid having negatives show up in equations. Suspicion of zero and negative numbers did eventually fade, partially because negatives are useful for expressing concepts like debt, but mostly because negatives just kept sneaking into mathematics.

It turns out there is just a whole lot of math you cannot do if you do not allow negative numbers to play. Before negatives were accepted, simple algebra problems like $x + 1 = 0$ have no solution, just like the equation $x^2 + 1 = 0$ seemed to have no solution before imaginary numbers were accepted either. The only difference here it that when the real and imaginary part are put together, the one-dimension number line transforms into a two-dimension number plane called the complex plane, whose main contribution comes from the works of Italian mathematicians Gerolamo Cardano (1501 - 1576) and Rafael Bombelli (1526 - 1572).

2.2 Quaternions[2]

Just as complex numbers are a two-dimensional extension of real numbers, ultra-complex numbers, also known as quaternions, are a four-dimensional extension. They are an absolutely fascinating and often underappreciated number system from math, much because it is impossible to visualize graphically a four-dimensional number in a three-dimensional world. But they are not just playful mathematical tools, they have a surprising pragmatic utility for describing rotation in space and even in quantum mechanics.

The story of their discovery is also quite famous in math. The Irish mathematician Willian Rowan Hamilton (1805 - 1865) spent much of his life seeking a three-dimension number system analogous to the complex numbers, and as the story goes, his son would ask him every morning whether or not he had figure out how to divide triples and he would always say "no, not yet". But one day, while crossing the Broome Bridge in Dublin, he realized, with a supposed flash of insight, that what he needed was not to add a single dimension to the complex numbers, but to add two more imaginary dimensions: three imaginary dimensions describing space and the real number sitting perpendicular to that in some kind of four-dimensional space. He carved the crucial equation describing these three imaginary units into the bridge (Fig. 3a), which today bears a plaque in his honor showing that equation (Fig. 3b).

(a) Bridge[3]

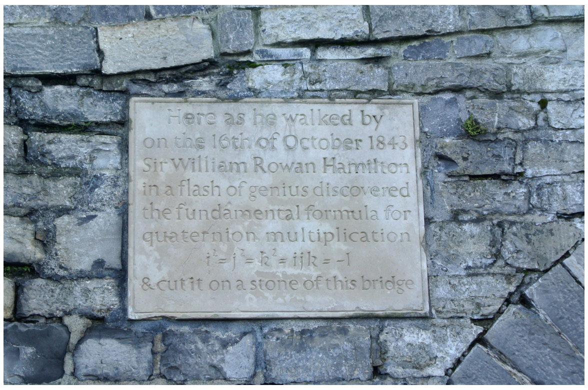

Here as he walked by
on the 16th of October 1843
Sir William Rowan Hamilton
in a flash of genius discovered
the fundamental formula for
quaternion multiplication
$i^2 = j^2 = k^2 = ijk = -1$
& cut it on a stone of this bridge
(b) Plaque[4]

Figure 3: Broom Bridge Plaque

[2]Text adapted from 3Blue1Brown YouTube Channel (https://youtu.be/d4EgbgTm0Bg) and authorized for reproduction by the author (Grant Sanderson)

[4]Image source: Flickr (https://www.flickr.com/photos/infomatique/44408787052)

[4]Image source: Wikimedia Commons (https://commons.wikimedia.org/wiki/File:Inscription_on_Broom_Bridge_(Dublin)_regarding_the_discovery_of_Quaternions_multiplication_by_Sir_William_Rowan_Hamilton.jpg)

Our modern notion of vectors, with their dot product, cross product and things like that, did not really exist in Hamilton time, at least not in a standardized form. So, after his discovery, he pushed hard for quaternions to be the primary language with which we teach students to describe three-dimensional space, even forming an official quaternion society to proselytize his discovery. However, this was balanced with mathematicians on the other side of the fence who believed that the confusing notion of quaternion multiplication was not necessary for describing three-dimensions.

It is even believed that the Mad Hatter scene from Alice in Wonderland, whose author Lewis Carroll (1832 - 1898) was an English mathematician, was written in reference to quaternions: that the chaotic table placement changes were mocking their multiplication, and that certain quotes were referencing their non-commutative nature:

"Why, you might just as well say that I see what I eat is the same thing as I eat what I see!"

Fast forward about a century and the computing industry gave quaternions a resurgence among programmers who work with graphics and robotics and anything involving orientation in space. And this is because they give an elegant way to describe and to compute three-dimensional rotations, which is computationally more efficient than other methods and which also avoids a lot of numerical errors that arise in these other methods.

Not too long ago, Apollo 11 Moon mission suffered an incident called gimbal lock. Gimbal lock is the loss of one degree of freedom in a three-dimensional system, usually called a singularity. Although engineers were aware of this problem, their lack of experience at that time prevented them on avoiding it, just 60 years ago. As it would become evident in the coming years, a potential solution to this problem is to represent the orientation in some other way such as quaternions.

The 20th century also brought quaternions some more attention from a completely different direction: quantum mechanics. The special actions the quaternions describe in four dimensions are actually quite relevant to the way that two-state systems like spin of an electron or the polarization of a photon are described mathematically.

To summarize, quaternions are a very recent innovation in math which have only now begun to demonstrate their real potential in practical applications. Just as negative numbers and zero made it possible for civilizations to advance as we never imagined, it is still unclear what kind of progress quaternions will enable us to do.

2.3 Satellites[5]

We do not think about them that often, but above us are hundreds of flying robots that play a large part in our lives on Earth. In 1957, lonely Sputnik circled the Earth by itself, but today, the worlds of communication, navigation, weather forecasting and aerial photography all rely heavily on satellites, as do many national militaries and government intelligence agencies.

The number of active satellites in space have balloned from 852 in 2004 to 3,368 in 2020[6], while the total market for satellite manufacturing, the launches that carry them to space, and related equipment and services has gone from USD 60 billion to USD 271 billion in the same period[7]. Satellite industry revenue today makes up only 16% of the global telecommunications industry but accounts for over 74% of space industry revenue.

Despite all this, the satellite industry still uses technological resources from 50 years ago, is terribly risk-averse and is dominated by few companies that are usually State suppliers. With the proportional size of a van, a conventional satellite takes an average of 10 years to build and costs more than USD 400 million. At the launch date, its technology is already completely out of date.

This scenario led to micro-satellites being developed. Smaller than a microwave, satellites of this type can be built in less than a year and cost USD 250 thousand. In what have been call the Small Sat Revolution, CubeSats lead the way. A CubeSat (Fig. 4) is a small and affordable satellite that can be developed and launched by college, high schools and even individuals. The specifications were developed by Cal Poly and Stanford University in 1999. Its basic structure is a 10cm cube (volume of 1 liter) weighting less than 1.33kg, which allows several of these standardized packages to be stacked together and launched as secondary payloads on other missions.

[5]Text adapted from Wait But Why blog (https://waitbutwhy.com/2015/08/how-and-why-spacex-will-colonize-mars.html) and authorized for reproduction by the author (Tim Urban)

[6]Data source: Statista (https://www.statista.com/statistics/897719/number-of-active-satellites-by-year/)

[7]Data source: SIA (https://sia.org/news-resources/state-of-the-satellite-industry-report/)

Figure 4: CubeSat[8]

A typical CubeSat is composed of three main parts: power, communication and control. Power deals with the generation, storage and distribution of energy for its components to operate; communication deals with the antennas and radio frequencies utilized to communicate with a ground station or even directly with other CubeSats in space; and control deals with the sensors and actuators needed to change the satellite's motion in space.

The translation motion of a satellite is mostly defined by its orbit, which in turn is mostly defined by the launch that has put it there first place. Although very little, there is still a bit of air resistance where satellites are, not enough to stop them but enough to make them slowly lose orbit and reentry earth atmosphere, disintegrating in this process. To keep it operational, they must constantly perform a few motion adjustments called orbit maneuvers. Not all satellites have the components needed to perform orbital maneuvers, some of them are just designed to operate for a few months or are located at higher orbits that would take years to decay.

The rotational motion control, on the other hand, is a must. Without it, a satellite will be at arbitrary orientations and would not be able to communicate with earth, harvest energy for the sun, take pictures of earth and constellations, among many other activities they are designed to do. In other words, the satellite would be useless. Luckily, controlling the orientation (usually called attitude) of a satellite is much easier than its position, and can even be done without propellant, just with electrical energy obtained directly from the sun in space.

[8]Image Source: ETH Zurich (https://www.ethz.ch/content/specialinterest/baug/institute-igp/mathematical-and-physical-geodesy/en/research/satellite-mission.html)

2.4 Attitude control

Spinning a satellite around one of its axis is the simplest control mechanism to keep it pointed in a desired direction. This technique is usually called spin stabilization because it uses the gyroscopic action of the rotating mass as the stabilizing mechanism. Propulsion system thrusters are fired only occasionally to make desired changes in spin rate, or in the spin-stabilized attitude. If desired, the spinning may be stopped through the use of the same thrusters or by yo-yo de-spin: two cables with weights on the ends that absorbs angular momentum when the cable length is increased.

An alternative control strategy is three-axis stabilization, where the satellite does not continuously rotate but keeps a fixed orientation in space. Techniques here are usually divided into active or passive, whether or not it depends on external sources to work.

The most common active technique is gyroscopic attitude control, where the desired attitude is maintained by the spin of small wheels within the satellite. These are called reaction wheels (Fig. 5) or momentum wheels and at least three of them are required to provide full three-dimensional control. If allowed to pivot relative to the satellite, they are known as control moment gyros. The attitude can be changed by varying the speed or orientation of these internal gyros. Small thrusters may also be used to supplement the gyroscopic attitude control or to hold the satellite orientation fixed when it is necessary to de-spin or reorient gyros that have become saturated (reached their maximum spin rate or deflection) over time.

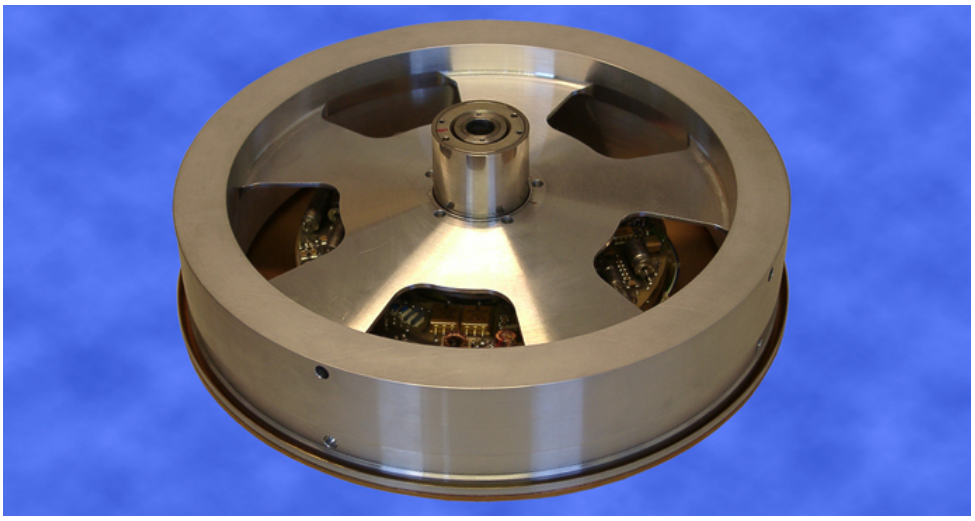


Figure 5: Reaction wheel[9]

[9]Image source: Collins Aerospace (https://www.rockwellcollins.com/Products-and-Services/Defense/Platforms/Space/High-Motor-Torque-Momentum-and-Reaction-Wheels.aspx)

As for passive techniques, the most common is magnetic torquers, which utilizes coils to create a magnetic field that exert a moment against the local magnetic field. This method works only where there is a magnetic field against which to react, such as Earth's orbit. However, only small satellites are able to interact in this way, since Earth has a very low magnetic field. Another example are solar sails, devices that produce thrust as a reaction force induced by reflecting incident light from the Sun. There are even purely passive mechanisms, such as gravity-gradient stabilization where Earth's gravitational field by itself can stabilize the attitude of a large satellite.

As it can be seen, the most interesting technique is gyroscopic attitude control, since it does not depend on external sources to work, and it can maneuver the satellite to any desired orientation. However, it is usually the most expensive of them (because of their need of reaction wheels and/or thrusters) as well as one of the most complicated to implement.

The advantage of reaction wheels over thrusters is that they form a class of electric actuators and so do not require any propellant to work. Its working principle is based on the conservation of angular momentum. The action of the reaction wheel on the satellite is performed by angular momentum, limited to the axis of rotation of the wheel. Due to the large difference between the inertia of the satellite and the reaction wheel, an attitude control with great precision is possible. Reaction wheels are particularly useful when the satellite must be rotated by very small amounts, such as to hold a telescope or a camera pointed in a certain direction. The drawback is that it can only rotate a satellite around its center of mass and are not able to move it from one place to another.

Reaction wheels are typically made of an electric motor, usually a brushless motor, and an inertia element. The inertia element and the motor are mounted on a bearing which must ensure precise rotation about an axis. The speed of rotation of the system is controlled by an electronic motor drive system. Reaction wheels can be driven in two different ways: by torque or speed. When driven by torque, the reaction wheels must be able to estimate the useful torque generated by it (the raw torque minus the friction losses).

Although initially used solely and exclusively for attitude control of satellites, reaction wheels are now used in a variety of fields, from Formula 1 cars to academic kits.

One hindrance of the gyroscopic attitude control is that there is virtually no way to test a prototype given that our current physical knowledge still does not allow us to develop a chamber on Earth with micro gravity. However, it is still possible to control something very similar to a satellite here on earth but taking into account Earth's gravity.

3 LITERATURE REVIEW

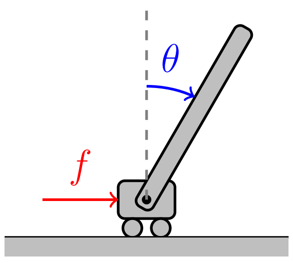
Now, the Cubli will be introduced, which is a test bed for satellite's attitude control system utilizing quaternions as feedback control states, thus, combining all previously described topics into a single system.

A greater focus will be given on the research being carry out in this field, with citations of relevant conferences and journal papers.

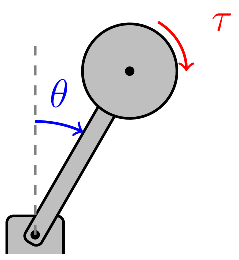
3.1 Inverted pendulums

Inverted pendulum systems have been a popular demonstration of using feedback control to stabilize open-loop unstable systems. Introduced back in 1908 by Stephenson [32], the first solution to this problem was presented only in 1960 with Roberge [26] and it is still widely used to test, demonstrate and benchmark new control concepts and theories [10, 11, 13, 29, 36, 38, 39].

Differently from cart-pole inverted pendulums, that have a controlled cart with linear motion (Fig. 6a), reaction wheel pendulums have a controlled rotating wheel that exchanges angular momentum with the pendulum (Fig. 6b). First introduced in 2001 by Spong [31], it was soon adapted to a 3D version by Lee and Goswami in 2007 [16].



(a) Cart-pole



(b) Reaction wheel

Figure 6: Inverted pendulum types

3.2 The Cubli

Perhaps, even most notable is the Cubli (Fig. 7). Originally developed and baptized in 2012 by Gajamohan [7,8] from the Institute for Dynamic Systems and Control of Zurich Federal Institute of Technology (ETH Zurich), the Cubli is a device that consists of a cube with three reaction wheels mounted on orthogonal faces. By positioning it in one of its vertices, it becomes a reaction wheel based 3D inverted pendulum. This method of utilizing reaction wheels is similar to the one used for decades to stabilize satellites and spacecraft in space [3,4,6], but due to gravity and surface friction, the dynamics of these systems are somewhat different.

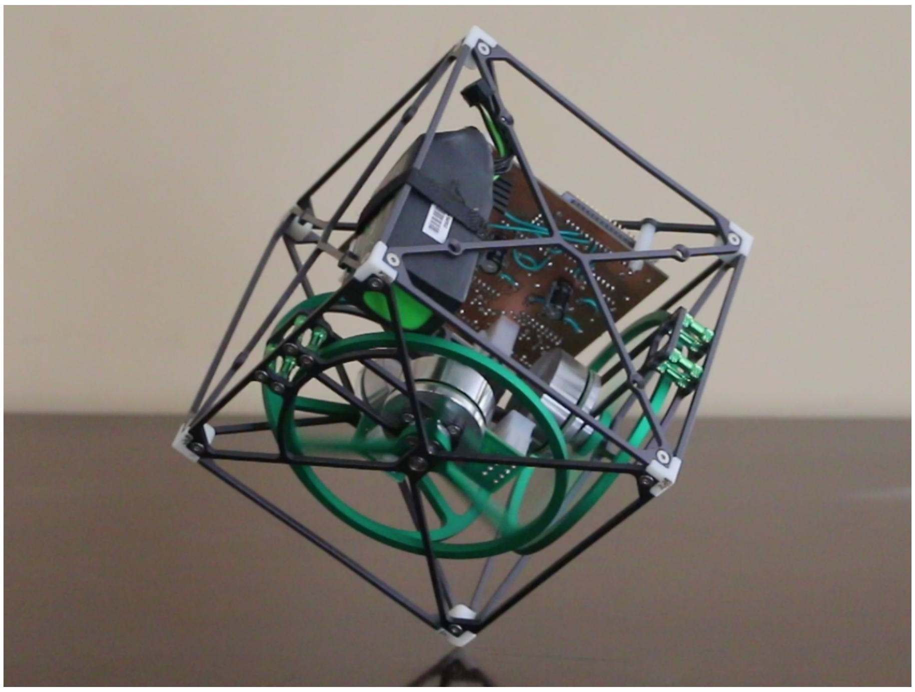


Figure 7: The Cubli

The purpose of this thesis is first to model the system, and then design and implement a nonlinear attitude controller for it. Although the ETH team has already done this [22, 24], the novelty of this work is the use of quaternions as feedback control states. In addition, ETH's attitude estimator utilizes just accelerometers [23, 35], which is only possible due to the fact that the Cubli has a non-accelerated pivot point and, thus, angular and centripetal acceleration terms can be dismembered. However, a disadvantage of this approach is that it requires six IMUs spread throughout the entire structure in previously known positions. The Cubli of this work utilizes an three-axis accelerometer together with a three-axis rate gyroscope in a quaternion based complementary filter, which requires only one IMU placed anywhere and yields equally satisfactory results.

3.3 Quaternion attitude control

Quaternions were used for simulation of the rotational motion of rigid bodies as early as 1958 by Robinson [27], but it was only a decade later, in 1968, that some early results on the use of quaternions as feedback control states was shown by Mortensen [21]. In fact, it was Meyer that introduced the attitude control theory a few years earlier, in 1966, but he utilized rotational matrices instead [20]. In 1985, Wie et al. proposed a linear decoupled model-independent control law, also utilizing quaternions [37]. Although different, all of them used Lyapunov control theory. A disadvantage of it is that the control law is based on intuition rather than fundamental principles. Moreover, important concepts such as damping and loop bandwidth are not well defined as in linear control theory.

Dwyer, in 1984 [5], and Slotine and Li, in 1991 [30], approached this problem utilizing a nonlinear transformation to realize an exact linear model of the rotational dynamics to which linear control can be applied, a method that is also called feedback linearization or dynamic inversion. The first utilized the Gibbs vector, whereas the second utilized Euler angles. The main problem here is that both parametrizations are singular at certain attitudes, even with no attitude error.

It was in 1993 that Paielli and Bach proposed an approach which incorporates features of these others while avoiding their main problems [1,25]. They utilized the same dynamic inversion as Dwyer, Slotine and Li, but with quaternions in the rotation dynamics, which are globally nonsingular, just as Mortensen and Wie et al. However, the Gibbs vector was utilized in the error dynamics because they have no nonlinear mathematical constraints to prevent the realization of linear error dynamics.

The control law derived in this thesis utilizes this same approach. Although it was proposed three decades ago, it continues to be considered state of the art control law in the field, with few practical applications even today. Most of nowadays aircraft and even some spacecraft are designed from a set of linear plant models and implemented with a gain-scheduled linear controller [28]. The first aircraft to utilize dynamic inversion control was the Lockheed Martin F-35, released in 2006 and currently the most advanced fighter jet in service. This scenario has started to change in the last years with the astonishing growth of nanosatellites and commercial UAVs [12,14,18,33,34].

Although this control technique is not novel, the implementation of it in the Cubli, as far as the author knows, has not been presented in the literature before. Moreover, a nice advantage of quaternions, besides the usual arguments to avoid singularities and trigonometric functions, is that it allows working out quite complex dynamic equations completely by hand utilizing vector notation [9]. This becomes evident in this thesis, where the entire modeling and nonlinear control law is derived without the need of any mathematical symbolic software in a very didactic and self-contained way, being a contribution to control education as well.

PART II

THEORY

4 MODELING

The first step in any control system problem is to deduct a mathematical representation of the system to be controlled. This is usually called modeling and implies in obtaining the differential equations that best describe the dynamics of the system.

Dynamics is that branch of mechanics which deals with the motion of bodies under the action of forces. The study of dynamics in engineering usually follows the study of statics, which deals with the effects of forces on bodies at rest. Dynamics has two distinct parts: kinematics, which is the study of motion without reference to the forces which cause motion; and kinetics, which relates the action of forces on bodies to their resulting motions.

Dealing with kinematics and kinetics as separate topics is beneficial in many ways. Primarily, the equation of motion becomes first order ordinary differential equation, which can be easily and numerically integrated for computer simulation. Furthermore, as the equation is separated by characteristics, it is convenient to further investigate the behavior of the system.

4.1 System definition

The Cubli is composed of four rigid bodies: a structure and three reaction wheels (Fig. 8). The structure rotates freely aroung the pivot point O (articulation vertex), whereas each reaction wheel, besides rotating together with the structure, also rotates around its axial axis.

There are other bodies, such as motors, batteries, microcontrollers, etc., that can be interpreted as being part of one or other of them. The only exception are the motors, whereby their stators are considered part of the structure whereas their rotors are considered part of the reaction wheels.

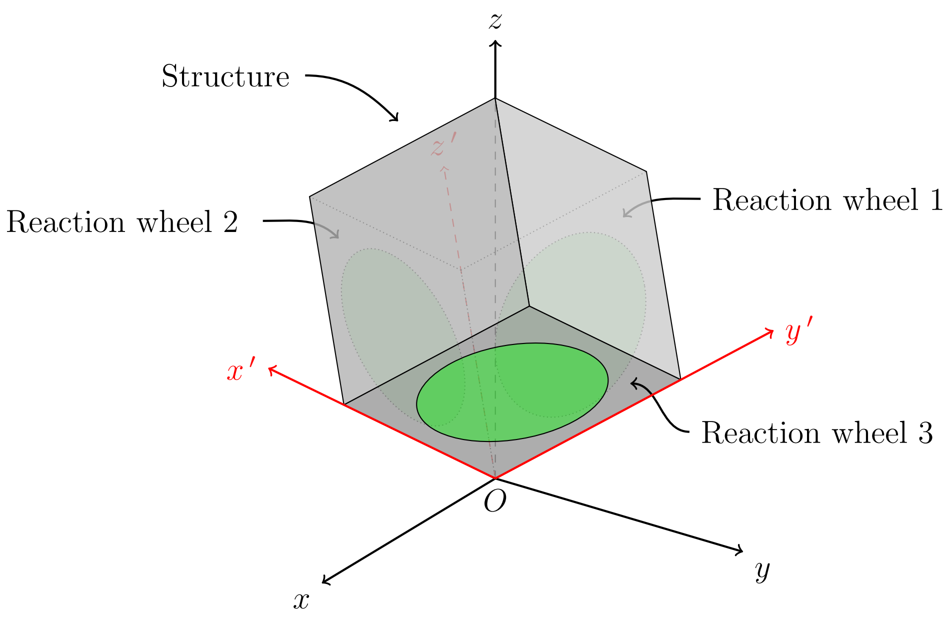


Figure 8: Cubli body parts

- Structure - Structure, motor stators, batteries, microcontrollers, sensors, PCBs, cables, screws and nuts
- Reaction wheel 1 - Reaction wheel and motor rotor that rotates around the x''-axis (parallel to the x'-axis)
- Reaction wheel 2 - Reaction wheel and motor rotor that rotates around the y''-axis (parallel to the y'-axis)
- Reaction wheel 3 - Reaction wheel and motor rotor that rotates around the z''-axis (parallel to the z'-axis)

Moreover, because of the reaction wheels masses, the Cubli center of mass is not exactly at the structure center of mass but somewhere between this point and the articulation vertex.

Although the Cubli center of mass presents translation motion, it is always rotating around the pivot point O (articulation vertex), so the whole motion can be interpreted only as a rotational motion around a fixed point.

4.2 Kinematics

In this section, the rotational kinematics equations of the Cubli will be derived. These equations consider only the orientation and angular velocities, not focusing on the moments applied on it.

Because the orientation of the structure is the same of the Cubli, and the relative angular displacement of the reaction wheels are not of interest, we will only focus on the orientation of the structure as a generic rigid body.

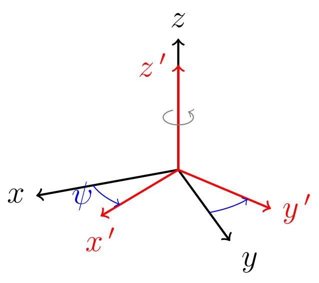
The kinematic equations will be derived both utilizing Euler angles and quaternions, so their pros and cons become more evident.

4.2.1 Euler angles

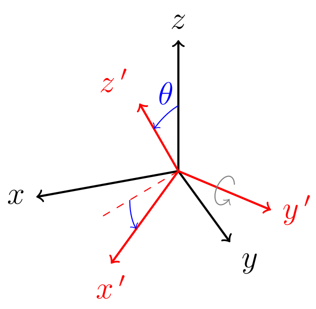
The most common representation for specifying the orientation of a rigid body are the Euler angles. The Euler angles are three angles introduced by Swiss mathematician Leonhard Euler (1707 - 1783) to describe the orientation of a coordinate frame, usually a body fixed coordinate frame, with respect to another coordinate frame, usually an inertial coordinate frame.

It is based on the property that any orientation can be described with a sequence of three rotations around predefined axis. There is no agreement on the sequence of rotations (there are 12 total possibilities), a fairly common one is the z-y-z sequence:

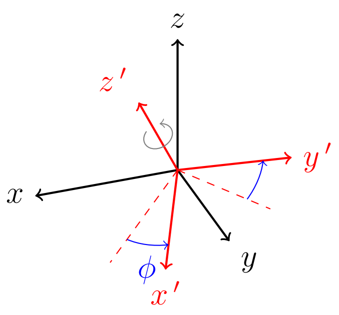
- Rotation of ψ (precession) around the z' axis (Fig. 9a)
- Rotation of θ (nutation) around the y' axis (Fig. 9b)
- Rotation of ϕ (spin) around the z' axis (Fig. 9c)



(a) First rotation



(b) Second rotation



(c) Third rotation

Figure 9: Euler angles (z-y-z sequence)

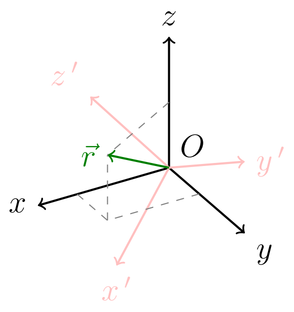
Note that all rotations are always performed around the body fixed coordinate frame axis (represented in red) and not the inertial coordinate frame axis (represented in black).

There are two main advantages on utilizing Euler angles: first is that they only need three parameters (the angles ψ, θ and ϕ) to specify the orientation of a rigid body; second is that these parameters have an intuitive physical meaning.

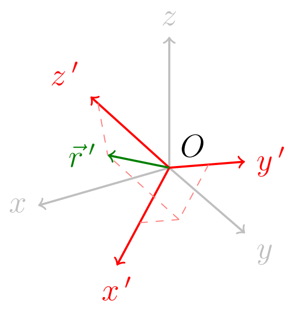
However, Euler angles also have their disadvantages, the most critical being the occurrence of singularities (also known as “gimbal lock”): orientations at which two Euler angles becomes undefined. For example, when the angle θ is equal to $0°$ or $\pm 180°$, the axes of the first and third rotation coincide and the values of ψ and ϕ become indistinguishable. The second disadvantage is that Euler ’s angles involves many trigonometric operations, which require a high computational capacity depending on the application.

4.2.1.1 Rotation matrix

Let $\vec{r}$ be an arbitrary fixed vector described in an inertial coordinate frame $O : \{x, y, z\}$ (Fig. 10a) and $\vec{r}\,'$ be the same vector but described in a body fixed coordinate frame $O : \{x', y', z'\}$ (Fig. 10b).



(a) Inertial coordinate frame



(b) Body fixed coordinate frame

Figure 10: Same vector described in different coordinate frames

To transform $\vec{r}$ into $\vec{r}\,'$, consider the following multiplication

$$\vec{r}\,' = R\vec{r}, \tag{4.1}$$

where R is the total rotation matrix

$$R = R_{z_\phi} R_{y_\theta} R_{z_\psi}\ , \tag{4.2}$$

and R_{z_ψ}, R_{y_θ} and R_{z_ϕ} are each individual rotation matrix given by

$$R_{z_\psi} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad R_{y_\theta} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}, \quad R_{z_\phi} = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.3}$$

Note that the matrix associated with the first rotation R_{z_ψ} goes to the rightmost side, while the matrix associated with the last rotation R_{z_ϕ} goes to the leftmost side. This happens because matrix multiplication is not commutative and happens from right to left.

Since the vector $\vec{r}$ is fixed in the inertial coordinate frame, the rotation matrix R can be used to describe the orientation of the body fixed coordinate frame with respect to the inertial coordinate frame

$$R = \begin{bmatrix} -\sin\phi\sin\psi + \cos\phi\cos\theta\cos\psi & \sin\phi\cos\psi + \cos\phi\cos\theta\sin\psi & -\cos\phi\sin\theta \\ -\cos\phi\sin\psi - \sin\phi\cos\theta\cos\psi & \cos\phi\cos\psi - \sin\phi\cos\theta\sin\psi & \sin\phi\sin\theta \\ \sin\theta\cos\psi & \sin\theta\sin\psi & \cos\theta \end{bmatrix}. \tag{4.4}$$

Moreover, a rotation matrix is an orthonormal matrix, which means that its inverse is equal to its transpose

$$R^{-1} = R^T\,. \tag{4.5}$$

So, in order to perform the inverse transformation, the following multiplication can be done

$$\vec{r} = R^T\vec{r}\,'\,. \tag{4.6}$$

4.2.1.2 Kinematic equation

Let us suppose now that the body fixed coordinate frame is in rotational motion around the origin O (Fig. 11).

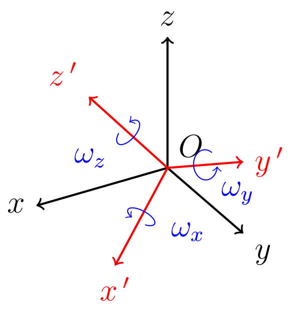


Figure 11: Euler angles and angular velocity

Its angular velocity vector $\vec{\omega}\,'$ is given by

$$\vec{\omega}\,' = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} . \tag{4.7}$$

Note that this is the angular velocity with respect to the inertial coordinate frame but described in the body fixed coordinate frame axes.

Since vector $\vec{r}$ is fixed in the inertial coordinate frame, its time derivative, as seen by the inertial coordinate frame, is zero

$$\dot{\vec{r}} = \vec{0}\,. \tag{4.8}$$

In turn, its time derivative, as seen by the body fixed coordinate frame, depends on the body fixed coordinate frame angular velocity vector

$$\dot{\vec{r}}\,' = -\vec{\omega}\,' \times \vec{r}\,'\,. \tag{4.9}$$

The minus sign appears because, if the body coordinate frame rotates in one direction, the vector will be seen by the body coordinate frame as rotating in the opposite direction.

Another way of representing (4.9) is

$$\dot{\vec{r}}\,' = -\tilde{\omega}\,'\vec{r}\,'\,, \tag{4.10}$$

where $\tilde{\omega}\,'$ is the angular velocity represented as a skew-symmetric matrix corresponding to its cross product

$$\tilde{\omega}\,' = \vec{\omega}\,'\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} . \tag{4.11}$$

Differentiating (4.1) and making use of (4.6) and (4.8), yields

$$\begin{aligned}\dot{\vec{r}}' &= \frac{d}{dt}\left(\vec{r}'\right)\\ &= \frac{d}{dt}\left(R\vec{r}\right)\\ &= \dot{R}\vec{r} + R\overset{\vec{0}}{\dot{\vec{r}}}\\ &= \dot{R}\left(R^T\vec{r}'\right)\\ &= \dot{R}R^T\vec{r}'\,. \end{aligned} \tag{4.12}$$

Comparing (4.12) with (4.10), it is possible to obtain the angular velocity skew-symmetric matrix in terms of the rotation matrix and its time derivative

$$\tilde{\omega}\,' = -\dot{R}R^T\,. \tag{4.13}$$

The Euler angles are not a vector and cannot be easily isolated. However, by substituting (4.4) into (4.13), the angular velocities can be written in terms of the Euler angles and their time derivatives in matrix notation

$$\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix} = \begin{bmatrix}0 & \sin\phi & -\cos\phi\sin\theta\\ 0 & \cos\phi & \sin\phi\sin\theta\\ 1 & 0 & \cos\theta\end{bmatrix}\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}\,. \tag{4.14}$$

Inverting the above matrix, the Euler angles time derivatives can be written in terms of themselves and the angular velocities

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \frac{1}{\sin\theta}\begin{bmatrix}\cos\phi\cos\theta & -\sin\phi\cos\theta & \sin\theta\\ \sin\phi\sin\theta & \cos\phi\sin\theta & 0\\ -\cos\phi & \sin\phi & 0\end{bmatrix}\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix}\,. \tag{4.15}$$

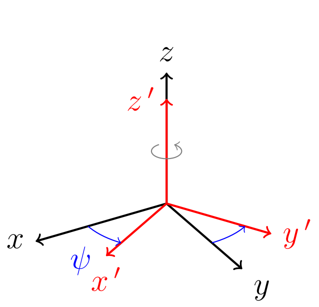
This last equation is the kinematic equation of a rigid body utilizing Euler angles with the *z*-*y*-*z* rotation sequence.

4.2.1.3 Singularities

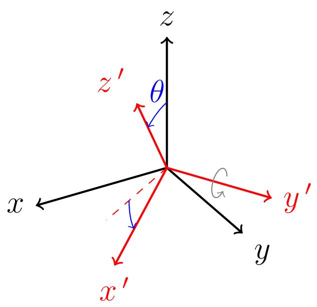
Analyzing (4.15), the singularity problem become evident in a mathematical fashion. When the angle θ is equal to $0°$ or $\pm 180°$, the term $\sin\theta$ goes to zero and the equation becomes invalid (because you cannot divide by zero). This inability of the Euler angles does not appear in the rotation matrix, but the kinematic equation clearly revels it.

Although singularities will always exist when utilizing Euler angles, one way to circumvent this problem is to adopt a sequence of rotations such that the orientation in which they occur is far from the nominal operating orientation of the system. For example, another sequence of rotations is the z-y-x sequence:

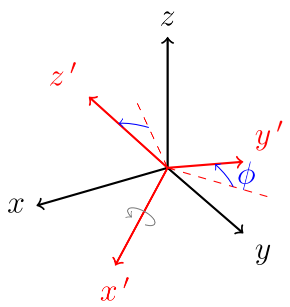
- Rotation of ψ (yaw) around the $z\,'$-axis (Fig. 12a)
- Rotation of θ (pitch) around the $y\,'$-axis (Fig. 12b)
- Rotation of ϕ (roll) around the $x\,'$-axis (Fig. 12c)



(a) First rotation



(b) Second rotation



(c) Third rotation

Figure 12: Euler angles (z-y-x sequence)

In this case, singularities still exist, but now they occur when the angle θ is equal to $\pm 90°$ and no longer $\pm 0°$ or $\pm 180°$. Regardless of the rotation sequence adopted, it is always the second rotation that defines the singularity.

The total rotation matrix R is now given by

$$R = R_{x_\phi} R_{y_\theta} R_{z_\psi} \,, \tag{4.16}$$

and R_{z_ψ}, R_{y_θ} and R_{x_ϕ} are each individual rotation matrix given by

$$R_{z_\psi} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} , \quad R_{y_\theta} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} , \quad R_{x_\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} . \tag{4.17}$$

This means that the rotation matrix is now different

$$R = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & \sin\phi\cos\theta \\ \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & \cos\phi\cos\theta \end{bmatrix}. \tag{4.18}$$

Doing the same procedure, but now with this new rotation matrix, the kinematic equation of a rigid body for the z-y-x rotation sequence is obtained

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \frac{1}{\cos\theta} \begin{bmatrix} \cos\theta & \sin\phi\sin\theta & \cos\phi\sin\theta \\ 0 & \cos\phi\cos\theta & -\sin\phi\cos\theta \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \tag{4.19}$$

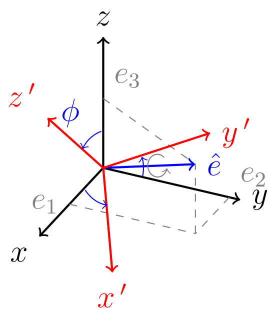
As can be seen from the kinematic equation, now there is a term $\cos\theta$ in the denominator, which will be zero when the angle θ is equal to $\pm 90°$. As it had already been anticipated, the singularity problem was not solved, it just changed from one condition to another. If it is desired to completely solve the singularity problem, there is only one thing to be done: utilize a different representation.

4.2.2 Quaternions

Another representation for specifying the orientation of a rigid body, not as used as the Euler angles but which has been growing quite a lot lately, are the quaternions.

Unlike the Euler angles, based on the property that any orientation of a rigid body can be described with a sequence of three consecutive rotations around predefined axis, quaternions are based on the property[1] that any orientation of a rigid body can be described with a single rotation around an arbitrary axis: the eigenaxis (the real eigenvector of the transformation matrix between body and inertial axes). For this, quaternions require four parameters: three to describe the eigenaxis $\hat{e}$ coordinates plus one to describe the rotation angle ϕ (Fig. 13a).

[1]Interestingly, this other property was also formulated by Leonard Euler.



(a) Single rotation

Figure 13: Quaternions

The disadvantages of Euler angles (singularities and trigonometric functions) does not exist when dealing with quaternions, but on the other hand, all its advantages (three intuitive parameters) do not exist either. So, you should be asking: why use quaternions then? The answer is simple: first, because the disadvantages of the Euler angles are much more critical than those of the quaternions; second, and most important, because quaternions can be represented utilizing vector notation, which allow one to write down quite complex dynamics completely by hand.

Quaternions possess a not very common and somewhat complex algebra, that is going to be briefly described in this section. Before diving into quaternion algebra [9], the rotation around a specific axis (the property which quaternions are based on) will be derived.

4.2.2.1 Spacial rotation

Let $\vec{r}$ be an arbitrary vector to be rotated around a unit vector $\hat{e}$ by an angle ϕ generating a rotated vector $\vec{r}'$ (Fig. 14).

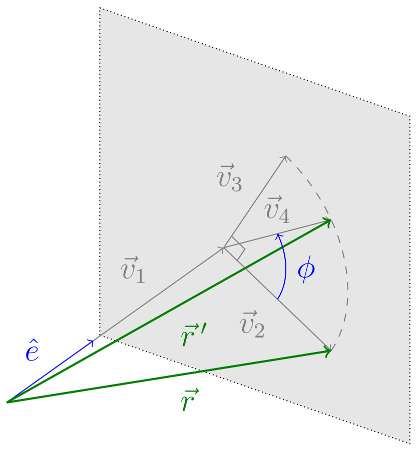


Figure 14: Rodrigues' rotation geometry

Projection vectors $\vec{v}_1$, $\vec{v}_2$, $\vec{v}_3$ and $\vec{v}_4$ can be written in terms of vector $\vec{r}$, unit vector $\hat{e}$ and angle ϕ as

$$\vec{v}_1 = (\vec{r} \cdot \hat{e})\hat{e}, \tag{4.20}$$

$$\vec{v}_2 = \vec{r} - \vec{v}_1, \tag{4.21}$$

$$\vec{v}_3 = \vec{v}_2 \times \hat{e}, \tag{4.22}$$

$$\vec{v}_4 = \vec{v}_2 \cos\phi + \vec{v}_3 \sin\phi. \tag{4.23}$$

The rotated vector $\vec{r}\,'$ can be written in terms of projection vectors such that

$$\vec{r}\,' = \vec{v}_1 + \vec{v}_4\,. \tag{4.24}$$

Substituting (4.20)–(4.23) in (4.24) results in

$$\begin{aligned}
\vec{r}\,' &= \vec{v}_1 + \vec{v}_4 \\
&= (\vec{r} \cdot \hat{e})\hat{e} + \vec{v}_2 \cos\phi + \vec{v}_3 \sin\phi \\
&= (\vec{r} \cdot \hat{e})\hat{e} + (\vec{r} - \vec{v}_1) \cos\phi + (\vec{v}_2 \times \hat{e}) \sin\phi \\
&= (\vec{r} \cdot \hat{e})\hat{e} + (\vec{r} - (\vec{r} \cdot \hat{e})\hat{e}) \cos\phi + ((\vec{r} - \vec{v}_1) \times \hat{e}) \sin\phi \\
&= (\vec{r} \cdot \hat{e})\hat{e} + (\vec{r} - (\vec{r} \cdot \hat{e})\hat{e}) \cos\phi + ((\vec{r} - (\vec{r} \cdot \hat{e})\hat{e}) \times \hat{e}) \sin\phi \\
&= (\vec{r} \cdot \hat{e})\hat{e} + \vec{r}\cos\phi - (\vec{r} \cdot \hat{e})\hat{e}\cos\phi + \vec{r} \times \hat{e}\sin\phi - \cancel{(\vec{r} \cdot \hat{e})\hat{e} \times \hat{e}\sin\phi} \\
&= (1 - \cos\phi)(\vec{r} \cdot \hat{e})\hat{e} + \cos\phi\vec{r} + \sin\phi(\vec{r} \times \hat{e})\,. \qquad (4.25)
\end{aligned}$$

Equation (4.25) is the Rodrigues' rotation formula that describes the rotation of a vector $\vec{r}$ by an angle ϕ along a unit vector $\hat{e}$. It was named after French mathematician Olinde Rodrigues (1795 - 1851).

4.2.2.2 Quaternion fundamentals

Quaternion algebra can be generated from the following properties

$$i^2 = j^2 = k^2 = ijk = -1\,. \tag{4.26}$$

By left and right multiplying (4.26), together with associativity and distributivity, the following multiplication rules arise

$$\begin{aligned} ij &= k\,, & ji &= -k\,, \\ jk &= i\,, & kj &= -i\,, \\ ki &= j\,, & ik &= -j\,, \end{aligned} \tag{4.27}$$

as it can be seen, the product is non-commutative.

4.2.2.3 Quaternion notation

A quaternion q is a set of four parameters, a real value q_0 and three imaginary values q_1, q_2 and q_3 such that

$$q = q_0 + q_1 i + q_2 j + q_3 k\,. \tag{4.28}$$

This notation proves itself to be very unpractical, this is why a quaternion can also be represented as a four dimension column vector composed of a real value q_0 and a vectorial imaginary value $\vec{q} = [q_1 \quad q_2 \quad q_3]^T$ such that

$$q = \begin{bmatrix} q_0 \\ \vec{q} \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}\,. \tag{4.29}$$

The conjugate of a quaternion is defined as

$$\bar{q} = \begin{bmatrix} q_0 \\ -\vec{q} \end{bmatrix} = \begin{bmatrix} q_0 \\ -q_1 \\ -q_2 \\ -q_3 \end{bmatrix}\,, \tag{4.30}$$

and its norm (a non-negative real value) as

$$|q| = \sqrt{q_0^2 + \vec{q} \cdot \vec{q}} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}\,. \tag{4.31}$$

4.2.2.4 Quaternion product

From the rules given in (4.26) and (4.27), the product of two quaternions q and r (represented by the $\circ$ operator) can be derived

$$\begin{aligned} q \circ r = &(q_0 + q_1 i + q_2 j + q_3 k)(r_0 + r_1 i + r_2 j + r_3 k) \\ =&q_0 r_0 + q_0 r_1 i + q_0 r_2 j + q_0 r_3 k + \\ &q_1 r_0 i + q_1 r_1 ii + q_1 r_2 ij + q_1 r_3 ik + \\ &q_2 r_0 j + q_2 r_1 ji + q_2 r_2 jj + q_2 r_3 jk + \\ &q_3 r_0 k + q_3 r_1 ki + q_3 r_2 kj + q_3 r_3 kk \\ =&q_0 r_0 + q_0 r_1 i + q_0 r_2 j + q_0 r_3 k + \\ &q_1 r_0 i - q_1 r_1 + q_1 r_2 k - q_1 r_3 j + \\ &q_2 r_0 j - q_2 r_1 k - q_2 r_2 + q_2 r_3 i + \\ &q_3 r_0 k + q_3 r_1 j - q_3 r_2 i - q_3 r_3 \,. \end{aligned} \tag{4.32}$$

Re-arranging the terms

$$\begin{aligned} q \circ r =&(q_0 r_0 - q_1 r_1 - q_2 r_2 - q_3 r_3) + \\ &(q_0 r_1 + q_1 r_0 + q_2 r_3 - q_3 r_2) i + \\ &(q_0 r_2 + q_2 r_0 + q_3 r_1 - q_1 r_3) j + \\ &(q_0 r_3 + q_3 r_0 + q_1 r_2 - q_2 r_1) k \,. \end{aligned} \tag{4.33}$$

By inspecting the above equation, a simplified notation of quaternion product can be defined utilizing vector algebra

$$q \circ r = \begin{bmatrix} q_0 r_0 - \vec{q} \cdot \vec{r} \\ q_0 \vec{r} + r_0 \vec{q} + \vec{q} \times \vec{r} \end{bmatrix} \tag{4.34}$$

Since (4.34) is linear in r, it can also be written in matrix-vector product form

$$q \circ r = \begin{bmatrix} q_0 & -\vec{q}^T \\ \vec{q} & q_0 I_{3\times3} + \tilde{q} \end{bmatrix} \begin{bmatrix} r_0 \\ \vec{r} \end{bmatrix} , \tag{4.35}$$

where $\tilde{q}$ is the rotation quaternion vector represented as a skew-symmetric matrix corresponding to its cross product

$$\tilde{q} = \vec{q}\times = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} . \tag{4.36}$$

Note that the matrix corresponding to its quaternion product can also be written as

$$q \circ = \begin{bmatrix} | & | \\ q & G(q)^T \\ | & | \end{bmatrix} , \tag{4.37}$$

where $G(q)$, usually called the Lagrange matrix, named after Italian mathematician Joseph-Louis Lagrange (1736 - 1813), is given by

$$G(q) = \begin{bmatrix} -\vec{q} & q_0 I_{3\times3} - \tilde{q} \end{bmatrix} . \tag{4.38}$$

From (4.34), it can be seen that

$$\overline{q \circ r} = \bar{r} \circ \bar{q} \,, \tag{4.39}$$

and

$$q \circ \bar{q} = \bar{q} \circ q = \begin{bmatrix} |q|^2 \\ \vec{0} \end{bmatrix} . \tag{4.40}$$

Moreover, if the quaternion has unit norm, *i.e.*, $|q| = 1$, two other properties are valid

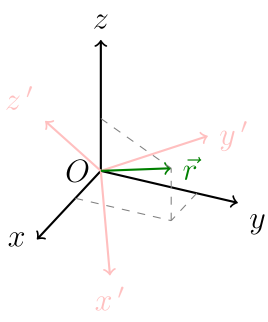
$$q \circ \bar{q} = \bar{q} \circ q = \begin{bmatrix} 1 \\ \vec{0} \end{bmatrix} , \tag{4.41}$$

and

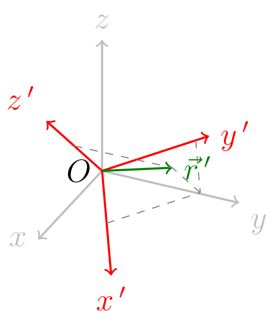
$$G(q)G(q)^T = I_{3\times3} \,. \tag{4.42}$$

4.2.2.5 Rotation quaternion

Let $\vec{r}$ be an arbitrary fixed vector described in an inertial coordinate frame $O : \{x, y, z\}$ (Fig. 15a) and $\vec{r}\,'$ be the same vector but described in a body fixed coordinate frame $O : \{x', y', z'\}$ (Fig. 15b).



(a) Inertial coordinate frame



(b) Body fixed coordinate frame

Figure 15: Same vector described in different coordinate frames

To transform $\vec{r}$ into $\vec{r}\,'$, consider the following quaternion multiplication

$$r\,' = \bar{q} \circ r \circ q\,, \tag{4.43}$$

where r and $r\,'$ are quaternions with zero real part and with the vectors $\vec{r}$ and $\vec{r}\,'$ in their imaginary part

$$r = \begin{bmatrix} 0 \\ \vec{r} \end{bmatrix}\,, \qquad r\,' = \begin{bmatrix} 0 \\ \vec{r}\,' \end{bmatrix}\,, \tag{4.44}$$

and q is the rotation quaternion whose components are defined in terms fo the eigenaxis $\hat{e}$ and rotation angle ϕ, such that

$$q = \begin{bmatrix} \cos\frac{\phi}{2} \\ \hat{e}\sin\frac{\phi}{2} \end{bmatrix} = \begin{bmatrix} \cos\frac{\phi}{2} \\ e_1\sin\frac{\phi}{2} \\ e_2\sin\frac{\phi}{2} \\ e_3\sin\frac{\phi}{2} \end{bmatrix}\,. \tag{4.45}$$

Note that, because the eigenaxis have unit norm, *i.e.*, $|\hat{e}| = 1$, the rotation quaternion also has unit norm

$$\begin{aligned}
|q| &= \sqrt{q_0^2 + \vec{q}\cdot\vec{q}} \\
&= \sqrt{\left(\cos\frac{\phi}{2}\right)^2 + \left(\hat{e}\sin\frac{\phi}{2}\right)\cdot\left(\hat{e}\sin\frac{\phi}{2}\right)} \\
&= \sqrt{\cos^2\frac{\phi}{2} + \cancelto{1}{|\hat{e}|}\sin^2\frac{\phi}{2}} \\
&= \sqrt{\cos^2\frac{\phi}{2} + \sin^2\frac{\phi}{2}} \\
&= 1\,, \qquad (4.46)
\end{aligned}$$

and differentiating (4.46), another property can be derived

$$\begin{aligned}
\frac{d}{dt}\left(|q|^2\right) &= \frac{d}{dt}(1) \\
\frac{d}{dt}\left(q_0^2 + \vec{q}\cdot\vec{q}\right) &= 0 \\
2q_0\dot{q}_0 + \dot{\vec{q}}\cdot\vec{q} + \vec{q}\cdot\dot{\vec{q}} &= 0 \\
\cancel{2}q_0\dot{q}_0 + \cancel{2}\vec{q}\cdot\dot{\vec{q}} &= 0 \\
q_0\dot{q}_0 + \vec{q}\cdot\dot{\vec{q}} &= 0\,. \qquad (4.47)
\end{aligned}$$

Going back to (4.43) and expanding it yields

$$\begin{aligned}
\bar{q}\circ r\circ q &= \begin{bmatrix} q_0 \\ -\vec{q} \end{bmatrix} \circ \begin{bmatrix} 0 \\ \vec{r} \end{bmatrix} \circ \begin{bmatrix} q_0 \\ \vec{q} \end{bmatrix} \\
&= \begin{bmatrix} \vec{q}\cdot\vec{r} \\ q_0\vec{r} - \vec{q}\times\vec{r} \end{bmatrix} \circ \begin{bmatrix} q_0 \\ \vec{q} \end{bmatrix} \\
&= \begin{bmatrix} (\vec{q}\cdot\vec{r})q_0 - (q_0\vec{r} - \vec{q}\times\vec{r})\cdot\vec{q} \\ (\vec{q}\cdot\vec{r})\vec{q} + q_0(q_0\vec{r} - \vec{q}\times\vec{r}) + (q_0\vec{r} - \vec{q}\times\vec{r})\times\vec{q} \end{bmatrix} \\
&= \begin{bmatrix} \cancel{(\vec{q}\cdot\vec{r})q_0} - \cancel{q_0(\vec{r}\cdot\vec{q})} + (\vec{q}\times\vec{r})\cdot\vec{q} \\ (\vec{q}\cdot\vec{r})\vec{q} + q_0^2\vec{r} - q_0(\vec{q}\times\vec{r}) + q_0(\vec{r}\times\vec{q}) - \vec{q}\times\vec{r}\times\vec{q} \end{bmatrix} \\
&= \begin{bmatrix} \cancelto{0}{(\vec{q}\times\vec{r})\cdot\vec{q}} \\ (\vec{q}\cdot\vec{r})\vec{q} + q_0^2\vec{r} + q_0(\vec{r}\times\vec{q}) + q_0(\vec{r}\times\vec{q}) - (\vec{q}\cdot\vec{q})\vec{r} + (\vec{q}\cdot\vec{r})\vec{q} \end{bmatrix} \\
&= \begin{bmatrix} 0 \\ 2(\vec{q}\cdot\vec{r})\vec{q} + q_0^2\vec{r} + 2q_0(\vec{r}\times\vec{q}) - (\vec{q}\cdot\vec{q})\vec{r} \end{bmatrix}\,. \qquad (4.48)
\end{aligned}$$

Substituting the rotation quaternion from (4.45) in (4.48) results in

$$\begin{aligned}
\bar{q} \circ r \circ q &= \begin{bmatrix} 0 \\ 2(\vec{q}\cdot\vec{r})\vec{q} + q_0^2\vec{r} + 2q_0(\vec{r}\times\vec{q}) - (\vec{q}\cdot\vec{q})\vec{r} \end{bmatrix} \\
&= \begin{bmatrix} 0 \\ 2(\hat{e}\sin\frac{\phi}{2}\cdot\vec{r})\hat{e}\sin\frac{\phi}{2} + \cos^2\frac{\phi}{2}\vec{r} + 2\cos\frac{\phi}{2}(\vec{r}\times\hat{e}\sin\frac{\phi}{2}) - (\hat{e}\sin\frac{\phi}{2}\cdot\hat{e}\sin\frac{\phi}{2})\vec{r} \end{bmatrix} \\
&= \begin{bmatrix} 0 \\ 2\sin^2\frac{\phi}{2}(\hat{e}\cdot\vec{r})\hat{e} + \cos^2\frac{\phi}{2}\vec{r} + 2\cos\frac{\phi}{2}\sin\frac{\phi}{2}(\vec{r}\times\hat{e}) - \sin^2\frac{\phi}{2}\vec{r} \end{bmatrix} \\
&= \begin{bmatrix} 0 \\ (1-\cos\phi)(\hat{e}\cdot\vec{r})\hat{e} + \cos\phi\vec{r} + \sin\phi(\vec{r}\times\hat{e}) \end{bmatrix}, \qquad (4.49)
\end{aligned}$$

which is the Rodrigues' rotation formula derived in (4.25).

For the inverse transformation, one just needs to swap the rotation quaternion with it conjugate

$$r = q \circ r' \circ \bar{q}\,. \qquad (4.50)$$

Moreover, since vector $\vec{r}$ is fixed in the inertial coordinate frame, a rotation quaternion q can be used to represent the rotation of the body fixed coordinate frame with respect to the inertial coordinate frame

$$\vec{r}' = R(q)\vec{r}, \qquad (4.51)$$

where $R(q)$ is the rotation matrix in terms of the rotation quaternion, given by

$$R(q) = \left[q_0 I_{3\times3} + \vec{q}\,\vec{q}^T + 2q_0\tilde{q} + \tilde{q}^2\right]. \qquad (4.52)$$

4.2.2.6 Kinematic equation

Let us suppose now that the body fixed coordinate frame is in rotational motion around the origin O (Fig. 16).

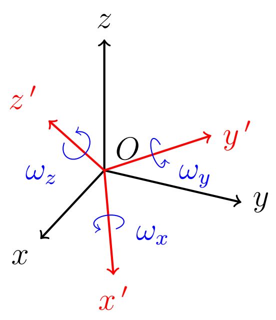


Figure 16: Quaternions and angular velocity

Its angular velocity vector $\vec{\omega}\,'$ is given by

$$\vec{\omega}\,' = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} . \tag{4.53}$$

Note that this is the angular velocity with respect to the inertial coordinate frame but described along the body fixed coordinate frame axes.

Let $\omega\,'$ be a quaternion with zero real part and with the vector $\vec{\omega}\,'$ in its imaginary part

$$\omega\,' = \begin{bmatrix} 0 \\ \vec{\omega}\,' \end{bmatrix} . \tag{4.54}$$

Since vector $\vec{r}$ is fixed in the inertial coordinate frame, its time derivative, as seen by the inertial coordinate frame, is zero

$$\dot{r} = \begin{bmatrix} 0 \\ \dot{\vec{r}} \end{bmatrix} = \begin{bmatrix} 0 \\ \vec{0} \end{bmatrix} . \tag{4.55}$$

In turn, its time derivative, as seen by the body fixed coordinate frame, depends on the body fixed coordinate frame angular velocity vector

$$\dot{r}\,' = \begin{bmatrix} 0 \\ \dot{\vec{r}}' \end{bmatrix} = \begin{bmatrix} 0 \\ -\vec{\omega}\,' \times \vec{r} \end{bmatrix} . \tag{4.56}$$

The minus sign appears because, if the body coordinate frame rotates in one direction, the vector will be seen by the body coordinate frame as rotating in the opposite direction.

Since quaternions $r\,'$ and $\omega\,'$ have zero real part, (4.56) is equivalent to

$$\dot{r}\,' = -\omega\,' \circ r\,' . \tag{4.57}$$

Differentiating (4.43) and using (4.50) and (4.55) yields

$$\begin{aligned}
\dot{r}' &= \frac{d}{dt}\,(r') \\
&= \frac{d}{dt}\,(\bar{q} \circ r \circ q) \\
&= \dot{\bar{q}} \circ r \circ q + \bar{q} \circ \cancelto{0}{\dot{r}} \circ q + \bar{q} \circ r \circ \dot{q} \\
&= \dot{\bar{q}} \circ (q \circ r' \circ \bar{q}) \circ q + \bar{q} \circ (q \circ r' \circ \bar{q}) \circ \dot{q} \\
&= \dot{\bar{q}} \circ q \circ r' \circ \cancel{\bar{q} \circ q} + \cancel{\bar{q} \circ q} \circ r' \circ \bar{q} \circ \dot{q} \\
&= \dot{\bar{q}} \circ q \circ r' + r' \circ \bar{q} \circ \dot{q}
\end{aligned} \tag{4.58}$$

Inspecting the terms $\bar{q} \circ \dot{q}$ and $\dot{\bar{q}} \circ q$ and taking into account (4.47)

$$\begin{aligned}
\bar{q} \circ \dot{q} &= \begin{bmatrix} q_0 \\ -\vec{q} \end{bmatrix} \circ \begin{bmatrix} \dot{q}_0 \\ \dot{\vec{q}} \end{bmatrix} \\
&= \begin{bmatrix} \cancelto{0}{q_0\dot{q}_0 + \vec{q} \cdot \dot{\vec{q}}} \\ q_0\dot{\vec{q}} - \dot{q}_0\vec{q} - \vec{q} \times \dot{\vec{q}} \end{bmatrix} \\
&= \begin{bmatrix} 0 \\ q_0\dot{\vec{q}} - \dot{q}_0\vec{q} - \vec{q} \times \dot{\vec{q}} \end{bmatrix},
\end{aligned} \tag{4.59}$$

$$\begin{aligned}
\dot{\bar{q}} \circ q &= \begin{bmatrix} \dot{q}_0 \\ -\dot{\vec{q}} \end{bmatrix} \circ \begin{bmatrix} q_0 \\ \vec{q} \end{bmatrix} \\
&= \begin{bmatrix} \cancelto{0}{\dot{q}_0 q_0 + \dot{\vec{q}} \cdot \vec{q}} \\ \dot{q}_0\vec{q} - q_0\dot{\vec{q}} - \dot{\vec{q}} \times \vec{q} \end{bmatrix} \\
&= \begin{bmatrix} 0 \\ -q_0\dot{\vec{q}} + \dot{q}_0\vec{q} + \vec{q} \times \dot{\vec{q}} \end{bmatrix},
\end{aligned} \tag{4.60}$$

it is possible to note that

$$\bar{q} \circ \dot{q} = -\dot{\bar{q}} \circ q\,. \tag{4.61}$$

By making use of this property, (4.58) can be written as:

$$\begin{aligned}
\dot{r}' &= \dot{\bar{q}} \circ q \circ r' + r' \circ \bar{q} \circ \dot{q} \\
&= -\bar{q} \circ \dot{q} \circ r' - \bar{q} \circ \dot{q} \circ r' \\
&= -2\bar{q} \circ \dot{q} \circ r'\,.
\end{aligned} \tag{4.62}$$

Comparing (4.62) with (4.57), it is possible to obtain the angular velocity quaternion in terms of the rotation quaternion and its time derivative:

$$\omega' = 2\bar{q} \circ \dot{q}\,, \tag{4.63}$$

which can also be rewritten by making use of (4.61) as

$$\omega' = -2\dot{\bar{q}} \circ q\,. \tag{4.64}$$

By left-multiplying both sides of (4.63) with q and using (4.41), the quaternion time derivative $\dot{q}$ can be isolated

$$\begin{aligned} q \circ \omega' &= 2\cancel{q \circ \bar{q}} \circ \dot{q} \\ q \circ \omega' &= 2\dot{q} \\ \dot{q} &= \frac{1}{2} q \circ \omega'\,. \end{aligned} \tag{4.65}$$

Equation (4.65) is the rotational kinematic equation of a rigid body utilizing quaternions. It can be seen that it is much simpler than the one utilizing Euler angles. Although now there is an additional term, all trigonometric operations no longer exist, and singularities are also gone.

Moreover, because quaternion ω has zero real part, (4.63), (4.64) and (4.65) can be written in vector notation utilizing (4.37) as

$$\underbrace{\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}}_{\vec{\omega}'} = 2 \underbrace{\begin{bmatrix} -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{bmatrix}}_{G(q)} \underbrace{\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}}_{\dot{q}}, \tag{4.66}$$

$$\underbrace{\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}}_{\vec{\omega}'} = -2 \underbrace{\begin{bmatrix} -\dot{q}_1 & \dot{q}_0 & \dot{q}_3 & -\dot{q}_2 \\ -\dot{q}_2 & -\dot{q}_3 & \dot{q}_0 & \dot{q}_1 \\ -\dot{q}_3 & \dot{q}_2 & -\dot{q}_1 & \dot{q}_0 \end{bmatrix}}_{G(\dot{q})} \underbrace{\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}}_{q}, \tag{4.67}$$

$$\underbrace{\begin{bmatrix}\dot{q}_0\\ \dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\end{bmatrix}}_{\dot{q}} = \frac{1}{2}\underbrace{\begin{bmatrix}-q_1 & -q_2 & -q_3\\ q_0 & -q_3 & q_2\\ q_3 & q_0 & -q_1\\ -q_2 & q_1 & q_0\end{bmatrix}}_{G(q)^T}\underbrace{\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix}}_{\vec{\omega}\,'}, \tag{4.68}$$

and also from (4.37), it is possible to demonstrate that

$$\underbrace{\begin{bmatrix}0 & -\omega_z & \omega_y\\ \omega_z & 0 & -\omega_x\\ -\omega_y & \omega_x & 0\end{bmatrix}}_{\tilde{\omega}\,'} = 2\underbrace{\begin{bmatrix}-q_1 & q_0 & q_3 & -q_2\\ -q_2 & -q_3 & q_0 & q_1\\ -q_3 & q_2 & -q_1 & q_0\end{bmatrix}}_{G(q)}\underbrace{\begin{bmatrix}-\dot{q}_1 & -\dot{q}_2 & -\dot{q}_3\\ \dot{q}_0 & -\dot{q}_3 & \dot{q}_2\\ \dot{q}_3 & \dot{q}_0 & -\dot{q}_1\\ -\dot{q}_2 & \dot{q}_1 & \dot{q}_0\end{bmatrix}}_{G(\dot{q})^T}. \tag{4.69}$$

4.3 Kinetics

In this section, the rotational kinetic equations of a the Cubli will be derived. These equations consider only the angular velocities and moments applied on the rigid body, not focusing on the orientation of it.

A convenient way to derive the kinetic equations is to utilize the Lagrange equations. This method allows one to deal with energy functions rather than forces and accelerations as with Newton-Euler equations. Because energy is a scalar while forces and accelerations are vectors, in a multi-body system, the kinetic and potential energy can be computed for each moving body independently and then added together to form the energy of the complete system. This is an important advantage of the Lagrange equations.

The first step is to define the total kinetic and potential energy of the system, and only then, the Lagrange equation will be introduced and utilized.

4.3.1 Kinetic Energy

Since the Cubli is rotating aroung the pivot point O (articulation vertex), all its kinetic energy can be interpreted as angular kinetic energy, as long as their inertia matrices are described accordingly.

The Cubli total kinetic energy is the sum of the kinetic energy of each moving body

$$T = T_s + \sum_{i=1}^{3} T_{wi}\,. \tag{4.70}$$

where T_s is the kinetic energy of the structure and T_{wi} is the kinetic energy of the i-th reaction wheel.

The angular kinetic energy depends on the angular velocity and inertia matrix. Both of these parameters will be defined for each one of the bodies.

4.3.1.1 Angular velocities

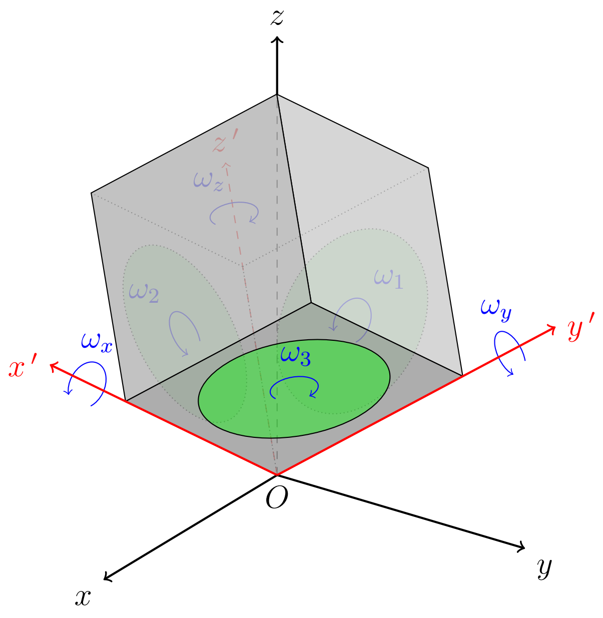


Figure 17: Cubli angular velocities

Structure

Let $\vec{\omega}_s{}'$ be the structure angular velocity vector described along the body fixed coordinate frame but with respect to the inertial coordinate frame (Fig. 17), given by:

$$\vec{\omega}_s{}' = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \tag{4.71}$$

Reaction wheels

Let $\vec{\omega}_{w1}{}'$, $\vec{\omega}_{w2}{}'$ and $\vec{\omega}_{w3}{}'$ be the reaction wheels relative angular velocity vectors described in and with respect to the body fixed coordinate frame (Fig. 17), given by:

$$\vec{\omega}_{w1}{}' = \begin{bmatrix} \omega_1 \\ 0 \\ 0 \end{bmatrix} \quad , \quad \vec{\omega}_{w2}{}' = \begin{bmatrix} 0 \\ \omega_2 \\ 0 \end{bmatrix} \quad , \quad \vec{\omega}_{w3}{}' = \begin{bmatrix} 0 \\ 0 \\ \omega_3 \end{bmatrix} \quad . \tag{4.72}$$

Note that this angular velocity vectors are relative; hence, to obtain the reaction wheel angular velocity vector with respect to the inertial coordinate frame, the structure angular velocity vector needs to be added (since the reaction wheels are rotating together with the structure).

4.3.1.2 Inertia matrices

The structure can be approximated to a cube of side length l, mass m_s and moment of inertia around its principal axes $I_{s_{xx}} = I_{s_{yy}} = I_{s_{zz}}$, whereas each reaction wheel can be approximated to a disc of mass m_w, moment of inertia around its axial principal axis $I_{w_{xx}}$ and moment of inertia around its perpendicular principal axes $I_{w_{yy}} = I_{w_{zz}}$. These parameters were obtained from the CAD version of the Cubli and are given in Table 1.

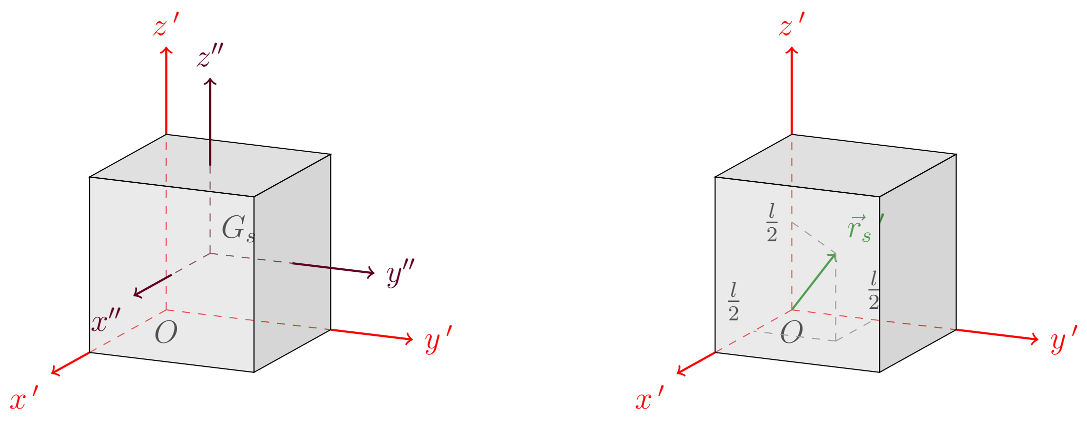
Table 1: Cubli parameters

Parameter	Value
l	0.15 m
m_s	0.40 kg
m_w	0.15 kg
$I_{s_{xx}}$	2.00×10^{-3} kg.m^2
$I_{w_{xx}}$	1.25×10^{-4} kg.m^2
$I_{w_{yy}}$	4.00×10^{-5} kg.m^2

Structure

Let I_{s_G} be the structure inertia tensor on its center of mass G_s with respect to the $x''y''z''$ axes (Fig. 18a) and $\vec{r}_s$ be the vector from the pivot point O to the structure center of mass G_s (Fig. 18b), given by

$$I_{s_G} = \begin{bmatrix} I_{s_{xx}} & 0 & 0 \\ 0 & I_{s_{xx}} & 0 \\ 0 & 0 & I_{s_{xx}} \end{bmatrix} \quad , \quad \vec{r}_s = \begin{bmatrix} l/2 \\ l/2 \\ l/2 \end{bmatrix} \quad . \tag{4.73}$$



(a) Moments of inertia (b) Vector

Figure 18: Structure parameters

Because of symmetry, all moments of inertia are the same.

With both of these values, it is possible to calculate I_{s_O}, the structure inertia matrix aroung the pivot point O with respect to the body fixed coordinate frame, by applying the Huygens-Steiner theorem named after Dutch mathematician Christiaan Huygens (1629-1695) and Swiss mathematician Jakob Steiner (1796 - 1863)

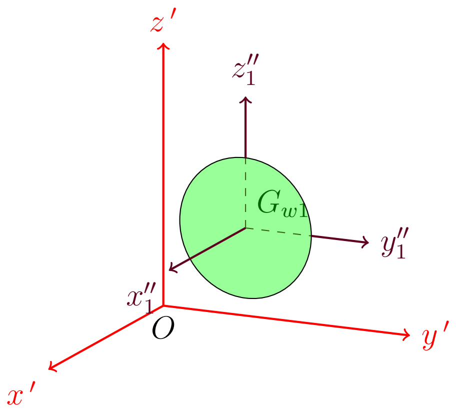
$$\begin{aligned}
I_{s_O} &= I_{s_G} + m_s \tilde{r}_s \tilde{r}_s^T \\
&= \begin{bmatrix} I_{s_{xx}} & 0 & 0 \\ 0 & I_{s_{xx}} & 0 \\ 0 & 0 & I_{s_{xx}} \end{bmatrix} + m_s \begin{bmatrix} 0 & -\frac{l}{2} & \frac{l}{2} \\ \frac{l}{2} & 0 & -\frac{l}{2} \\ -\frac{l}{2} & \frac{l}{2} & 0 \end{bmatrix} \begin{bmatrix} 0 & \frac{l}{2} & -\frac{l}{2} \\ -\frac{l}{2} & 0 & \frac{l}{2} \\ \frac{l}{2} & -\frac{l}{2} & 0 \end{bmatrix} \\
&= \begin{bmatrix} I_{s_{xx}} & 0 & 0 \\ 0 & I_{s_{xx}} & 0 \\ 0 & 0 & I_{s_{xx}} \end{bmatrix} + m_s \begin{bmatrix} \frac{l^2}{2} & -\frac{l^2}{4} & -\frac{l^2}{4} \\ -\frac{l^2}{4} & \frac{l^2}{2} & -\frac{l^2}{4} \\ -\frac{l^2}{4} & -\frac{l^2}{4} & \frac{l^2}{2} \end{bmatrix} \\
&= \begin{bmatrix} I_{s_{xx}} + m_s \frac{l^2}{2} & -m_s \frac{l^2}{4} & -m_s \frac{l^2}{4} \\ -m_s \frac{l^2}{4} & I_{s_{xx}} + m_s \frac{l^2}{2} & -m_s \frac{l^2}{4} \\ -m_s \frac{l^2}{4} & -m_s \frac{l^2}{4} & I_{s_{xx}} + m_s \frac{l^2}{2} \end{bmatrix} . \qquad (4.74)
\end{aligned}$$

Reaction wheels

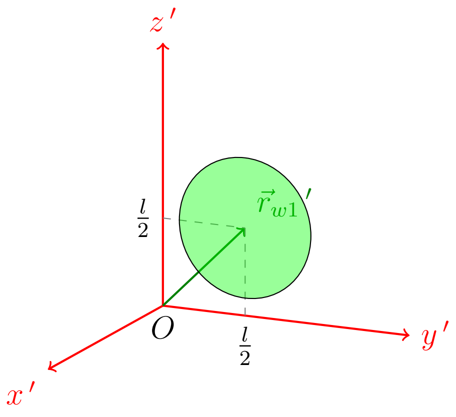
Let I_{w1_G} be reaction wheel 1 inertia tensor on its center of mass G_{w1} with respect to the $x_1''y_1''z_1''$ axes (Fig. 19a) and $\vec{r}_{w1}$ be the vector from the pivot point O to reaction wheel

1 center of mass G_{w1} (Fig. 19b), given by

$$I_{w1_G} = \begin{bmatrix} I_{w_{xx}} & 0 & 0 \\ 0 & I_{w_{yy}} & 0 \\ 0 & 0 & I_{w_{yy}} \end{bmatrix}, \quad \vec{r}_{w1} = \begin{bmatrix} 0 \\ {}^{l}\!/_{2} \\ {}^{l}\!/_{2} \end{bmatrix}. \tag{4.75}$$



(a) Moments of inertia



(b) Vector

Figure 19: Reaction wheel 1 parameters

Because of symmetry, the moment of inertia around the y''_1 and z''_1 axes are the same. Moreover, in reality, vector $\vec{r}_{w1}{}'$ should have a small displacement in the x' direction, but this is being neglected.

With both of these values, it is possible to calculate I_{w1_O}, the reaction wheel 1 inertia matrix around the pivot point O with respect to the body fixed coordinate frame, the same way it was done for the structure

$$\begin{aligned} I_{w1_O} &= I_{w1_G} + m_w \tilde{r}_{w1} \tilde{r}_{w1}^T \\ &= \begin{bmatrix} I_{w_{xx}} & 0 & 0 \\ 0 & I_{w_{yy}} & 0 \\ 0 & 0 & I_{w_{yy}} \end{bmatrix} + m_w \begin{bmatrix} 0 & -\frac{l}{2} & \frac{l}{2} \\ \frac{l}{2} & 0 & 0 \\ -\frac{l}{2} & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & \frac{l}{2} & -\frac{l}{2} \\ -\frac{l}{2} & 0 & 0 \\ \frac{l}{2} & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} I_{w_{xx}} & 0 & 0 \\ 0 & I_{w_{yy}} & 0 \\ 0 & 0 & I_{w_{yy}} \end{bmatrix} + m_w \begin{bmatrix} \frac{l^2}{2} & 0 & 0 \\ 0 & \frac{l^2}{4} & -\frac{l^2}{4} \\ 0 & -\frac{l^2}{4} & \frac{l^2}{4} \end{bmatrix} \\ &= \begin{bmatrix} I_{w_{xx}} + m_w \frac{l^2}{2} & 0 & 0 \\ 0 & I_{w_{yy}} + m_w \frac{l^2}{4} & -m_w \frac{l^2}{4} \\ 0 & -m_w \frac{l^2}{4} & I_{w_{yy}} + m_w \frac{l^2}{4} \end{bmatrix}. \end{aligned} \tag{4.76}$$

Since all three reaction wheels are identical and differ only in their position, orientation and axis around which they rotate, it can be inferred that

$$I_{w2_G} = \begin{bmatrix} I_{w_{yy}} & 0 & 0 \\ 0 & I_{w_{xx}} & 0 \\ 0 & 0 & I_{w_{yy}} \end{bmatrix} \,, \quad \vec{r}_{w2} = \begin{bmatrix} l/2 \\ 0 \\ l/2 \end{bmatrix} \,, \tag{4.77}$$

$$I_{w3_G} = \begin{bmatrix} I_{w_{yy}} & 0 & 0 \\ 0 & I_{w_{yy}} & 0 \\ 0 & 0 & I_{w_{xx}} \end{bmatrix} \,, \quad \vec{r}_{w3} = \begin{bmatrix} l/2 \\ l/2 \\ 0 \end{bmatrix} \,. \tag{4.78}$$

Repeating the same procedure for the other two reaction wheels yields

$$I_{w2_O} = \begin{bmatrix} I_{w_{yy}} + m_w\frac{l^2}{4} & 0 & -m_w\frac{l^2}{4} \\ 0 & I_{w_{xx}} + m_w\frac{l^2}{2} & 0 \\ -m_w\frac{l^2}{4} & 0 & I_{w_{yy}} + m_w\frac{l^2}{4} \end{bmatrix} \,, \tag{4.79}$$

$$I_{w3_O} = \begin{bmatrix} I_{w_{yy}} + m_w\frac{l^2}{4} & -m_w\frac{l^2}{4} & 0 \\ -m_w\frac{l^2}{4} & I_{w_{yy}} + m_w\frac{l^2}{4} & 0 \\ 0 & 0 & I_{w_{xx}} + m_w\frac{l^2}{2} \end{bmatrix} \,. \tag{4.80}$$

4.3.1.3 Kinetic Energy

Now that the angular velocities and inertia matrices of each moving body have been defined, it is possible to write down the total kinetic energy of the system

$$\begin{aligned} T =& T_s + \sum_{i=0}^{3} T_{wi} \\ =& \frac{1}{2}\vec{\omega}_s{}'^T I_{s_O} \vec{\omega}_s{}' + \sum_{i=0}^{3}\left(\frac{1}{2}(\vec{\omega}_s{}' + \vec{\omega}_{wi}{}')^T I_{wi_G}(\vec{\omega}_s{}' + \vec{\omega}_{wi}{}') + \frac{1}{2}(\vec{\omega}_s{}' \times \vec{r}_{wi}{}')^T m_w\,(\vec{\omega}_s{}' \times \vec{r}_{wi}{}')\right) \\ =& \frac{1}{2}\vec{\omega}_s{}'^T I_{s_O} \vec{\omega}_s{}' + \sum_{i=0}^{3}\left(\frac{1}{2}(\vec{\omega}_s{}' + \vec{\omega}_{wi}{}')^T I_{wi_G}(\vec{\omega}_s{}' + \vec{\omega}_{wi}{}') + \frac{1}{2}\vec{\omega}_s{}'^T\left(I_{wi_O} - I_{wi_G}\right)\vec{\omega}_s{}'\right) \\ =& \frac{1}{2}\vec{\omega}_s{}'^T\left(I_{s_O} + \sum_{i=0}^{3}\left(I_{wi_O} - I_{wi_G}\right)\right)\vec{\omega}_s{}' + \sum_{i=0}^{3}\left(\frac{1}{2}(\vec{\omega}_s{}' + \vec{\omega}_{wi}{}')^T I_{wi_G}(\vec{\omega}_s{}' + \vec{\omega}_{wi}{}')\right) \,. \end{aligned} \tag{4.81}$$

Because each reaction wheel rotates around an axis orthogonal to each other, (4.81) can be simplified to

$$T = \frac{1}{2}\vec{\omega}_c{}'^T \bar{I}_c \vec{\omega}_c{}' + \frac{1}{2}(\vec{\omega}_c{}' + \vec{\omega}_w{}')^T I_w(\vec{\omega}_c{}' + \vec{\omega}_w{}') \,, \tag{4.82}$$

where $\vec{\omega}_c{}'$ is the Cubli angular velocity vector, which is the same as the structure

$$\begin{aligned}\vec{\omega}_c{}' &= \vec{\omega}_s{}' \\ &= \begin{bmatrix}\omega_x \\ \omega_y \\ \omega_z\end{bmatrix} ,\end{aligned} \tag{4.83}$$

$\vec{\omega}_w{}'$ is the composition of all three relative angular velocities vectors of the reaction wheels

$$\begin{aligned}\vec{\omega}_w{}' &= \sum_{i=1}^{3} \vec{\omega}_{wi}{}' \\ &= \begin{bmatrix}\omega_1 \\ \omega_2 \\ \omega_3\end{bmatrix} ,\end{aligned} \tag{4.84}$$

I_w is the net inertia tensor of the three reaction wheels around each of their individual rotational axis

$$\begin{aligned}I_w &= I_{w_{xx}} I_{3\times 3} \\ &= \begin{bmatrix}I_{w_{xx}} & 0 & 0 \\ 0 & I_{w_{xx}} & 0 \\ 0 & 0 & I_{w_{xx}}\end{bmatrix} ,\end{aligned} \tag{4.85}$$

and $\bar{I}_c$ is the Cubli total inertia matrix around the pivot point O only without the reaction wheels inertias around each of their individual rotational axis:

$$\begin{aligned}\bar{I}_c &= \underbrace{\left(I_{s_O} + \sum_{i=1}^{3} I_{wi_O} \right)}_{I_{c_O}} - I_w \\ &= \begin{bmatrix}\bar{I}_{c_{xx}} & \bar{I}_{c_{xy}} & \bar{I}_{c_{xy}} \\ \bar{I}_{c_{xy}} & \bar{I}_{c_{xx}} & \bar{I}_{c_{xy}} \\ \bar{I}_{c_{xy}} & \bar{I}_{c_{xy}} & \bar{I}_{c_{xx}}\end{bmatrix}\end{aligned} \tag{4.86}$$

where

$$\bar{I}_{c_{xx}} = I_{s_{xx}} + 2I_{w_{yy}} + (m_s + 2m_w)\frac{l^2}{2} , \tag{4.87}$$

$$\bar{I}_{c_{xy}} = -(m_s + m_w)\frac{l^2}{4} . \tag{4.88}$$

4.3.2 Potential Energy

Since the Cubli is rotating aroung the pivot point O (articulation vertex), the potential energy depends only on its bodies center of mass vectors. It can be analyzed from two different perspectives: in the inertial coordinate frame, where the gravity vector remains constant, but the bodies center of mass vectors varies according to the orientation of the Cubli; and in the body fixed coordinate frame, where the gravity vector varies according to the orientation of the Cubli, but the bodies center of mass vectors remains constant.

Both of these approaches are analogous to each other, but since the kinetic energy of the system have already been analyzed in the body fixed coordinate frame, the same will be done for the potential energy.

The Cubli total potential energy is the sum of the potential energy of each moving body:

$$V = V_s + \sum_{i=1}^{3} V_{wi}\,, \tag{4.89}$$

where V_s is the potential energy of the structure and V_{wi} is the potential energy of the i-th reaction wheel.

The potential energy depends on the masses, center of mass vectors and gravity vector. The first two parameters have already been defined, only the third one is missing.

4.3.2.1 Gravity

Let $\vec{g}$ be the gravity vector described in the inertial coordinate frame

$$\vec{g} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}\,. \tag{4.90}$$

The gravity vector $\vec{g}\,'$ in the body fixed coordinate frame is simply the rotation of the previous vector

$$\vec{g}\,' = R(q)\vec{g}\,. \tag{4.91}$$

4.3.2.2 Potential Energy

Now that the masses and center of mass vectors of each moving body and gravity vector have been defined, it is possible to write down the total potential energy of the system

$$\begin{aligned} V &= V_s + \sum_{i=1}^{3} V_{wi} \\ &= m_s \vec{r}_s{}'^T \vec{g}\,' + \sum_{i=1}^{3} m_w \vec{r}_{wi}{}'^T \vec{g}\,' \\ &= m_s \vec{r}_s{}'^T R(q)\vec{g} + \sum_{i=1}^{3} m_w \vec{r}_{wi}{}'^T R(q)\vec{g}\,, \end{aligned} \tag{4.92}$$

which can be simplified to

$$V = m_c \vec{r}_c{}'^T R(q)\vec{g}\,, \tag{4.93}$$

where m_c is the Cubli's total mass (structure with all three reaction wheels) given by

$$m_c = m_s + 3m_w \tag{4.94}$$

and $\vec{r}_c{}'$ is the vector from pivot point O to the Cubli center of mass G_c (with all three reaction wheels), described in the body fixed coordinate frame

$$\vec{r}_c{}' = \frac{m_s \vec{r}_s{}' + m_w \sum_{i=1}^{3} \vec{r}_{wi}{}'}{m_s + 3m_w}\,. \tag{4.95}$$

4.3.3 Lagrange equations

Once the system kinetic and potential energy have been defined, *i.e.*, (4.82) and (4.93), the Lagrangian is simply the difference between the two

$$\begin{aligned} L &= T - V \\ &= \frac{1}{2}\vec{\omega}_c{}'^T \bar{I}_c \vec{\omega}_c{}' + \frac{1}{2}(\vec{\omega}_c{}' + \vec{\omega}_w{}')^T I_w (\vec{\omega}_c{}' + \vec{\omega}_w{}') - m_c \vec{r}_c{}'^T R(q)\vec{g}\,, \end{aligned} \tag{4.96}$$

The kinetic equations of the system can then be obtained applying the Lagrange equations

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{Q}_i}\right) - \frac{\partial L}{\partial Q_i} = \sum F_{Q_i}\,. \tag{4.97}$$

where Q_i is the i-th generalized coordinate of the system[2], and F_{q_i} is the generalized force in the Q_i direction.

Generalized coordinates are typically position coordinates (distances or angles). There

[2]Literature usually uses the lowercase letter q_i to denote the generalized coordinates, but we are using the uppercase letter Q_i so as not to confuse it with the rotation quarternion q.

are two of them of interest in the Cubli:

- The rotation quaternion q, that describes the Cubli orientation
- The vector $\vec{\theta_w}\,'$, that describes the reaction wheels relative angular displacement

Generalized forces are external inputs to the system (forces or torques). There is only the vector of torques $\vec{\tau}$ from the motors applied on each of the three reaction wheels, given by

$$\vec{\tau} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} . \tag{4.98}$$

These torques occur in the same direction as the reaction wheels relative angular displacement.

For each one of these generalized coordinates, there will be one kinetic equation to be calculated separately.

4.3.3.1 Generalized coordinate q

To make it easier, the Lagrange equation will be divided into four terms and each one of them will be calculated individually

$$\underbrace{\frac{d}{dt}\underbrace{\left(\frac{\partial L}{\partial \dot{q}}\right)}_{\text{Term 1}}}_{\text{Term 2}} - \underbrace{\frac{\partial L}{\partial q}}_{\text{Term 3}} = \underbrace{\sum F_q}_{\text{Term 4}} . \tag{4.99}$$

Term 1

First, the Lagrangian will be rewritten and rearranged substituting the angular velocity vector with (4.66), so that $\dot{q}$ is isolated

$$\begin{aligned}
L &= \frac{1}{2}\vec{\omega}_c{}'^T \bar{I}_c \vec{\omega}_c{}' + \frac{1}{2}(\vec{\omega}_c{}' + \vec{\omega}_w{}')^T I_w (\vec{\omega}_c{}' + \vec{\omega}_w{}') - m_c \vec{r}_c{}'^T R(q)\vec{g} \\
&= \frac{1}{2}(2G(q)\dot{q})^T \bar{I}_c \left(2G(q)\dot{q}\right) + \frac{1}{2}(2G(q)\dot{q} + \vec{\omega}_w{}')^T I_w \left(2G(q)\dot{q} + \vec{\omega}_w{}'\right) - m_c \vec{r}_c{}'^T R(q)\vec{g} \\
&= 2(G(q)\dot{q})^T \bar{I}_c \left(G(q)\dot{q}\right) + 2\left(G(q)\left(\dot{q} + \frac{G(q)^{-1}}{2}\vec{\omega}_w{}'\right)\right)^T I_w \left(G(q)\left(\dot{q} + \frac{G(q)^{-1}}{2}\vec{\omega}_w{}'\right)\right) \\
&\qquad - m_c \vec{r}_c{}'^T R(q)\vec{g} \\
&= 2\left(\dot{q}^T G(q)^T\right) \bar{I}_c \left(G(q)\dot{q}\right) + 2\left(\left(\dot{q} + \frac{G(q)^{-1}}{2}\vec{\omega}_w{}'\right)^T G(q)^T\right) I_w \left(G(q)\left(\dot{q} + \frac{G(q)^{-1}}{2}\vec{\omega}_w{}'\right)\right) \\
&\qquad - m_c \vec{r}_c{}'^T R(q)\vec{g} \\
&= 2\dot{q}^T \left(G(q)^T \bar{I}_c G(q)\right)\dot{q} + 2\left(\dot{q} + \frac{G(q)^{-1}}{2}\vec{\omega}_w{}'\right)^T \left(G(q)^T I_w G(q)\right)\left(\dot{q} + \frac{G(q)^{-1}}{2}\vec{\omega}_w{}'\right) \\
&\qquad - m_c \vec{r}_c{}'^T R(q)\vec{g}\,.
\end{aligned} \tag{4.100}$$

With $\dot{q}$ isolated, it is straightforward to perform the partial derivative with respect to it

$$\begin{aligned}
\frac{\partial L}{\partial \dot{q}} &= 4\left(G(q)^T \bar{I}_c G(q)\right)\dot{q} + 4\left(G(q)^T I_w G\right)\left(\dot{q} + \frac{G(q)^{-1}}{2}\vec{\omega}_w{}'\right) \\
&= 2G(q)^T \bar{I}_c \left(2G(q)\dot{q}\right) + 2G(q)^T I_w \left(2G(q)\left(\dot{q} + \frac{G(q)^{-1}}{2}\vec{\omega}_w{}'\right)\right) \\
&= 2G(q)^T \bar{I}_c \left(2G(q)\dot{q}\right) + 2G(q)^T I_w \left(2G(q)\dot{q} + \vec{\omega}_w{}'\right)\,.
\end{aligned} \tag{4.101}$$

Finally, it will be rewritten by making use of the same property as before

$$\begin{aligned}
\frac{\partial L}{\partial \dot{q}} &= 2G(q)^T \bar{I}_c \left(2G(q)\dot{q}\right) + 2G(q)^T I_w \left(2G(q)\dot{q} + \vec{\omega}_w{}'\right) \\
&= 2G(q)^T \bar{I}_c \vec{\omega}_c{}' + 2G(q)^T I_w \left(\vec{\omega}_c{}' + \vec{\omega}_w{}'\right)\,.
\end{aligned} \tag{4.102}$$

Term 2

With term 1 rewritten in this form, its time derivative is also straightforward

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) = 2G(q)^T \bar{I}_c \dot{\vec{\omega}}_c{}' + 2G(\dot{q})^T \bar{I}_c \omega_c' + 2G(q)^T I_w \left(\dot{\vec{\omega}}_c{}' + \dot{\vec{\omega}}_w{}'\right) + 2G(\dot{q})^T I_w \left(\omega_c' + \omega_w'\right)\,. \tag{4.103}$$

Term 3

First, the Lagrangian will be rewritten and rearranged substituting the angular velocity vector with (4.67), so that q is isolated

$$\begin{aligned}
L &= \frac{1}{2}\vec{\omega}_c{}'^T \bar{I}_c \vec{\omega}_c{}' + \frac{1}{2}(\vec{\omega}_c{}' + \vec{\omega}_w{}')^T I_w (\vec{\omega}_c{}' + \vec{\omega}_w{}') - m_c \vec{r}_c{}'^T R(q)\vec{g} \\
&= \frac{1}{2}(-2G(\dot{q})q)^T \bar{I}_c\,(-2G(\dot{q})q) + \frac{1}{2}(-2G(\dot{q})q + \vec{\omega}_w{}')^T I_w\,(-2G(\dot{q})q + \vec{\omega}_w{}') - m_c \vec{r}_c{}'^T R(q)\vec{g} \\
&= 2(G(\dot{q})q)^T \bar{I}_c\,(G(\dot{q})q) + 2\left(G(\dot{q})\left(q - \frac{G(\dot{q})^{-1}}{2}\vec{\omega}_w{}'\right)\right)^T I_w \left(G(\dot{q})\left(q - \frac{G(\dot{q})^{-1}}{2}\vec{\omega}_w{}'\right)\right) \\
&\qquad - m_c \vec{r}_c{}'^T R(q)\vec{g} \\
&= 2\left(q^T G(\dot{q})^T\right) \bar{I}_c\,(G(\dot{q})q) + 2\left(\left(q - \frac{G(\dot{q})^{-1}}{2}\vec{\omega}_w{}'\right)^T G(\dot{q})^T\right) I_w \left(G(\dot{q})\left(q - \frac{G(\dot{q})^{-1}}{2}\vec{\omega}_w{}'\right)\right) \\
&\qquad - m_c \vec{r}_c{}'^T R(q)\vec{g} \\
&= 2q^T\left(G(\dot{q})^T \bar{I}_c G(\dot{q})\right) q + 2\left(q - \frac{G(\dot{q})^{-1}}{2}\vec{\omega}_w{}'\right)^T \left(G(\dot{q})^T I_w G(\dot{q})\right)\left(q - \frac{G(\dot{q})^{-1}}{2}\vec{\omega}_w{}'\right) \\
&\qquad - m_c \vec{r}_c{}'^T R(q)\vec{g}\,.
\end{aligned} \tag{4.104}$$

With q isolated, it is straightforward to perform the partial derivative with respect to it

$$\begin{aligned}
\frac{\partial L}{\partial q} &= 4\left(G(\dot{q})^T \bar{I}_c G(\dot{q})\right) q + 4\left(G(\dot{q})^T I_w G(\dot{q})\right)\left(q - \frac{G(\dot{q})^{-1}}{2}\vec{\omega}_w{}'\right) - 2m_c \Delta q \\
&= 2G(\dot{q})^T \bar{I}_c\,(2G(\dot{q})q) + 2G(\dot{q})^T I_w \left(2G(\dot{q})\left(q - \frac{G(\dot{q})^{-1}}{2}\vec{\omega}_w{}'\right)\right) - 2m_c \Delta q \\
&= 2G(\dot{q})^T \bar{I}_c\,(2G(\dot{q})q) + 2G(\dot{q})^T I_w\,(2G(\dot{q})q - \vec{\omega}_w{}') - 2m_c \Delta q\,.
\end{aligned} \tag{4.105}$$

Note that, the rotation matrix $R(q)$ depends on q but it cannot be isolated with matrix notation. That is why its partial derivative with respect to q depends on Δ, given by

$$\Delta = \begin{bmatrix} \vec{g}^T \vec{r}_c{}' & -(\vec{g} \times \vec{r}_c{}')^T \\ -\vec{g} \times \vec{r}_c{}'^T & \vec{g}\vec{r}_c{}'^T + \vec{r}_c{}'\vec{g}^T - I_{3\times 3}(\vec{g}^T \vec{r}_c{}')\,. \end{bmatrix} \tag{4.106}$$

Finally, it will be rewritten by making use of the same property as before

$$\begin{aligned}\frac{\partial L}{\partial q} &= 2G(\dot{q})^T \bar{I}_c \left(2G(\dot{q})q\right) + 2G(\dot{q})^T I_w \left(2G(\dot{q})q - \vec{\omega}_w{}'\right) - 2m_c\Delta q \\ &= 2G(\dot{q})^T \bar{I}_c \left(-\vec{\omega}_c{}'\right) + 2G(\dot{q})^T I_w \left(-\vec{\omega}_c{}' - \vec{\omega}_w{}'\right) - 2m_c\Delta q \\ &= -2G(\dot{q})^T \bar{I}_c \vec{\omega}_c{}' - 2G(\dot{q})^T I_w \left(\vec{\omega}_c{}' + \vec{\omega}_w{}'\right) - 2m_c\Delta q\,. \end{aligned} \tag{4.107}$$

Term 4

The electric motors apply torques in the reaction wheels and thus a reaction torque in applied in the Cubli. However, since one is dealing with relative angular displacements of the reaction wheels, the generalized forces in the direction of the rotation quaternion are zero

$$\sum F_q = \vec{0}\,. \tag{4.108}$$

Lagrange equation

Now that all terms have been calculated, they can be put together into the Lagrange equation

$$\begin{aligned} &\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = \sum F_q \\ &2G(q)^T \bar{I}_c \dot{\vec{\omega}}_c{}' + 2G(\dot{q})^T \bar{I}_c \vec{\omega}_c{}' + 2G(q)^T I_w \left(\dot{\vec{\omega}}_c{}' + \dot{\vec{\omega}}_w{}'\right) + 2G(\dot{q})^T I_w \left(\vec{\omega}_c{}' + \vec{\omega}_w{}'\right) \\ &\qquad + 2G(\dot{q})^T \bar{I}_c \vec{\omega}_c{}' + 2G(\dot{q})^T I_w \left(\vec{\omega}_c{}' + \vec{\omega}_w{}'\right) + 2m_c\Delta q = \vec{0} \\ &2G(q)^T \bar{I}_c \dot{\vec{\omega}}_c + 4G(\dot{q})^T \bar{I}_c \vec{\omega}_c{}' + 2G(q)^T I_w \left(\dot{\vec{\omega}}_c{}' + \dot{\vec{\omega}}_w{}'\right) + 4G(\dot{q})^T I_w \left(\vec{\omega}_c{}' + \vec{\omega}_w{}'\right) \\ &\qquad + 2m_c\Delta q = \vec{0}\,. \end{aligned} \tag{4.109}$$

By left multiplying all terms by $\frac{1}{2}G(q)$ and making use of (4.42) and (4.69), the above equation can be further simplified

$$\begin{aligned} &\frac{1}{2}G(q)2G(q)^T \bar{I}_c \dot{\vec{\omega}}_c{}' + \frac{1}{2}G(q)4G(\dot{q})^T \bar{I}_c \vec{\omega}_c{}' + \frac{1}{2}G(q)2G(q)^T I_w \left(\dot{\vec{\omega}}_c{}' + \dot{\vec{\omega}}_w{}'\right) \\ &\qquad + \frac{1}{2}G(q)4G(\dot{q})^T I_w \left(\vec{\omega}_c{}' + \vec{\omega}_w{}'\right) + \frac{1}{2}G(q)2m_c\Delta q = \frac{1}{2}G(q)\vec{0} \\ &GG(q)^T \bar{I}_c \dot{\vec{\omega}}_c{}' + 2G(q)G(\dot{q})^T \bar{I}_c \vec{\omega}_c{}' + GG(q)^T I_w \left(\dot{\vec{\omega}}_c{}' + \dot{\vec{\omega}}_w{}'\right) \\ &\qquad + 2G(q)G(\dot{q})^T I_w \left(\vec{\omega}_c{}' + \vec{\omega}_w{}'\right) + m_c G(q)\Delta q = \vec{0} \\ &\bar{I}_c \dot{\vec{\omega}}_c{}' + \tilde{\omega}_c' \left(\bar{I}_c \vec{\omega}_c{}'\right) + I_w \left(\dot{\vec{\omega}}_c{}' + \dot{\vec{\omega}}_w{}'\right) + \tilde{\omega}_c' \left(I_w \left(\vec{\omega}_c{}' + \vec{\omega}_w{}'\right)\right) + m_c G(q)\Delta q = \vec{0}\,. \end{aligned} \tag{4.110}$$

This is the kinetic equation for the rotation quaternion q generalized coordinate. It can also be written as

$$\bar{I}_c\dot{\vec{\omega}}_c{}' + \tilde{\omega}_c' \left(\bar{I}_c\vec{\omega}_c{}'\right) + I_w \left(\dot{\vec{\omega}}_c{}' + \dot{\vec{\omega}}_w{}'\right) + \tilde{\omega}_c' \left(I_w \left(\vec{\omega}_c{}' + \vec{\omega}_w{}'\right)\right) + \bar{m}_c g l G(q)\Gamma q = \vec{0}\,, \quad (4.111)$$

where

$$\bar{m}_c = m_s + 2m_w\,, \qquad \Gamma = \begin{bmatrix} 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ -1 & 0 & -1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}. \quad (4.112)$$

4.3.3.2 Reaction wheels angular displacement

As considered for the rotation quaternion, the Lagrange equation will be divided into four terms and each one of them will be calculated individually

$$\underbrace{\frac{d}{dt}\underbrace{\left(\frac{\partial L}{\partial \dot{\vec{\theta}}_w{}'}\right)}_{\text{Term 1}}}_{\text{Term 2}} - \underbrace{\frac{\partial L}{\partial \vec{\theta}_w{}'}}_{\text{Term 3}} = \underbrace{\sum F_i}_{\text{Term 4}}\,. \quad (4.113)$$

Term 1

First, the Lagrangian will be rewritten substituting the reaction wheels angular velocity vector, so that $\dot{\vec{\theta}}_w{}'$ is isolated

$$\begin{aligned} L &= \frac{1}{2}\vec{\omega}_c{}'^T \bar{I}_c \vec{\omega}_c{}' + \frac{1}{2}(\vec{\omega}_c{}' + \vec{\omega}_w{}')^T I_w (\vec{\omega}_c{}' + \vec{\omega}_w{}') - m_c \vec{r}_c{}'^T R(q)\vec{g} \\ &= \frac{1}{2}\vec{\omega}_c{}'^T \bar{I}_c \vec{\omega}_c{}' + \frac{1}{2}(\vec{\omega}_c{}' + \dot{\vec{\theta}}_w{}')^T I_w (\vec{\omega}_c{}' + \dot{\vec{\theta}}_w{}') - m_c \vec{r}_c{}'^T R(q)\vec{g}\,. \end{aligned} \quad (4.114)$$

With $\dot{\vec{\theta}}_w{}'$ isolated, it is straightforward to perform the partial derivative with respect to it

$$\frac{\partial L}{\partial \dot{\vec{\theta}}_w{}'} = I_w \left(\vec{\omega}_c{}' + \dot{\vec{\theta}}_w{}'\right)\,. \quad (4.115)$$

Finally, it will be rewritten by making use of the same property as before

$$\begin{aligned} \frac{\partial L}{\partial \dot{\vec{\theta}}_w{}'} &= I_w \left(\vec{\omega}_c{}' + \dot{\vec{\theta}}_w{}'\right) \\ &= I_w \left(\vec{\omega}_c{}' + \vec{\omega}_w{}'\right)\,. \end{aligned} \quad (4.116)$$

Term 2

With term 1 rewritten in this form, it is straightforward to perform the time derivative

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\vec{\theta}}_w{}'}\right) = I_w\left(\dot{\vec{\omega}}_c{}' + \dot{\vec{\omega}}_w{}'\right) + \tilde{\omega}_c'\left(I_w\left(\vec{\omega}_c{}' + \vec{\omega}_w{}'\right)\right)\,.$$

Term 3

Since the Lagrangian does not depend on $\vec{\theta}_w{}'$, its partial derivative with respect to it is zero

$$\frac{\partial L}{\partial \vec{\theta}_w{}'} = \vec{0}\,.$$

Term 4

The electric motors apply torques in the reaction wheels in the same direction as the reaction wheels relative angular displacement, so the generalized forces are exactly these torques

$$\sum F_{\vec{\theta}_w{}'} = \vec{\tau}\,'\,. \tag{4.117}$$

Lagrange equation

Now that all terms have been calculated, they can be put together into the Lagrange equation

$$\begin{aligned}\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} &= \sum F_q\\ I_w\left(\dot{\vec{\omega}}_c{}' + \dot{\vec{\omega}}_w{}'\right) + \tilde{\omega}_c'\left(I_w\left(\vec{\omega}_c{}' + \vec{\omega}_w{}'\right)\right) - \vec{0} &= \vec{\tau}\,'\\ I_w\left(\dot{\vec{\omega}}_c{}' + \dot{\vec{\omega}}_w{}'\right) + \tilde{\omega}_c'\left(I_w\left(\vec{\omega}_c{}' + \vec{\omega}_w{}'\right)\right) &= \vec{\tau}\,'\,. \end{aligned} \tag{4.118}$$

This is the kinetic equation for the reaction wheels relative angular displacement $\vec{\theta}_w{}'$ generalized coordinate.

4.3.4 Kinetic equations

The system kinetic equations are composed of kinetic equations obtained with the Lagrange equations, one for each generalized coordinate

$$\begin{cases} \bar{I}_c\dot{\vec{\omega}}_c{}' + \tilde{\omega}_c\left(\bar{I}_c\vec{\omega}_c{}'\right) + I_w\left(\dot{\vec{\omega}}_c{}' + \dot{\vec{\omega}}_w{}'\right) + \tilde{\omega}_c\left(I_w\left(\vec{\omega}_c{}' + \vec{\omega}_w{}'\right)\right) + \bar{m}_c g l G(q)\Gamma q = \vec{0} \\ I_w\left(\dot{\vec{\omega}}_c{}' + \dot{\vec{\omega}}_w{}'\right) + \tilde{\omega}_c'\left(I_w\left(\vec{\omega}_c{}' + \vec{\omega}_w{}'\right)\right) = \vec{\tau}\,' \end{cases} . \tag{4.119}$$

Because $\vec{\omega}_w{}' \gg \vec{\omega}_c{}'$, the approximation $(\vec{\omega}_c{}' + \vec{\omega}_w{}') \approx \vec{\omega}_w{}'$ can be applied

$$\begin{cases} \bar{I}_c\dot{\vec{\omega}}_c{}' + I_w\dot{\vec{\omega}}_w{}' + \tilde{\omega}_c\left(\bar{I}_c\vec{\omega}_c{}' + I_w\vec{\omega}_w{}'\right) + \bar{m}_c g l G(q)\Gamma q = \vec{0} \\ I_w\dot{\vec{\omega}}_w{}' + \tilde{\omega}_c'\left(I_w\vec{\omega}_w{}'\right) = \vec{\tau}\,' \end{cases} . \tag{4.120}$$

Moreover, since the reaction wheels angular velocity are relative (measured with respect to the structure), their gyroscopic torques have no influence on them, only on the Cubli. Hence, the reaction wheels gyroscopic torques can be disregarded

$$\begin{cases} \bar{I}_c\dot{\vec{\omega}}_c{}' + I_w\dot{\vec{\omega}}_w{}' + \tilde{\omega}_c\left(\bar{I}_c\vec{\omega}_c{}' + I_w\vec{\omega}_w{}'\right) + \bar{m}_c g l G(q)\Gamma q = \vec{0} \\ I_w\dot{\vec{\omega}}_w{}' = \vec{\tau}\,' \end{cases} . \tag{4.121}$$

These equations can then be rewritten isolating the time derivative terms

$$\begin{cases} \dot{\vec{\omega}}_c{}' = \bar{I}_c^{-1}\left(-\tilde{\omega}_c\left(\bar{I}_c\vec{\omega}_c{}' + I_w\vec{\omega}_w{}'\right) - \bar{m}_c g l G(q)\Gamma q - \vec{\tau}\,'\right) \\ \dot{\vec{\omega}}_w{}' = I_w^{-1}\vec{\tau}\,' \end{cases} . \tag{4.122}$$

These are the kinetic equations of the system with vector notation utilizing quaternions.

4.3.5 Friction forces

For the model to become even more realistic, it is necessary to include friction forces. There are two main frictions in this model: the friction between the Cubli and the surface at pivot point O, and the friction of the motors.

The surface friction can be modeled by a viscous friction coefficient b. However, since it occurs only in the direction orthogonal to the gravity vector, it depends on the orientation of the Cubli.

The motor friction is somewhat more complicated. Besides having a viscous friction, it also has a static friction (that generates a dead zone) and an aerodynamic drag (since the reaction wheels are hollow). It can be modeled as

$$\tau_f(\omega_i) = \text{sign}(\omega_i)\left(\tau_c + b_w|\omega_i| + c_d|\omega_i|^2\right) , \tag{4.123}$$

where τ_c is the Coulomb friction, b_w is the viscous friction coefficient and c_d is the aerodynamic drag coefficient. Those parameters were identified experimentally with a torque controller by varying the torque reference, registering the equivalent steady-state velocity (where the input torque equals the friction torque), and curve fitting of the data (Fig. 20). The identified parameters are given in Table 2.

Table 2: Friction torque parameters

Parameter	Value
τ_c	2.46×10^{-3} N.m
b_w	1.06×10^{-5} N.m.s.rad^{-1}
c_d	1.70×10^{-8} N.m.s^2.rad^{-2}

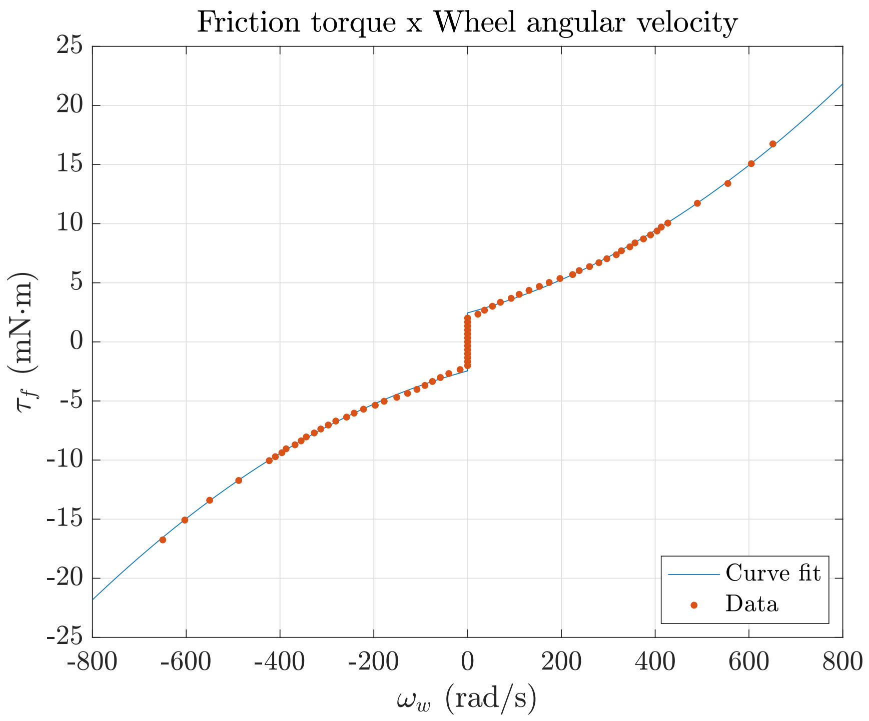


Figure 20: Friction torque data and curve fitting

4.4 Dynamics

Once the kinematics, which is the study of motion without reference to the forces which cause motion, and kinetics, which relates the action of forces on bodies to their resulting motion, are complete, the full dynamic of the system is complete.

4.4.1 Dynamic equations

Including the kinematic (4.65) and friction forces in (4.122), the full equations of motion are obtained

$$\begin{cases} \dot{q} = \frac{1}{2}G^T(q)\vec{\omega}_c{}' \\ \dot{\vec{\theta}}_w{}' = \vec{\omega}_w{}' \\ \dot{\vec{\omega}}_c{}' = \bar{I}_c^{-1}\left(-\vec{\omega}_c{}' \times \left(\bar{I}_c\vec{\omega}_c{}' + I_w\vec{\omega}_w{}'\right) + \bar{m}_c g l\left(G(q)\Gamma q\right)\right. \\ \qquad\qquad \left. -b\left(G(q)\Lambda q\left(G(q)\Lambda q\right)^T\right)\vec{\omega}_c{}' + \vec{\tau}_f(\vec{\omega}_w{}') - \vec{\tau}{}'\right) \\ \dot{\vec{\omega}}_w{}' = I_w^{-1}\left(-\vec{\tau}_f(\vec{\omega}_w{}') + \vec{\tau}{}'\right) \end{cases}, \tag{4.124}$$

where

$$\vec{\tau}_f(\vec{\omega}_w) = \begin{bmatrix} \tau_f(\omega_1) \\ \tau_f(\omega_2) \\ \tau_f(\omega_3) \end{bmatrix}, \qquad \Lambda = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix}. \tag{4.125}$$

The system can also be represented in a block diagram (Fig. 21), where it is easier to interpret the gyroscopic terms, gravity torque, surface friction and motor friction.

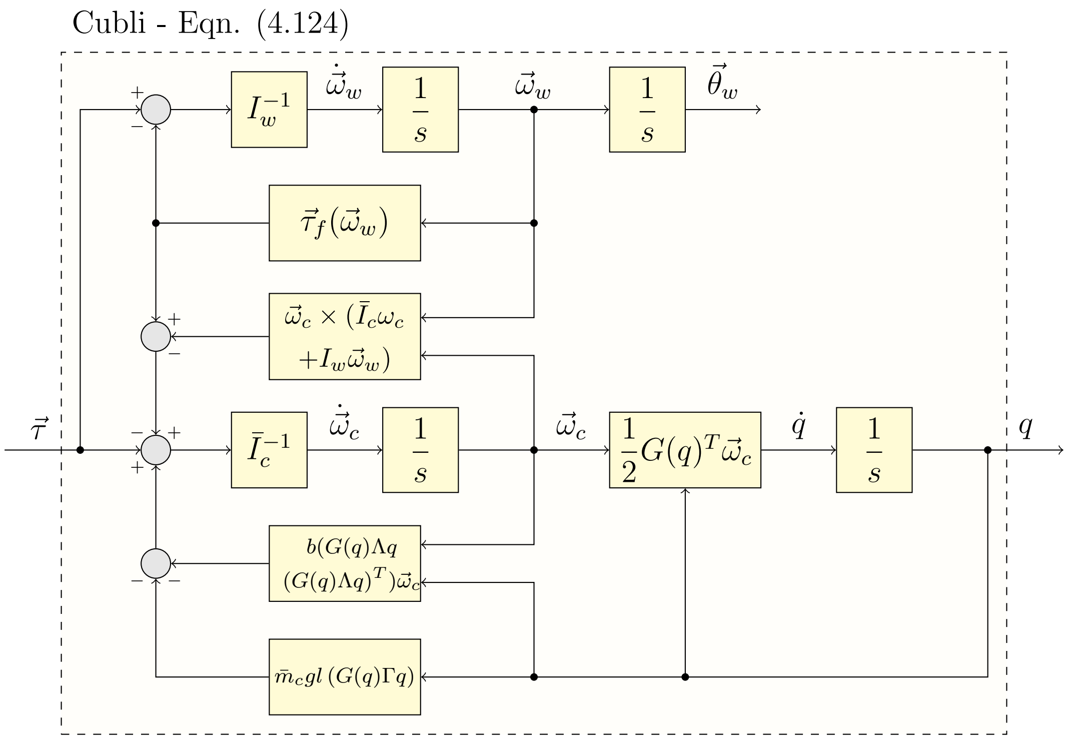


Figure 21: Cubli dynamics

Note that the Cubli and the reaction wheel dynamics are coupled by the gyroscopic terms and motor friction. In steady-state conditions, angular velocities are close to zero and thus they are coupled only by the viscous friction of the motors.

4.5 Validation

A mathematical model is only useful as long as it is a good representation of the system. Of course, what constitutes a good representation is subjective, but from a simplistic point of view, our validation criteria would be to confirm that it is correctly implemented with respect to the conceptual model.

This will be done by means of computer simulations, and they will be divided in three main types: invariant analysis, singular motions and poinsot trajectories. In all of them, friction forces are being neglected.

4.5.1 Invariant analysis

Invariant analysis considers parameters that must remain unchanged with time when no forces are being applied. The Cubli has three invariants:

- Total mechanical energy E
- Angular momentum projection in the gravitational field direction H_z
- Angular momentum projection in the gyroscopic axis direction (diagonal axis of the Cubli) $H_{z'}$

The total mechanical energy, $E = T + V$, is given from (4.82) and (4.93)

$$\begin{aligned} E &= T + V \\ &= \frac{1}{2}\vec{\omega}_c{}'^T \bar{I}_c \vec{\omega}_c{}' + \frac{1}{2}(\vec{\omega}_c{}' + \vec{\omega}_w{}')^T I_w (\vec{\omega}_c{}' + \vec{\omega}_w{}') + m_c \vec{r}_c^T R(q)\vec{g}\,. \end{aligned} \tag{4.126}$$

Note that this is not the same as the Lagrangian, where the potential energy is subtracted and not added.

Assuming the Cubli is initially aligned with the inertial coordinate frame with no angular velocities, *i.e.*,

$$q(0) = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \qquad \vec{\omega}_c{}'(0) = \begin{bmatrix}0\\0\\0\end{bmatrix}, \qquad \vec{\omega}_w{}'(0) = \begin{bmatrix}0\\0\\0\end{bmatrix}. \tag{4.127}$$

It results in the motion presented in Fig. 22.

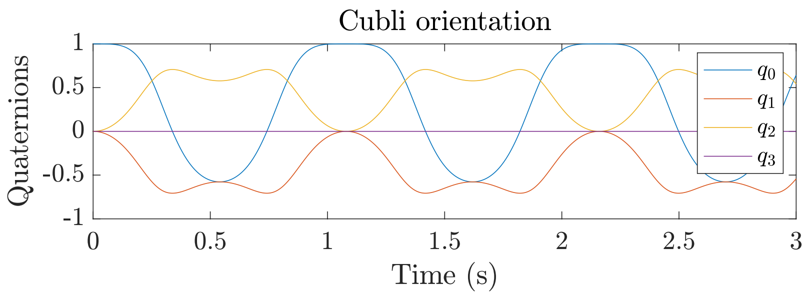


Figure 22: Simulation 1a - Invariant analysis (quaternion)

Because quaternions do not have an intuitive physical meaning, it is just possible to infer that the Cubli presented some kind of periodic motion. However, since the objective is only to analyze its energy, this is not a problem. The mechanical energy, presented in Fig. 23, remained unchanged. As Cubli lost potential energy, it acquired the same amount of kinetic energy, and vice-versa. This not only confirms the hypothesis of periodic motion, but also ensures that the model is consistent.

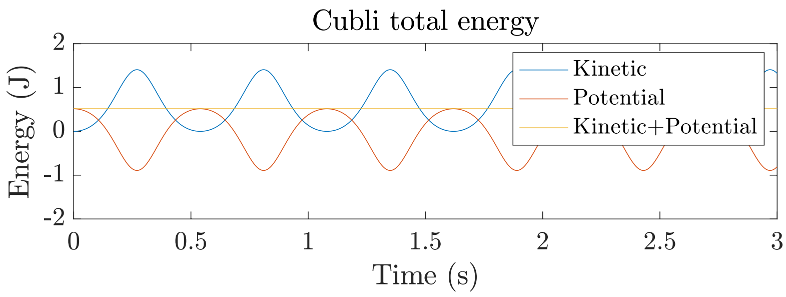


Figure 23: Simulation 1a - Invariant analysis (energy)

For the angular momentum projection invariants, the reaction wheels were assumed to be fixed. The angular momentum vector is

$$\vec{H} = \bar{I}_c \vec{\omega}_c\,, \tag{4.128}$$

so that its projections are simply given by

$$H_z = \bar{I}_c \vec{\omega}_c \cdot \vec{g}' \,, \qquad H_{z'} = \mathrm{sum}(\bar{I}_c \vec{\omega}_c) \,. \tag{4.129}$$

Let us assume the same initial conditions, but now with an arbitrary initial angular velocity, i.e

$$q(0) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \,, \qquad \vec{\omega}_c{}'(0) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \,. \tag{4.130}$$

As it can be seen in Fig. 24, the angular momentum projection on the gravitational field and gyroscopic axis directions remained unchanged.

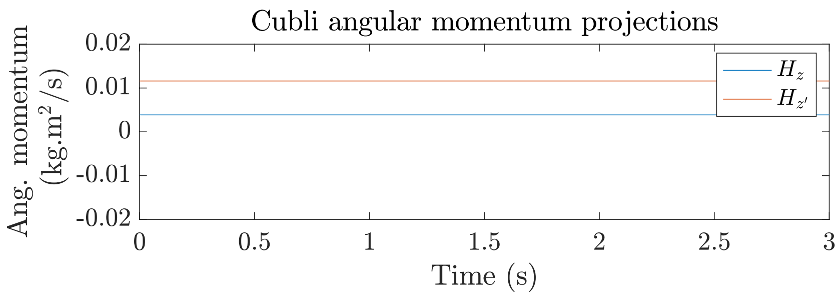


Figure 24: Simulation 2 - Invariant analysis (angular momentum)

4.5.2 Singular motions

Singular motions consider pre-defined initial conditions in which the behavior of the system can be predicted. They will be divided into static equilibrium, whereby the system states must remain unchanged, and dynamic equilibrium, whereby the system states change as expected.

4.5.2.1 Static equilibrium

The Cubli has two static equilibrium positions: an stable one and an unstable one. The stable occurs when its center of mass vector $\vec{r}_c$ is pointing down aligned with the z axis in the inertial coordinate frame (Fig. 25a), while the unstable occurs when this same vector is pointing up (Fig. 25b).

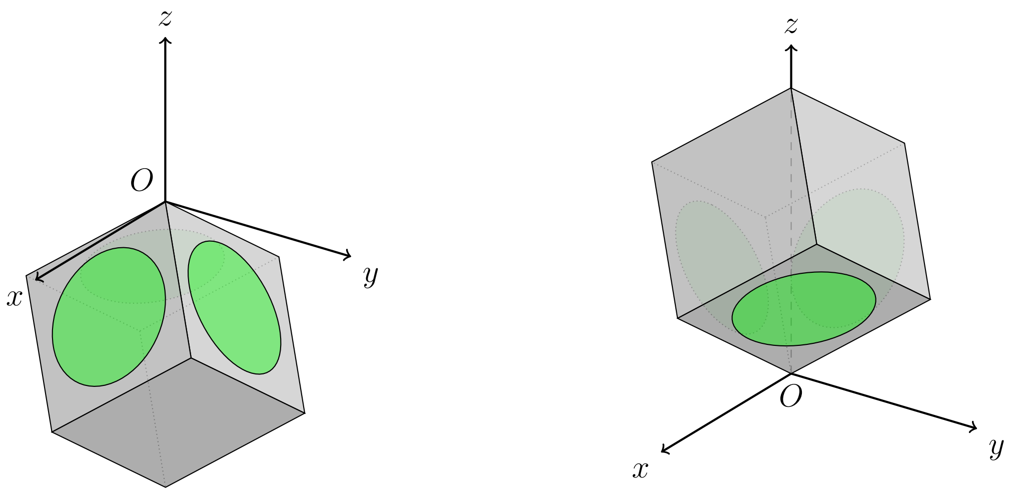


Figure 25: Cubli static equilibrium positions

Note that the stable equilibrium position is only being considered for simulation purposes, since the Cubli would never be under the xy plane.

By drawing the structure center of mass vector $\vec{r}_{s_0}$, when the inertial coordinate system and body coordinate system are aligned, together with the vector $\vec{r}_{s_s}$ or $\vec{r}_{s_u}$, when the Cubli is in its stable (Fig. 26a) or unstable (Fig. 26b) positions, the rotation quaternions for those positions can be determined.

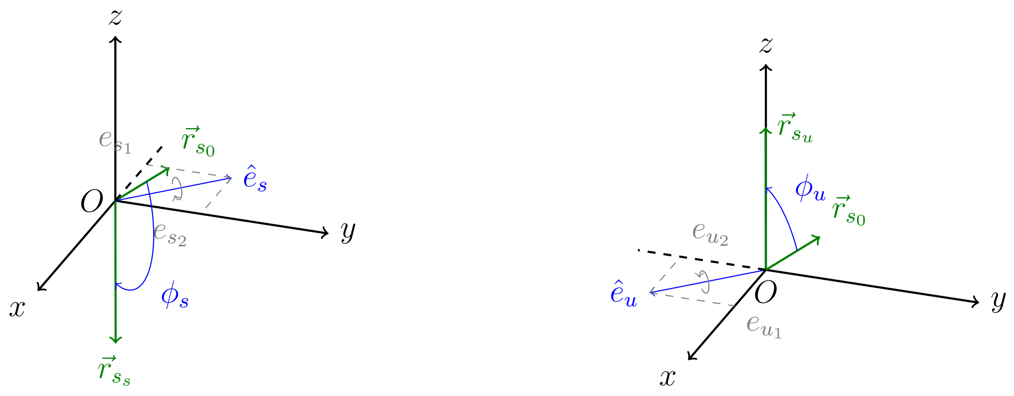


Figure 26: Cubli static equilibrium positions rotation quaternions

Where

$$\vec{r}_{s_0} = \begin{bmatrix} \frac{l}{2} \\ \frac{l}{2} \\ \frac{l}{2} \end{bmatrix}, \qquad \vec{r}_{s_s} = \begin{bmatrix} 0 \\ 0 \\ -\frac{l\sqrt{3}}{2} \end{bmatrix}, \qquad \vec{r}_{s_u} = \begin{bmatrix} 0 \\ 0 \\ \frac{l\sqrt{3}}{2} \end{bmatrix}. \tag{4.131}$$

The eigenaxes can be determined from the cross product of the structure center of mass vectors

$$\hat{e}_s = \frac{\vec{r}_{s0} \times \vec{r}_{s_s}}{|\vec{r}_{s0} \times \vec{r}_{s_s}|} = \begin{bmatrix} -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \end{bmatrix}, \qquad \hat{e}_u = \frac{\vec{r}_{s0} \times \vec{r}_{s_u}}{|\vec{r}_{s0} \times \vec{r}_{s_u}|} = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} \\ 0 \end{bmatrix}, \tag{4.132}$$

while the angles can be determined from the dot product of the structure center of mass vectors

$$\phi_s = \cos^{-1}\left(\frac{\vec{r}_{s0} \cdot \vec{r}_{s_s}}{|\vec{r}_{s0}||\vec{r}_{s_s}|}\right) \qquad \phi_u = \cos^{-1}\left(\frac{\vec{r}_{s0} \cdot \vec{r}_{s_u}}{|\vec{r}_{s0}||\vec{r}_{s_u}|}\right)$$

$$\phi_s = \cos^{-1}\left(-\frac{\sqrt{3}}{3}\right), \tag{4.133}$$

$$\phi_u = \cos^{-1}\left(\frac{\sqrt{3}}{3}\right). \tag{4.134}$$

Once both of these values have been determined, the rotation quaternions can also be determined

$$q_s = \begin{bmatrix} \cos\frac{\phi_s}{2} \\ \vec{e}_s \sin\frac{\phi_s}{2} \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{\frac{2}{3}\left(3-\sqrt{3}\right)}}{2} \\ -\frac{\sqrt{\frac{1}{3}\left(3+\sqrt{3}\right)}}{2} \\ \frac{\sqrt{\frac{1}{3}\left(3+\sqrt{3}\right)}}{2} \\ 0 \end{bmatrix}, \qquad q_u = \begin{bmatrix} \cos\frac{\phi_u}{2} \\ \vec{e}_u \sin\frac{\phi_u}{2} \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{\frac{2}{3}\left(3+\sqrt{3}\right)}}{2} \\ \frac{\sqrt{\frac{1}{3}\left(3-\sqrt{3}\right)}}{2} \\ -\frac{\sqrt{\frac{1}{3}\left(3-\sqrt{3}\right)}}{2} \\ 0 \end{bmatrix}. \tag{4.135}$$

Since the Cubli can rotate around its diagonal axis and still be in an equilibrium position, there are infinite other equivalent rotation quaternions.

In one simulation, the Cubli was considered initially on its stable equilibrium position with no initial angular velocities, *i.e.*,

$$q(0) = q_s = \begin{bmatrix} \frac{\sqrt{\frac{2}{3}\left(3-\sqrt{3}\right)}}{2} \\ -\frac{\sqrt{\frac{1}{3}\left(3+\sqrt{3}\right)}}{2} \\ \frac{\sqrt{\frac{1}{3}\left(3+\sqrt{3}\right)}}{2} \\ 0 \end{bmatrix}, \qquad \vec{\omega}_c{}'(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \qquad \vec{\omega}_w{}'(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.136}$$

whereas in the other, it was considered in its unstable equilibrium position, also no initial angular velocities, *i.e.*,

$$q(0) = q_u = \begin{bmatrix} \frac{\sqrt{\frac{2}{3}\left(3+\sqrt{3}\right)}}{2} \\ \frac{\sqrt{\frac{1}{3}\left(3-\sqrt{3}\right)}}{2} \\ -\frac{\sqrt{\frac{1}{3}\left(3-\sqrt{3}\right)}}{2} \\ 0 \end{bmatrix}, \qquad \vec{\omega}_c{}'(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \qquad \vec{\omega}_w{}'(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.137}$$

The results of both simulations can be seen in Fig. 27 and 28. In both cases, rotation quaternions remained unchanged, confirming that these are in fact static stable positions.

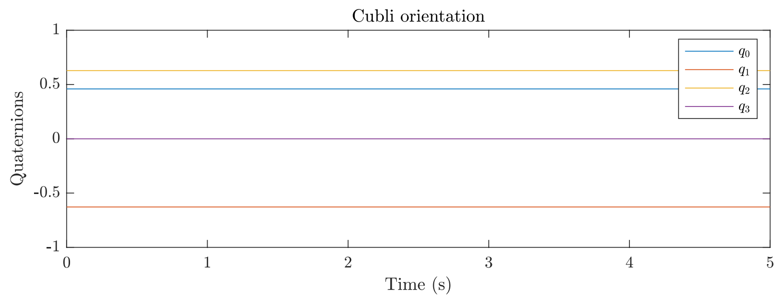


Figure 27: Simulation 3 - Static equilibrium stable (quaternion)

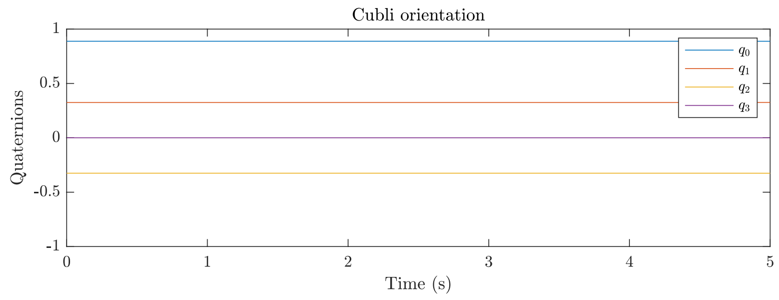
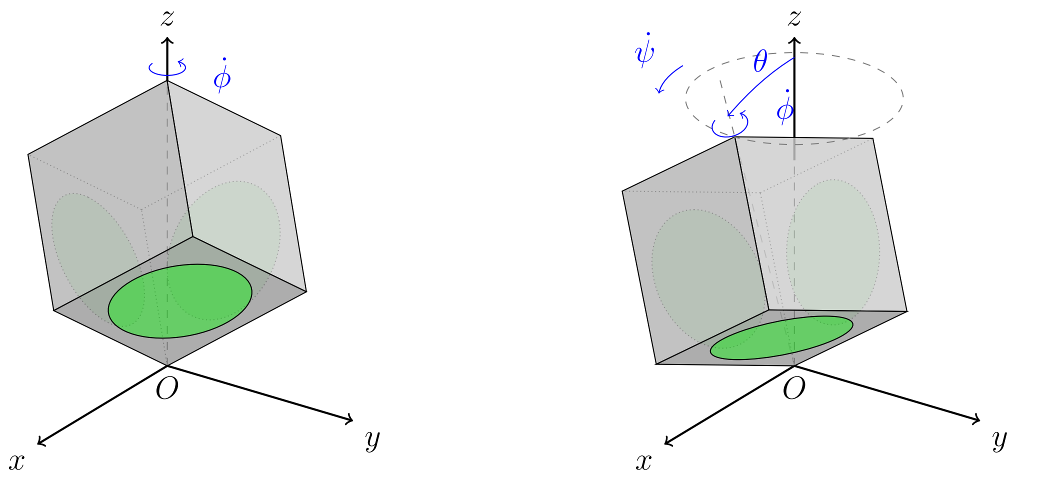


Figure 28: Simulation 4 - Static equilibrium unstable (quaternion)

4.5.2.2 Dynamic equilibrium

The Cubli has many dynamic equilibrium motions, the most well-known being those like the spinning top motion. Two of them will be analyzed: the single spin motion and the precession, nutation, and spin motion. The first occurs when the Cubli is in its static equilibrium position (either stable or unstable) but spinning around its diagonal axis (Fig. 29a), whereas the second occurs when the Cubli center of mass vector $\vec{r}_c$ is not perfectly aligned with the z axis in the inertial coordinate frame, so it spins around its diagonal axis and also precesses around the z axis in the inertial coordinate frame (Fig. 29b).



(a) Spin (b) Precession, nutation and spin

Figure 29: Cubli dynamic equilibrium motions

All these simulations were performed utilizing quaternions, but for the ease of representation, the results were converted to precession, nutation and spin angles. Moreover, the reaction wheels were assumed to be fixed.

For the single spin motion, the same initial conditions as previous simulation were assumed, but now with an initial angular velocity, *i.e.*,

$$q(0) = q_u = \begin{bmatrix} \frac{\sqrt{\frac{2}{3}\left(3+\sqrt{3}\right)}}{2} \\ \frac{\sqrt{\frac{1}{3}\left(3-\sqrt{3}\right)}}{2} \\ -\frac{\sqrt{\frac{1}{3}\left(3-\sqrt{3}\right)}}{2} \\ 0 \end{bmatrix}, \qquad \vec{\omega}_c{}'(0) = 2\pi\frac{1}{\sqrt{3}}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \qquad \vec{\omega}_w{}'(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.138}$$

meaning it is spinning. Results are presented in Fig. 30. The spin angle kept increasing whereas the precession and nutation angles remained unchanged, meaning that the Cubli only rotated around its diagonal axis. Moreover, Cubli rotated at exactly 2π rad/s (1Hz), which agrees with the initial angular velocities.

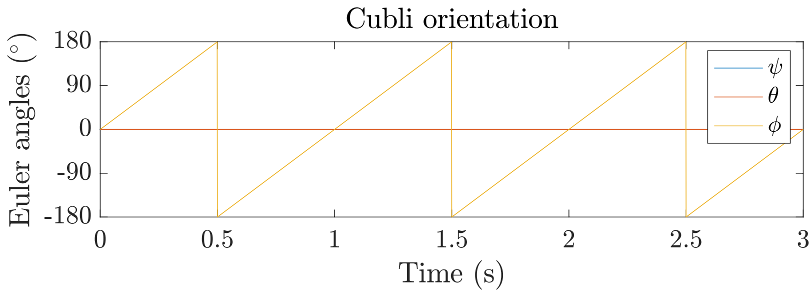


Figure 30: Simulation 5 - Dynamic equilibrium spin (Euler angles)

To simulate the precession, nutation and spin motion, a non-equilibrium rotation quaternion q_{ne} was calculated considering a somewhat small nutation angle (10°). This rotation quaternion was calculated the same way it was done for the static equilibrium unstable rotation quaternion, but now considering a small deviation from the vertical axis (Fig. 31).

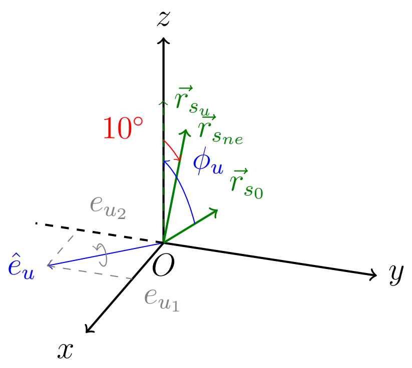


Figure 31: Cubli non-equilibrium position rotation quaternion

$$q_{ne} = \begin{bmatrix} \cos\frac{\phi_u - 10°}{2} \\ \vec{e}_u \sin\frac{\phi_u - 10°}{2} \end{bmatrix} . \tag{4.139}$$

Considering this new rotation quaternion as an initial condition with initial angular velocities 10 times faster (to guarantee it precesses), *i.e.*,

$$q(0) = q_{ne} , \qquad \vec{\omega}_c(0)' = 20\pi\frac{1}{\sqrt{3}}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} , \qquad \vec{\omega}_w(0)' = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} . \tag{4.140}$$

Fig. 32 show the results.

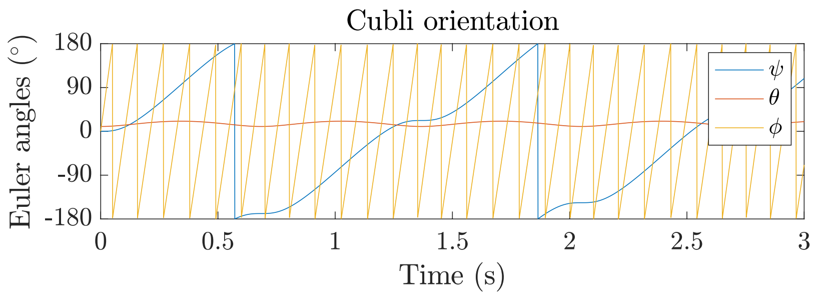


Figure 32: Simulation 6 - Dynamic equilibrium precession, nutation and spin (Euler angles)

Now all the three angles are changing, but in a standardized way. The nutation angle keeps oscillating around 10°, whereas the precession and spin angle kept increasing. Moreover, the spin velocity is clearly higher than the precession velocity, which is, in fact, expected in the spinning top motion. Although the initial spin velocity is 10 times that of the previous simulation, the frequency is not 10 times higher, meaning that the spin is now somewhat slower. This is because the Cubli is now also performing a gyroscopic precession.

Another interesting graph of the same simulation is a three-dimensional position of the Cubli center of mass, which can be seen in Fig. 33. Although not in scale, it gives a clear perspective of the spinning top motion.

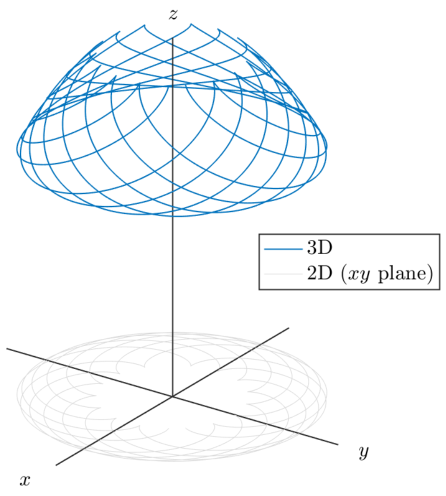


Figure 33: Simulation 6 - Dynamic equilibrium (center of mass)

To completely validate this motion, they will be compared to the well-known general equations of rotation of a symmetrical body about a fixed point O [19]

$$\begin{cases} I_o\left(\ddot{\psi}\sin\theta + 2\dot{\psi}\dot{\theta}\cos\theta\right) - I\dot{\theta}\left(\dot{\psi}\cos\theta + \dot{\phi}\right) = 0 \\ I_o\left(\ddot{\theta} - \dot{\psi}^2\sin\theta\cos\theta\right) + I\dot{\psi}\left(\dot{\psi}\cos\theta + \dot{\phi}\right)\sin\theta = m_c g|\vec{r}_c|\sin\theta \\ I\left(\ddot{\phi} + \ddot{\psi}\cos\theta - \dot{\psi}\dot{\theta}\sin\theta\right) = 0 \end{cases}, \tag{4.141}$$

where I_o is the Cubli maximum principal moment of inertia, around axis 1 and 2, whereas I is the minimum principal moment of inertia, around axis 3 (Fig. 34).

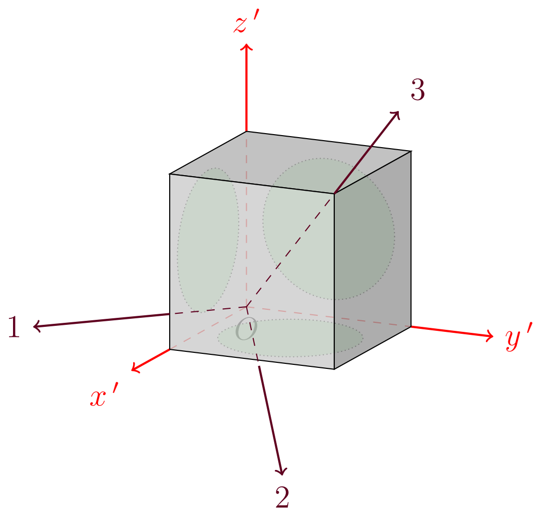


Figure 34: Cubli principal moment of inertia

These moments of inertia are the eigenvalues of the Cubli inertia tensor $\bar{I}_c$ and are given by

$$I = \bar{I}_{c_{xx}} - \bar{I}_{c_{xy}} \,, \qquad I_o = \bar{I}_{c_{xx}} + 2\bar{I}_{c_{xy}} \,. \tag{4.142}$$

Considering the same previous initial conditions, but now in terms of Euler angles, *i.e.*,

$$\begin{aligned} \phi(0) &= 0\,, & \dot{\phi}(0) &= 20\pi\,, \\ \theta(0) &= 10^\circ\,, & \dot{\theta}(0) &= 0\,, \\ \psi(0) &= 0\,, & \dot{\psi}(0) &= 0\,, \end{aligned} \tag{4.143}$$

and simulating (4.141), the result is the same of Fig. 32, confirming that the dynamic equations are consistent.

Next, the steady precession case is considered. For this motion to happen, the Cubli should have constant spin and precession velocities, and also a constant nutation angle, *i.e.*, $\ddot{\psi} = \ddot{\phi} = \dot{\theta} = 0$. This simplifies (4.141) to a single equation

$$(Io - I)\,\dot{\psi}^2 \cos\theta - I\dot{\psi}\dot{\phi} + m_c g|\vec{r}_c| = 0\,. \tag{4.144}$$

From (4.144), it is possible to calculate the precession velocity in terms of the spin velocity and nutation angle

$$\dot{\psi} = \frac{I\dot{\phi} \pm \sqrt{I^2\dot{\phi}^2 - 4\,(Io - I)\cos\theta m_c g|\vec{r}_c|}}{2\,(Io - I)\cos\theta}\,. \tag{4.145}$$

Note that, for this equation to be valid, the square root term must be real, meaning that there is a minimum spin velocity needed for steady precession

$$\dot{\phi} \geq \frac{2}{I}\sqrt{(Io - I)\cos\theta m_c g|\vec{r}_c|}\,. \tag{4.146}$$

The spin velocity and nutation angle of the previous simulation satisfy (4.146), but for the Cubli to present steady precession, the precession velocity from (4.145) should be $\psi_1 = 4.40$ rad/s or $\psi_2 = 22.19$ rad/s. Assuming the same initial conditions but now with $\psi(0) = \psi_1$, which means that $\vec{\omega}_c(0) = \begin{bmatrix}38.47 & 38.47 & 39.40\end{bmatrix}^T$, and simulating (4.124) instead of (4.141), yield the results presented in Fig. 35. As it can be seen, the Cubli is now clearly in steady precession, showing once again the consistency of the model.

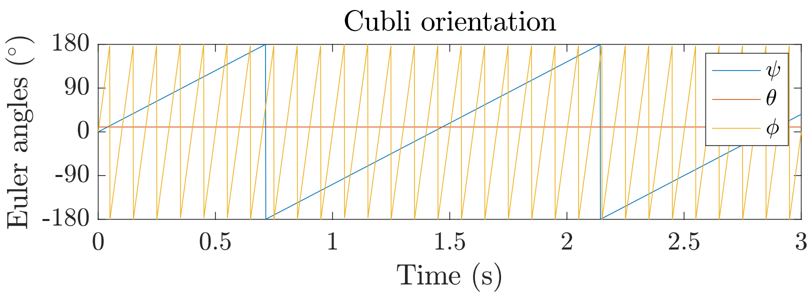


Figure 35: Simulation 8 - Dynamic equilibrium (Euler angles)

4.5.3 Poinsot trajectories

Poinsot trajectories are a geometrical method for visualizing the torque-free motion of a rotating rigid body. Since the system needs to be in torque-free motion, gravity will be neglected. The conservation of angular momentum implies that in the absence of applied torques, $\vec{H}$ is conserved in an inertial coordinate frame ($\frac{d\vec{H}}{dt} = 0$). The conservation of energy implies that in the absence of input torques and energy dissipation, T is also conserved ($\frac{dT}{dt} = 0$). Considering the principal axes, it is possible to write $\vec{H} = \begin{bmatrix}I_o\omega_1 & I_o\omega_2 & I\omega_3\end{bmatrix}^T$, so that the total angular momentum is simply the magnitude of this vector

$$H = \sqrt{I_o^2\omega_1^2 + I_o^2\omega_2^2 + I^2\omega_3^2}\,. \tag{4.147}$$

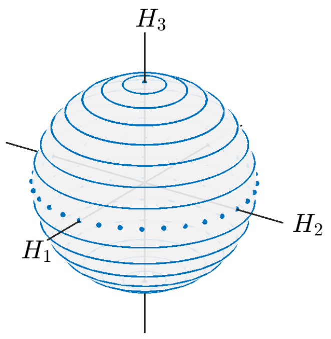
The angular kinetic energy, also considering the principal axes, is given by

$$T = \frac{1}{2}I_o\omega_1^2 + \frac{1}{2}I_o\omega_2^2 + \frac{1}{2}I\omega_3^2\,. \tag{4.148}$$

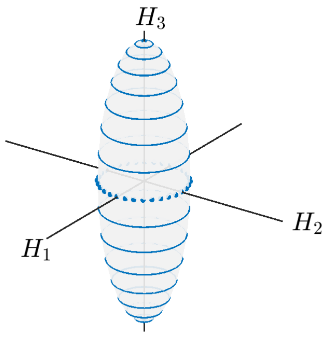
Writing (4.147) and (4.148) in terms of the angular momentum vector components along the principal axes yields

$$\begin{cases} H^2 = H_1^2 + H_2^2 + H_3^2 \\ 2T = \dfrac{H_1^2}{I_o} + \dfrac{H_2^2}{I_o} + \dfrac{H_3^2}{I} \end{cases}, \tag{4.149}$$

which are equivalent to two constraints for the 3D angular momentum vector $\vec{H}$. The angular momentum constrains $\vec{H}$ to lie on a sphere, whereas the kinetic energy constrains $\vec{H}$ to lie on an ellipsoid. These two surfaces intersections define the possible solutions for $\vec{H}$.



(a) Constant angular momentum



(b) Constant kinetic energy

Figure 36: Simulations 9 and 10 - Poinsot trajectories

Simulations in Fig. 36a considered various initial angular velocities, all with the same total H. Each line or dot is a different simulation and represents an intersection with the kinetic energy ellipsoid. The surface created is clearly a sphere, which is expected for a constant angular momentum. Moreover, because each simulation has constant H_3, the body is axisymmetric along this axis, which is in fact the case for the Cubli.

Considering now the same kinetic energy T, yields Fig. 36b. In this case, the surface is an ellipsoid, which is also expected for constant kinetic energy. Now each line or dot represents an intersection with the angular momentum sphere. Moreover, because two moments of inertia are the same and the third one is smaller than the other two, the shape is in fact a prolate spheroid, which is a particular case of an ellipsoid.

5 ANALYSIS

Before diving into the control of the Cubli, its stability and controllability properties will be analyzed.

5.1 Linearized dynamics without reaction wheels

When the Cubli is at rest perfectly balanced on its unstable equilibrium position, *i.e.*,

$$q(0) = q_u = \begin{bmatrix} \frac{\sqrt{\frac{2}{3}\left(3+\sqrt{3}\right)}}{2} \\ \frac{\sqrt{\frac{1}{3}\left(3-\sqrt{3}\right)}}{2} \\ -\frac{\sqrt{\frac{1}{3}\left(3-\sqrt{3}\right)}}{2} \\ 0 \end{bmatrix}, \qquad \vec{\omega}_c{}'(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \tag{5.1}$$

the linearized dynamics without the reaction wheels are

$$\begin{bmatrix} \dot{q} \\ \dot{\vec{\omega}}_c \end{bmatrix} = \left[\begin{array}{c:c} 0_{4\times4} & \frac{1}{2}G^T \\ \hdashline \bar{I}_c^{-1}K & -\bar{I}_c^{-1}B \end{array}\right] \begin{bmatrix} q \\ \vec{\omega}_c \end{bmatrix} + \begin{bmatrix} 0_{4\times3} \\ \bar{I}_c^{-1} \end{bmatrix} \vec{\tau}, \tag{5.2}$$

where

$$\begin{aligned} G &= G(q)\Big|_{q=q_u} \\ &= G(q_u) \\ &= \begin{bmatrix} q_{u_1} & q_{u_0} & -q_{u_3} & q_{u_2} \\ q_{u_2} & q_{u_3} & q_{u_0} & -q_{u_1} \\ q_{u_3} & -q_{u_2} & q_{u_1} & q_{u_0} \end{bmatrix}, \end{aligned} \tag{5.3}$$

$$\begin{aligned} K &= \frac{\partial}{\partial q}\left(\bar{m}_c g l \left(G(q)\Gamma q\right)\right)\Big|_{q=q_u} \\ &= \bar{m}_c g l \left(G(q)\Gamma - G(\Gamma q)\right)\Big|_{q=q_u} \\ &= \bar{m}_c g l \left(G(q_u)\Gamma - G(\Gamma q_u)\right)\ , \end{aligned} \tag{5.4}$$

and

$$\begin{aligned} B &= \frac{\partial}{\partial \vec{\omega}_c}\left(b\left(G(q)\Lambda q \left(G(q)\Lambda q\right)^T\right)\vec{\omega}_c\right)\Big|_{q=q_u} \\ &= b\left[G(q)\Lambda q\left(G(q)\Lambda q\right)^T\right]\Big|_{q=q_u} \\ &= b\left[G(q_u)\Lambda q_u\left(G(q_u)\Lambda q_u\right)^T\right] \\ &= b 1_{3\times 3}\,, \end{aligned} \tag{5.5}$$

being $1_{3\times3}$ a 3×3 matrix with all elements equal to one.

Its characteristic equation is given by

$$\begin{aligned} \det\left(sI_{7\times7} - A\right) &= 0 \\ \det\left(\left[\begin{array}{c|c} sI_{4\times4} & 0_{4\times3} \\ \hline 0_{3\times4} & sI_{3\times3} \end{array}\right] - \left[\begin{array}{c|c} 0_{4\times4} & \frac{1}{2}G^T \\ \hline \bar{I}_c^{-1}K & -\bar{I}_c^{-1}B \end{array}\right]\right) &= 0 \\ \det\left(\left[\begin{array}{c|c} sI_{4\times4} & -\frac{1}{2}G^T \\ \hline -\bar{I}_c^{-1}K & sI_{3\times3} + \bar{I}_c^{-1}B \end{array}\right]\right) &= 0 \\ \det\left(sI_{4\times4}\right)\det\left(sI_{3\times3} + \bar{I}_c^{-1}B - \bar{I}_c^{-1}K[sI_{4\times4}]^{-1}\frac{1}{2}G^T\right) &= 0 \\ s^4\det\left(\frac{1}{s}\left(s^2 I_{3\times3} + s\bar{I}_c^{-1}B - \frac{1}{2}\bar{I}_c^{-1}KG^T\right)\right) &= 0 \\ s^{\cancel{4}}\frac{1}{\cancel{s^3}}\det\left(s^2 I_{3\times3} + \frac{\omega_{n_1}}{3}s\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} - 2\frac{\omega_{n_0}^2}{3}\begin{bmatrix}2&-1&-1\\-1&2&-1\\-1&-1&2\end{bmatrix}\right) &= 0 \\ \underbrace{s}_{\substack{\text{quaternion}\\\text{redundancy}}}\quad\underbrace{s\left(s+\omega_{n_1}\right)}_{\substack{\text{yaw}\\\text{dynamics}}}\underbrace{\left(s^2-\omega_{n_0}^2\right)^2}_{\substack{\text{roll/pitch}\\\text{dynamics}}} &= 0\,, \end{aligned} \tag{5.6}$$

where ω_{n_0} is the natural frequency of the roll and pitch dynamics, whereas ω_{n_1} is the natural frequency of the yaw dynamics, given by

$$\omega_{n_0} = \sqrt{\frac{\bar{m}_c g l \sqrt{3}}{\bar{I}_{c_{xx}} - \bar{I}_{c_{xy}}}}, \qquad \omega_{n_1} = \frac{b}{\bar{I}_{c_{xx}} + 2\bar{I}_{c_{xy}}}. \tag{5.7}$$

Note that $\bar{I}_{c_{xx}} - \bar{I}_{c_{xy}}$ and $\bar{I}_{c_{xx}} + 2\bar{I}_{c_{xy}}$ are the Cubli principal moments of inertia derived in (4.142), which are, in fact, the moments of inertia around the roll and pitch motion and around the yaw motion, respectively.

Although quaternions have been utilized the whole time, the characteristic equation of the linearized dynamics is clearly described in terms of Euler angles. Roll and pitch dynamics are unstable due to its poles being located at $\pm\omega_{n_0}$, whereas yaw dynamics is marginally stable due to its poles being located at 0 and $-\omega_{n_1}$. Moreover, there is also an extra pole at 0, which is inherited from the rotation quaternion kinematic equation since a rotation quaternion is a redundant way to describe an orientation.

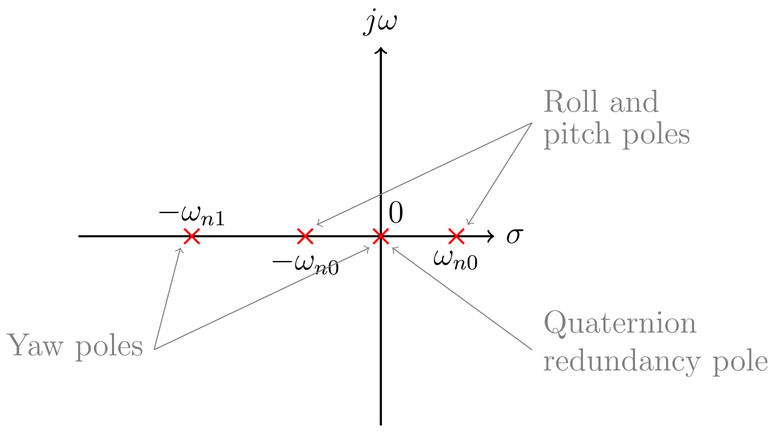


Figure 37: Open loop poles (without reaction wheels)

The controllability matrix has $\text{rank}(\mathcal{C}) = 6$, whereas the system has dimension $n = 7$. However, even that $\text{rank}(\mathcal{C}) \neq n$, the system is fully controllable since one of the system states is redundant due to quaternion representation. In other words, although quaternions are being utilized (which includes an extra redundant state), the system still has 3 d.o.f. and thus its "physical" dimension remains $n = 6$.

5.2 Linearized dynamics with reaction wheels angular velocity

Let us now consider the linearized dynamics with the reaction wheels angular velocity

$$\begin{bmatrix} \dot{q} \\ \dot{\vec{\omega}}_c \\ \dot{\vec{\omega}}_w \end{bmatrix} = \left[\begin{array}{c:c:c} 0_{4\times4} & \frac{1}{2}G^T & 0_{4\times3} \\ \hdashline \bar{I}_c^{-1}K & -\bar{I}_c^{-1}B & \bar{I}_c^{-1}F \\ \hdashline 0_{3\times4} & 0_{3\times3} & -\bar{I}_w^{-1}F \end{array}\right] \begin{bmatrix} q \\ \vec{\omega}_c \\ \vec{\omega}_w \end{bmatrix} + \left[\begin{array}{c} 0_{4\times3} \\ \hdashline \bar{I}_c^{-1} \\ \hdashline I_w^{-1} \end{array}\right] \vec{\tau}\,, \tag{5.8}$$

where

$$F = b_w I_{3\times3}\,. \tag{5.9}$$

The characteristic equation has an extra term

$$\begin{aligned}
\det\left(sI_{10\times10} - A\right) &= 0 \\
\det\left(\left[\begin{array}{c:c:c} sI_{4\times4} & 0_{4\times3} & 0_{4\times3} \\ \hdashline 0_{3\times4} & sI_{3\times3} & 0_{3\times3} \\ \hdashline 0_{3\times4} & 0_{3\times3} & sI_{3\times3} \end{array}\right] - \left[\begin{array}{c:c:c} 0_{4\times4} & \frac{1}{2}G^T & 0_{4\times3} \\ \hdashline \bar{I}_c^{-1}K & -\bar{I}_c^{-1}B & \bar{I}_c^{-1}F \\ \hdashline 0_{3\times4} & 0_{3\times3} & -\bar{I}_w^{-1}F \end{array}\right]\right) &= 0 \\
\det\left(\left[\begin{array}{c:c:c} sI_{4\times4} & -\frac{1}{2}G^T & 0_{4\times3} \\ \hdashline -\bar{I}_c^{-1}K & sI_{3\times3} + \bar{I}_c^{-1}B & -\bar{I}_c^{-1}F \\ \hdashline 0_{3\times4} & 0_{3\times3} & sI_{3\times3} + \bar{I}_w^{-1}F \end{array}\right]\right) &= 0 \\
\det\left(sI_{4\times4}\right)\det\left(\left[\begin{array}{c:c} sI_{3\times3} + \bar{I}_c^{-1}B & -\bar{I}_c^{-1}F \\ \hdashline 0_{3\times3} & sI_{3\times3} + \bar{I}_w^{-1}F \end{array}\right] - \left[\begin{array}{c} -\bar{I}_c^{-1}K \\ \hdashline 0_{3\times4} \end{array}\right] {sI_{4\times4}}^{-1} \left[\begin{array}{c:c} -\frac{1}{2}G^T & 0_{4\times3} \end{array}\right]\right) &= 0 \\
\det\left(sI_{4\times4}\right)\det\left(\left[\begin{array}{c:c} sI_{3\times3} + \bar{I}_c^{-1}B & -\bar{I}_c^{-1}F \\ \hdashline 0_{3\times3} & sI_{3\times3} + \bar{I}_w^{-1}F \end{array}\right] - \frac{1}{s}\left[\begin{array}{c:c} \frac{1}{2}\bar{I}_c^{-1}KG^T & 0_{3\times3} \\ \hdashline 0_{3\times3} & 0_{3\times3} \end{array}\right]\right) &= 0 \\
s^4\det\left(\frac{1}{s}\left[\begin{array}{c:c} s^2 I_{3\times3} + s\bar{I}_c^{-1}B - \frac{1}{2}\bar{I}_c^{-1}KG^T & -s\bar{I}_c^{-1}F \\ \hdashline 0_{3\times3} & s^2 I_{3\times3} + s\bar{I}_w^{-1}F \end{array}\right]\right) &= 0 \\
\cancel{s^4}\frac{1}{s^{\cancel{6}2}}\det\left(\left[s^2 I_{3\times3} + s\bar{I}_c^{-1}B - \frac{1}{2}\bar{I}_c^{-1}KG^T\right]\right)\det\left(\left[s^2 I_{3\times3} + s\bar{I}_w^{-1}F\right]\right) &= 0 \\
\frac{1}{s^2}\det\left(\left[s^2 I_{3\times3} + \frac{\omega_{n_1}}{3}s\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} - 2\frac{\omega_{n0}^2}{3}\begin{bmatrix}2&-1&-1\\-1&2&-1\\-1&-1&2\end{bmatrix}\right]\right)\det\left(\left[s^2 I_{3\times3} + \omega_{n_2} s I_{3\times3}\right]\right) &= 0 \\
\underbrace{s}_{\substack{\text{quaternion}\\ \text{redundancy}}}\ \underbrace{\left(s+\omega_{n_2}\right)^3}_{\substack{\text{reac. wheel}\\ \text{dynamics}}}\ \underbrace{s\left(s+\omega_{n_1}\right)}_{\substack{\text{yaw}\\ \text{dynamics}}}\underbrace{\left(s^2-\omega_{n0}^2\right)^2}_{\substack{\text{roll/pitch}\\ \text{dynamics}}} &= 0\,,
\end{aligned} \tag{5.10}$$

where ω_{n_2} is the natural frequency of the reaction wheel dynamics, given by

$$\omega_{n_2} = \frac{b_w}{I_{w_{xx}}} \,. \tag{5.11}$$

The reaction wheels angular velocities dynamics is stable due to its pole being located at $-\omega_{n_2}$. Although the viscous friction of the motors couples the Cubli and the reaction wheel dynamics, it does not interfere in the linearized roll, pitch and yaw dynamics, since its poles remained unchanged.

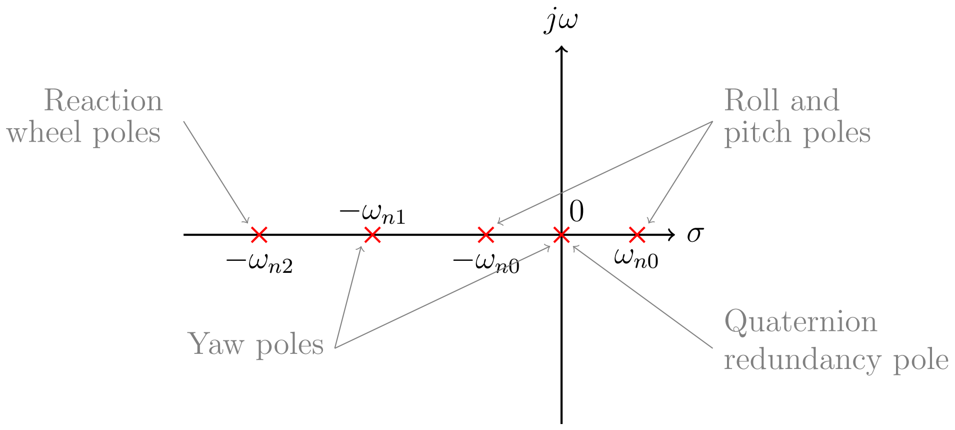


Figure 38: Open loop poles (with reaction wheels angular velocity)

The controllability matrix now has $\text{rank}(\mathcal{C}) = 9$, whereas the system now has dimension $n = 10$. Disregarding the quaternion redundancy, its "physical" dimension is $n = 9$, which means that the system is still fully controllable.

5.3 Linearized dynamics with reaction wheels angular velocity and displacement

Let us now consider the full linearized dynamics, that is, with the reaction wheels angular velocity and displacement

$$\begin{bmatrix} \dot{q} \\ \dot{\vec{\theta}}_w \\ \dot{\vec{\omega}}_c \\ \dot{\vec{\omega}}_w \end{bmatrix} = \left[\begin{array}{c:c:c:c} 0_{4\times4} & 0_{4\times3} & \frac{1}{2}G^T & 0_{4\times3} \\ \hdashline 0_{3\times4} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} \\ \hdashline \bar{I}_c^{-1}K & 0_{3\times3} & -\bar{I}_c^{-1}B & \bar{I}_c^{-1}F \\ \hdashline 0_{3\times4} & 0_{3\times3} & 0_{3\times3} & -\bar{I}_w^{-1}F \end{array}\right] \begin{bmatrix} q \\ \vec{\theta}_w \\ \vec{\omega}_c \\ \vec{\omega}_w \end{bmatrix} + \left[\begin{array}{c} 0_{4\times3} \\ \hdashline 0_{3\times3} \\ \hdashline \bar{I}_c^{-1} \\ \hdashline I_w^{-1} \end{array}\right] \vec{\tau} \,, \tag{5.12}$$

The characteristic equation has another extra term

$$
\begin{aligned}
&\det\left(sI_{13\times 13}-A\right)=0\\
&\det\left(\left[\begin{array}{c|c|c|c} sI_{4\times 4} & 0_{4\times 3} & 0_{4\times 3} & 0_{4\times 3}\\ \hline 0_{3\times 4} & sI_{3\times 3} & 0_{3\times 3} & 0_{3\times 3}\\ \hline 0_{3\times 4} & 0_{3\times 3} & sI_{3\times 3} & 0_{3\times 3}\\ \hline 0_{3\times 4} & 0_{3\times 3} & 0_{3\times 3} & sI_{3\times 3}\end{array}\right]-\left[\begin{array}{c|c|c|c} 0_{4\times 4} & 0_{4\times 3} & \frac{1}{2}G^T & 0_{4\times 3}\\ \hline 0_{3\times 4} & 0_{3\times 3} & 0_{3\times 3} & I_{3\times 3}\\ \hline \bar{I}_c^{-1}K & 0_{3\times 3} & -\bar{I}_c^{-1}B & \bar{I}_c^{-1}F\\ \hline 0_{3\times 4} & 0_{3\times 3} & 0_{3\times 3} & -\bar{I}_w^{-1}F\end{array}\right]\right)=0\\
&\det\left(\left[\begin{array}{c|c|c|c} sI_{4\times 4} & 0_{4\times 3} & -\frac{1}{2}G^T & 0_{4\times 3}\\ \hline 0_{3\times 4} & sI_{3\times 3} & 0_{3\times 3} & -I_{3\times 3}\\ \hline -\bar{I}_c^{-1}K & 0_{3\times 3} & sI_{3\times 3}+\bar{I}_c^{-1}B & -\bar{I}_c^{-1}F\\ \hline 0_{3\times 4} & 0_{3\times 3} & 0_{3\times 3} & sI_{3\times 3}+\bar{I}_w^{-1}F\end{array}\right]\right)=0\\
&\det\left(\left[\begin{array}{c|c} sI_{4\times 4} & 0_{4\times 3}\\ \hline 0_{3\times 4} & sI_{3\times 3}\end{array}\right]\right)\det\left(\left[\begin{array}{c|c} sI_{3\times 3}+\bar{I}_c^{-1}B & -\bar{I}_c^{-1}F\\ \hline 0_{3\times 3} & sI_{3\times 3}+\bar{I}_w^{-1}F\end{array}\right]\right.\\
&\qquad\left.-\left[\begin{array}{c|c} -\bar{I}_c^{-1}K & 0_{3\times 3}\\ \hline 0_{3\times 4} & 0_{3\times 3}\end{array}\right]\left[\begin{array}{c|c} sI_{4\times 4} & 0_{4\times 3}\\ \hline 0_{3\times 4} & sI_{3\times 3}\end{array}\right]^{-1}\left[\begin{array}{c|c} -\frac{1}{2}G^T & 0_{4\times 3}\\ \hline 0_{3\times 3} & -I_{3\times 3}\end{array}\right]\right)=0\\
&\det\left(s\left[I_{7\times 7}\right]\right)\det\left(\left[\begin{array}{c|c} sI_{3\times 3}+\bar{I}_c^{-1}B & -\bar{I}_c^{-1}F\\ \hline 0_{3\times 3} & sI_{3\times 3}+\bar{I}_w^{-1}F\end{array}\right]-\frac{1}{s}\left[\begin{array}{c|c} \frac{1}{2}\bar{I}_c^{-1}KG^T & 0_{3\times 3}\\ \hline 0_{3\times 3} & 0_{3\times 3}\end{array}\right]\right)=0\\
&s^7\det\left(\frac{1}{s}\left[\begin{array}{c|c} s^2I_{3\times 3}+s\bar{I}_c^{-1}B-\frac{1}{2}\bar{I}_c^{-1}KG^T & -s\bar{I}_c^{-1}F\\ \hline 0_{3\times 3} & s^2I_{3\times 3}+s\bar{I}_w^{-1}F\end{array}\right]\right)=0\\
&s^{\cancel{7}}\frac{1}{\cancel{s^6}}\det\left(\left[s^2I_{3\times 3}+s\bar{I}_c^{-1}B-\frac{1}{2}\bar{I}_c^{-1}KG^T\right]\right)\det\left(\left[s^2I_{3\times 3}+s\bar{I}_w^{-1}F\right]\right)=0\\
&s\det\left(\left[s^2I_{3\times 3}+\frac{\omega_{n_1}}{3}s\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}-2\frac{\omega_{n_0}^2}{3}\begin{bmatrix}2&-1&-1\\-1&2&-1\\-1&-1&2\end{bmatrix}\right]\right)\det\left(\left[s^2I_{3\times 3}+\omega_{n_2}sI_{3\times 3}\right]\right)=0\\
&\underbrace{s}_{\substack{\text{quaternion}\\ \text{redundancy}}}\ \underbrace{\left(s\left(s+\omega_{n_2}\right)\right)^3}_{\substack{\text{reac. wheel}\\ \text{dynamics}}}\underbrace{s\left(s+\omega_{n_1}\right)}_{\substack{\text{yaw}\\ \text{dynamics}}}\underbrace{\left(s^2-\omega_{n_0}^2\right)^2}_{\substack{\text{roll/pitch}\\ \text{dynamics}}}=0\,,
\end{aligned}
\tag{5.13}
$$

Which makes the reaction wheel dynamics marginally stable now due to an extra pole located at 0.

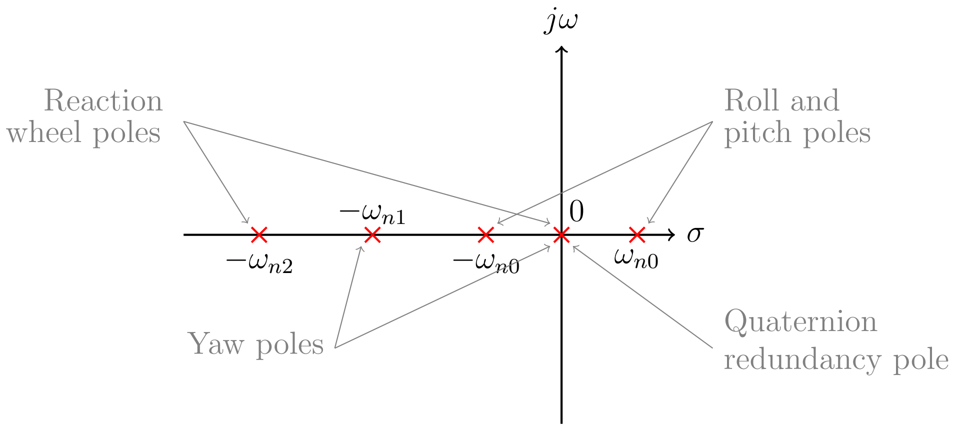


Figure 39: Open loop poles (with reaction wheels angular velocity and displacement)

The controllability matrix now has $\text{rank}(\mathcal{C}) = 11$, whereas the system now has dimension $n = 13$. Disregarding the quaternion redundancy, its "physical" dimension is $n = 12$, which means that now the system is no longer fully controllable.

That is, the inclusion of the angular velocities of the reaction wheels does not make the system uncontrollable, but the inclusion of the angular displacements does.

6 CONTROL

Once a mathematical representation of the system is developed, verified and analyzed, the next step is the analysis and design of a feedback control algorithm.

Much of control system design theory have been developed for linear systems, but nearly all real systems are nonlinear. What is often done is the linearization of the nonlinear system into a linear one, which works very well in conditions close to linearization.

However, in recent years, the availability of powerful low-cost microprocessor has spurred on great advances in the theory and applications of nonlinear control. Because of this, the focus will be to design a nonlinear control for the Cubli.

6.1 Control strategy

Although it is possible to control the yaw position or reaction wheel position, it is impossible to control both simultaneously. As will be shown further, this is due to the Cubli symmetry around the yaw axis.

One way to deal with this problem is to decouple the yaw dynamics and do not try to control it, thus, viscous friction with the surface will make the open-loop yaw dynamics marginally stable, as shown in [22].

Another way is is to control only the angular velocities of the reaction wheels, not concerning about their angular displacements. This makes the reaction wheels to never stop, but at least they keep turning at constant speed, without saturating.

An third way is to implement a trajectory control for the yaw axis to track a sinusoidal like signal with zero-mean. This causes the wheels to turn in one direction and then in the other periodically, such that the yaw dynamics, while never stops, stay bounded.

The last two strategies will be adressed in this study.

6.2 Attitude Controller

Initially, the focus will be only on the Cubli dynamics, without caring about controlling the wheels.

6.2.1 Feedback linearization

Adopting a new input vector $\vec{u}$ and making the input torque $\vec{\tau}$ equal to

$$\vec{\tau} = -\,\vec{\omega}_c \times \left(\bar{I}_c\vec{\omega}_c + I_w\vec{\omega}_w\right) - \bar{m}_c g l\,(G(q)\Gamma q) - b\left(G(q)\Lambda q\,(G(q)\Lambda q)^T\right)\vec{\omega}_c + \vec{\tau}_f(\vec{\omega}_w) - \bar{I}_c\vec{u}\,, \tag{6.1}$$

a feedback linearization law that cancels out all the gyroscopic terms, gravity torque, surface friction and motor friction is obtained (Fig. 40).

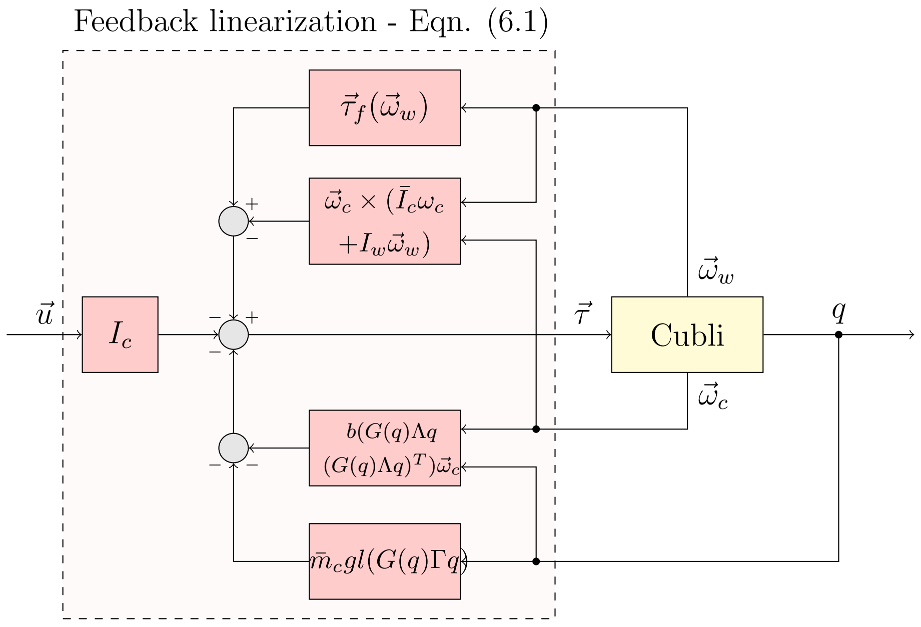

Feedback linearization - Eqn. (6.1)
$\vec{\tau}_f(\vec{\omega}_w)$
$\vec{\omega}_c \times (\bar{I}_c\omega_c$
$+I_w\vec{\omega}_w)$
$\vec{u}$
I_c
$\vec{\tau}$
Cubli
q
$\vec{\omega}_w$
$\vec{\omega}_c$
$b(G(q)\Lambda q$
$(G(q)\Lambda q)^T)\vec{\omega}_c$
$\bar{m}_c gl(G(q)\Gamma q)$
+
−

Figure 40: The Cubli with feedback linearization

By substituting (6.1) into (4.124), it reduces the system to

$$\begin{cases} \dot{q} = \frac{1}{2}G(q)^T\vec{\omega}_c \\ \dot{\vec{\omega}}_c = \vec{u} \end{cases} . \tag{6.2}$$

Although the angular velocity differential equation is now linear, the rotation quaternion differential equation is still nonlinear.

6.2.2 State regulator

Let q_r be an orientation quaternion reference and q_e be an orientation quaternion error, such that

$$q_r = \begin{bmatrix} q_{r0} \\ \vec{q}_r \end{bmatrix}, \qquad q_e = \begin{bmatrix} q_{e0} \\ \vec{q}_e \end{bmatrix}. \tag{6.3}$$

The orientation error represents the rotation needed from current orientation to match the orientation reference. In quaternion notation, consecutive rotations can be represented as multiplications between respective orientation quaternions, which means that

$$q_r = q \circ q_e \,. \tag{6.4}$$

By left-multiplying both sides of (6.4) with $\bar{q}$ and using (4.41), it is possible to isolate the orientation quaternion error

$$\begin{aligned} \bar{q} \circ q_r &= \cancel{\bar{q} \circ q} \circ q_e \\ q_e &= \bar{q} \circ q_r \,. \end{aligned} \tag{6.5}$$

Let $\vec{\omega}_r$ be an angular velocity vector reference and $\vec{\omega}_e$ be an angular velocity vector error, which can be represented as quaternions with zero real parts, such that

$$\omega_r = \begin{bmatrix} 0 \\ \vec{\omega}_r \end{bmatrix}, \qquad \omega_e = \begin{bmatrix} 0 \\ \vec{\omega}_e \end{bmatrix}. \tag{6.6}$$

The angular velocity error represents the difference between angular velocity reference and current angular velocity. However, because there is also a difference between orientations, the angular velocity reference must be rotated from orientation reference to the current orientation

$$\omega_e = q_e \circ \omega_r \circ \bar{q}_e - \omega_c \,, \tag{6.7}$$

which, in vector format, is the same as

$$\vec{\omega}_e = R(q_e)^T \vec{\omega}_r - \vec{\omega}_c \,. \tag{6.8}$$

When current orientation matches the orientation reference, no additional rotation is needed and thus the orientation quaternion error is $q_e = \begin{bmatrix} 1 & \vec{0} \end{bmatrix}^T$. Because q_e is not zero (and will never be since a orientation quaternion always have unit norm), (6.9) could not be used to guarantee asymptotically stable error dynamics

$$\ddot{q}_e + k_d \dot{q}_e + k_p q_e \neq \begin{bmatrix} 0 \\ \vec{0} \end{bmatrix} . \tag{6.9}$$

However, the vector part of the orientation quaternion error will be zero, which means that (6.10) could be used instead

$$\ddot{\vec{q}}_e + k_d \dot{\vec{q}}_e + k_p \vec{q}_e = \vec{0} , \tag{6.10}$$

an equation that ensures that the orientation quaternion error q_e converges exponentially to zero (for any positive value of k_p and k_d, the state regulator gains).

The first time derivative of q_e is obtained by differentiating (6.5)

$$\begin{aligned}
\dot{q}_e &= \frac{d}{dt} \left(\bar{q} \circ q_r \right) \\
\dot{q}_e &= \dot{\bar{q}} \circ q_r + \bar{q} \circ \dot{q}_r \\
\dot{q}_e &= \dot{\bar{q}} \circ \left(q \circ q_e \right) + \bar{q} \circ \left(\frac{1}{2} q_r \circ \omega_r \right) \\
\dot{q}_e &= \left(\dot{\bar{q}} \circ q \right) \circ q_e + \frac{1}{2} \left(\bar{q} \circ q_r \right) \circ \omega_r \\
\dot{q}_e &= -\frac{1}{2} \omega_c \circ q_e + \frac{1}{2} q_e \circ \omega_r \\
\dot{q}_e \circ \bar{q}_e &= -\frac{1}{2} \omega_c \circ \cancel{q_e \circ \bar{q}_e} + \frac{1}{2} q_e \circ \omega_r \circ \bar{q}_e \\
\dot{q}_e \circ \bar{q}_e &= \frac{1}{2} \left(q_e \circ \omega_r \circ \bar{q}_e - \omega_c \right) \\
\dot{q}_e \circ \bar{q}_e &= \frac{1}{2} \omega_e \\
\dot{q}_e \circ \cancel{\bar{q}_e \circ q_e} &= \frac{1}{2} \omega_e \circ q_e \\
\dot{q}_e &= \frac{1}{2} \omega_e \circ q_e ,
\end{aligned} \tag{6.11}$$

where its vector part is given by

$$\dot{\vec{q}}_e = \frac{1}{2} \left(q_{e0} I_{3\times3} - \tilde{q}_e \right) \vec{\omega}_e . \tag{6.12}$$

On the other hand, the second time derivative of q_e can be calculated by differentiating

(6.11)

$$\begin{aligned}
\ddot{q}_e &= \frac{d}{dt}\left(\frac{1}{2}\omega_e \circ q_e\right) \\
\ddot{q}_e &= \frac{1}{2}\dot{\omega}_e \circ q_e + \frac{1}{2}\omega_e \circ \dot{q}_e \\
\ddot{q}_e &= \frac{1}{2}\dot{\omega}_e \circ q_e + \frac{1}{2}\omega_e \circ \left(\frac{1}{2}\omega_e \circ q_e\right) \\
\ddot{q}_e &= \frac{1}{2}\dot{\omega}_e \circ q_e + \frac{1}{4}\omega_e \circ \omega_e \circ q_e\,, \qquad (6.13)
\end{aligned}$$

where its vector part is given by

$$\ddot{\vec{q}}_e = \frac{1}{2}\left(q_{e_0} I_{3\times3} - \tilde{q}_e\right)\dot{\vec{\omega}}_e - \frac{1}{4}\vec{\omega}_e^T \vec{\omega}_e \vec{q}_e\,. \qquad (6.14)$$

Substituting (6.12) and (6.14) into (6.10) yields

$$\begin{aligned}
\ddot{\vec{q}}_e + k_d \dot{\vec{q}}_e + k_p \vec{q}_e &= \vec{0} \\
\left(\frac{1}{2}\left(q_{e_0} I_{3\times3} - \tilde{q}_e\right)\dot{\vec{\omega}}_e - \frac{1}{4}\vec{\omega}_e^T \vec{\omega}_e \vec{q}_e\right) + k_d\left(\frac{1}{2}\left(q_{e_0} I_{3\times3} - \tilde{q}_e\right)\vec{\omega}_e\right) + k_p \vec{q}_e &= \vec{0} \\
\frac{1}{2}\left(q_{e_0} I_{3\times3} - \tilde{q}_e\right)\left(\dot{\vec{\omega}}_e + k_d \vec{\omega}_e\right) + \left(k_p - \frac{1}{4}\vec{\omega}_e^T \vec{\omega}_e\right)\vec{q}_e &= \vec{0} \\
\dot{\vec{\omega}}_e + k_d \vec{\omega}_e + 2(q_{e_0} - \tilde{q}_e)^{-1}\left(k_p - \frac{1}{4}\vec{\omega}_e^T \vec{\omega}_e\right)\vec{q}_e &= \vec{0} \\
\dot{\vec{\omega}}_e + k_d \vec{\omega}_e + 2\left(k_p - \frac{1}{4}\vec{\omega}_e^T \vec{\omega}_e\right)\frac{\vec{q}_e}{q_{e_0}} &= \vec{0}\,. \qquad (6.15)
\end{aligned}$$

The term $\frac{\vec{q}_e}{q_{e_0}}$ is the Gibbs vector error $\vec{\sigma}_e$, given by

$$\vec{\sigma}_e = \begin{bmatrix} e_{e_x} \tan\frac{\phi_e}{2} \\ e_{e_y} \tan\frac{\phi_e}{2} \\ e_{e_z} \tan\frac{\phi_e}{2} \end{bmatrix}\,. \qquad (6.16)$$

This vector is singular for $\phi_e = \pm 180°$, which appears to be a disadvantage of utilizing rotation quaternions as feedback control states. However, the biggest possible attitude error between two orientations is 180°, and if it is necessary to go to a reference in the longest path, a trajectory control may be utilized.

The time derivative of ω_e is obtained by differentiating (6.7)

$$
\begin{aligned}
\dot{\omega}_e &= \frac{d}{dt}\left(q_e \circ \omega_r \circ \bar{q}_e - \omega_c\right)\\
\dot{\omega}_e &= \dot{q}_e \circ \omega_r \circ \bar{q}_e + q_e \circ \left(\dot{\omega}_r \circ \bar{q}_e + \omega_r \circ \dot{\bar{q}}_e\right) - \dot{\omega}_c\\
\dot{\omega}_e &= \dot{q}_e \circ \omega_r \circ \bar{q}_e + q_e \circ \omega_r \circ \dot{\bar{q}}_e + q_e \circ \dot{\omega}_r \circ \bar{q}_e - \dot{\omega}_c\\
\dot{\omega}_e &= \left(\frac{1}{2}\omega_e \circ q_e\right) \circ \omega_r \circ \bar{q}_e + q_e \circ \omega_r \circ \left(-\frac{1}{2}\bar{q}_e \circ \omega_e\right) + q_e \circ \dot{\omega}_r \circ \bar{q}_e - \dot{\omega}_c\\
\dot{\omega}_e &= \frac{1}{2}\omega_e \circ \left(q_e \circ \omega_r \circ \bar{q}_e\right) + -\frac{1}{2}\left(q_e \circ \omega_r \circ \bar{q}_e\right) \circ \omega_e + q_e \circ \dot{\omega}_r \circ \bar{q}_e - \dot{\omega}_c\\
\dot{\omega}_e &= \frac{1}{2}\omega_e \circ \left(\omega_e + \omega_c\right) - \frac{1}{2}\left(\omega_e + \omega_c\right) \circ \omega_e + q_e \circ \dot{\omega}_r \circ \bar{q}_e - \dot{\omega}_c\\
\dot{\omega}_e &= \cancel{\frac{1}{2}\omega_e \circ \omega_e} + \frac{1}{2}\omega_e \circ \omega_c - \cancel{\frac{1}{2}\omega_e \circ \omega_e} - \frac{1}{2}\omega_c \circ \omega_e + q_e \circ \dot{\omega}_r \circ \bar{q}_e - \dot{\omega}_c\\
\dot{\omega}_e &= \frac{1}{2}\omega_e \circ \omega_c + \frac{1}{2}\omega_e \circ \omega_c + q_e \circ \dot{\omega}_r \circ \bar{q}_e - \dot{\omega}_c\\
\dot{\omega}_e &= \omega_e \circ \omega_c + q_e \circ \dot{\omega}_r \circ \bar{q}_e - \dot{\omega}_c\,, \qquad (6.17)
\end{aligned}
$$

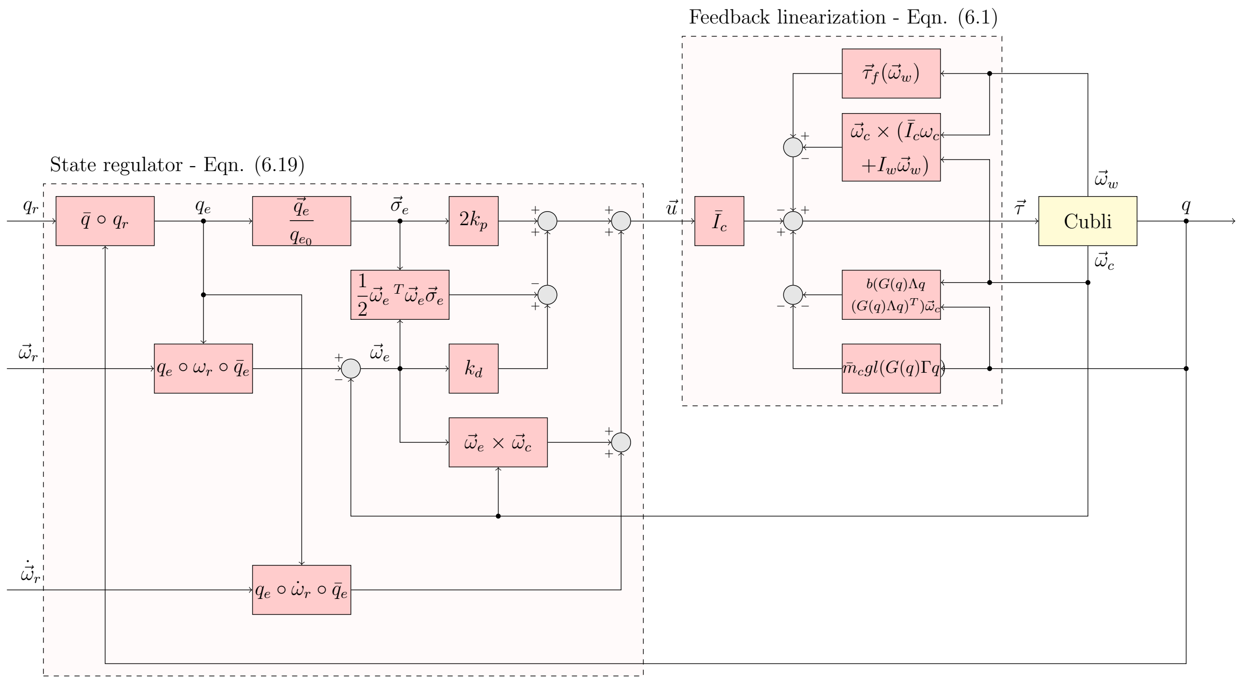
where its vector part is given by

$$\dot{\vec{\omega}}_e = \vec{\omega}_e \times \vec{\omega}_c + R(q_e)^T \dot{\vec{\omega}}_r - \dot{\vec{\omega}}_c\,. \qquad (6.18)$$

Isolating $\dot{\vec{\omega}}_c$ in (6.18) and substituting (6.15) in it yields the following control law

$$\vec{u} = \underbrace{2\left(k_p - \frac{1}{4}\vec{\omega}_e^T\vec{\omega}_e\right)\frac{\vec{q}_e}{q_{e_0}} + k_d\vec{\omega}_e}_{\text{Feedback}} \underbrace{+\vec{\omega}_e \times \vec{\omega}_c + R(q_e)^T\dot{\vec{\omega}}_r}_{\text{Feedfoward}}\,. \qquad (6.19)$$

This control law was derived at NASA Ames Research Center back in 1993 by Paielli and Bach [1, 25] for spacecraft attitude control (Fig. 41).



Feedback linearization - Eqn. (6.1)
State regulator - Eqn. (6.19)
q_r
$\bar{q} \circ q_r$
q_e
$\frac{\vec{q}_e}{q_{e_0}}$
$\vec{\sigma}_e$
$2k_p$
$\vec{u}$
$\bar{I}_c$
$\vec{\tau}$
Cubli
q
$\vec{\omega}_w$
$\vec{\omega}_c$
$\vec{\tau}_f(\vec{\omega}_w)$
$\vec{\omega}_c \times (\bar{I}_c\omega_c$
$+I_w\vec{\omega}_w)$
$b(G(q)\Lambda q$
$(G(q)\Lambda q)^T)\vec{\omega}_c$
$\bar{m}_c gl(G(q)\Gamma q)$
$\frac{1}{2}\vec{\omega}_e{}^T\vec{\omega}_e\vec{\sigma}_e$
$\vec{\omega}_r$
$q_e \circ \omega_r \circ \bar{q}_e$
$\vec{\omega}_e$
k_d
$\vec{\omega}_e \times \vec{\omega}_c$
$\dot{\vec{\omega}}_r$
$q_e \circ \dot{\omega}_r \circ \bar{q}_e$

Figure 41: The Cubli with state regulator and feedback linearization

For small rotations, the term $\vec{\omega}_e^T\vec{\omega}_e$ is close to zero, q_{e_0} is close to one, $\vec{\omega}_e \times \vec{\omega}_c$ is close to zero and $R(q_e)^T$ is close to identity, which further simplifies the control law

$$\vec{u} \approx 2k_p\vec{q}_e + k_d\vec{\omega}_e + \dot{\vec{\omega}}_r\,. \tag{6.20}$$

Moreover, the orientation quaternion error and angular velocity error vectors are approximate to the Euler angles errors

$$\vec{q}_e = \begin{bmatrix} e_x \sin\frac{\phi_e}{2} \\ e_y \sin\frac{\phi_e}{2} \\ e_z \sin\frac{\phi_e}{2} \end{bmatrix} \approx \begin{bmatrix} \frac{\phi_e}{2} \\ \frac{\theta_e}{2} \\ \frac{\psi_e}{2} \end{bmatrix}, \quad \vec{\omega}_e = \begin{bmatrix} \omega_{e_x} \\ \omega_{e_y} \\ \omega_{e_z} \end{bmatrix} \approx \begin{bmatrix} \dot{\phi}_e \\ \dot{\theta}_e \\ \dot{\psi}_e \end{bmatrix}. \tag{6.21}$$

Substituting (6.21) into (6.20) yields a state regulator that is equal to the one commonly utilized with Euler angles when dealing with small rotations

$$\vec{u} \approx \begin{bmatrix} k_p(\phi_r - \phi) + k_d(\dot{\phi}_r - \dot{\phi}) + \ddot{\phi}_r \\ k_p(\theta_r - \theta) + k_d(\dot{\theta}_r - \dot{\theta}) + \ddot{\theta}_r \\ k_p(\psi_r - \psi) + k_d(\dot{\psi}_r - \dot{\psi}) + \ddot{\psi}_r \end{bmatrix}. \tag{6.22}$$

This means that, for small rotations, the derived nonlinear control law of (6.19) is equivalent to a linear one dynamically linearized at the reference position.

6.2.3 Controller gains

Substituting (6.19) into (6.2) and rewriting the first differential equation in terms of the Gibbs vector error and no longer in terms of quaternions yields

$$\begin{cases} \dot{\vec{\sigma}}_e = \frac{1}{2}\left(I_{3\times3} - \tilde{\sigma}_e + \vec{\sigma}_e\vec{\sigma}_e^T\right)\vec{\omega}_e \\ \dot{\vec{\omega}}_e = -2\left(k_p - \frac{1}{4}\vec{\omega}_e^T\vec{\omega}_e\right)\vec{\sigma}_e - k_d\vec{\omega}_e \end{cases} . \tag{6.23}$$

When the Cubli is in its unstable equilibrium position, *i.e.*, $\vec{\sigma}_e = \vec{\omega}_e = \dot{\vec{\omega}}_r = \vec{0}$, the closed loop linearized dynamics is

$$\begin{bmatrix} \dot{\vec{\sigma}}_e \\ \dot{\vec{\omega}}_e \end{bmatrix} = \left[\begin{array}{c:c} 0_{3\times3} & \frac{1}{2}I_{3\times3} \\ \hdashline -2k_pI_{3\times3} & -k_dI_{3\times3} \end{array}\right] \begin{bmatrix} \vec{\sigma}_e \\ \vec{\omega}_e \end{bmatrix} . \tag{6.24}$$

Its characteristic equation is

$$\begin{aligned}
\det\left(sI_{6\times6} - A\right) &= 0 \\
\det\left(\left[\begin{array}{c:c} sI_{3\times3} & 0 \\ \hdashline 0 & sI_{3\times3} \end{array}\right] - \left[\begin{array}{c:c} 0_{3\times3} & \frac{1}{2}I_{3\times3} \\ \hdashline -2k_pI_{3\times3} & -k_dI_{3\times3} \end{array}\right]\right) &= 0 \\
\det\left(\left[\begin{array}{c:c} sI_{3\times3} & -\frac{1}{2}I_{3\times3} \\ \hdashline 2k_pI_{3\times3} & \left(s + k_d\right)I_{3\times3} \end{array}\right]\right) &= 0 \\
\det\left(\left[sI_{3\times3}\right]\right)\det\left(\left[\left(s+k_d\right)I_{3\times3}\right] - \left[2k_pI_{3\times3}\right]\left[sI_{3\times3}\right]^{-1}\left[-\frac{1}{2}I_{3\times3}\right]\right) &= 0 \\
s^3\det\left(\left[I_{3\times3}\right]\right)\det\left(\left[\left(s+k_d\right)I_{3\times3}\right] + \left[\frac{k_p}{s}I_{3\times3}\right]\right) &= 0 \\
s^3\det\left(\left[\left(s + k_d + \frac{k_p}{s}\right)I_{3\times3}\right]\right) &= 0 \\
s^3\left(s + k_d + \frac{k_p}{s}\right)^3\det\left(\left[I_{3\times3}\right]\right) &= 0 \\
s^3\left(\frac{1}{s}\left(s^2 + k_ds + k_p\right)\right)^3 &= 0 \\
s^3\frac{1}{s^3}\left(s^2 + k_ds + k_p\right)^3 &= 0 \\
\left(s^2 + k_ds + k_p\right)^3 &= 0 \, . \qquad (6.25)
\end{aligned}$$

Comparing (6.25) with the characteristic equation of a generic 2nd order system with two complex poles

$$s^2 + 2\zeta\omega_n s + \omega_n = 0 \, , \tag{6.26}$$

yields the following values for the controller gains in terms of the desired closed-loop parameters ζ and ω_n

$$\begin{cases} k_p = \omega_n^2 \\ k_d = 2\zeta\omega_n \end{cases} . \tag{6.27}$$

6.3 Attitude and Wheel Velocity Controller

Since the Cubli is influenced by the acceleration of the reaction wheels, it may happen that one reaction wheel velocity saturates for a while. Moreover, the Cubli construction may be imperfect, or attitude sensors may be misaligned, so what seems to be an equilibrium position may not be, and the reaction wheels will keep accelerating trying to maintain that erroneous equilibrium. It is thus desirable to achieve the dual goals of stabilizing the Cubli and keep the reaction wheels velocities small.

6.3.1 State regulator

To achieve this, the control law of (6.19) must be slightly modified by also having feedback from the reaction wheels angular velocities, such that

$$\vec{u} = \underbrace{2\left(k_p - \frac{1}{4}\vec{\omega}_e^T\vec{\omega}_e\right)\frac{\vec{q}_e}{q_{e_0}} + k_d\vec{\omega}_e}_{\text{Attitude feedback}} \underbrace{+\vec{\omega}_e \times \vec{\omega}_c + R(q_e)^T\dot{\vec{\omega}}_r}_{\text{Attitude feedfoward}} \underbrace{-k_{d_w}\vec{\omega}_w}_{\text{Wheel feedback}} . \tag{6.28}$$

This heuristic strategy was considered, for instance, by Block et al. [2], creators of the original 2D reaction wheel pendulum.

The full nonlinear control law (Fig. 42) is composed of the feedback linearization from (6.1) and the state regulator from (6.28).

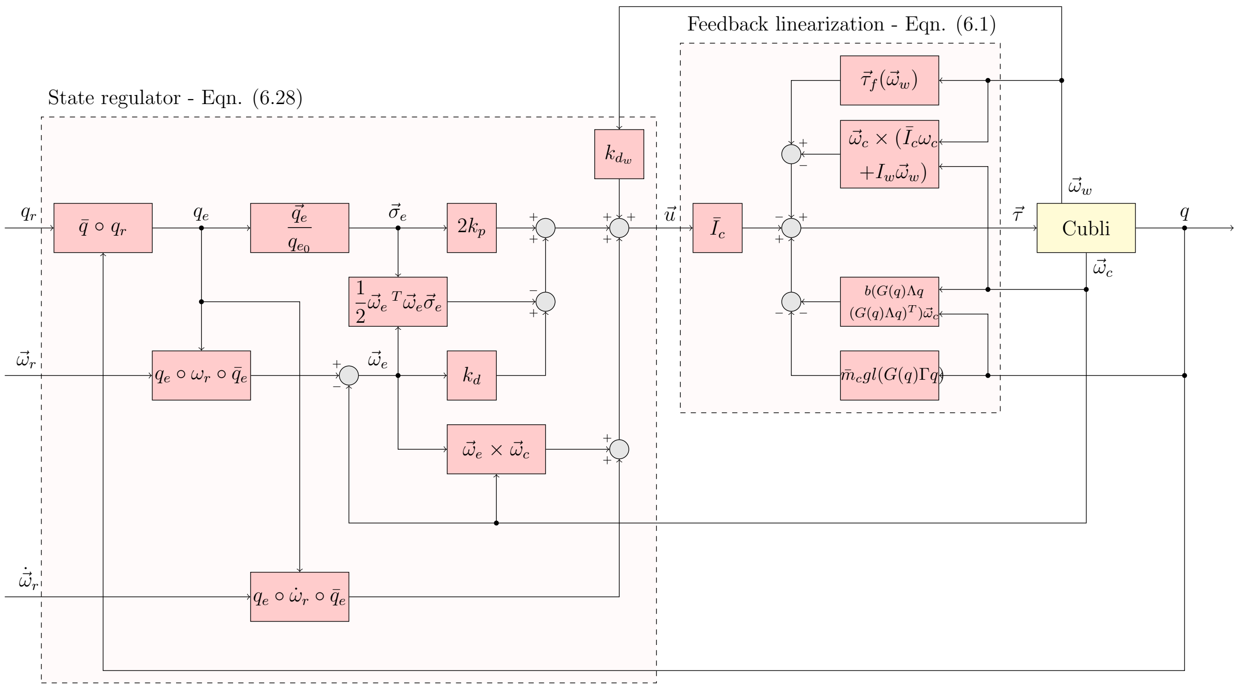

Feedback linearization - Eqn. (6.1)
State regulator - Eqn. (6.28)
q_r
$\bar{q} \circ q_r$
q_e
$\frac{\vec{q}_e}{q_{e_0}}$
$\vec{\sigma}_e$
$2k_p$
k_{d_w}
$\vec{u}$
$\bar{I}_c$
$\vec{\tau}$
Cubli
q
$\vec{\omega}_w$
$\vec{\omega}_c$
$\vec{\tau}_f(\vec{\omega}_w)$
$\vec{\omega}_c \times (\bar{I}_c\vec{\omega}_c + I_w\vec{\omega}_w)$
$b(G(q)\Lambda q (G(q)\Lambda q)^T)\vec{\omega}_c$
$\bar{m}_c g l (G(q)\Gamma q)$
$\frac{1}{2}\vec{\omega}_e{}^T\vec{\omega}_e\vec{\sigma}_e$
$\vec{\omega}_r$
$q_e \circ \omega_r \circ \bar{q}_e$
$\vec{\omega}_e$
k_d
$\vec{\omega}_e \times \vec{\omega}_c$
$\dot{\vec{\omega}}_r$
$q_e \circ \dot{\omega}_r \circ \bar{q}_e$

Figure 42: The Cubli with state regulator and feedback linearization (with wheels)

Note that, if the objective is just to stabilize the Cubli, *i.e.*, $\vec{\omega}_r = \dot{\vec{\omega}}_r = \vec{0}$, the control law reduces to

$$\vec{u} = \underbrace{2\left(k_p - \frac{1}{4}\vec{\omega}_c^T\vec{\omega}_c\right)\frac{\vec{q}_e}{q_{e_0}} - k_d\vec{\omega}_c}_{\text{Attitude feedback}} \quad \underbrace{-k_{d_w}\vec{\omega}_w}_{\text{Wheel feedback}} \quad . \tag{6.29}$$

6.3.2 Controller gains

Substituting (6.28) into (6.2) yields

$$\begin{cases} \dot{\vec{\sigma}}_e = \frac{1}{2}\left(I_{3\times3} - \tilde{\sigma}_e + \vec{\sigma}_e\vec{\sigma}_e^T\right)\vec{\omega}_e \\ \dot{\vec{\omega}}_e = -2\left(k_p - \frac{1}{4}\vec{\omega}_e^T\vec{\omega}_e\right)\frac{\vec{q}_e}{q_{e_0}} - k_d\vec{\omega}_e + k_{d_w}\vec{\omega}_w \\ \dot{\vec{\omega}}_w = -I_w^{-1}\Big(\vec{\omega}_c \times \left(\bar{I}_c\vec{\omega}_c + I_w\vec{\omega}_w\right) + \bar{m}_c g l\left(G(q)\Gamma q\right) + b\left(G(q)\Lambda q\left(G(q)\Lambda q\right)^T\right) \\ \qquad -\bar{I}_c\left(2\left(k_p - \frac{1}{4}\vec{\omega}_e^T\vec{\omega}_e\right)\vec{\sigma}_e + k_d\vec{\omega}_w + R(q_e)^T\dot{\vec{\omega}}_r + \vec{\omega}_e \times \vec{\omega}_c - k_{d_w}\vec{\omega}_w\right)\Big) \end{cases} . \tag{6.30}$$

When the Cubli is in its unstable equilibrium position, *i.e.*, $\vec{\sigma}_e = \vec{\omega}_e = \vec{\omega}_w = \dot{\vec{\omega}}_r = \vec{0}$, the closed loop linearized dynamics is

$$\begin{bmatrix} \dot{\sigma}_e \\ \dot{\vec{\omega}}_e \\ \dot{\vec{\omega}}_w \end{bmatrix} = \left[\begin{array}{c:c:c} 0_{3\times3} & \frac{1}{2}I_{3\times3} & 0_{3\times3} \\ \hdashline -2k_pI_{3\times3} & -k_dI_{3\times3} & k_{d_w}I_{3\times3} \\ \hdashline -2K_w - 2k_pI_w^{-1}\bar{I}_c & B_w - k_dI_w^{-1}\bar{I}_c & k_{d_w}I_w^{-1}\bar{I}_c \end{array}\right] \begin{bmatrix} \sigma_e \\ \vec{\omega}_e \\ \vec{\omega}_w \end{bmatrix} , \tag{6.31}$$

where

$$I_w^{-1}\bar{I}_c = \frac{1}{3}\begin{bmatrix}\beta+2\gamma & \beta-\gamma & \beta-\gamma\\ \beta-\gamma & \beta+2\gamma & \beta-\gamma\\ \beta-\gamma & \beta-\gamma & \beta+2\gamma\end{bmatrix}, \quad K_w = \frac{\delta}{3}\begin{bmatrix}2 & -1 & -1\\ -1 & 2 & -1\\ -1 & -1 & 2\end{bmatrix}, \quad B_w = \frac{\varepsilon}{3}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}, \tag{6.32}$$

and

$$\beta = \frac{\bar{I}_{c_{xx}} + 2\bar{I}_{c_{xy}}}{I_{w_{xx}}}, \qquad \gamma = \frac{\bar{I}_{c_{xx}} - \bar{I}_{c_{xy}}}{I_{w_{xx}}}, \qquad \delta = \frac{\bar{m}_c g l\sqrt{3}}{I_{w_{xx}}}, \qquad \varepsilon = \frac{b}{I_{w_{xx}}}. \tag{6.33}$$

Its characteristic equation is

$$\det\left(sI_{9\times 9} - A\right) = 0$$

$$\det\left(\left[\begin{array}{c:c:c} sI_{3\times3} & 0_{3\times3} & 0_{3\times3}\\ \hdashline 0_{3\times3} & sI_{3\times3} & 0_{3\times3}\\ \hdashline 0_{3\times3} & 0_{3\times3} & sI_{3\times3}\end{array}\right] - \left[\begin{array}{c:c:c} 0_{3\times3} & \frac{1}{2}I_{3\times3} & 0_{3\times3}\\ \hdashline -2k_pI_{3\times3} & -k_dI_{3\times3} & k_{d_w}I_{3\times3}\\ \hdashline -2K_w - 2k_pI_w^{-1}\bar{I}_c & B_w - k_dI_w^{-1}\bar{I}_c & k_{d_w}I_w^{-1}\bar{I}_c\end{array}\right]\right) = 0$$

$$\det\left(\left[\begin{array}{c:c:c} sI_{3\times3} & -\frac{1}{2}I_{3\times3} & 0_{3\times3}\\ \hdashline 2k_pI_{3\times3} & (s+k_d)I_{3\times3} & -k_{d_w}I_{3\times3}\\ \hdashline 2K_w + 2k_pI_w^{-1}\bar{I}_c & -B_w + k_dI_w^{-1}\bar{I}_c & sI_{3\times3} - k_{d_w}I_w^{-1}\bar{I}_c\end{array}\right]\right) = 0$$

$$\det\left(sI_{3\times3}\right)\det\left(\left[\begin{array}{c:c} (s+k_d)I_{3\times3} & -k_{d_w}I_{3\times3}\\ \hdashline -B_w + k_dI_w^{-1}\bar{I}_c & sI_{3\times3} - k_{d_w}I_w^{-1}\bar{I}_c\end{array}\right]\right.$$
$$\left. - \left[\begin{array}{c} 2k_pI_{3\times3}\\ \hdashline 2K_w + 2k_pI_w^{-1}\bar{I}_c\end{array}\right]\left[sI_{3\times3}\right]^{-1}\left[\begin{array}{c:c} -\frac{1}{2}I_{3\times3} & 0_{3\times3}\end{array}\right]\right) = 0$$

$$\det\left(sI_{3\times3}\right)$$
$$\det\left(\left[\begin{array}{c:c} (s+k_d)I_{3\times3} & -k_{d_w}I_{3\times3}\\ \hdashline -B_w + k_dI_w^{-1}\bar{I}_c & sI_{3\times3} - k_{d_w}I_w^{-1}\bar{I}_c\end{array}\right] - \frac{1}{s}\left[\begin{array}{c:c} -k_pI_{3\times3} & 0_{3\times3}\\ \hdashline -K_w - k_pI_w^{-1}\bar{I}_c & 0_{3\times3}\end{array}\right]\right) = 0$$

$$s^3\det\left(\frac{1}{s}\left[\begin{array}{c:c} (s^2+k_ds+k_p)I_{3\times3} & -k_{d_w}sI_{3\times3}\\ \hdashline K_w - B_ws + (k_ds+k_p)I_w^{-1}\bar{I}_c & s^2I_{3\times3} - k_{d_w}sI_w^{-1}\bar{I}_c\end{array}\right]\right) = 0$$

$$\cancel{s^3}\frac{1}{s^{\cancel{6}3}}\det\left(\left[(s^2+k_ds+k_p)I_{3\times3}\right]\right)\det\left(\left[s^2I_{3\times3} - k_{d_w}sI_w^{-1}\bar{I}_c\right]\right.$$
$$\left. - \left[K_w - B_ws + (k_ds+k_p)I_w^{-1}\bar{I}_c\right]\left[(s^2+k_ds+k_p)I_{3\times3}\right]^{-1}\left[(-k_{d_w}s)I_{3\times3}\right]\right) = 0$$

$$\frac{1}{s^3}(s^2+k_ds+k_p)^3\det\left(\left[s^2I_{3\times3} - k_{d_w}sI_w^{-1}\bar{I}_c\right]\right.$$
$$\left. + \frac{k_{d_w}s}{(s^2+k_ds+k_p)}\left[K_w - B_ws + (k_ds+k_p)I_w^{-1}\bar{I}_c\right]\right) = 0$$

$$\begin{aligned}
&\frac{1}{s^3}(s^2+k_ds+k_p)^3 \det\left(\frac{1}{(s^2+k_ds+k_p)}\left[s^2(s^2+k_ds+k_p)I_{3\times 3}\right.\right. \\
&\qquad \left.\left.+k_{d_w}s\left(K_w - B_w + \left((k_ds+k_p)+(s^2+k_ds+k_p)\right)I_w^{-1}\bar{I}_c\right)\right]\right) = 0 \\
&\qquad\qquad \frac{1}{s^3}\cancel{(s^2+k_ds+k_p)^3}\,\frac{1}{\cancel{(s^2+k_ds+k_p)^3}} \\
&\qquad \det\left(\left[s^2(s^2+k_ds+k_p)I_{3\times 3} + k_{d_w}s\left(K_w - B_ws - s^2I_w^{-1}\bar{I}_c\right)\right]\right) = 0 \\
&\frac{1}{s^3}\det\left(\left[s^2(s^2+k_ds+k_p)I_{3\times 3} + k_{d_w}s\left(K_w - B_ws - s^2I_w^{-1}\bar{I}_c\right)\right]\right) = 0 \\
&\underbrace{s\left(s^2+\left(k_d-\beta k_{d_w}\right)s+\left(k_p-\varepsilon k_{d_w}\right)\right)}_{\text{Yaw closed-loop dynamics}}\underbrace{\left(s^3+\left(k_d-\gamma k_{d_w}\right)s^2+k_ps+\delta k_{d_w}\right)^2}_{\text{Roll/Pitch closed-loop dynamics}} = 0\,. \qquad (6.34)
\end{aligned}$$

Note that, by varying just the controller gains k_p, k_d and k_{d_w}, it is possible to freely allocate yaw and roll and pitch closed-loop poles. However, because of the negative sign in the reaction wheel gain k_{d_w}, this gain cannot be too big as it can maket one or both dynamics unstable.

Comparing the roll and pitch terms of (6.34) with the characteristic equation of a generic 3th order system with two complex poles and one real pole, given by $(s^2+2\zeta\omega_ns+\omega_n^2)$ $(s+\alpha\zeta\omega_n)=0$, which is equivalent to

$$s^3+\zeta\omega_n\left(2+\alpha\right)s^2+\omega_n^2\left(1+2\alpha\zeta^2\right)s+\alpha\zeta\omega_n^3 = 0\,, \qquad (6.35)$$

yields the following values for the controller gains in terms of the desired closed-loop dynamics ζ, ω_n and α and parameters β, γ and δ

$$\begin{cases} k_p = \omega_n^2\left(1+2\alpha\zeta^2\right) \\ k_d = \zeta\omega_n\left(2+\alpha\right)+\gamma\dfrac{\alpha\zeta\omega_n^3}{\delta} \\ k_{d_w} = \dfrac{\alpha\zeta\omega_n^3}{\delta} \end{cases}. \qquad (6.36)$$

Note that, if $\alpha = 0$, the controller gains k_p and k_d are equal to those ones derived in (6.27), whereas the controller gain k_{d_w} is equal to zero. By choosing a small enough value of α, one guarantees that the reaction wheel dynamics would be slow enough to not interfere in the Cubli dynamics. In other words, the Cubli roll and pitch closed-loop poles will be sufficient faster than the reaction wheel closed-loop poles.

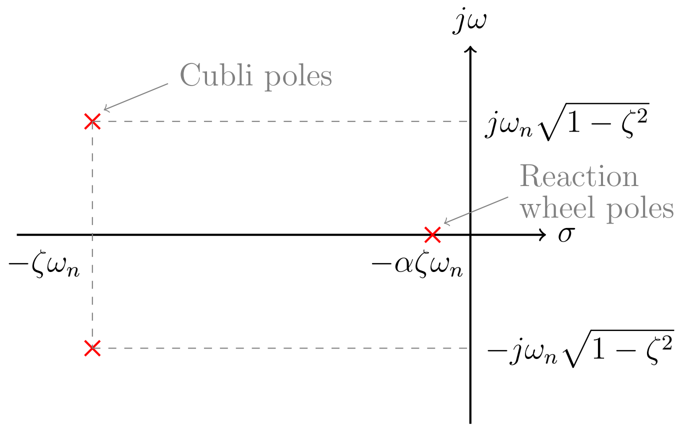


Figure 43: Closed loop poles

6.4 Attitude and Wheel Angle Controller

To keep tighter control of the reaction wheels, the angular displacement, in addition to angular velocity, can also be feedback. This will cause the angular velocity to converge to zero and not just stabilize at a constant value.

This scheme was also used by Block et al. [2], creators of the original 2D reaction wheel pendulum. However, there is an important difference here: the 2D reaction wheel pendulum, unlike the 3D version of this thesis, is fully controllable even with angular displacement feedback. Therefore, it is expected that this strategy will not work properly, as will be proved later through simulations and experiments.

Nonetheless, as discussed at the beginning of this chapter, the controllability problem can be mitigated, but not eliminated, with an strategy of sinusoidal trajectory control.

6.4.1 State regulator

The control law of (6.28) must be slightly modified by also having feedback from the reaction wheels angular displacement, such that

$$\vec{u} = \underbrace{2\left(k_p - \frac{1}{4}\vec{\omega}_e^T\vec{\omega}_e\right)\frac{\vec{q}_e}{q_{e_0}} + k_d\vec{\omega}_e}_{\text{Attitude feedback}} \underbrace{+\vec{\omega}_e \times \vec{\omega}_c + R(q_e)^T\dot{\vec{\omega}}_r}_{\text{Attitude feedfoward}} \underbrace{-k_{p_w}\vec{\theta}_w - k_{d_w}\vec{\omega}_w}_{\text{Wheel feedback}} . \tag{6.37}$$

The full nonlinear control law (Fig. 44) is composed of the feedback linearization from (6.1) and the state regulator from (6.37).

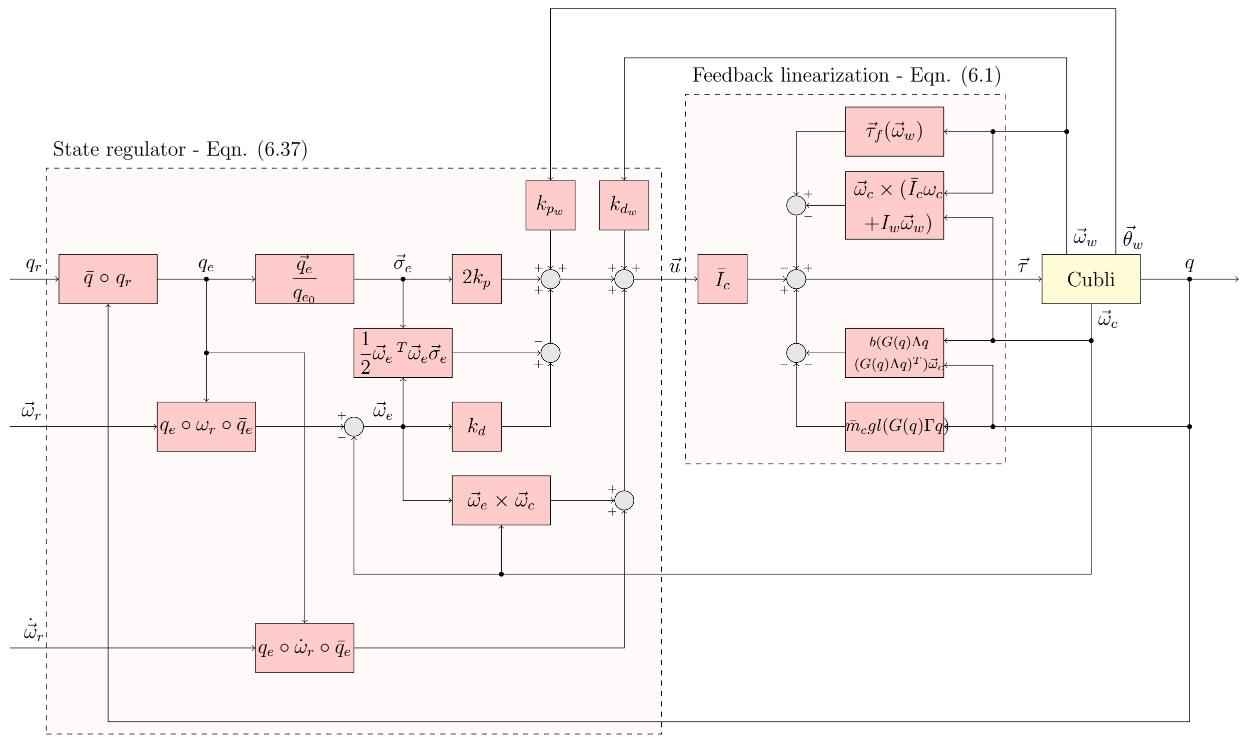

Feedback linearization - Eqn. (6.1)
State regulator - Eqn. (6.37)
q_r
$\bar{q} \circ q_r$
q_e
$\frac{\vec{q}_e}{q_{e_0}}$
$\vec{\sigma}_e$
$2k_p$
k_{p_w}
k_{d_w}
$\vec{u}$
$\bar{I}_c$
$\vec{\tau}_f(\vec{\omega}_w)$
$\vec{\omega}_c \times (\bar{I}_c\omega_c + I_w\vec{\omega}_w)$
$\vec{\tau}$
Cubli
$\vec{\omega}_w$
$\vec{\theta}_w$
$\vec{\omega}_c$
q
$\frac{1}{2}\vec{\omega}_e{}^T\vec{\omega}_e\vec{\sigma}_e$
$b(G(q)\Lambda q\,(G(q)\Lambda q)^T)\vec{\omega}_c$
$\vec{\omega}_r$
$q_e \circ \omega_r \circ \bar{q}_e$
$\vec{\omega}_e$
k_d
$\bar{m}_c gl(G(q)\Gamma q)$
$\vec{\omega}_e \times \vec{\omega}_c$
$\dot{\vec{\omega}}_r$
$q_e \circ \dot{\omega}_r \circ \bar{q}_e$

Figure 44: The Cubli with state regulator and feedback linearization (with wheels)

Once again, if the objective is just to stabilize the Cubli, *i.e.*, $\vec{\omega}_r = \dot{\vec{\omega}}_r = \vec{0}$, the control law reduces to

$$\vec{u} = \underbrace{2\left(k_p - \frac{1}{4}\vec{\omega}_c^T\vec{\omega}_c\right)\frac{\vec{q}_e}{q_{e_0}} - k_d\vec{\omega}_c}_{\text{Attitude feedback}} \underbrace{-k_{p_w}\vec{\theta}_w - k_{d_w}\vec{\omega}_w}_{\text{Wheel feedback}}\ . \tag{6.38}$$

6.4.2 Controller gains

Substituting (6.37) into (6.2) yields

$$\begin{cases} \dot{\vec{\sigma}}_e = \frac{1}{2}\left(I_{3\times3} - \tilde{\sigma}_e + \vec{\sigma}_e\vec{\sigma}_e^T\right)\vec{\omega}_e \\ \dot{\vec{\theta}}_w = \vec{\omega}_w \\ \dot{\vec{\omega}}_e = -2\left(k_p - \frac{1}{4}\vec{\omega}_e^T\vec{\omega}_e\right)\frac{\vec{q}_e}{q_{e_0}} - k_d\vec{\omega}_e + k_{p_w}\vec{\theta}_w + k_{d_w}\vec{\omega}_w \\ \dot{\vec{\omega}}_w = -I_w^{-1}\Big(\vec{\omega}_c \times \left(\bar{I}_c\vec{\omega}_c + I_w\vec{\omega}_w\right) + \bar{m}_c gl\left(G(q)\Gamma q\right) + b\left(G(q)\Lambda q\left(G(q)\Lambda q\right)^T\right) \\ \qquad -\bar{I}_c\left(2\left(k_p - \frac{1}{4}\vec{\omega}_e^T\vec{\omega}_e\right)\vec{\sigma}_e + k_d\vec{\omega}_w + R(q_e)^T\dot{\vec{\omega}}_r + \vec{\omega}_e \times \vec{\omega}_c - k_{p_w}\vec{\theta}_w - k_{d_w}\vec{\omega}_w\right)\Big) \end{cases} . \tag{6.39}$$

When the Cubli is in its unstable equilibrium position, *i.e.*, $\vec{\sigma}_e = \vec{\theta}_w = \vec{\omega}_e = \vec{\omega}_w =$

$\dot{\vec{\omega}}_r = \vec{0}$, the closed loop linearized dynamics is

$$\begin{bmatrix} \dot{\sigma}_e \\ \dot{\vec{\theta}}_w \\ \dot{\vec{\omega}}_e \\ \dot{\vec{\omega}}_w \end{bmatrix} = \left[\begin{array}{c:c:c:c} 0_{3\times3} & 0_{3\times3} & \frac{1}{2}I_{3\times3} & 0_{3\times3} \\ \hdashline 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} \\ \hdashline -2k_pI_{3\times3} & k_{p_w}I_{3\times3} & -k_dI_{3\times3} & k_{d_w}I_{3\times3} \\ \hdashline -2K_w - 2k_pI_w^{-1}\bar{I}_c & k_{p_w}I_w^{-1}\bar{I}_c & B_w - k_dI_w^{-1}\bar{I}_c & k_{d_w}I_w^{-1}\bar{I}_c \end{array}\right] \begin{bmatrix} \sigma_e \\ \vec{\theta}_w \\ \vec{\omega}_e \\ \vec{\omega}_w \end{bmatrix}, \tag{6.40}$$

Its characteristic equation is

$$\det\left(sI_{12\times12} - A\right) = 0$$

$$\det\left(\left[\begin{array}{c:c:c:c} sI_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ \hdashline 0_{3\times3} & sI_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ \hdashline 0_{3\times3} & 0_{3\times3} & sI_{3\times3} & 0_{3\times3} \\ \hdashline 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & sI_{3\times3} \end{array}\right] - \left[\begin{array}{c:c:c:c} 0_{3\times3} & 0_{3\times3} & \frac{1}{2}I_{3\times3} & 0_{3\times3} \\ \hdashline 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} \\ \hdashline -2k_pI_{3\times3} & k_{p_w}I_{3\times3} & -k_dI_{3\times3} & k_{d_w}I_{3\times3} \\ \hdashline -2K_w - 2k_pI_w^{-1}\bar{I}_c & k_{p_w}I_w^{-1}\bar{I}_c & B_w - k_dI_w^{-1}\bar{I}_c & k_{d_w}I_w^{-1}\bar{I}_c \end{array}\right]\right) = 0$$

$$\det\left(\left[\begin{array}{c:c:c:c} sI_{3\times3} & 0_{3\times3} & -\frac{1}{2}I_{3\times3} & 0_{3\times3} \\ \hdashline 0_{3\times3} & sI_{3\times3} & 0_{3\times3} & -I_{3\times3} \\ \hdashline 2k_pI_{3\times3} & -k_{p_w}I_{3\times3} & (s+k_d)I_{3\times3} & -k_{d_w}I_{3\times3} \\ \hdashline 2K_w + 2k_pI_w^{-1}\bar{I}_c & -k_{p_w}I_w^{-1}\bar{I}_c & -B_w + k_dI_w^{-1}\bar{I}_c & sI_{3\times3} - k_{d_w}I_w^{-1}\bar{I}_c \end{array}\right]\right) = 0$$

$$\det\left(\left[\begin{array}{c:c} sI_{3\times3} & 0_{3\times3} \\ \hdashline 0_{3\times3} & sI_{3\times3} \end{array}\right]\right)\det\left(\left[\begin{array}{c:c} (s+k_d)I_{3\times3} & -k_{d_w}I_{3\times3} \\ \hdashline -B_w + k_dI_w^{-1}\bar{I}_c & sI_{3\times3} - k_{d_w}I_w^{-1}\bar{I}_c \end{array}\right]\right.$$

$$\left. - \left[\begin{array}{c:c} 2k_pI_{3\times3} & -k_{p_w}I_{3\times3} \\ \hdashline 2K_w + 2k_pI_w^{-1}\bar{I}_c & -k_{p_w}I_w^{-1}\bar{I}_c \end{array}\right] \left[\begin{array}{c:c} sI_{3\times3} & 0_{3\times3} \\ \hdashline 0_{3\times3} & sI_{3\times3} \end{array}\right]^{-1} \left[\begin{array}{c:c} -\frac{1}{2}I_{3\times3} & 0_{3\times3} \\ \hdashline 0_{3\times3} & -I_{3\times3} \end{array}\right]\right) = 0$$

$$\det\left(s\left[I_{6\times6}\right]\right)$$

$$\det\left(\left[\begin{array}{c:c} (s+k_d)I_{3\times3} & -k_{d_w}I_{3\times3} \\ \hdashline -B_w + k_dI_w^{-1}\bar{I}_c & sI_{3\times3} - k_{d_w}I_w^{-1}\bar{I}_c \end{array}\right] - \frac{1}{s}\left[\begin{array}{c:c} -k_pI_{3\times3} & k_{p_w}I_{3\times3} \\ \hdashline -K_w - k_pI_w^{-1}\bar{I}_c & k_{p_w}I_w^{-1}\bar{I}_c \end{array}\right]\right) = 0$$

$$s^6\det\left(\frac{1}{s}\left[\begin{array}{c:c} (s^2 + k_ds + k_p)I_{3\times3} & -(k_{d_w}s + k_{p_w})I_{3\times3} \\ \hdashline K_w - B_ws + (k_ds + k_p)I_w^{-1}\bar{I}_c & s^2I_{3\times3} - (k_{d_w}s + k_{p_w})I_w^{-1}\bar{I}_c \end{array}\right]\right) = 0$$

$$
\begin{aligned}
&\cancel{s^6}\frac{1}{\cancel{s^6}}\det\left(\left[(s^2+k_ds+k_p)I_{3\times3}\right]\right)\det\left(\left[s^2I_{3\times3}-(k_{d_w}s+k_{p_w})I_w^{-1}\bar{I}_c\right]\right. \\
&\left.-\left[K_w-B_ws+(k_ds+k_p)I_w^{-1}\bar{I}_c\right]\left[(s^2+k_ds+k_p)I_{3\times3}\right]^{-1}\left[-(k_{d_w}s+k_{p_w})I_{3\times3}\right]\right)=0 \\
&(s^2+k_ds+k_p)^3\det\left(\left[s^2I_{3\times3}-(k_{d_w}s+k_{p_w})I_w^{-1}\bar{I}_c\right]\right. \\
&\left.+\frac{(k_{d_w}s+k_{p_w})}{(s^2+k_ds+k_p)}\left[K_w-B_ws+(k_ds+k_p)I_w^{-1}\bar{I}_c\right]\right)=0 \\
&(s^2+k_ds+k_p)^3\det\left(\frac{1}{(s^2+k_ds+k_p)}\left[s^2(s^2+k_ds+k_p)I_{3\times3}\right.\right. \\
&\left.\left.+(k_{d_w}s+k_{p_w})\left(K_w-B_w+\left((k_ds+k_p)+(s^2+k_ds+k_p)\right)I_w^{-1}\bar{I}_c\right)\right]\right)=0 \\
&\cancel{(s^2+k_ds+k_p)^3}\frac{1}{\cancel{(s^2+k_ds+k_p)^3}} \\
&\det\left(\left[s^2(s^2+k_ds+k_p)I_{3\times3}+(k_{d_w}s+k_{p_w})\left(K_w-B_ws-s^2I_w^{-1}\bar{I}_c\right)\right]\right)=0 \\
&\det\left(\left[s^2(s^2+k_ds+k_p)I_{3\times3}+(k_{d_w}s+k_{p_w})\left(K_w-B_ws-s^2I_w^{-1}\bar{I}_c\right)\right]\right)=0 \\
&\overbrace{s\left(s^3+(k_d-\beta k_{d_w})s^2+(k_p-\beta k_{p_w}-\varepsilon k_{d_w})s-\varepsilon k_{p_w}\right)}^{\text{Yaw closed-loop dynamics}} \\
&\underbrace{\left(s^4+(k_d-\gamma k_{d_w})s^3+(k_p-\gamma k_{p_w})s^2+\delta k_{d_w}s+\delta k_{p_w}\right)^2}_{\text{Roll/Pitch closed-loop dynamics}}=0\,.
\end{aligned}
\tag{6.41}
$$

Note that, by varying just the controller gains k_p, k_d, k_{p_w} and k_{d_w}, it is possible to freely allocate yaw or roll and pitch closed-loop poles, but never both simultaneously. What could be done is to allocate the roll and pitch poles in the desired locations and just ensure that the yaw dynamics remains stable. However, even this is not possible. Due to the negative sign in term ϵk_{p_w}, reaction wheel gain k_{p_w} would need to be negative to make yaw dynamics stable, but then roll and pitch dynamics would become unstable. There is no way out. Physically speaking, gravity acts as a non-restorative force in the roll and pitch dynamics, whereas surface friction acts as a restorative force in the yaw dynamics. This inversion of signs is exactly what makes the system not controllable.

Nonetheless, if the parameter ϵ is equal to zero, the yaw dynamics would be marginally stable. So, to reduce this problem, two things should be done: first is to position the Cubli on a smooth surface with the least possible friction to make b, and consequently ϵ, small; second is to reduce k_{p_w} the maximum, to make it similar to the previous case where there was no angle feedback and the system was controllable. However, this will only increase the time the system is controllable, but never extend it to infinity.

Comparing the roll and pitch terms of (6.41) with the characteristic equation of a generic 4th order system with two complex poles and two repeated real poles, given by

$\left(s^2+2\zeta\omega_n s+\omega_n^2\right)\left(s+\alpha\zeta\omega_n\right)^2=0$, which is equivalent to

$$s^4+2\zeta\omega_n\left(1+\alpha\right)s^3+\omega_n^2\left(1+\alpha\zeta^2\left(4+\alpha\right)\right)s^2+\left(2\alpha\zeta\omega_n^3\left(1+\alpha\zeta^2\right)\right)s+\alpha^2\zeta^2\omega_n^4=0\,, \quad (6.42)$$

yields the following values for the controller gains in terms of the desired closed-loop dynamics ζ, ω_n and α and parameters β, γ and δ

$$\begin{cases} k_p = \omega_n^2\left(1+\alpha\zeta^2\left(4+\alpha\right)\right)+\gamma\dfrac{\alpha^2\zeta^2\omega_n^4}{\delta} \\ k_d = 2\zeta\omega_n\left(1+\alpha\right)+\gamma\dfrac{2\alpha\zeta\omega_n^3\left(1+\alpha\zeta^2\right)}{\delta} \\ k_{p_w} = \dfrac{\alpha^2\zeta^2\omega_n^4}{\delta} \\ k_{d_w} = \dfrac{2\alpha\zeta\omega_n^3\left(1+\alpha\zeta^2\right)}{\delta} \end{cases}. \quad (6.43)$$

Once again, if $\alpha = 0$, the controller gains k_p and k_d are equal to those ones derived in (6.27), whereas the controller gains k_{p_w} and k_{d_w} are equal to zero. By choosing a small enough value of α, one guarantees that the reaction wheel dynamics would be slow enough to not interfere in the Cubli dynamics. In other words, the Cubli roll and pitch closed-loop poles will be sufficient faster than the reaction wheel closed-loop poles.

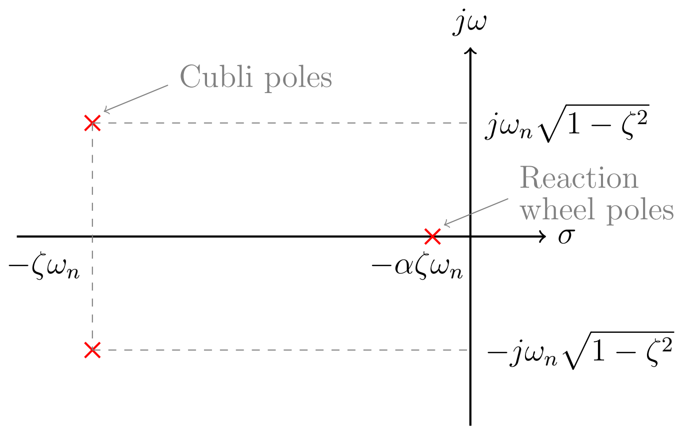


Figure 45: Closed loop poles

7 ESTIMATION

A controller is only useful as long as the states available for feedback are trustworthy. Typically, measured states have a lot of noise and uncertainty, and need to be handled smartly to generate what are called estimated states. We do this through filters and estimators, which generates “supposedly perfect” values. This is called “observed-based control” and is a very typical method for structuring control systems.

Each reaction wheel angular velocity and displacement was estimated from a dedicated 1st order state observer, which takes into account not only the angular velocity readings from the motors hall sensors but the applied torques and friction forces as well.

To estimate the Cubli angular velocity and orientation, a quaternion based complementary filter was developed that fuses rate gyroscope readings, which has low-frequency noise due to a constant bias being integrated over time, with accelerometer readings, which has high-frequency noise due to centripetal and tangential accelerations.

7.1 Reaction wheels angular velocity and displacement

The reaction wheels dynamics can be represented by the block diagram in Fig. 46.

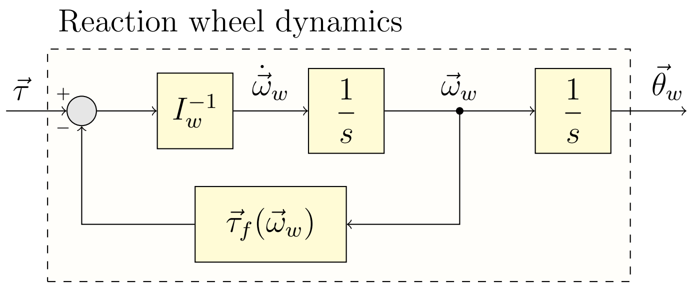


Figure 46: Reaction wheels dynamics

The only measured state is the angular velocity $\vec{\omega}_w$ through the motor hall sensors.

7.1.1 Low-pass filter

One way to estimate the reaction wheels angular velocity is using a low-pass filter. A low-pass filter attenuates signals above a certain cut-off frequency ω_c. It is widely used to filter noise, as they usually have a higher frequency than the signal being measured.

In order to estimate the angular velocity $\hat{\vec{\omega}}_w$ without or with reduced noise, the measured angular velocity $\vec{\omega}_w$ is passed through a low-pass filter. This can be represented by the block diagram in Fig. 47.

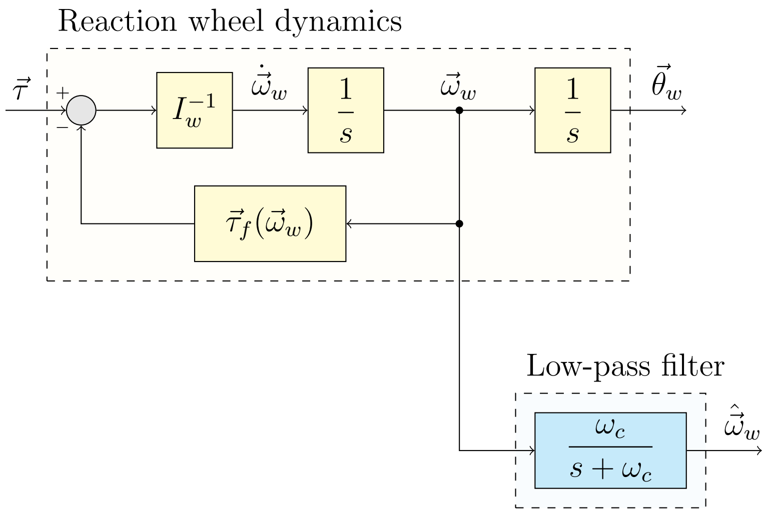


Figure 47: Reaction wheels low-pass filter

Since this filter will be implemented in a microcontroller, it becomes necessary to determine its discrete counterpart. First, let us obtain the corresponding differential equation, using the inverse Laplace transform

$$\begin{aligned}
\frac{\hat{\vec{\Omega}}_w(s)}{\vec{\Omega}_w(s)} &= \frac{\omega_c}{s+\omega_c} \\
(s+\omega_c)\,\hat{\vec{\Omega}}_w(s) &= \omega_c\vec{\Omega}_w(s) \\
s\hat{\vec{\Omega}}_w(s) + \omega_c\hat{\vec{\Omega}}_w(s) &= \omega_c\vec{\Omega}_w(s) \\
\frac{d}{dt}\hat{\vec{\omega}}_w(t) + \omega_c\hat{\vec{\omega}}_w(t) &= \omega_c\vec{\omega}_w(t)\,. \qquad (7.1)
\end{aligned}$$

Then, (7.1) will be discretized using the implicit Euler method[1]

$$\begin{aligned}
\frac{\hat{\vec{\omega}}_w(t) - \hat{\vec{\omega}}_w(t-\Delta t)}{\Delta t} + \omega_c \hat{\vec{\omega}}_w(t) &= \omega_c \vec{\omega}_w(t) \\
\hat{\vec{\omega}}_w(t) - \hat{\vec{\omega}}_w(t-\Delta t) + \omega_c \Delta t \hat{\vec{\omega}}_w(t) &= \omega_c \Delta t \vec{\omega}_w(t) \\
(1+\omega_c \Delta)\, \hat{\vec{\omega}}_w(t) &= \hat{\vec{\omega}}_w(t-\Delta t) + \omega_c \Delta t \vec{\omega}_w(t) \\
\hat{\vec{\omega}}_w(t) &= \underbrace{\frac{1}{1+\omega_c\Delta t}}_{(1-\alpha)} \hat{\vec{\omega}}_w(t-\Delta t) + \underbrace{\frac{\omega_c \Delta t}{1+\omega_c \Delta t}}_{\alpha} \vec{\omega}_w(t) \\
\hat{\vec{\omega}}_w(t) &= (1-\alpha)\, \hat{\vec{\omega}}_w(t-\Delta t) + \alpha \vec{\omega}_w(t)\,. \qquad (7.2)
\end{aligned}$$

Note that a discretized low-pass filter is nothing more than a weighted average between the old estimated value and the new measured value, and the constant α is exactly that weighting factor. It can be represented by the block diagram in Fig. 48.

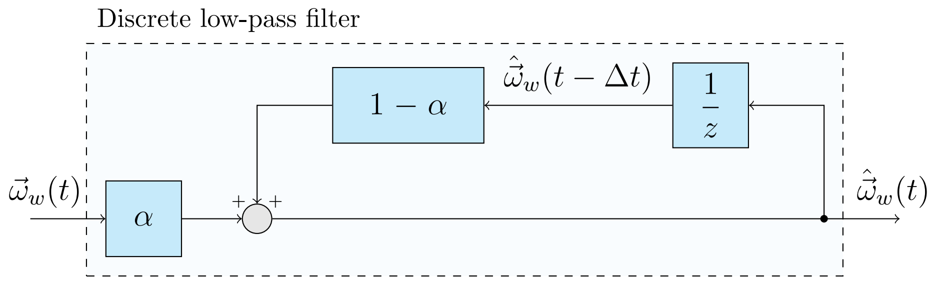


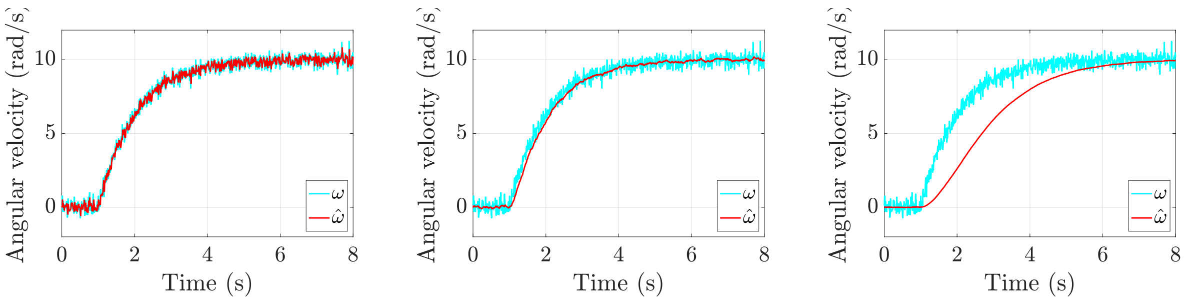
Figure 48: Discrete low-pass filter

The constant α is called the "smoothing factor", which depends on the cut-off frequency ω_c and the time interval Δt of the discretization

$$\alpha = \frac{\omega_c \Delta t}{1+\omega_c \Delta t}\,. \qquad (7.3)$$

- The higher the cut-off frequency ω_c, the closer to 1 the smoothing factor α will be and, thus, more weight will be given to the new measured value. This is advantageous as it ensures that the estimated signal converges faster, however, it also lets more noise through (Fig. 49a).
- The lower the cut-off frequency ω_c, the closer to 0 the smoothing factor α will be and, thus, more weight will be given to the old estimated value. This is advantageous as it lets less noise get through, however, it makes the estimated signal converge more slowly (Fig. 49c).

[1]The explicit ("forward") Euler method uses the approximation $\frac{d}{dt}x(t) \approx \frac{x(t+\Delta t)-x(t)}{\Delta t}$, whereas the implicit ("backwards") Euler method uses the approximation $\frac{d}{dt}x(t) \approx \frac{x(t)-x(t-\Delta t)}{\Delta t}$



(a) High ω_c (b) Optimal ω_c (c) Low ω_c

Figure 49: Cut-off frequency ω_c influence in a low-pass filter

Determining the optimal cut-off frequency ω_c (Fig. 49b), that is, one that guarantees a good trade-off between noise filter and delay, is the biggest challenge in implementing a low-pass filter.

7.1.2 State observer

Another way to estimate the reaction wheels angular velocity is through a state observer. A state observer is a copy of a system that, based on the input and output values of the real system (plant), provides us with estimates of the states of that plant.

Initially, let us consider an 1st order state observer in which the estimated angular velocity $\hat{\vec{\omega}}_w$ is assumed to be constant

$$\dot{\hat{\vec{\omega}}}_w = 0\,. \tag{7.4}$$

If the difference between the measured angular velocity $\vec{\omega}_w$ and the estimated angular velocity $\hat{\vec{\omega}}_w$ is feedback

$$\dot{\hat{\vec{\omega}}}_w = 0 + l\left(\vec{\omega}_w - \hat{\vec{\omega}}_w\right)\,, \tag{7.5}$$

it is ensured that the orientation quaternion error q_e converges exponentially to zero (for any positive value of l, known as the state observer gain).

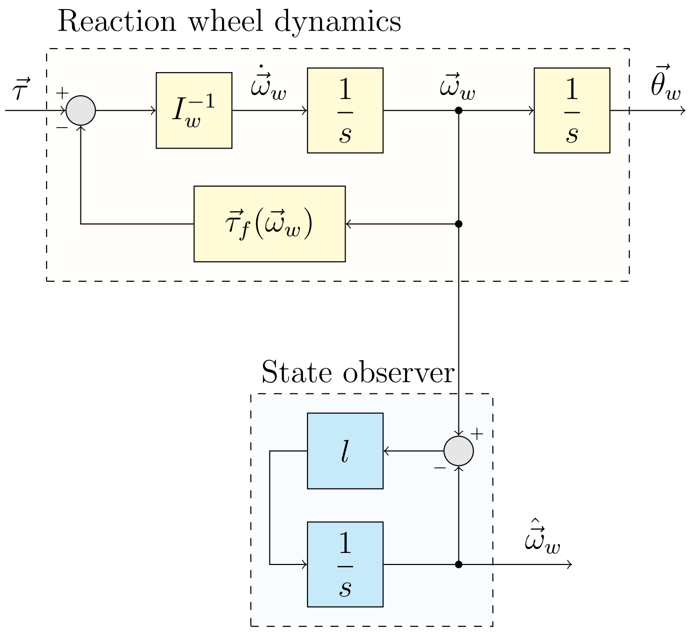


Figure 50: Reaction wheels state observer

If this is not clear from Fig. 50, perhaps it becomes when the state observer is isolated and rearranged, as shown in Fig. 51.

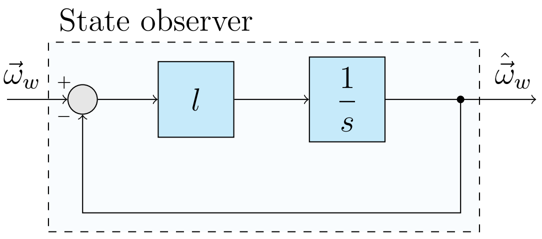


Figure 51: Reaction wheels state observer rearranged

It is evident that a state observer is analogous to a state regulator, where the reference becomes the measured state and the output becomes the estimated state. Its transfer function is

$$\frac{\hat{\vec{\Omega}}_w(s)}{\vec{\Omega}_w(s)} = \frac{l}{s+l} \tag{7.6}$$

Note that this is the same transfer function as a low-pass filter, where the gain of the estimator l is equal to the cut-off frequency ω_c. That is, a 1st order state observer is exactly the same thing as a low-pass filter.

Let us now consider the same 1st order state observer but, instead of considering the angular velocity as being constant, let us consider the influence of the input torque and friction on it

$$\dot{\hat{\vec{\omega}}}_w = I_w^{-1}\left(-\hat{\vec{\tau}}_f(\vec{\omega}_w) + \vec{\tau}\right) \,, \tag{7.7}$$

and feedback again the difference between the measured angular velocity $\vec{\omega}_w$ and the estimated angular velocity $\hat{\vec{\omega}}_w$

$$\dot{\hat{\vec{\omega}}}_w = I_w^{-1}\left(-\hat{\vec{\tau}}_f(\vec{\omega}_w) + \vec{\tau}\right) + l\left(\vec{\omega}_w - \hat{\vec{\omega}}_w\right) \,. \tag{7.8}$$

Note that now the state observer becomes a perfect copy of the plant, as shown in Fig. 52.

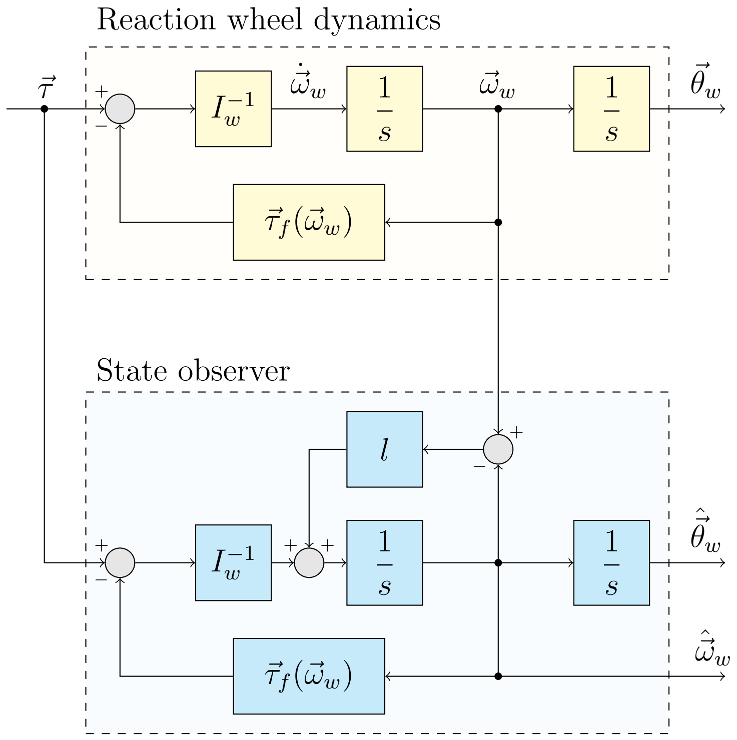


Figure 52: Reaction wheels state observer considering internal dynamics

In orther to estimate also the angular displacement $\hat{\vec{\theta}}_w$, one just need to add an integrator. Note, however, that because the angular displacement is not being measured and feedback, its estimate tends to diverge with time and is therefore considered a “pseudo-estimate”.

7.2 Cubli angular velocity and orientation

The Cubli orientation kinematics can be represented by the block diagram in Fig. 53.

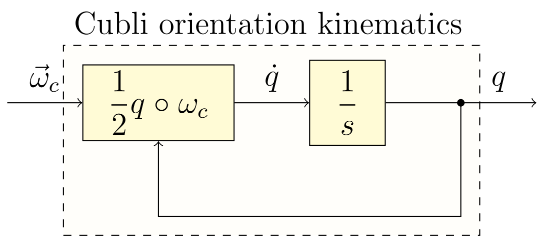


Figure 53: Cubli orientation kinematics

The angular velocity $\vec{\omega}_c$ is measured directly through the rate gyroscope, while the orientation quaternion q is measured indirectly through the accelerometer.

7.2.1 State observer

Initially, let us consider a 1st order state observer in which the estimated orientation quaternion $\hat{q}$ is assumed to be constant

$$\dot{\hat{q}} = 0\,, \tag{7.9}$$

and, thus, the angular velocity $\hat{\vec{\omega}}_c$ is assumed to be zero

$$\hat{\vec{\omega}}_c = 0\,. \tag{7.10}$$

Let $\hat{q}$ be an orientation quaternion estimation and q_e be an orientation quaternion error, such that

$$\hat{q} = \begin{bmatrix} \hat{q}_0 \\ \hat{\vec{q}}_r \end{bmatrix} \,, \qquad q_e = \begin{bmatrix} q_{e_0} \\ \vec{q}_e \end{bmatrix} \,. \tag{7.11}$$

The orientation error represents the rotation needed from orientation estimation to match the orientation measurement. In quaternion notation, consecutive rotations can be represented as multiplications between respective orientation quaternions, which means that

$$q = \hat{q} \circ q_e\,. \tag{7.12}$$

By left-multiplying both sides of (7.12) with $\bar{\hat{q}}$ and using (4.41), it is possible to isolate orientation quaternion error

$$\begin{aligned} \bar{\hat{q}} \circ q &= \cancel{\bar{\hat{q}} \circ \hat{q}} \circ q_e \\ q_e &= \bar{\hat{q}} \circ q\,. \end{aligned} \tag{7.13}$$

When orientation estimation matches the orientation measurement, no additional rotation is needed and thus the orientation quaternion error is $q_e = \begin{bmatrix} 1 & \vec{0} \end{bmatrix}^T$. Because q_e is not zero (and will never be since an orientation quaternion always have unit norm), (7.14) could not be used to guarantee asymptotically stable error dynamics

$$\dot{q}_e + l q_e \neq \begin{bmatrix} 0 \\ \vec{0} \end{bmatrix} . \tag{7.14}$$

However, the vector part of the orientation quaternion error will be zero, which means that (7.15) could be used instead

$$\dot{\vec{q}}_e + l \vec{q}_e = \vec{0} , \tag{7.15}$$

The first time derivative of q_e is obtained by differentiating (7.13)

$$\begin{aligned} \dot{q}_e &= \frac{d}{dt} \left(\bar{\hat{q}} \circ q \right) \\ \dot{q}_e &= \dot{\bar{\hat{q}}} \circ q + \bar{\hat{q}} \circ \cancelto{0}{\dot{q}} \\ \dot{q}_e &= \dot{\bar{\hat{q}}} \circ \left(\hat{q} \circ q_e \right) \\ \dot{q}_e &= \left(\dot{\bar{\hat{q}}} \circ \hat{q} \right) \circ q_e \\ \dot{q}_e &= -\frac{1}{2} \hat{\omega}_c \circ q_e , \end{aligned} \tag{7.16}$$

where its vector part is given by

$$\dot{\vec{q}}_e = -\frac{1}{2} \left(q_{e0} I_{3\times3} - \tilde{q}_e \right) \vec{\omega}_c . \tag{7.17}$$

Substituting (7.17) in (7.15) yields

$$\begin{aligned} \dot{\vec{q}}_e + l \vec{q}_e &= \vec{0} \\ -\frac{1}{2} \left(q_{e0} I_{3\times3} - \tilde{q}_e \right) \hat{\vec{\omega}}_c + l \vec{q}_e &= \vec{0} \\ \hat{\vec{\omega}}_c - 2 (q_{e0} I_{3\times3} - \tilde{q}_e)^{-1} l \vec{q}_e &= \vec{0} \\ \hat{\vec{\omega}}_c - 2l \frac{\vec{q}_e}{q_{e0}} &= \vec{0} \\ \hat{\vec{\omega}}_c &= 2l \frac{\vec{q}_e}{q_{e0}} . \end{aligned} \tag{7.18}$$

This equation ensures that the orientation quaternion error q_e converges exponentially to zero (for any positive value of l, the state observer gain).

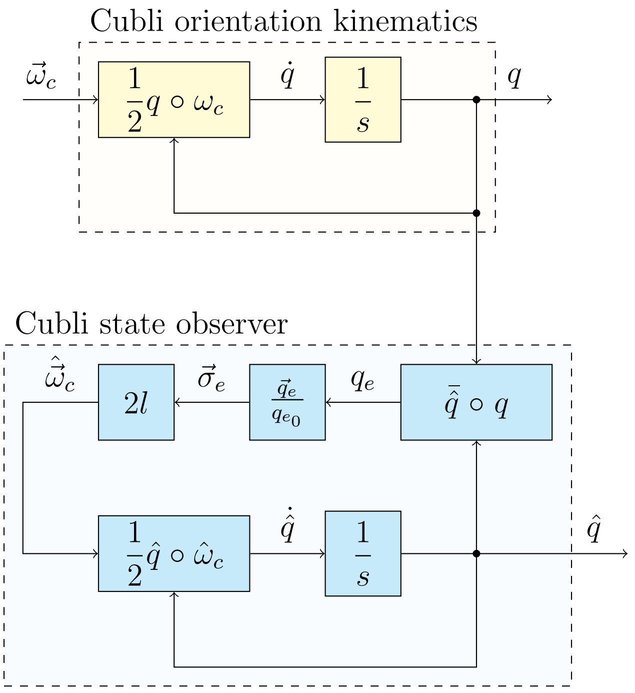


Figure 54: Cubli state observer

Fig. 54 represents this state observer scheme.

Let us now consider the same 1st order state observer but, instead of considering the angular velocity as being zero, it will be equal to its measurements

$$\hat{\vec{\omega}}_c = \vec{\omega}_c \, . \tag{7.19}$$

Although simple, this modification guarantees that both the angular velocity $\vec{\omega}_c$ and orientation quaternion q measurements are being considered into the state observer, as shown in Fig. 55.

The angular velocity $\vec{\omega}_c$ is measured directly thought the rate gyroscope, while the orientation quaternion q is measured indirectly though the accelerometer. Because the rate gyroscope has low-frequency noise, due to a constant bias being integrated over time, and the accelerometer has high-frequency noise, due to centripetal and tangential accelerations, those sensors are complementary to each other. That is why the derived state estimator is also called a complementary filter.

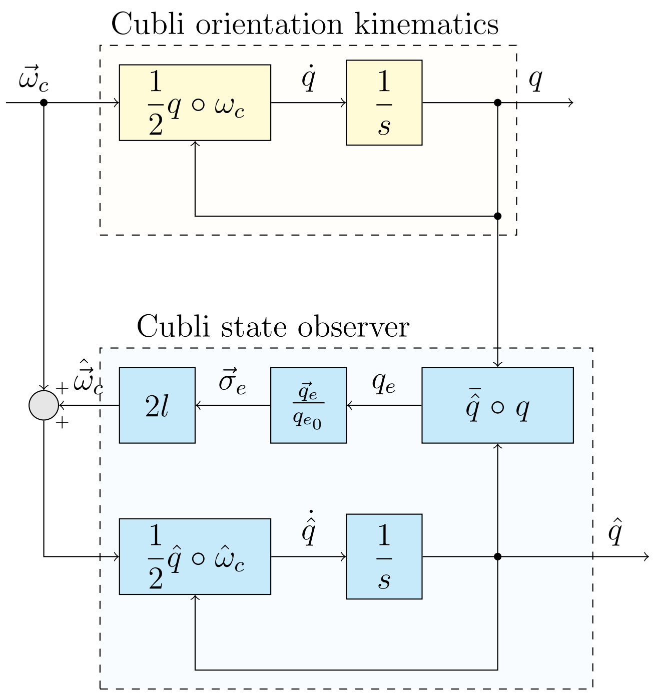


Figure 55: Cubli state observer considering angular velocity

Let us understand better how these two sensors work and are used together.

7.2.1.1 Rate gyroscope

Rate gyroscopes measures angular velocity. They are composed of a proof mass connected to an enclosure with springs and dampers, as exemplified in Fig. 56. In the x'-axis, a vibration is forced with constant amplitude and fequency $f = f_0 \sin(\omega_0 t)$. When the enclosure have an angular velocity $\dot{\theta}$, due to the Coriollis acceleration, a vibration will be induced in the proof mass in the y'-axis. By measuring the amplitude of this induced vibration, it is possible to infer the angular velocity $\dot{\theta}$.

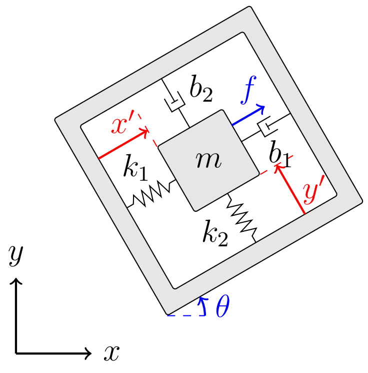


Figure 56: Rate gyroscope

In order to measure the angular velocity around all three axes, as shown in Fig. 57, three rate gyroscopes are positioned orthogonally to each other.

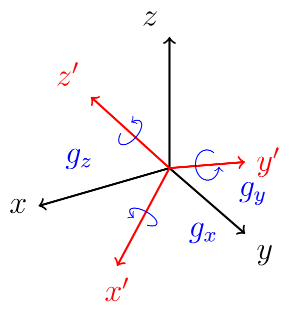


Figure 57: Three-axis rate gyroscope

where

$$\vec{g} = \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} . \tag{7.20}$$

Three-axis rate gyroscopes possess fixed errors, namely, zero offset (bias), scale factor and non-orthogonality between axes. A calibration model can be expressed to convert the measured angular velocity $\vec{g}$ into a corrected angular velocity $\vec{g}_c$

$$\vec{g}_c = F_g \left(\vec{g} - \vec{b}_g \right) , \tag{7.21}$$

where $\vec{b}_g$ is a vector that consists of the zero offset (bias) in each axis and F_g is a matrix that consists of the scale factors (diagonal terms) and non-orthogonal error coefficients

$$F_g = \begin{bmatrix} f_{g_x} & f_{g_{xy}} & f_{g_{xz}} \\ f_{g_{xy}} & f_{g_y} & f_{g_{yz}} \\ f_{g_{xz}} & f_{g_{yz}} & f_{g_z} \end{bmatrix} , \qquad b_g = \begin{bmatrix} b_{g_x} \\ b_{g_y} \\ b_{g_z} \end{bmatrix} . \tag{7.22}$$

The scale factor and non-orthogonality between axes are usually neglected, meaning that $F_g = I_{3\times3}$, while the zero offset (bias) is calculated at static condition during initialization.

Assuming the corrected rate gyroscope angular velocity $\vec{g}_c$ is a perfect estimation of the angular velocity $\hat{\vec{\omega}}_c$, it can be used in the state observer, as shown in Fig. 58.

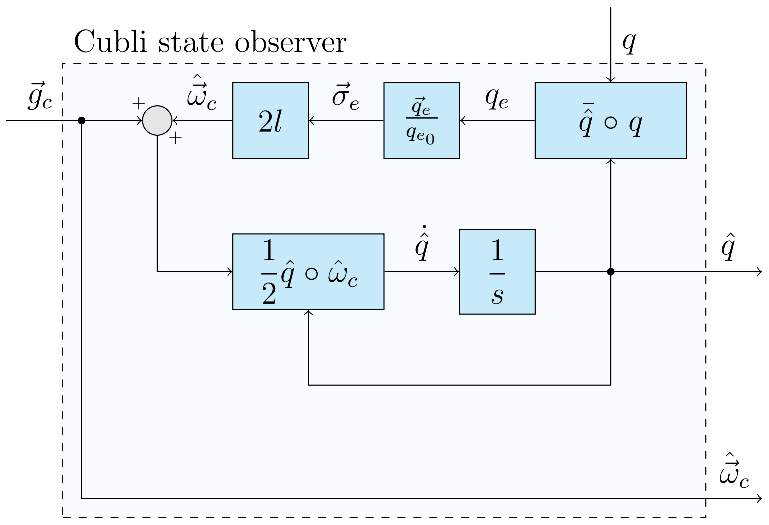


Figure 58: Cubli state observer considering angular velocity with rate gyroscope

7.2.1.2 Accelerometer

Accelerometers measures linear acceleration. They are composed of a proof mass connected to an enclosure with springs and dampers, as is exemplified in Fig. 59. When the enclosure has an acceleration $\ddot{x}$, due to Newton's first law, a displacement will be induced in the proof mass in the x'-axis. By measuring this displacement, it is possible to infer the acceleration.

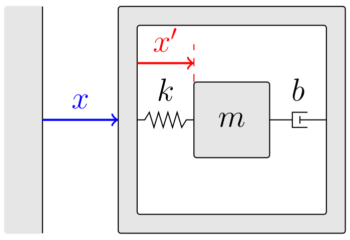


Figure 59: Accelerometer

In order to measure the linear acceleration through all three axes, as shown in Fig. 60, three accelerometers are positioned orthogonally to each other.

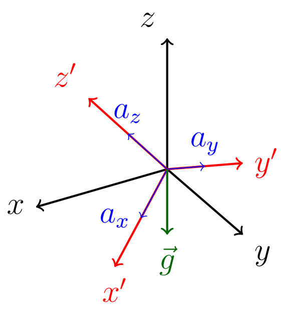


Figure 60: Three-axis accelerometer

where

$$\vec{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} . \tag{7.23}$$

Three-axis accelerometer possess fixed errors, namely, zero offset (bias), scale factor and non-orthogonality between axes. A calibration model can be expressed to convert the measured acceleration $\vec{a}$ into the corrected acceleration $\vec{a}_c$

$$\vec{a}_c = F_a \left(\vec{a} - \vec{b}_a \right) \tag{7.24}$$

where $\vec{b}_a$ is a vector that consists of the zero offset (bias) in each axis and F_a is a matrix that consists of the scale factors (diagonal terms) and non-orthogonal error coefficients

$$F_a = \begin{bmatrix} f_{a_x} & f_{a_{xy}} & f_{a_{xz}} \\ f_{a_{xy}} & f_{a_y} & f_{a_{yz}} \\ f_{a_{xz}} & f_{a_{yz}} & f_{a_z} \end{bmatrix} , \qquad b_a = \begin{bmatrix} b_{a_x} \\ b_{a_y} \\ b_{a_z} \end{bmatrix} \tag{7.25}$$

The calibration process of an accelerometer consists of identifying those nine coefficients. For this, the sensor is placed at different stationary orientations where only the acceleration of gravity $\vec{g}$ should be present. The measured values of acceleration are recorded and then an optimization algorithm finds the values of F_a and $\vec{b}_a$ that minimize the cost function given by

$$J = \left(|\vec{g}| - \left| F_a \left(\vec{a} - \vec{b}_a \right) \right| \right)^2 \tag{7.26}$$

Let a_c and g be quaternions with no real part and with the vectors $\vec{a}_c$ and $\vec{g}$ in their imaginary part

$$a_c = \begin{bmatrix} 0 \\ \vec{a}_c \end{bmatrix} = \begin{bmatrix} 0 \\ a_{c_x} \\ a_{c_y} \\ a_{c_z} \end{bmatrix} , \qquad g = \begin{bmatrix} 0 \\ \vec{g} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g \end{bmatrix} . \tag{7.27}$$

Assuming the sensor is stationary, the corrected accelerometer linear acceleration a_c is equal to the gravity vector g rotated by the orientation quaternion q

$$a_c = \bar{q} \circ g \circ q \, . \tag{7.28}$$

By left multiplying both sides of (7.28) with q and $\bar{g}$ and using (4.41), this yield

$$\begin{aligned} a_c &= \bar{q} \circ g \circ q \\ q \circ a_c &= \cancel{q \circ \bar{q}} \circ g \circ q \\ \bar{g} \circ q \circ a_c &= \cancel{\bar{g} \circ g} \circ q \\ \bar{g} \circ q \circ a_c &= q \,. \end{aligned} \tag{7.29}$$

Assuming rotation quaternion measurement q_m is equal to (7.29), it can be used in the state observer, as shown in Fig. 61.

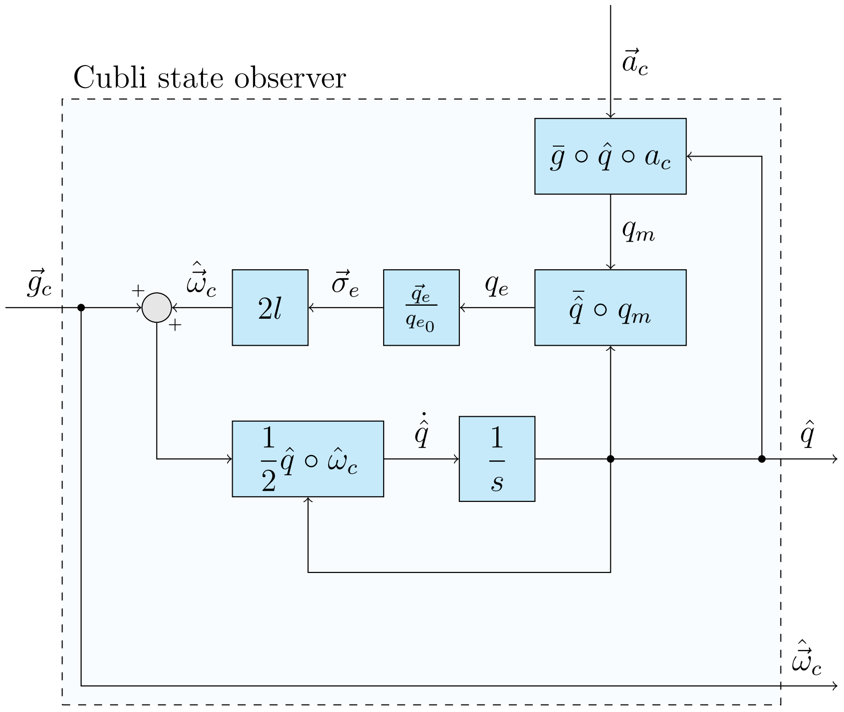


Figure 61: Cubli state observer considering angular velocity with rate gyroscope and accelerometer

As we have already seen, a 1st order state observer is analogous to a low-pass filter, where the estimator gain l corresponds to the cut-off frequency ω_c. Thus, the greater the gain l, more weight will be given to the accelerometer and less to the rate gyroscope. On the other hand, the smaller the gain l, more weight will be given to the rate gyroscope and less to the accelerometer.

Usually, a very small value of l is used [17], so that the greatest weight is given to the rate gyroscope and the role of the accelerometer is just to ensure that the orientation estimate does not diverge with time.

PART III

RESULTS

8 SIMULATIONS

To validate the derived control law, simulations were carried out in a MATLAB/Simulink model, as shown in Fig. 62, which has the same structure as the block diagrams from Fig. 21 and 44.

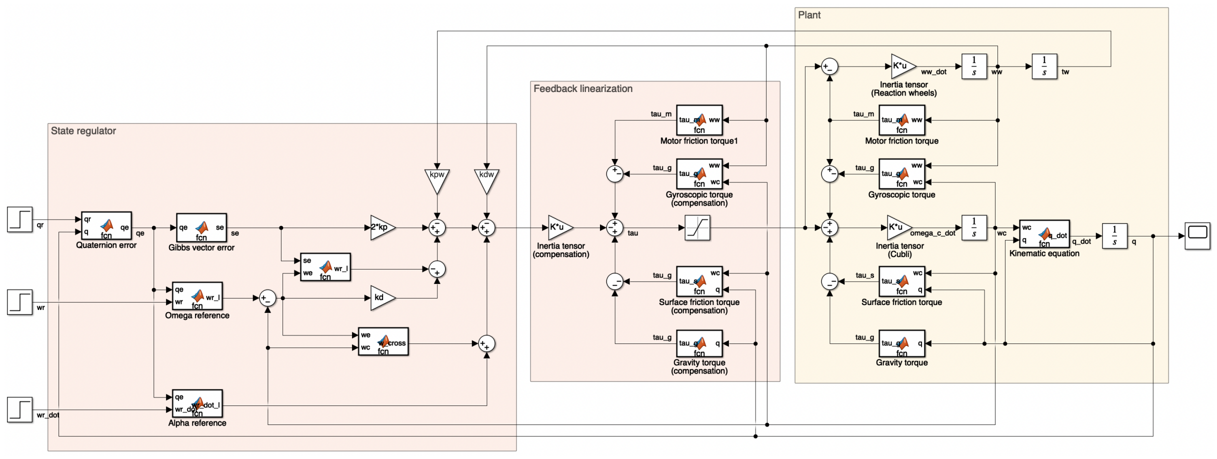


Figure 62: MATLAB/Simulink model

Note that the estimators are not being considered. In addition, as it is a simulation environment, the feedback linearization terms perfectly cancels all nonlinearities of the system. This means that the effects of an imperfect estimation or nonlinearity cancellation is not analyzed here. On the other hand, it makes it clear that any strange behavior that appears in the simulation environment are not due to these effects either.

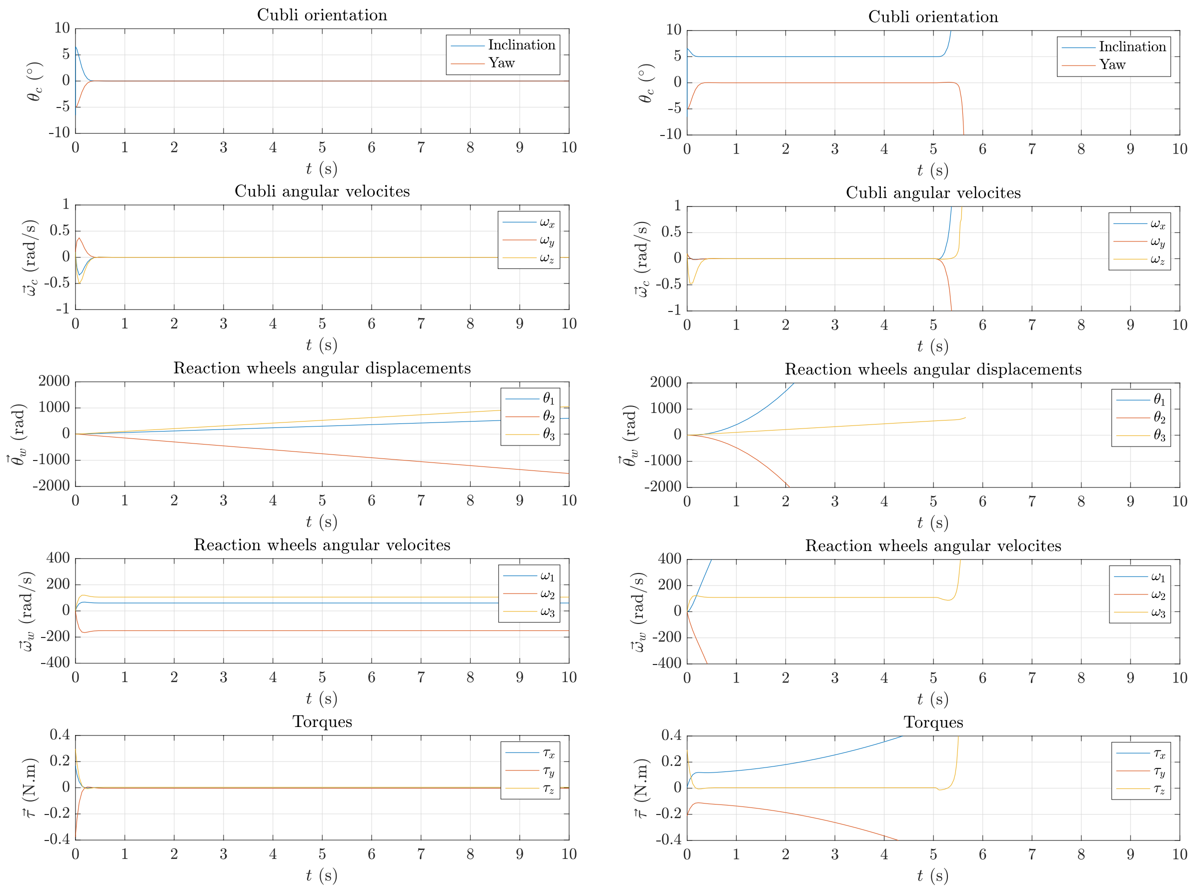
Results were obtained adopting $\zeta = \frac{\sqrt{2}}{2}$, $\omega_n = \omega_{n_0}$ and $\alpha = 0.2$ for the controller gains.

Three main simulations were carried:

- Simulation without reaction wheels feedback
- Simulation with reaction wheels angular velocities feedback
- Simulation with reaction wheels angular velocities and displacements feedback

8.1 Simulation without reaction wheels feedback

Setting the rotation quaternion reference to the Cubli unstable equilibrium position, *i.e.*, $q_r = q_u$, yields Fig. 63a.



(a) Unstable equilibrium position reference ($q_r = q_u$)

(b) Nearly unstable equilibrium position reference ($q_r = q_{ne}$)

Figure 63: Simulation 1 - Without reaction wheels feedback

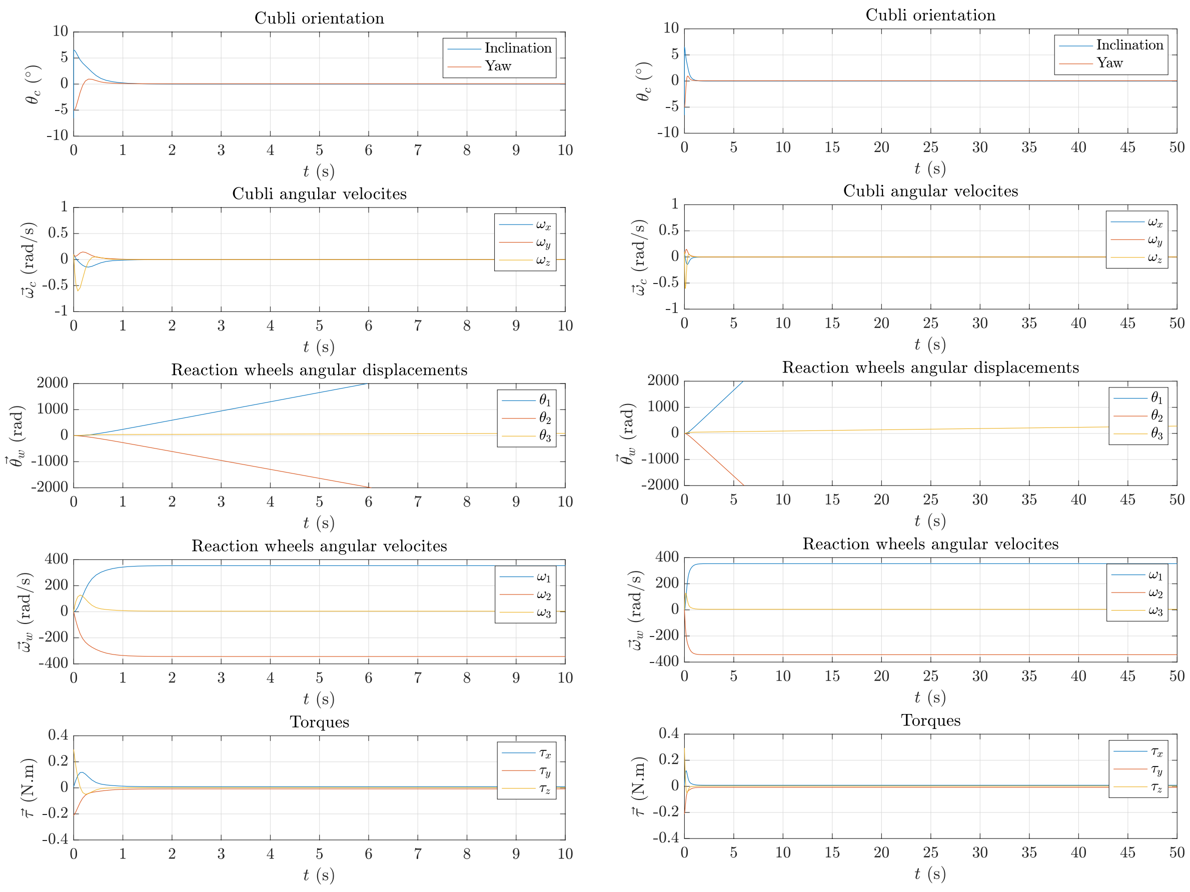
As can be seen, the Cubli is indeed controlled, even without the reaction wheels feedback. However, this only occurs because of an idealized simulation scenario.

In a more real scenario, the Cubli center of mass would not be perfectly aligned, or its orientation sensor could also be slightly misaligned. This effect can be verified by setting the rotation quaternion reference to the Cubli nearly unstable equilibrium position, *i.e.*, $q_r = q_{ne}$, which yields Fig. 63b.

As the reaction wheels are not being feedback, the controller tried to keep the Cubli in a position that is not its real unstable equilibrium position. At first, this was not a problem, as the motors still had useful torque to keep it in that position. However, as torque is product of acceleration, the reaction wheels were constantly accelerating, until reaching a point, at around 5.5s, that they saturated (reached their maximum velocity) and the system lost its controllability.

8.2 Simulation with reaction wheels angular velocities feedback

The same problem would not occur if the reaction wheels angular velocities were being feedback, as can be seen in Fig. 64a.



(a) Short time window

(b) Long time window

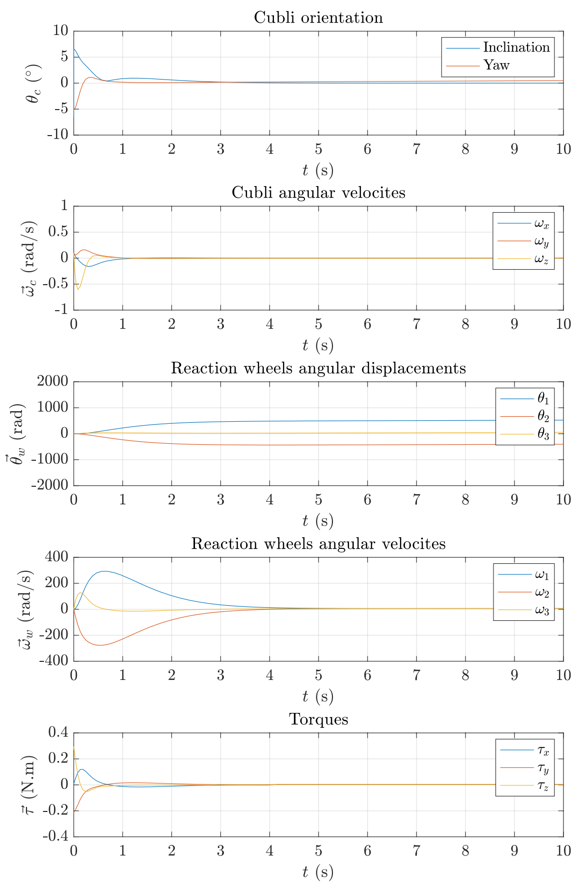
Figure 64: Simulation 2 - With reaction wheels angular velocities feedback

The the rotation quaternion reference is still the Cubli nearly unstable equilibrium position, *i.e.*, $q_r = q_{ne}$, but as the reaction wheels are now being feedback, the controller has led the Cubli to the unstable equilibrium position by itself.

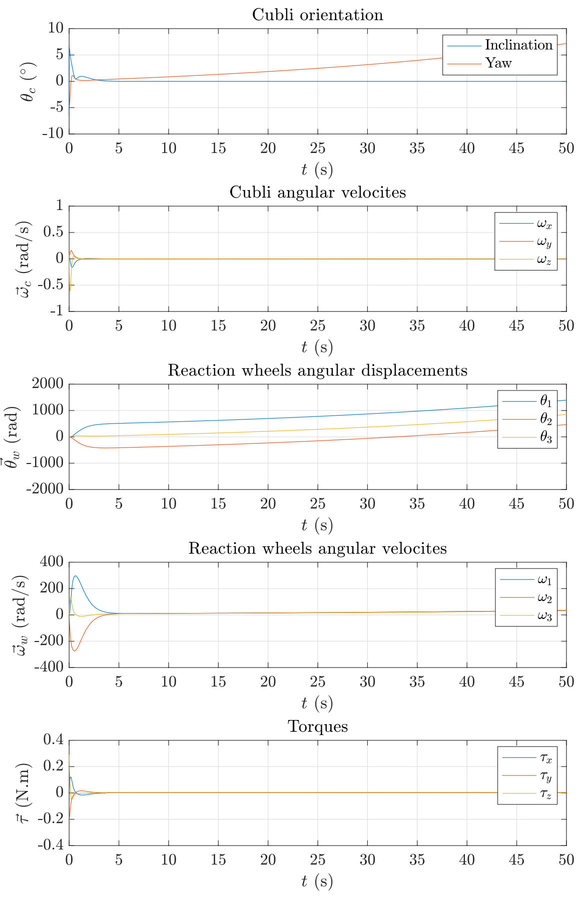
Figure 64b shows the same simulation but for a longer period. Although the Cubli is indeed controlled, as can be seen from its inclination and yaw angles at zero, the reaction wheels are constantly rotating at a constant speed.

8.3 Simulation with reaction wheels angular velocities and displacements feedback

In order for the reaction wheels angular velocities converge to zero, their angular displacements need to be feedback, as shown in Fig. 65a.



(a) Short time window



(b) Long time window

Figure 65: Simulation 3 - With reaction wheels angular velocities and displacements feedback

However, as it was expected, the Cubli controlability is now lost. As can be seen in Fig. 65b, although at first the reaction wheels angular velocities have in fact converged to zero, over time they begin to increase. The same can be observed in the yaw angle, which does not remain at zero.

9 EXPERIMENTS

To fully validate the derived control law, experiments were carried out with the Cubli prototype, whose physical construction and firmware implementation are detailed in appendix A and B.

Results were obtained adopting $\zeta = \frac{\sqrt{2}}{2}$, $\omega_n = \omega_{n_0}$ and $\alpha = 0.2$ for the controller gains and setting the rotation quaternion reference to the Cubli unstable equilibrium position, *i.e.*, $q_r = q_u$.

Four main experiments were carried out and are detailed below:

- Disturbance rejection
- Wheel velocity and angle feedback comparison
- Sinusoidal tracking reference
- Linear and nonlinear comparison

A video of the real system, including the experiments presented here and some other cases, is available at `https://youtu.be/AWEWNBDW6CM` and `https://youtu.be/Q2rAg7tytjw`.

9.1 Disturbance rejection

In the first experiment, presented in Fig. 66, the Cubli was released around 10° from its equilibrium position and it was stabilized in less than 1 second, as it can be seen from its angular velocities quickly decaying to zero. The reaction wheels angular velocities also decayed to zero, but at a much slower rate of around 5 seconds. This makes sense since $\alpha = 0.2$, which means the Cubli dynamics should be 5 times faster than the reaction wheel dynamics.

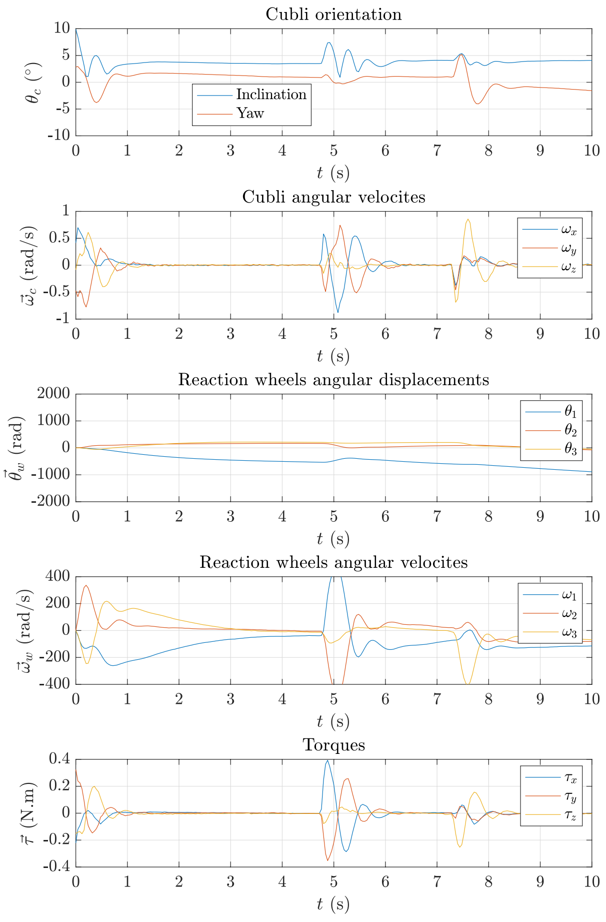


Figure 66: Experiment 1 - Disturbance rejection

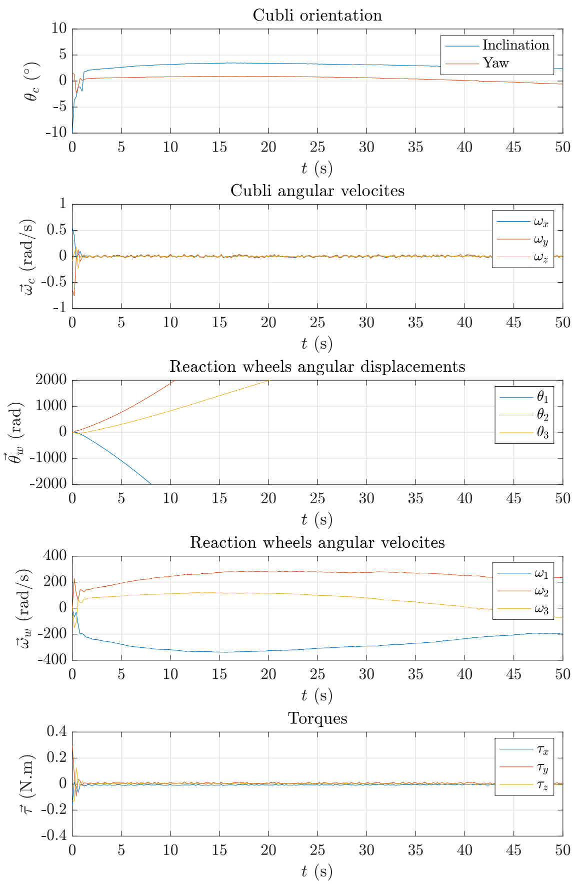
Two external disturbances were applied, one around 4.7 seconds in the inclination angle and another one around 7.3 seconds in the yaw angle. In both cases, the control system quickly rejected the disturbance without oscillating too much or saturating the actuators, which would happen for torques above 0.5N.m.

Moreover, the Cubli did not stabilize at 0° inclination angle but around 4°. This probably happened due to construction imperfections or sensor misalignment. However, because the reaction wheels angular displacements and velocities are also being feed-backed, the controller was able to find the real equilibrium position.

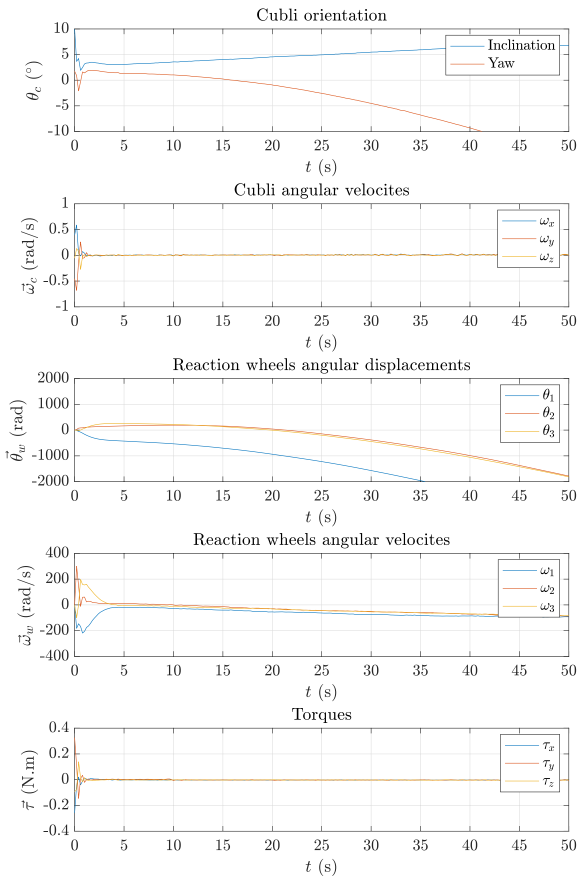
The experimental data is practically noise free due to the implementation and tuning of the state estimators previously described.

9.2 Wheel velocity and angle feedback comparison

The second experiments shows the difference between feedback only the reaction wheels angular velocities, as shown in Fig. 67a, or also their angular displacements, as shown in Fig. 67b.



(a) Wheel velocity feedback



(b) Wheel velocity and angle feedback

Figure 67: Experiment 1 - Wheel velocity and angle feedback comparison

Both experiments were performed for a much longer period of nearly 1 minute, and in both cases the Cubli remained stable for the whole time.

In the first case, the reaction wheels did not stop turning but the yaw angle remained stable. In the second case, however, the reaction wheels converged to zero initially but in the long run both they and the yaw angle diverged.

Because the reaction wheels velocities are increasing in modulus, even if slowly, this means that, at some point, they will saturate and the Cubli will lose its inclination stability. As mentioned before, this is due to the loss of controllability when the reaction wheels angular displacements are feedback, and one way to mitigate this problem is through a reference strategy.

9.3 Sinusoidal tracking reference

Third experiment is presented in Fig. 68. By making the reference follow a sinusoidal motion in the yaw axis, the yaw angle and reaction wheels angular displacement still increase in modulus, but now the reaction wheels angular velocities stay bounded. This does not mean that the yaw angle will be controllable, but at least it will remain stable longer, not leading to the inclination instability mentioned earlier.

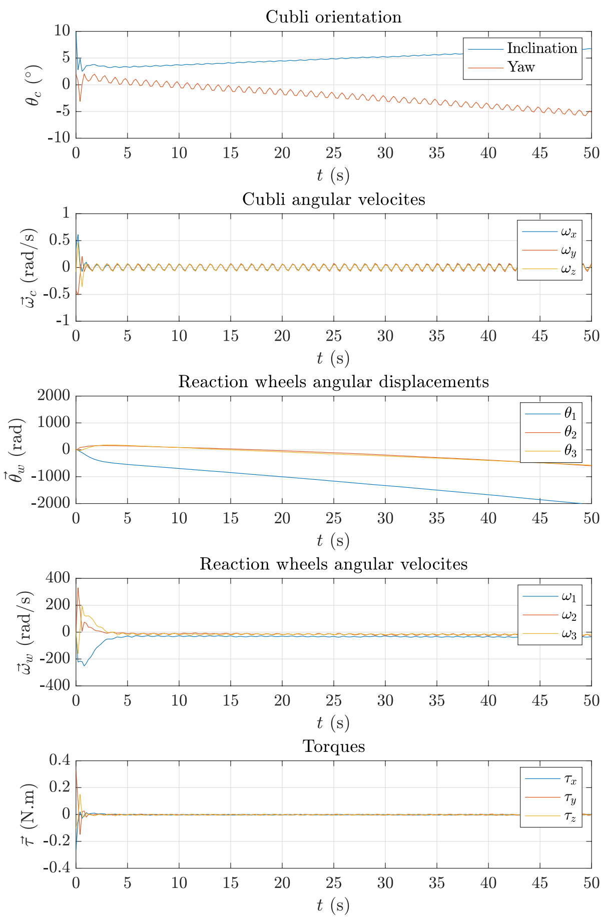
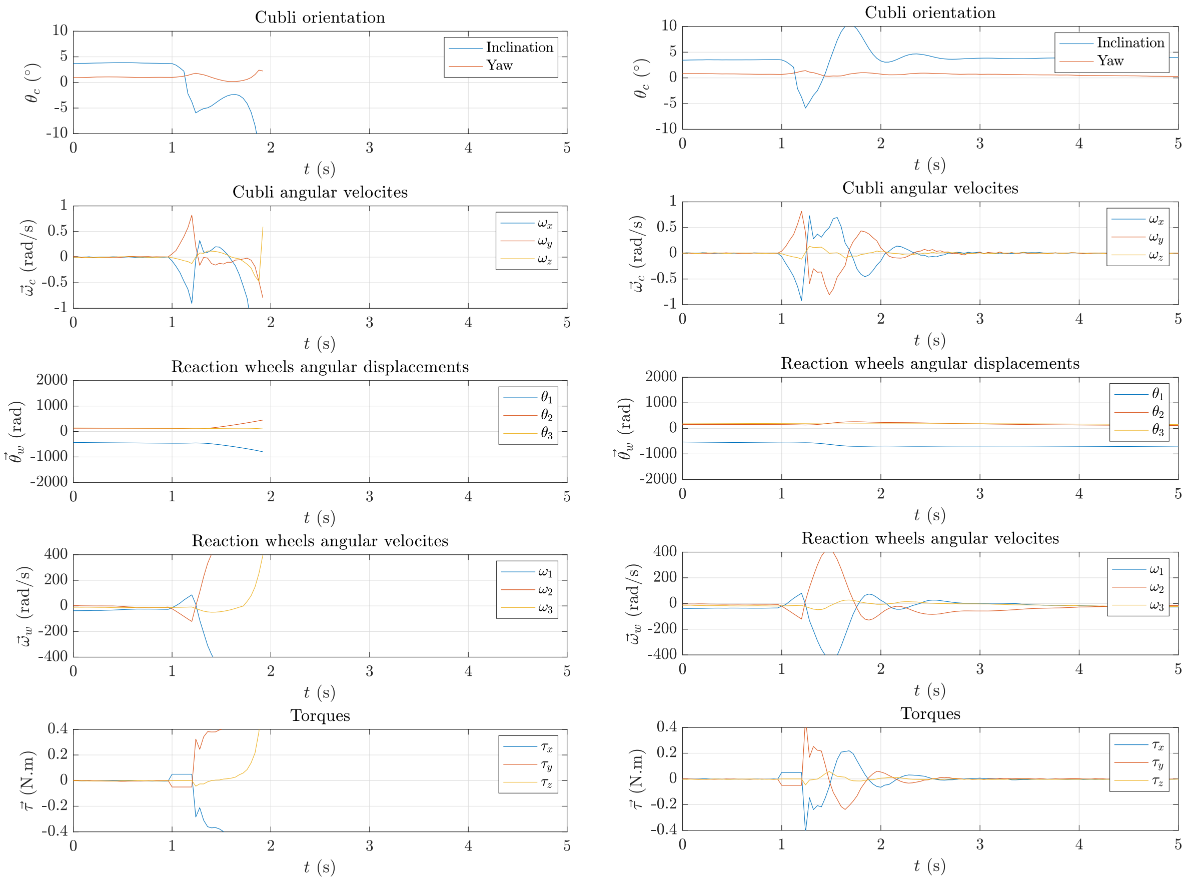


Figure 68: Experiment 3 - Sinusoidal tracking reference

Obviously another way to solve this problem would be to simply not feedback the reaction wheels angular displacement. But notice how, in this case, it was possible to keep the Cubli stable and still make the reaction wheels angular velocities converge to zero. In practice, they are oscillating above and below zero.

9.4 Linear and nonlinear comparison

Another experiment is a comparison between the derived nonlinear controller (6.19), utilizing quaternions, with a linear controller (6.22), utilizing Euler angles . To generate the same disturbance in both cases, a software-generated force disturbance was applied, *i.e.*, a constant torque of 0.2N.m for 0.2 seconds on the x and y motors.



(a) Linear controller (b) Nonlinear controller

Figure 69: Experiment 4 - Linear and nonlinear comparison

Figure 69a shows the result for the linear controller, while Fig. 69b shows the result for the nonlinear controller. As it can be seen, the forced input applied at 1 second caused an almost 10° disturbance in the inclination angle. While the nonlinear controller managed to recover, the linear controller did not. Note that this only occurs for large disturbances, where the Cubli leaves its equilibrium position. For small disturbances, not shown in the figure, the performance of both controllers would be virtually identical.

10 CONCLUSIONS

By utilizing quaternions instead of Euler angles, modeling, control and estimation design could be performed utilizing vector notation. Although somewhat complex at a first view, in the end, the control law was quite compact and obtained completely by hand, without the need for any mathematical symbolic software. Moreover, its implementation is computationally inexpensive, which makes it effective even with low-performance microcontrollers. Computer simulations showed that the model is consistent, whereas Poinsot trajectories presented a geometrical approach that also validated the model. Experimental results showed that the designed nonlinear control law is consistent and much more robust than a linear one.

The reaction wheels feedback, despite not being necessary in a simulation environment, is crucial for the real system, where imperfections and misalignments, no matter how small, are always present. However, unlike the two-dimensional reaction wheel pendulum, the three-dimensional system can not have the angle of the wheels feedback, only the speed. This is due to controllability problem inherent to Cubli's geometry, which, despise not being solved, was mitigated with a sinusoidal reference strategy.

Because the Cubli has only one equilibrium position, it must always work close to that position. This makes a nonlinear controller not so advantageous over a linear one, although, even so, it was possible to demonstrate that the nonlinear control was more robust. However, the entire developed control strategy could be replicated with small adjustments in systems where the nonlinear controller is much better, such as aircraft, spacecraft and satellites.

The challenge of building a physical prototype was as great, if not greater, than that of developing the entire modeling, control, and estimation theory behind this project. Perhaps the biggest contribution of this work ends up being the development and construction of a control testbed where several other techniques can be tested and validated.

Although this thesis has been successfully completed, there are several future opportunities. One of them is to develop a braking mechanism for the reaction wheels, so that, when suddenly braking, the rotation kinetic energy of the wheels is converted into rotation kinetic energy of the Cubli, making it able to “jump up” by itself. This mechanism is present in the original Cubli. Another opportunity is to apply the derived nonlinear control law on an orbiting CubeSat, which is under the effect of microgravity and, therefore, can rotate freely in all directions. Although the control is even simpler in this case, the estimation becomes more complicated as different orientation sensors are needed in space.

REFERENCES

[1] Ralph E. Bach and Russell A. Paielli. Linearization of attitude-control error dynamics. *IEEE Transactions on Automatic Control*, 38(10):1521–1525, 1993.

[2] Daniel J. Block, Karl J. Åström, and Mark W. Spong. The reaction wheel pendulum. *Synthesis Lectures on Control and mechatronics*, 1(1):1–105, 2007.

[3] Robert H. Cannon Jr. Some basic response relations for reaction-wheel attitute control. *ARS Journal*, 32(1):61–74, 1962.

[4] M. L. Dertouzos and J. K. Roberge. High-capacity reaction-wheel attitude control. *IEEE Transactions on Applications and Industry*, 83(71):99–104, 1964.

[5] T. Dwyer. Exact nonlinear control of large angle rotational maneuvers. *IEEE Transactions on Automatic Control*, 29(9):769–774, 1984.

[6] R. Froelich and H. Papapoff. Reaction wheel attitude control for space vehicles. *IRE Transactions on Automatic Control*, 4(3):139–149, 1959.

[7] Mohanarajah Gajamohan, Michael Merz, Igor Thommen, and Raffaello D'Andrea. The cubli: A cube that can jump up and balance. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3722–3727. IEEE, 2012.

[8] Mohanarajah Gajamohan, Michael Muehlebach, Tobias Widmer, and Raffaello D'Andrea. The cubli: A reaction wheel based 3d inverted pendulum. In *2013 European Control Conference*, pages 268–274. IEEE, 2013.

[9] Basile Graf. Quaternions and dynamics. *arXiv preprint arXiv:0811.2889*, 2008.

[10] Jian Huang, Songhyok Ri, Lei Liu, Yongji Wang, Jiyong Kim, and Gyongchol Pak. Nonlinear disturbance observer-based dynamic surface control of mobile wheeled inverted pendulum. *IEEE Transactions on Control Systems Technology*, 23(6):2400–2407, 2015.

[11] Shiuh-Jer Huang and Chien-Lo Huang. Control of an inverted pendulum using grey prediction model. *IEEE Transactions on Industry Applications*, 36(2):452–458, 2000.

[12] S. M. Joshi, A. G. Kelkar, and J. T.-Y. Wen. Robust attitude stabilization of spacecraft using nonlinear quaternion feedback. *IEEE Transactions on Automatic Control*, 40(10):1800–1803, 1995.

[13] Seul Jung and Sung Su Kim. Control experiment of a wheel-driven mobile inverted pendulum using neural network. *IEEE Transactions on Control Systems Technology*, 16(2):297–303, 2008.

[14] Raymond Kristiansen, Per Johan Nicklasson, and Jan Tommy Gravdahl. Satellite attitude control by quaternion-based backstepping. *IEEE Transactions on Control Systems Technology*, 17(1):907–912, 2009.

[15] Lev Davidovi Landau and Evgenii Mikhailovich Lifshitz. *Mechanics and electrodynamics*, volume 1. Elsevier, 2013.

[16] Sung-Hee Lee and Ambarish Goswami. Reaction mass pendulum (rmp): An explicit model for centroidal angular momentum of humanoid robots. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4667–4672. IEEE, 2007.

[17] Sebastian O. H. Madgwick, Andrew J. L. Harrison, and Ravi Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. In *2011 IEEE international conference on rehabilitation robotics*, pages 1–7. IEEE, 2011.

[18] Christopher G. Mayhew, Ricardo G. Sanfelice, and Andrew R. Teel. Quaternion-based hybrid control for robust global attitude tracking. *IEEE Transactions on Automatic Control*, 56(11):2555–2566, 2011.

[19] James L. Meriam, L. Glenn Kraige, and Jeff N. Bolton. *Engineering mechanics: dynamics*, volume 2. John Wiley & Sons, 2020.

[20] George Meyer. On the use of euler's theorem on rotations for the synthesis of attitude control systems. Technical Report NASA TN-D-3643, 1966.

[21] Richard E. Mortensen. A globally stable linear attitude regulator. *International Journal of Control*, 8(3):297–302, 1968.

[22] Michael Muehlebach and Raffaello D'Andrea. Nonlinear analysis and control of a reaction-wheel-based 3-d inverted pendulum. *IEEE Transactions on Control Systems Technology*, 25(1):235–246, 2016.

[23] Michael Muehlebach and Raffaello D'Andrea. Accelerometer-based tilt determination for rigid bodies with a nonaccelerated pivot point. *IEEE Transactions on Control Systems Technology*, 26(6):2106–2120, 2017.

[24] Michael Muehlebach, Mohanarajah Gajamohan, and Raffaello D'Andrea. Nonlinear analysis and control of a reaction wheel-based 3d inverted pendulum. In *52nd IEEE Conference on Decision and Control*, pages 1283–1288. IEEE, 2013.

[25] Russell A. Paielli and Ralph E. Bach. Attitude control with realization of linear error dynamics. *AIAA Journal of Guidance, Control, and Dynamics*, 16(1):182–189, 1993.

[26] James Kerr Roberge. *The Mechanical Seal.* PhD thesis, Massachusetts Institute of Technology, 1960.

[27] Alfred C. Robinson. On the use of quaternions in simulation of rigid-body motion. Technical Report WADC Technical Report 58-17, 1958.

[28] Tariq Samad and A. M. Annaswamy. The impact of control technology. *IEEE Control Systems Society*, (2nd edition), 2014.

[29] Peng Shi, Huijiao Wang, and Cheng-Chew Lim. Network-based event-triggered control for singular systems with quantizations. *IEEE Transactions on Industrial Electronics*, 63(2):1230–1238, 2015.

[30] Jean-Jacques E. Slotine and Weiping Li. *Applied Nonlinear Control*. Prentice Hall, 1991.

[31] Mark W. Spong, Peter Corke, and Rogelio Lozano. Nonlinear control of the reaction wheel pendulum. *Automatica*, 37(11):1845–1851, 2001.

[32] Andrew Stephenson. On a new type of dynamical stability. *Memoirs and procsatedings of the Manchester Literary & Philosophical Society*, 52:1–10, 1908.

[33] Abdelhamid Tayebi. Unit quaternion-based output feedback for the attitude tracking problem. *IEEE Transactions on Automatic Control*, 53(6):1516–1520, 2008.

[34] Abdelhamid Tayebi and Stephen McGilvray. Attitude stabilization of a vtol quadrotor aircraft. *IEEE Transactions on Control Systems Technology*, 14(3):562–571, 2006.

[35] Sebastian Trimpe and Raffaello D'Andrea. Accelerometer-based tilt estimation of a rigid body with only rotational degrees of freedom. In *2010 IEEE International Conference on Robotics and Automation*, pages 2630–2636. IEEE, 2010.

[36] Rong-Jong Wai and Li-Jung Chang. Adaptive stabilizing and tracking control for a nonlinear inverted-pendulum system via sliding-mode technique. *IEEE Transactions on Industrial Electronics*, 53(2):674–692, 2006.

[37] Bong Wie and Peter M. Barba. Quaternion feedback for spacecraft large angle maneuvers. *AIAA Journal of Guidance, Control, and Dynamics*, 8(3):360–365, 1985.

[38] Xin Xu, Chuanqiang Lian, Lei Zuo, and Haibo He. Kernel-based approximate dynamic programming for real-time online learning control: An experimental study. *IEEE Transactions on Control Systems Technology*, 22(1):146–156, 2013.

[39] Chenguang Yang, Zhijun Li, Rongxin Cui, and Bugong Xu. Neural network-based motion control of an underactuated wheeled inverted pendulum model. *IEEE Transactions on Neural Networks and Learning Systems*, 25(11):2004–2016, 2014.

APPENDIX A – CONSTRUCTION

To validate the derived control law, a Cubli physical prototype was built (Fig. 70).

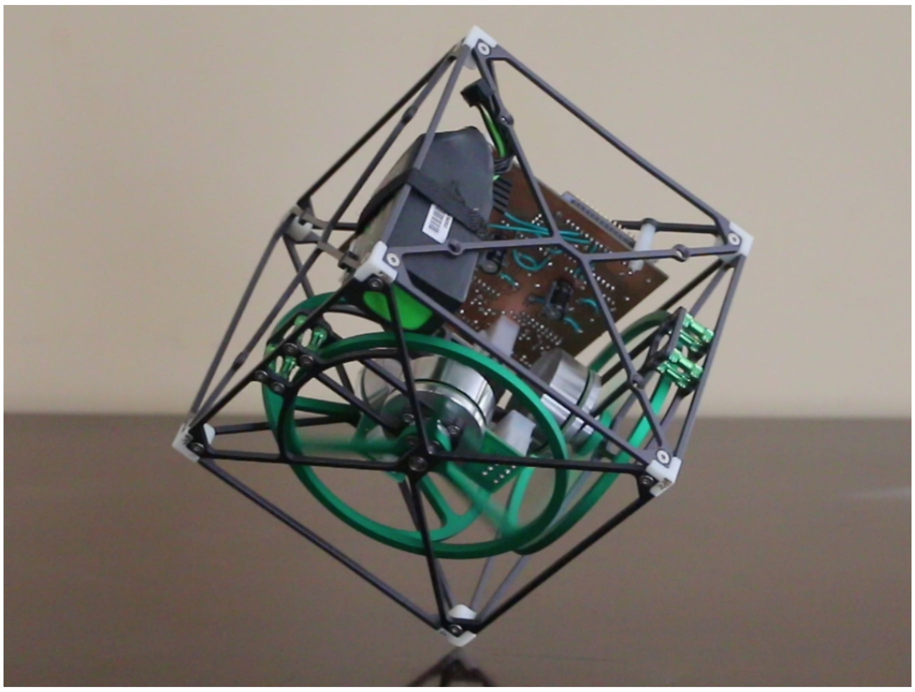


Figure 70: The Cubli

Its construction will be the focus of this appendix, justifying the choice of components and materials, and also detailing the technical drawings of the items that had to be custom-built. It is divided into electrical and electronics and mechanics.

A.1 Electrical and Electronics

Its electrical and electronics was made of off-the-shelf components and a dedicated PCB that was specially designed to interface all these components.

A.1.1 Off-the-shelf components

Five main components were needed for the Cubli:

- 1 microcontroler (MCU)
- 1 inertial measurement unit (IMU)
- 3 motors
- 3 motor controllers
- 1 battery

The selected development board was the STM NUCLEO-L432KC (Fig. 71), which has an ARM 32-bit Cortex-M4 microcontroller from STM. It was chosen because of its small footprint and high processing power (80MHz clock with 256KB of flash memory): among the STM32 Nucleo Boards, it has the greatest processing power within the 32-pin size type (which is the smallest one). Many other boards from many other vendors, such as Arduino, NXP, etc. could have been chosen instead, even with lower processing capabilities. STM was chosen because of its compatibility with Mbed Studio IDE.

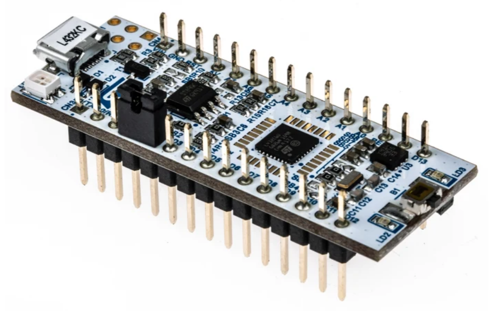


Figure 71: STM NUCLEO-L432KC[1]

The inertial measurement unit was the SparkFun 9DoF Sensor Stick (Fig. 72), which has an LSM9DS1 IMU also from STM. It was chosen because of its low cost and easy availability. At the end of the project, many development boards with integrated IMUs were being released, perhaps if the work was redone today, such an option would make more sense.

[1]Image source: STM (https://www.st.com/en/evaluation-tools/nucleo-l432kc.html)

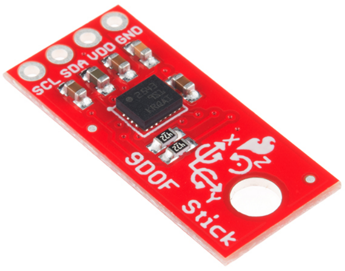


Figure 72: SparkFun 9DoF Sensor Stick[2]

The chosen motor was the Maxon EC 45 Flat 50W brushless motors (Fig. 73) from Maxon Group. It was considered for a few reasons: first, because of it small-sized and therefore being especially suitable for confined space installation; second, because it has hall sensors, which are essential in determining the displacement and angular velocity of the reaction wheels; third, because Maxon Group is world leader in the production of precision and high quality motors, thus passing a lot of reliability; and fourth, because all of the self-balanced cubes developed worldwide (including the original Cubli [7, 8, 22, 24]) utilizes this exact same motor. The chosen model was the 25601, which has a good compromise of torque and speed and supports 24V.

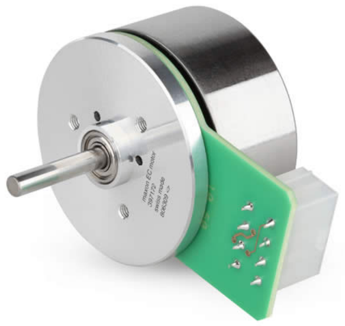


Figure 73: Maxon EC 45 Flat 50W[3]

Brushless motors are quite complex to control. As this is not the focus of this thesis, a dedicated motor controller was utilized. The chosen one was a Maxon ESCON Module 50/5 (Fig. 74), also from Maxon Group. The choice here was much simpler: because they are also made by Maxon Group, they have great compatibility and integration with the chosen motor. Moreover, it has current control (and therefore torque control) besides speed control, which makes it much easier for controlling the Cubli.

[2]Image source: SparkFun (https://www.sparkfun.com/products/13944)
[3]Image source: Maxon Group (https://www.maxongroup.com/maxon/view/product/251601)

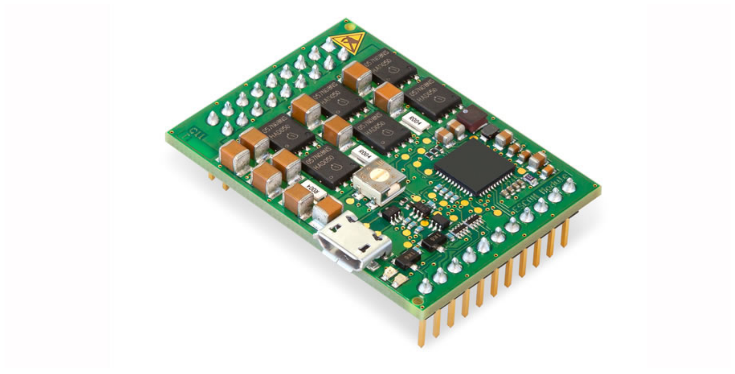


Figure 74: Maxon ESCON Module 50/5[4]

The selected battery was the Turnigy Graphene Panther 1000mAh 6S LiPo (Fig. 73) from HobbyKing. It was chosen for a few reasons: first, because it is compact and fits inside the Cubli; second, because it has 6 cells, which means that it operates from 22.2V to 25.6V (within the Motor limitation); and third, for withstanding peak currents of up to 75A (which means 25A per motor, way above the maximum supported by the motor controller of 15A).

Figure 75: Turnigy Graphene Panther 1000mAh 6S LiPo[5]

A.1.2 Printed circuit board

The microcontroller (MCU) runs ARM Mbed OS open-source operating system, communicates with the inertial measurement unit (IMU) by I2C serial communication protocol and with the motor controllers by PWM signal, for current set-point, and analog signals,

[4]Image source: Maxon Group (https://www.maxongroup.com/maxon/view/product/control/4-Q-Servokontroller/438725)

[5]Image source: HobbyKing (https://hobbyking.com/pt_pt/turnigy-graphene-1000mah-6s-75c-lipo-pack-w-xt60 html)

for hall sensor readings. The motor controllers, on the other hand, actuate the electric motors through a 3-phase PWM, while also reading the hall sensors. All components are powered by 5V, with the exception of the electric motors which are powered by 24V. As the battery has 24V, a voltage regulator was used. To integrate all those components, a printed circuited board was designed specifically for this (Fig. 76).

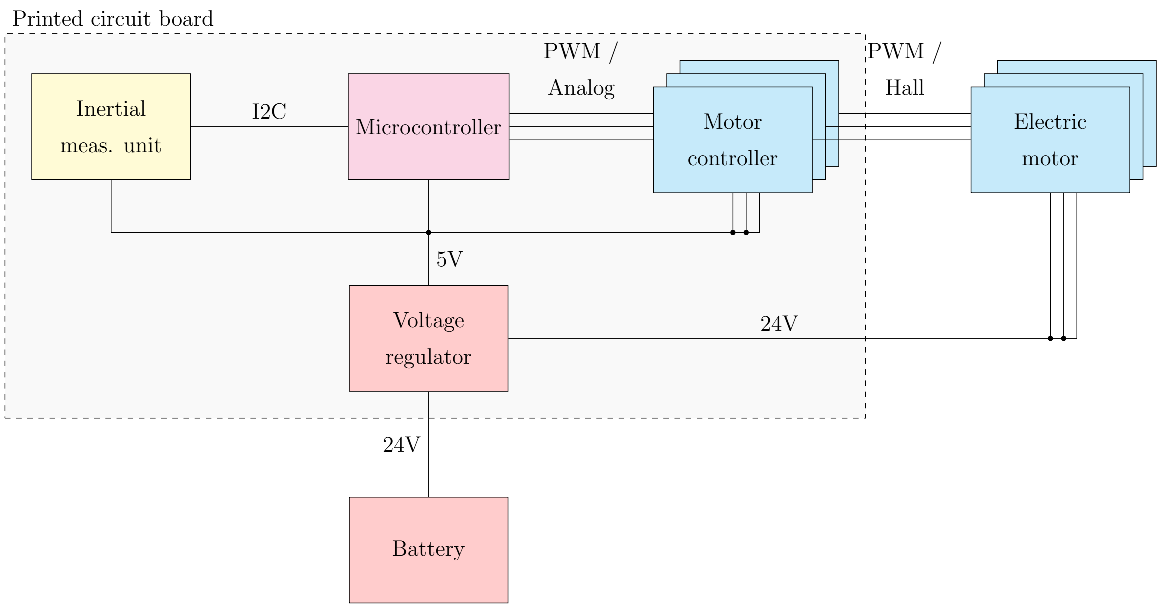


Figure 76: Electronic circuit block diagram

The PCB was designed utilizing Autodesk Eagle. Its board (Fig. 77) and schematics (Fig. 78) are available below.

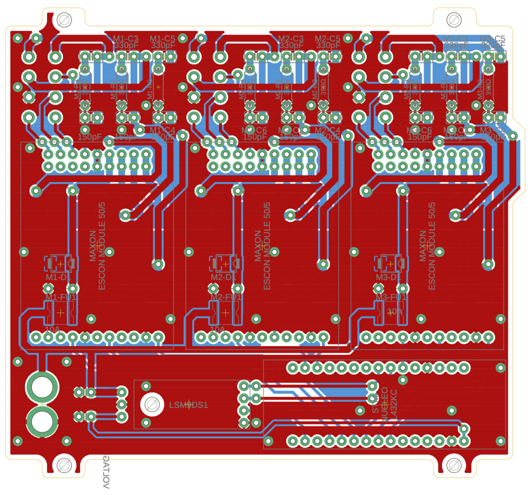


Figure 77: PCB board

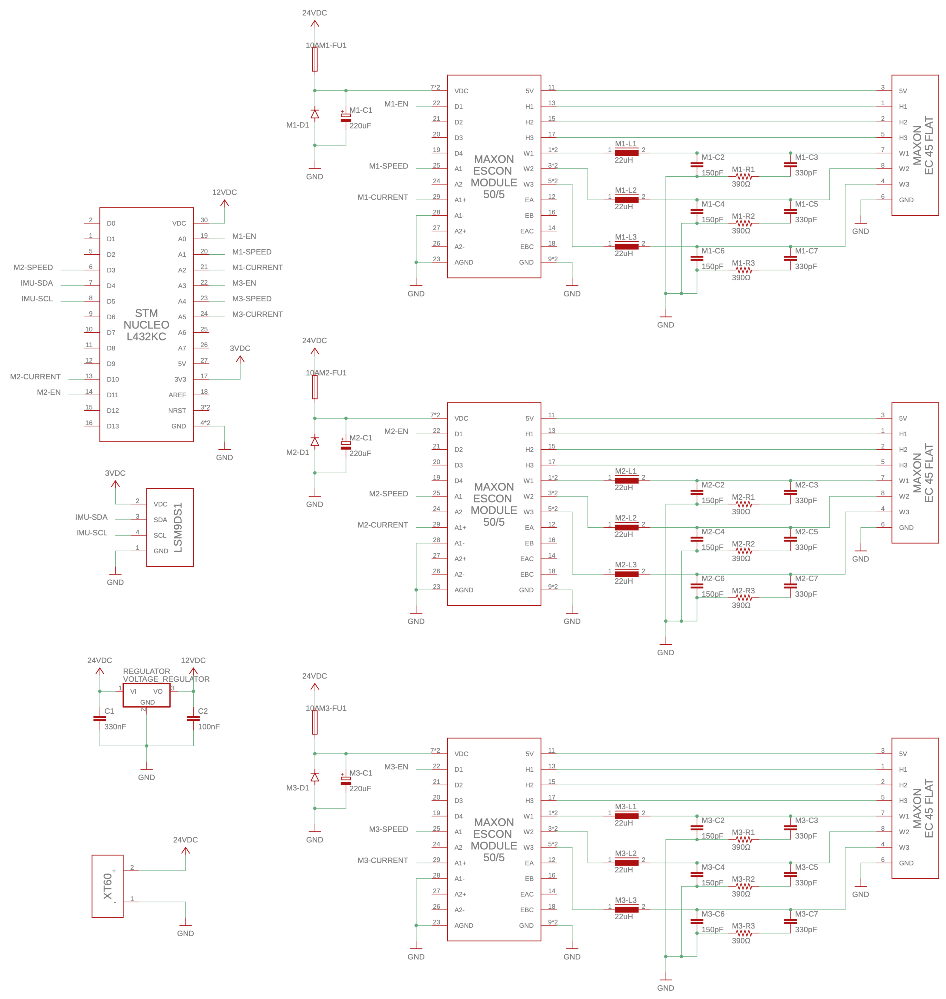
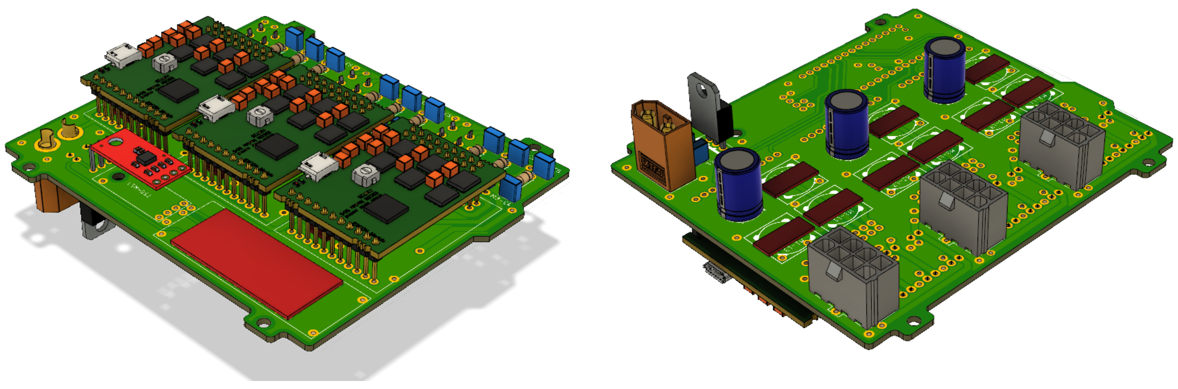


Figure 78: PCB schematics

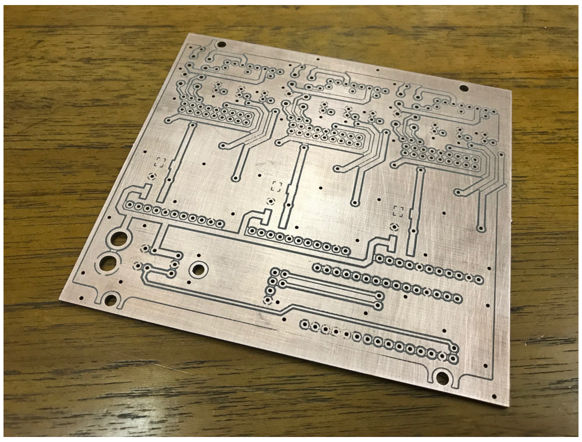
There is also a 3D render of the printed circuit board (Fig. 79), together with some real world pictures of it (Fig. 80). The PCB was manufactured in a CNC router.



(a) Top layer

(b) Bottom layer

Figure 79: PCB renders


(a) Top layer

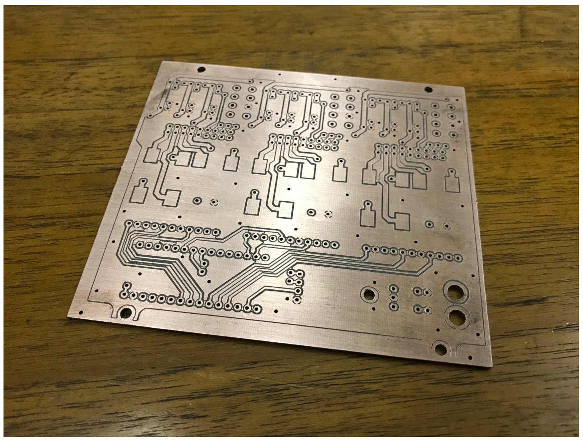

(b) Bottom layer

Figure 80: PCB pictures

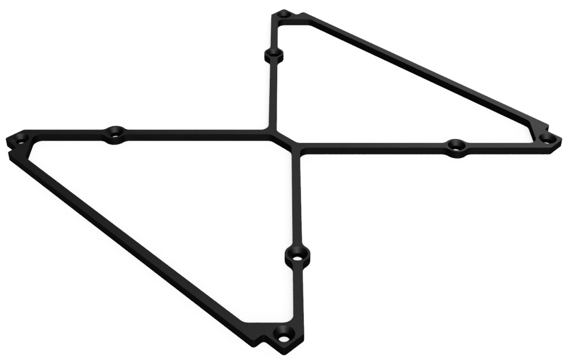
A.2 Mechanics

The mechanical parts of the Cubli were made of laser cut aluminum and 3D printed ABS.

A.2.1 Laser cut aluminum

Although a lightweight structure would result in high recovery angles, it must be strong enough to withstand sudden motor torque changes. That is why aluminum was chosen for almost all mechanical parts, so that only the most complex manufacturing parts were made in plastic.

The six aluminum faces were divided in two types: three of them were designed to support the PCB or battery (Fig. 81a), while three of them were designed to support the motors (Fig. 81b). These faces were designed in such a way that they could be manufactured by laser cutting a 3mm thick aluminum plate, which greatly simplifies the manufacturing process. They were also anodized in gray color.


(a) Regular

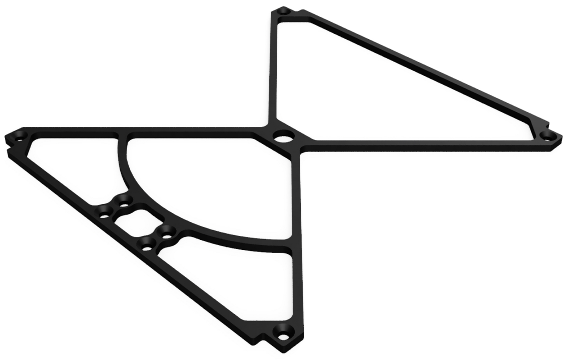

(b) Motor

Figure 81: Face

The reaction wheels (Fig. 82) were manufactured the same way, but with a 5mm thick aluminum plate and anodized in green. They were designed to be hollow and not solid in order to increase the moment of inertia of the reaction wheels without greatly increasing the mass of the Cubli as a whole.

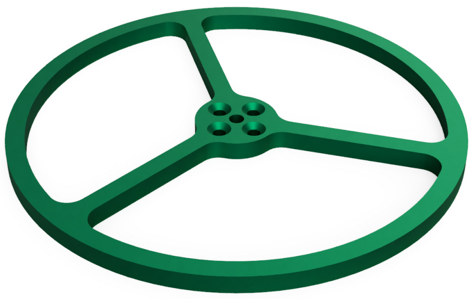
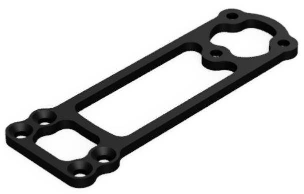


Figure 82: Reaction wheel

There is also the support of the electric motors (Fig. 83a), which are responsible for joining together the motor face, reaction wheels and motors (Fig. 83b).


(a) Support

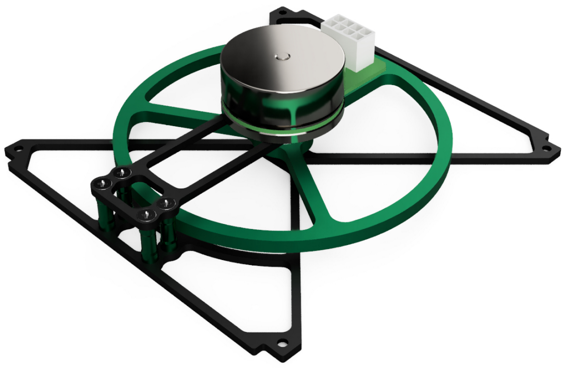

(b) Motor face assembly

Figure 83: Motor support

A.2.2 3D printed ABS

To join all six faces, eight black plastic vertices (Fig. 84) were designed and 3D printed in a way such that one face does not touch the other, even at the edges or vertices.

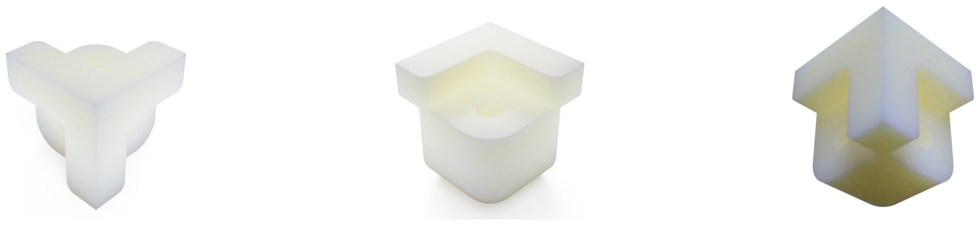

(a) View 1 (b) View 2 (c) View 3

Figure 84: Vertex

Although it would be better to use aluminum here too, due to manufacturing complexity, 3D printed ABS was preferred.

A.2.3 Screws and standoffs

To connect all those parts, ordinary M3 screws and some aluminum (for the motor support) and nylon (for the printed circuit board) standoffs were used.

APPENDIX B – FIRMWARE

To implement the derived estimation and control laws, an algorithm that provides the low-level control of the Cubli hardware is needed. This is usually called firmware.

The Cubli firmware was developed with C++ general-purpose programming language utilizing ARM Mbed OS. Since C++ is a object-oriented programming language, the source code was developed with several independent classes.

These classes were divided into three main types: modules, drivers and utilities (Fig. 85).

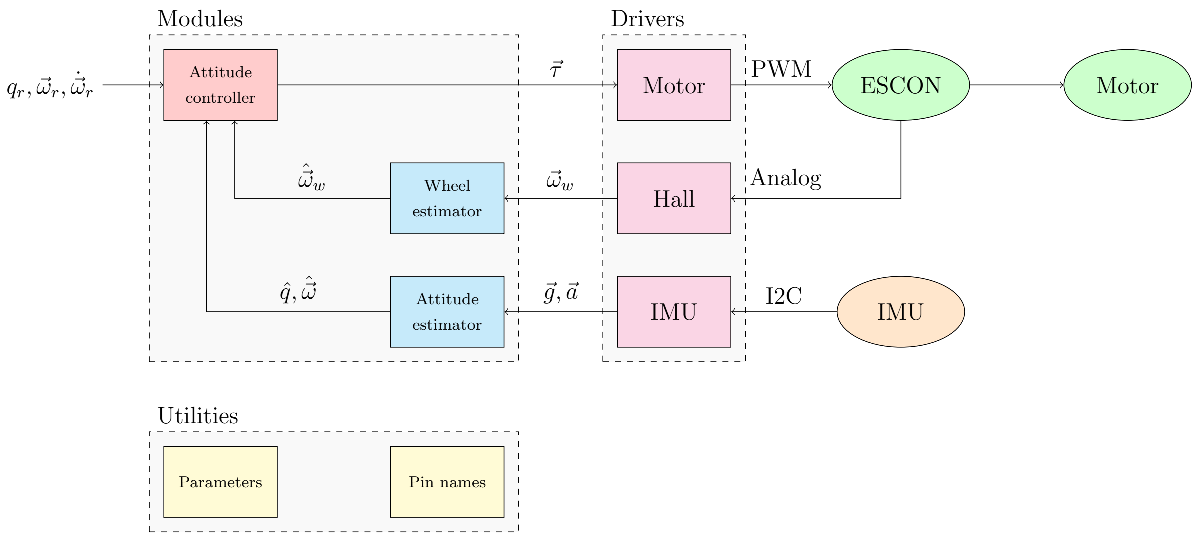


Figure 85: Firmware block diagram

Each class is composed of two files:

- A header file `.h`, that defines the constructor, variables and functions of the class
- An implementation file `.cpp`, where the source codes of the functions are implemented

All these files and folder structure are available at https://github.com/fbobrow/cubli-firmware/. In any case, they will be detailed below.

B.1 Modules

Modules contains the estimation and control classes.

B.1.1 Attitude and reaction wheel controller

The attitude and reaction wheel controller class implements the nonlinear control law derived in Sec. 6.2.

Its header file `controller_attitude_wheel.h` is:

```cpp
#ifndef controller_attitude_wheel_h
#define controller_attitude_wheel_h

#include "mbed.h"
#include "src/cubli.h"

// Attitude and wheel controller class
class AttitudeWheelController
{
  public:
    // Constructor
    AttitudeWheelController();
    // Control step
    void control(float qr0, float qr1, float qr2, float qr3, float q0, float q1, float q2, float q3, float omega_r_x,
         float omega_r_y, float omega_r_z, float omega_x, float omega_y, float omega_z, float alpha_r_x, float
         alpha_r_y, float alpha_r_z, float theta_1, float theta_2, float theta_3, float omega_1, float omega_2, float
         omega_3);
    // Quaternion error
    float qe0, qe1, qe2, qe3;
    // Torque [N.m]
    float tau_1, tau_2, tau_3;
  private:
    // State regulator step
    void state_regulator(float qr0, float qr1, float qr2, float qr3, float q0, float q1, float q2, float q3, float
         omega_r_x, float omega_r_y, float omega_r_z, float omega_x, float omega_y, float omega_z, float alpha_r_x,
         float alpha_r_y, float alpha_r_z, float theta_1, float theta_2, float theta_3, float omega_1, float omega_2,
         float omega_3);
    // Feedback linearization step
    void feedback_linearization(float q0, float q1, float q2, float q3, float omega_x, float omega_y, float omega_z,
         float omega_1, float omega_2, float omega_3);
    // Linear regulator
    void linear_regulator(float qr0, float qr1, float qr2, float qr3, float q0, float q1, float q2, float q3, float
         omega_r_x, float omega_r_y, float omega_r_z, float omega_x, float omega_y, float omega_z, float alpha_r_x,
         float alpha_r_y, float alpha_r_z, float theta_1, float theta_2, float theta_3, float omega_1, float omega_2,
         float omega_3);
    // Linearized input
    float u_1, u_2, u_3;


};

#endif
```

Whereas its implementation file `controller_attitude_wheel.cpp` is:

```cpp
#include "controller_attitude_wheel.h"

// Constructor
AttitudeWheelController::AttitudeWheelController()
{
    // Set initial quaternion error
    qe0 = 0.0;
    qe1 = 0.0;
    qe2 = 0.0;
    qe3 = 0.0;
    // Set initial torque
    tau_1 = 0.0;
    tau_2 = 0.0;
    tau_3 = 0.0;
    // Set initial linearized input
    u_1 = 0.0;
    u_2 = 0.0;
    u_3 = 0.0;
}

// Control step
void AttitudeWheelController::control(float qr0, float qr1, float qr2, float qr3, float q0, float q1, float q2, float q3, float omega_r_x, float omega_r_y, float omega_r_z, float omega_x, float omega_y, float omega_z, float alpha_r_x, float alpha_r_y, float alpha_r_z, float theta_1, float theta_2, float theta_3, float omega_1, float omega_2, float omega_3)
{
    // Non-linear controller
    state_regulator(qr0,qr1,qr2,qr3,q0,q1,q2,q3,omega_r_x,omega_r_y,omega_r_z,omega_x,omega_y,omega_z,alpha_r_x,alpha_r_y,alpha_r_z,theta_1,theta_2,theta_3,omega_1,omega_2,omega_3);
    feedback_linearization(q0,q1,q2,q3,omega_x,omega_y,omega_z,omega_1,omega_2,omega_3);
    // Linear controller
    // linear_regulator(qr0,qr1,qr2,qr3,q0,q1,q2,q3,omega_r_x,omega_r_y,omega_r_z,omega_x,omega_y,omega_z,alpha_r_x,alpha_r_y,alpha_r_z,theta_1,theta_2,theta_3,omega_1,omega_2,omega_3);
}

// State regulator step
void AttitudeWheelController::state_regulator(float qr0, float qr1, float qr2, float qr3, float q0, float q1, float q2, float q3, float omega_r_x, float omega_r_y, float omega_r_z, float omega_x, float omega_y, float omega_z, float alpha_r_x, float alpha_r_y, float alpha_r_z, float theta_1, float theta_2, float theta_3, float omega_1, float omega_2, float omega_3)
{
    // Calculate rotation quaternion error
    qe0 = q0*qr0 + q1*qr1 + q2*qr2 + q3*qr3;
    qe1 = q0*qr1 - q1*qr0 - q2*qr3 + q3*qr2;
    qe2 = q0*qr2 - q2*qr0 + q1*qr3 - q3*qr1;
    qe3 = q0*qr3 - q1*qr2 + q2*qr1 - q3*qr0;
    // Normalize rotation quaternion error
    float qe_norm = sqrt(qe0*qe0+qe1*qe1+qe2*qe2+qe3*qe3);
    qe0 /= qe_norm;
    qe1 /= qe_norm;
    qe2 /= qe_norm;
    qe3 /= qe_norm;
    // Auxiliary variables to avoid double arithmetic
    float qe0qe1 = qe0*qe1;
    float qe0qe2 = qe0*qe2;
    float qe0qe3 = qe0*qe3;
    float qe1qe1 = qe1*qe1;
    float qe1qe2 = qe1*qe2;
    float qe1qe3 = qe1*qe3;
    float qe2qe2 = qe2*qe2;
    float qe2qe3 = qe2*qe3;
    float qe3qe3 = qe3*qe3;
    // Calculate angular velocity error
    float omega_e_x = omega_r_x + 2.0*(omega_r_x*(- qe2qe2 - qe3qe3) + omega_r_y*(- qe0qe3 + qe1qe2) + omega_r_z*(qe0qe2 + qe1qe3)) - omega_x;
    float omega_e_y = omega_r_y + 2.0*(omega_r_x*(  qe0qe3 + qe1qe2) + omega_r_y*(- qe1qe1 - qe3qe3) + omega_r_z*(-qe0qe1 + qe2qe3)) - omega_y;
    float omega_e_z = omega_r_z + 2.0*(omega_r_x*(- qe0qe2 + qe1qe3) + omega_r_y*(  qe0qe1 + qe2qe3) + omega_r_z*(-qe1qe1 - qe2qe2)) - omega_z;
    // Auxiliary variable to avoid double arithmetic
    float _2_kp_omega_e_omega_e_4 = 2.0*(kp - (omega_e_x*omega_e_x + omega_e_y*omega_e_y + omega_e_z*omega_e_z)/4.0);
    // Attitude feedback
    u_1 = _2_kp_omega_e_omega_e_4*(qe1)/qe0 + kd*omega_e_x;
    u_2 = _2_kp_omega_e_omega_e_4*(qe2)/qe0 + kd*omega_e_y;
    u_3 = _2_kp_omega_e_omega_e_4*(qe3)/qe0 + kd*omega_e_z;
    // Attitude feedforward
    u_1 += omega_e_y*omega_z - omega_e_z*omega_y + alpha_r_x + 2.0*(alpha_r_x*(- qe2qe2 - qe3qe3) + alpha_r_y*(- qe0qe3 + qe1qe2) + alpha_r_z*(  qe0qe2 + qe1qe3));
    u_2 += omega_e_z*omega_x - omega_e_x*omega_z + alpha_r_y + 2.0*(alpha_r_x*(  qe0qe3 + qe1qe2) + alpha_r_y*(- qe1qe1 - qe3qe3) + alpha_r_z*(- qe0qe1 + qe2qe3));
    u_3 += omega_e_x*omega_y - omega_e_y*omega_x + alpha_r_z + 2.0*(alpha_r_x*(- qe0qe2 + qe1qe3) + alpha_r_y*(  qe0qe1 + qe2qe3) + alpha_r_z*(- qe1qe1 - qe2qe2));
    // Wheel feedback
    u_1 += - kpw*theta_1 - kdw*omega_1;
    u_2 += - kpw*theta_2 - kdw*omega_2;
    u_3 += - kpw*theta_3 - kdw*omega_3;
}

// Feedback linearization step
void AttitudeWheelController::feedback_linearization(float q0, float q1, float q2, float q3, float omega_x, float omega_y, float omega_z, float omega_1, float omega_2, float omega_3)
{
    // Calculate friction torque
    float sign_1 = (0.0<omega_1)-(omega_1<0.0);
    float sign_2 = (0.0<omega_2)-(omega_2<0.0);
    float sign_3 = (0.0<omega_3)-(omega_3<0.0);
    float tau_f_1 = sign_1*(tau_c + bw*abs(omega_1) + cd*omega_1*omega_1);
    float tau_f_2 = sign_2*(tau_c + bw*abs(omega_2) + cd*omega_2*omega_2);
    float tau_f_3 = sign_3*(tau_c + bw*abs(omega_3) + cd*omega_3*omega_3);
    // Auxiliary variable to avoid double arithmetic
    float omega_x_omega_y_omega_z = omega_x + omega_y + omega_z;
    // // Feedback linearization
    tau_1 = - I_c_xy_bar*(omega_y - omega_z)*omega_x_omega_y_omega_z - I_w_xx*(omega_3*omega_y - omega_2*omega_z) + m_c_bar_g_l*(0.5 - q0*q0 + q0*q1 - q3*q3 + q2*q3) + tau_f_1 - I_c_xx_bar*u_1 - I_c_xy_bar*(u_2 + u_3);
    tau_2 = - I_c_xy_bar*(omega_z - omega_x)*omega_x_omega_y_omega_z - I_w_xx*(omega_1*omega_z - omega_3*omega_x) + m_c_bar_g_l*(0.5 + q0*q2 - q1*q1 - q1*q3 - q2*q2) + tau_f_2 - I_c_xx_bar*u_2 - I_c_xy_bar*(u_1 + u_3);
    tau_3 = - I_c_xy_bar*(omega_x - omega_y)*omega_x_omega_y_omega_z - I_w_xx*(omega_2*omega_x - omega_1*omega_y) - m_c_bar_g_l*(      q0*q1 + q0*q2 - q1*q3 + q2*q3) + tau_f_3 - I_c_xx_bar*u_3 - I_c_xy_bar*(u_1 + u_2);
}

// Linear regulator
void AttitudeWheelController::linear_regulator(float qr0, float qr1, float qr2, float qr3, float q0, float q1, float q2, float q3, float omega_r_x, float omega_r_y, float omega_r_z, float omega_x, float omega_y, float omega_z, float alpha_r_x, float alpha_r_y, float alpha_r_z, float theta_1, float theta_2, float theta_3, float omega_1, float omega_2, float omega_3)
{
    // Calculate rotation quaternion error
    qe0 = q0*qr0 + q1*qr1 + q2*qr2 + q3*qr3;
    qe1 = q0*qr1 - q1*qr0 - q2*qr3 + q3*qr2;
    qe2 = q0*qr2 - q2*qr0 + q1*qr3 - q3*qr1;
    qe3 = q0*qr3 - q1*qr2 + q2*qr1 - q3*qr0;
    // Normalize rotation quaternion error
    float qe_norm = sqrt(qe0*qe0+qe1*qe1+qe2*qe2+qe3*qe3);
    qe0 /= qe_norm;
    qe1 /= qe_norm;
    qe2 /= qe_norm;
    qe3 /= qe_norm;
    // Calculate angle error
    float theta_e_x = 2*asin(qe1);
    float theta_e_y = 2*asin(qe2);
    float theta_e_z = 2*asin(qe3);
    // Calculate angular velocity error
    float omega_e_x = omega_r_x - omega_x;
    float omega_e_y = omega_r_y - omega_y;
    float omega_e_z = omega_r_z - omega_z;
    // Attitude feedback
    u_1 = kp*theta_e_x + kd*omega_e_x;
    u_2 = kp*theta_e_y + kd*omega_e_y;
    u_3 = kp*theta_e_z + kd*omega_e_z;
    // Attitude feedforward
    u_1 += alpha_r_x;
    u_2 += alpha_r_y;
    u_3 += alpha_r_z;
    // Wheel feedback
    u_1 += - kpw*theta_1 - kdw*omega_1;
    u_2 += - kpw*theta_2 - kdw*omega_2;
    u_3 += - kpw*theta_3 - kdw*omega_3;
    // Convert angular velocities in torques
    tau_1 = - I_c_xx_bar*u_1 - I_c_xy_bar*(u_2 + u_3);
    tau_2 = - I_c_xx_bar*u_2 - I_c_xy_bar*(u_1 + u_3);
    tau_3 = - I_c_xx_bar*u_3 - I_c_xy_bar*(u_1 + u_2);
}
```

B.1.2 Reaction wheel estimator

The reaction wheel estimator class implements the state observer derived in Sec. 7.1. Its header file `estimator_wheel.h` is:

```cpp
#ifndef estimator_wheel_h
#define estimator_wheel_h

#include "mbed.h"
#include "src/cubli.h"

// Wheel estimator class
class WheelEstimator
{
  public:
    // Constructor
    WheelEstimator(PinName PIN_SPEED);
    // Initializer
    void init();
    // Estimate step
    void estimate(float tau = 0.0);
    // Angular displacement [rad] and angular velocity [rad/s] estimations
    float theta_w, omega_w;
  private:
    // Motor hall sensor object
    Hall hall;
    // Angular velocity bias calibration
    void calibrate();
    // Predict step
    void predict(float tau);
    // Correct step
    void correct(float omega_w_m);
    // Angular velocity (rad/s) bias
    float b_omega_w;
};

#endif
```

Whereas its implementation file `estimator_wheel.cpp` is:

```cpp
#include "estimator_wheel.h"

// Constructor
WheelEstimator::WheelEstimator(PinName PIN_SPEED) : hall(PIN_SPEED)
{
    // Set initial angular displacement and angular velocity
    theta_w = 0.0;
    omega_w = 0.0;
}

// Initializer
void WheelEstimator::init()
{
    calibrate();
}

// Angular velocity bias calibration
void WheelEstimator::calibrate()
{
    // Calculate angular velocity bias by averaging n samples durint 0,5 second
    int n = f/2;
    for(int i = 0; i<n;i++)
    {
        hall.read();
        b_omega_w += hall.omega/n;
        wait_us(dt_us);
    }
}

// Estimate step
void WheelEstimator::estimate(float tau)
{
    // Predict step
    predict(tau);

    // Get angular velocity measurement from hall sensor
    hall.read();
    float omega_w_m = hall.omega-b_omega_w;
    // Correct step
    correct(omega_w_m);
}

// Predict step
void WheelEstimator::predict(float tau)
{
    // Calculate friction torque
    float sign = (0.0<omega_w)-(omega_w<0.0);
    float tau_f = sign*(tau_c+b*abs(omega_w)+cd*omega_w*omega_w);
    // Calculate angular acceleration
    float omega_w_dot = (1.0/I_w_xx)*(-tau_f+tau);
    // Predict angular displacement and angular velocity
    theta_w += omega_w*dt+omega_w_dot*dt*dt/2.0;
    omega_w += omega_w_dot*dt;
}

// Correct step
void WheelEstimator::correct(float omega_w_m)
{
    // Correct angular velocity with measurement
    omega_w += ldw*dt*(omega_w_m-omega_w);
}
```

B.1.3 Attitude estimator

The attitude estimator class implements the state observer derived in Sec. 7.2.

Its header file `estimator_attitude.h` is:

```
#ifndef estimator_attitude_h
#define estimator_attitude_h

#include "mbed.h"
#include "src/cubli.h"

// Attitude estimator class
class AttitudeEstimator
{
  public:
    // Constructor
    AttitudeEstimator(PinName PIN_SDA, PinName PIN_SCL);
    // Initializer
    void init();
    // Estimate step
    void estimate();
    // Rotation quaternion estimations
    float q0, q1, q2, q3;
    // Angular velocity (rad/s) estimations
    float omega_x, omega_y, omega_z;
  private:
    // IMU sensor object
    LSM9DS1 imu;
    // Angular velocity bias calibration
    void calibrate();
    // Predict step
    void predict(float omega_x, float omega_y, float omega_z);
    // Correct step
    void correct(float ax, float ay, float az);
    // Angular velocity (rad/s) bias
    float b_omega_x, b_omega_y, b_omega_z;
};

#endif
```

Whereas its implementation file `estimator_attitude.cpp` is:

```
#include "estimator_attitude.h"

// Constructor
AttitudeEstimator::AttitudeEstimator(PinName PIN_SDA, PinName PIN_SCL) : imu(PIN_SDA,PIN_SCL)
{
    // Set initial rotation quaternion
    q0 = 1.0;
    q1 = 0.0;
    q2 = 0.0;
    q3 = 0.0;
    // Set initial angular velocity
    omega_x = 0.0;
    omega_y = 0.0;
    omega_z = 0.0;
    // Set initial angular velocity bias
    b_omega_x = 0.0;
    b_omega_y = 0.0;
    b_omega_z = 0.0;
}

// Initializer
void AttitudeEstimator::init()
{
    // Initialize IMU sensor object
    imu.init();
    // Angular velocity bias calibration
    calibrate();
}

// Angular velocity bias calibration
void AttitudeEstimator::calibrate()
{
    // Calculate angular velocity bias by averaging n samples during 0,5 seconds
    int n = f/2;
    for(int i = 0; i<f;i++)
    {
        imu.read();
        b_omega_x += imu.gx/f;
        b_omega_y += imu.gy/f;
        b_omega_z += imu.gz/f;
        wait_us(dt_us);
    }
}

// Estimate step
void AttitudeEstimator::estimate()
{
    // Get angular velocity from IMU gyroscope data
    imu.read_gyr();
    omega_x = imu.gx-b_omega_x;
    omega_y = imu.gy-b_omega_y;
    omega_z = imu.gz-b_omega_z;
    // Predict step
    predict(omega_x,omega_y,omega_z);

    // Get linear acceleration from IMU accelerometer data
    imu.read_acc();
    float ax = f_ax*(imu.ax-b_ax);
    float ay = f_ay*(imu.ay-b_ay);
    float az = f_az*(imu.az-b_az);
    // Normalize linear acceleration
    float a_norm = sqrt(ax*ax+ay*ay+az*az);
    ax /= a_norm;
    ay /= a_norm;
    az /= a_norm;

    // Correct step
    correct(ax,ay,az);

    // Normalize rotation quaternion
    float q_norm = sqrt(q0*q0+q1*q1+q2*q2+q3*q3);
    q0 /= q_norm;
    q1 /= q_norm;
    q2 /= q_norm;
    q3 /= q_norm;
}

// Estimate step
void AttitudeEstimator::predict(float omega_x, float omega_y, float omega_z)
{
    // Predict rotation quaternion time derivative
    float q0_dot = 0.5*( - q1*omega_x - q2*omega_y - q3*omega_z);
    float q1_dot = 0.5*(   q0*omega_x - q3*omega_y + q2*omega_z);
    float q2_dot = 0.5*(   q3*omega_x + q0*omega_y - q1*omega_z);
    float q3_dot = 0.5*( - q2*omega_x + q0*omega_z + q1*omega_y);
    // Predict rotation quaternion
    q0 += q0_dot*dt;
    q1 += q1_dot*dt;
    q2 += q2_dot*dt;
    q3 += q3_dot*dt;
}

// Correct step
void AttitudeEstimator::correct(float ax, float ay, float az)
{
    // // Calculate rotation quaternion measurement
    // float qm0 =   ax*q2 - ay*q1 - az*q0;
    // float qm1 = - ax*q3 - ay*q0 + az*q1;
    // float qm2 =   ax*q0 - ay*q3 + az*q2;
    // float qm3 = - ax*q1 - ay*q2 - az*q3;
    // // Correct rotation quaternion
    // q0 += lds*dt*(qm0-q0);
    // q1 += lds*dt*(qm1-q1);
    // q2 += lds*dt*(qm2-q2);
    // q3 += lds*dt*(qm3-q3);

    // Calculate rotation quaternion measurement
    float qm0 =   ax*q2 - ay*q1 - az*q0;
    float qm1 = - ax*q3 - ay*q0 + az*q1;
    float qm2 =   ax*q0 - ay*q3 + az*q2;
    float qm3 = - ax*q1 - ay*q2 - az*q3;
    // Calculate rotation quaternion error
    float qe0 = q0*qm0 + q1*qm1 + q2*qm2 + q3*qm3;
    float qe1 = q0*qm1 - q1*qm0 - q2*qm3 + q3*qm2;
    float qe2 = q0*qm2 + q1*qm3 - q2*qm0 - q3*qm1;
    float qe3 = q0*qm3 - q1*qm2 + q2*qm1 - q3*qm0;
    // Calculate rotation Gibbs-vector error
    float se1 = qe1/qe0;
    float se2 = qe2/qe0;
    float se3 = qe3/qe0;
    // Calculate rotation quaternion error time derivative
    float qe0_dot = - q1*se1 - q2*se2 - q3*se3;
    float qe1_dot =   q0*se1 - q3*se2 + q2*se3;
    float qe2_dot =   q3*se1 + q0*se2 - q1*se3;
    float qe3_dot = - q2*se1 + q1*se2 + q0*se3;
    // Correct rotation quaternion
    q0 += lds*dt*qe0_dot;
    q1 += lds*dt*qe1_dot;
    q2 += lds*dt*qe2_dot;
    q3 += lds*dt*qe3_dot;
}
```

B.2 Drivers

Drivers contains the sensors and actuators classes.

B.2.1 Motor

The motor class interfaces with the motor controller for current/torque set-point.

Its header file `motor.h` is:

```
#ifndef motor_h
#define motor_h

#include "mbed.h"
#include "src/cubli.h"

// Motor class
class Motor
{
  public:
    // Class constructor
    Motor(PinName PIN_ENABLE, PinName PIN_CURRENT);
    // Set current [A]
    void set_current(float ia);
    // Set torque [N.m]
    void set_torque(float tau);
  private:
    // Objects
    DigitalOut enable;
    PwmOut     current;
};

#endif
```

Whereas its implementation file `motor.cpp` is:

```
#include "motor.h"

// Class constructor
Motor::Motor(PinName PIN_ENABLE, PinName PIN_CURRENT) : enable(PIN_ENABLE), current(PIN_CURRENT)
{
    current.period_ms(1);
}

// Set current [A]
void Motor::set_current(float ia)
{
    if (ia == 0.0)
    {
        enable = false;
        current = 0.5;
    }
    else
    {
        enable = true;
        if(ia > ia_max)
        {
            current = 0.9;
        }
        else if (ia < -ia_max)
        {
            current = 0.1;
        }
        else
        {
            current = 0.5+ia*(0.8/(2.0*ia_max));
        }
    }
}

// Set torque [N.m]
void Motor::set_torque(float tau)
{
    set_current(tau/Km);
}
```

B.2.2 Hall sensor

The hall sensor class interfaces with the motor controller for angular velocity reading. Its header file `hall.h` is:

```
#ifndef hall_h
#define hall_h

#include "mbed.h"
#include "src/cubli.h"

// Hall class
class Hall
{
  public:
    // Class constructor
    Hall(PinName PIN_SPEED);
    // Read angular velocity
    void read();
    // Angular velocity [rad/s]
    float omega;
  private:
    // Objects
    AnalogIn   speed;
};

#endif
```

Whereas its implementation file `hall.cpp` is:

```
#include "hall.h"

// Class constructor
Hall::Hall(PinName PIN_SPEED) : speed(PIN_SPEED)
{
}

// Read angular velocity
void Hall::read()
{
    omega = (speed.read()-0.5)*(2.0*omega_nl);
}
```

B.2.3 Inertial measurement unit

The inertial measurement unit class interfaces with the LSM9DS1 sensor for gyroscope and accelrometer readings.

Its header file `lsm9ds1.h` is:

```
#ifndef lsm9ds1_h
#define lsm9ds1_h

#include "mbed.h"
#include "src/cubli.h"

// LSM9DS1 I2C bus address
#define LSM9DS1_ADDRESS_ACC_GYR 0x6B << 1 // (0xD6) Shift 1 bit left because mbed utilizes 8-bit addresses and not 7-bit
#define LSM9DS1_ADDRESS_MAG     0x1E << 1 // (0x3C) Shift 1 bit left because mbed utilizes 8-bit addresses and not 7-bit

// Device identity
#define WHO_AM_I     0x0F
#define WHO_AM_I_M   0x0F

// Gyroscope configuration registers addresses
#define CTRL_REG1_G  0x10
// Gyroscope output register addresses
#define OUT_X_L_G    0x18
#define OUT_X_H_G    0x19
#define OUT_Y_L_G    0x1A
#define OUT_Y_H_G    0x1B
#define OUT_Z_L_G    0x1C
#define OUT_Z_H_G    0x1D

// Accelerometer configuration registers addresses
#define CTRL_REG6_XL 0x20
// Accelerometer output register addresses
#define OUT_X_L_XL   0x28
#define OUT_X_H_XL   0x29
#define OUT_Y_L_XL   0x2A
#define OUT_Y_H_XL   0x2B
#define OUT_Z_L_XL   0x2C
#define OUT_Z_H_XL   0x2D

// Magnetometer configuration registers addresses
#define CTRL_REG1_M  0x20
#define CTRL_REG2_M  0x21
// Magnetometer output register addresses
#define OUT_X_L_M    0x28
#define OUT_X_H_M    0x29
#define OUT_Y_L_M    0x2A
#define OUT_Y_H_M    0x2B
#define OUT_Z_L_M    0x2C
#define OUT_Z_H_M    0x2D

// Gyroscope full-scale ranges
enum gyr_scale
{
    GYR_SCALE_245DPS = 0b00,
    GYR_SCALE_500DPS = 0b01,
    GYR_SCALE_2000DPS = 0b11
};

// Accelerometer full-scale ranges
enum acc_scale
{
    ACC_SCALE_2G = 0b00,
    ACC_SCALE_4G = 0b10,
    ACC_SCALE_8G = 0b11,
    ACC_SCALE_16G = 0b01
};
```

```
// Magnetometer full-scale ranges
enum mag_scale
{
    MAG_SCALE_4G = 0b00,
    MAG_SCALE_8G = 0b01,
    MAG_SCALE_12G = 0b10,
    MAG_SCALE_16G = 0b11
};

// LSM9DS1 class
class LSM9DS1
{
    public:

        // Class constructor
        LSM9DS1(PinName sda, PinName scl);

        // Initialize sensor
        bool init();
        // Read sensor data
        void read();

        // Read gyroscope data
        void read_gyr();
        // Read accelerometer data
        void read_acc();
        // Read magnetometer data
        void read_mag();

        // Gyroscope data in x, y and z axis [rad/s]
        float gx, gy, gz;
        // Accelerometer data x, y and z axis [m/s^2]
        float ax, ay, az;
        // Magnetometer data x, y and z axis [uT]
        float mx, my, mz;

    private:

        // I2C bus
        I2C i2c;

        // Setup I2C bus
        void setup_i2c();
        // Test I2C bus
        bool test_i2c();

        // Setup gyroscope configurations (full-scale range)
        void setup_gyr(gyr_scale g_scale = GYR_SCALE_2000DPS);
        // Setup accelerometer configurations (full-scale range)
        void setup_acc(acc_scale a_scale = ACC_SCALE_2G);
        // Setup magnetometer configurations (full-scale range)
        void setup_mag(mag_scale m_scale = MAG_SCALE_4G);

        // Gyroscope resolution [rad/s / bit]
        float g_res;
        // Accelerometer resolution [m/s^2 / bit]
        float a_res;
        /// Magnetometers resolution [uT / bit]
        float m_res;

};

#endif
```

Whereas its implementation file `lsm9ds1.cpp` is:

```
#include "lsm9ds1.h"

// Class constructor
LSM9DS1::LSM9DS1(PinName sda, PinName scl) : i2c(sda, scl)
{
}

// Initialize sensor
bool LSM9DS1::init()
{
    // Setup I2C bus
    setup_i2c();
    // Test I2C bus
    if (test_i2c()) {
        // Setup gyroscope, accelerometer and magnetometer
        setup_gyr();
        setup_acc();
        setup_mag();
        return true;
    } else {
        return false;
    }
}

// Read sensor data
void LSM9DS1::read()
{
    // Read accelerometer and gyroscope data
    read_acc();
    read_gyr();
    //read_mag();
}

// Setup I2C bus
void LSM9DS1::setup_i2c()
{
    // Setup I2C bus frequency to 100kHz
    i2c.frequency(400000);
}

// Test I2C bus
bool LSM9DS1::test_i2c()
{
    // Register addresses
    char reg_acc_gyr[1] = {WHO_AM_I};
    char reg_mag[1] = {WHO_AM_I_M};
    // Data that we're going to read
    char data_acc_gyr[1];
    char data_mag[1];

    // Point to register address
    i2c.write(LSM9DS1_ADDRESS_ACC_GYR, reg_acc_gyr, 1);
    // Read data from this address
    i2c.read(LSM9DS1_ADDRESS_ACC_GYR, data_acc_gyr, 1);

    // Point to register address
    i2c.write(LSM9DS1_ADDRESS_MAG, reg_mag, 1);
    // Read data from this address
    i2c.read(LSM9DS1_ADDRESS_MAG, data_mag, 1);

    // Check if device identity is 0x68 (acc/gyr) and 0x3D (mag)
    if ((data_acc_gyr[0] == 0x68) && (data_mag[0] == 0x3D)) {
        return true;
    } else {
        return false;
    }
}

// Setup gyroscope configurations (full-scale range)
void LSM9DS1::setup_gyr(gyr_scale g_scale)
{
    // Register address and data that will be writed
    char reg_data[2] = {CTRL_REG1_G, (uint8_t) ((0b011 << 5) | (g_scale << 3) | 0b000)};

    // Point to register address and write data
    i2c.write(LSM9DS1_ADDRESS_ACC_GYR, reg_data, 2);
    // Adjust resolution [rad/s / bit] accordingly to choose scale
    switch (g_scale) {
        case GYR_SCALE_245DPS:
            g_res = 8.75f;
            break;
        case GYR_SCALE_500DPS:
            g_res = 17.50f;
            break;
        case GYR_SCALE_2000DPS:
            g_res = 70.0f;
            break;
    }
    // Convert resolution to SI (mdps / bit -> rad/s / bit)
    g_res = (g_res*1.0e-3f)*pi/180.0f;
}

// Setup accelerometer configurations (full-scale range)
void LSM9DS1::setup_acc(acc_scale a_scale)
{
    // Register address and data that will be writed
    char reg_data[2] = {CTRL_REG6_XL, (uint8_t) ((0b011 << 5) | (a_scale << 3) | 0b000)};

    // Point to register address and write data
    i2c.write(LSM9DS1_ADDRESS_ACC_GYR, reg_data, 2);

    // Adjust resolution [mg / bit] accordingly to choosed scale
    switch (a_scale) {
        case ACC_SCALE_2G:
            a_res = 0.061f;
            break;
        case ACC_SCALE_4G:
            a_res = 0.122f;
            break;
        case ACC_SCALE_8G:
            a_res = 0.244f;
            break;
        case ACC_SCALE_16G:
            a_res = 0.732f;
            break;
    }
    // Convert resolution to SI (mg / bit -> m/s^2 / bit)
    a_res = (a_res*1.0e-3f)*g;
}

// Setup magnetometer configurations (full-scale range)
void LSM9DS1::setup_mag(mag_scale m_scale)
{
    // Register address and data that will be writed
    char cmd[4] = {CTRL_REG1_M, 0x10, (uint8_t) (m_scale << 5), 0 };

    // Write the data to the mag control registers
    i2c.write(LSM9DS1_ADDRESS_MAG, cmd, 4);

    // Adjust resolution [mgauss / bit] accordingly to choosed scale
    switch (m_scale) {
        case MAG_SCALE_4G:
            m_res = 0.14f;
            break;
        case MAG_SCALE_8G:
            m_res = 0.29f;
            break;
        case MAG_SCALE_12G:
            m_res = 0.43f;
            break;
        case MAG_SCALE_16G:
            m_res = 0.58f;
            break;
    }
    // Convert resolution to SI (mgauss / bit -> uT / bit)
    m_res = ((m_res*1.0e-3f)*1.0e-4f)*1e6f;
}

// Read gyroscope data
void LSM9DS1::read_gyr()
{
    // LSM9DS1 I2C bus address
    char address = LSM9DS1_ADDRESS_ACC_GYR;
    // Register address
    char reg[1] = {OUT_X_L_G};
    // Data that we're going to read
    char data[6];

    // Point to register address
    i2c.write(address, reg, 1);
    // Read data from this address (register address will auto-increment and all three axis information (two 8 bit data
        each) will be read)
    i2c.read(address, data, 6);

    // Reassemble the data (two 8 bit data into one 16 bit data)
    int16_t gx_raw = data[0] | ( data[1] << 8 );
    int16_t gy_raw = data[2] | ( data[3] << 8 );
    int16_t gz_raw = data[4] | ( data[5] << 8 );
    // Convert to SI units [rad/s]
    gx = gy_raw * g_res;
    gy = gx_raw * g_res;
    gz = gz_raw * g_res;
}

// Read accelerometer output data
void LSM9DS1::read_acc()
{
    // LSM9DS1 I2C bus address
    char address = LSM9DS1_ADDRESS_ACC_GYR;
    // Register address
    char reg[1] = {OUT_X_L_XL};
    // Data that we're going to read
    char data[6];

    // Point to register address
    i2c.write(address, reg, 1);
    // Read data from this address (register address will auto-increment and all three axis information (two 8 bit data
        each) will be read)
    i2c.read(address, data, 6);

    // Reassemble the data (two 8 bit data into one 16 bit data)
    int16_t ax_raw = data[0] | ( data[1] << 8 );
    int16_t ay_raw = data[2] | ( data[3] << 8 );
    int16_t az_raw = data[4] | ( data[5] << 8 );
    // Convert to SI units [m/s^2]
    ax = -ay_raw * a_res;
    ay = -ax_raw * a_res;
    az = -az_raw * a_res;
}

// Read magnetometer output data
void LSM9DS1::read_mag()
{
    // LSM9DS1 I2C bus address
    char address = LSM9DS1_ADDRESS_MAG;
    // Register address
    char reg[1] = {OUT_X_L_M};
    // Data that we're going to read
    char data[6];

    // Point to register address
    i2c.write(address, reg, 1);
    // Read data from this address (register address will auto-increment and all three axis information (two 8 bit data
        each) will be read)
    i2c.read(address, data, 6);

    // Reassemble the data (two 8 bit data into one 16 bit data)
    int16_t mx_raw = data[0] | ( data[1] << 8 );
    int16_t my_raw = data[2] | ( data[3] << 8 );
    int16_t mz_raw = data[4] | ( data[5] << 8 );
    // Convert to SI units [uT]
    mx = my_raw * m_res;
    my = -mx_raw * m_res;
    mz = mz_raw * m_res;
}
```

B.3 Utilities

Utilities contains two auxiliary files that are not classes.

B.3.1 Parameters

The `parameters.h` file contains all the parameters used by the control and estimation algorithm.

```
#ifndef parameters_h
#define parameters_h

#include "cmath"

// Interrupt frequencies
const float f = 1000.0;                 // Controller interrupt frequency [Hz]
const float f_log = 25.0;               // Log data interrupt frequency [Hz]
const float f_blink = 1.0;              // Led blink interrupt frequency [Hz]
const float f_print = 10.0;             // Serial print interrupt frequency [Hz]
const float dt = 1.0/f;
const float dt_log = 1.0/f_log;
const float dt_blink = 1.0/f_blink;
const float dt_print = 1.0/f_print;
const int dt_us = dt*1e6;
const int dt_log_us = dt_log*1e6;
const int dt_blink_us = dt_blink*1e6;
const int dt_print_us = dt_print*1e6;

// Acelerometer bias and scale factor
const float b_ax = -0.0664;
const float b_ay = 0.1602;
const float b_az = 0.5595;
const float f_ax = 1.0065;
const float f_ay = 1.0086;
const float f_az = 0.9925;

// Physical parameters
const float pi = 3.141516;
const float g = 9.80665;                // Accelation of gravity [m/s^2]

// Surface parameters
const float b = 0;                      // Surface viscuous friction coefficient [N.m.s/rad]

// Motor (electrical) parameters
const float Ra = 1.03;                  // Armature resistance [Omega]
const float La = 0.572e-3;              // Armature inductance [H]
const float Km = 33.5e-3;               // Torque constant [N.m/A]
const float ia_max = 15.0;              // Stall current [A]
const float omega_nl = 6710.0*(pi/30.0); // No load speed [rpm -> rad/s]

// Motor (mechanical) parameters
const float tau_c = 2.46e-3;            // Coulomb friction torque [N.m]
const float bw = 1.06e-5;               // Rotational viscuous friction coefficient [N.m.s/rad]
const float cd = 1.70e-8;               // Rotational drag coefficient [N.m.s^2/rad^2]

// Structure parameters
const float l = 0.15;                   // Structure side length [m]
const float m_s = 0.40;                 // Structure mass [kg]
const float I_s_xx = 2.0e-3;            // Structure moment of inertia around x-y-z axis at center of mass [kg.m^2]

// Reaction wheel parameters
const float m_w = 0.15;                 // Reaction wheel mass [kg]
const float I_w_xx = 1.25e-4;           // Reaction wheel moment of inertia around x axis at center of mass [kg.m^2]
const float I_w_yy = 4.0e-5;            // Reaction wheel moment of inertia around y-z axis at center of mass [kg.m^2]
```

```
// Cubli (structure + reaction wheels) parameters
const float m_c = m_s+3*m_w;             // Cubli total mass [kg]
const float I_c_xx = I_s_xx+I_w_xx+2*I_w_yy+(m_s+2.0*m_w)*l*l/2.0; // Cubli moment of inertia around x-y-z axis at
     pivot point [kg.m^2]
const float I_c_xy = -(m_s+m_w)*l*l/4.0; // Cubli product of inertia at pivot point [kg.m^2]

// Cubli (auxiliary) parameters
const float m_c_bar = m_c - m_w;
const float I_c_xx_bar = I_c_xx - I_w_xx;
const float I_c_xy_bar = I_c_xy;
const float m_c_bar_g_l = m_c_bar*g*l;
const float omega_n_0 = sqrt(m_c_bar_g_l*sqrt(3.0)/(I_c_xx_bar-I_c_xy_bar));
const float omega_n_1 = b/(I_c_xx_bar+2*I_c_xy_bar);
const float beta = (I_c_xx_bar+2*I_c_xy_bar)/I_w_xx;
const float gamma = (I_c_xx_bar-I_c_xy_bar)/I_w_xx;
const float delta = m_c_bar_g_l*sqrt(3.0)/I_w_xx;
const float epsilon = b/I_w_xx;

// Estimator gains
const float lds = 0.02;
const float ldw = 50.0;

// Controller gains (speed+angle)
const float alpha = 0.2;
const float zeta = sqrt(2.0)/2.0; // 1.0;
const float omega_n = omega_n_0;
const float kpw = pow(alpha,2)*pow(zeta,2)*pow(omega_n,4)/delta;
const float kdw = 2.0*alpha*zeta*pow(omega_n,3)*(1.0+alpha*pow(zeta,2))/delta;
const float kp = pow(omega_n,2)*(1.0+alpha*pow(zeta,2)*(4.0+alpha))+gamma*kpw;
const float kd = 2.0*zeta*omega_n*(1.0+alpha)+gamma*kdw;

// Quaternion reference (Cubli in vertex fancing up minus phi_e - corresponding to center os mass disalignment)
const float phi_e = -0.0*pi/180.0;
const float qu0 =                cos(phi_e/2.0 + acos(sqrt(3.0)/3.0)/2.0);
const float qu1 =  sqrt(2.0)/2.0*sin(phi_e/2.0 + acos(sqrt(3.0)/3.0)/2.0);
const float qu2 = -sqrt(2.0)/2.0*sin(phi_e/2.0 + acos(sqrt(3.0)/3.0)/2.0);
const float qu3 =  0.0;

// Quaternion reference (Cubli in x-edge minus phi_e - corresponding to center os mass disalignment)
// const float phi_e = -5*pi/180.0;
// const float qu0 = cos(phi_e/2.0 -     pi/8.0);
// const float qu1 = cos(phi_e/2.0 + 3.0*pi/8.0);
// const float qu2 = 0.0;
// const float qu3 = 0.0;

// Quaternion reference (Cubli in y-edge minus phi_e - corresponding to center os mass disalignment)
// const float phi_e = 3.0*pi/180.0;
// const float qu0 = cos(phi_e/2.0 -     pi/8.0);
// const float qu1 = 0.0;
// const float qu2 = -cos(phi_e/2.0 + 3.0*pi/8.0);
// const float qu3 = 0.0;

//
const float qu0_qu0 = qu0*qu0;
const float qu0_qu1 = qu0*qu1;
const float qu0_qu2 = qu0*qu2;
const float qu0_qu3 = qu0*qu3;
const float qu1_qu1 = qu1*qu1;
const float qu1_qu2 = qu1*qu2;
const float qu1_qu3 = qu1*qu3;
const float qu2_qu2 = qu2*qu2;
const float qu2_qu3 = qu2*qu3;
const float qu3_qu3 = qu3*qu3;

// Minimum and maximum error limits (for control safety)
const float phi_min = 10.0*pi/180.0;
const float phi_max = 30.0*pi/180.0;

// Minimum jerk trajectory parameters
const float pos_traj = 2.0*pi;          // Trajectory path [rad]
const float t_rest = 5.0;               // Rest time [s]
const float t_traj = 20.0;              // Trajectory time [s]
const float cra_0 = 720.0*pos_traj/pow(t_traj,5);
const float sna_0 = 360.0*pos_traj/pow(t_traj,4);
const float jer_0 =  60.0*pos_traj/pow(t_traj,3);
```

```
// Sinusoidal trajectory parameters
const float A_traj = pi/36.0/100.0;
const float T_traj = 10.0/100.0;

#endif
```

B.3.2 Pin names

The `pin_names.h` file contains a correspondence between utilized pins and chosed development board.

```
#ifndef pin_names_h
#define pin_names_h

#include "mbed.h"

// Motor controller pins
const PinName M1_ENABLE = A3;
const PinName M2_ENABLE = D11;
const PinName M3_ENABLE = A0;
const PinName M1_CURRENT = A5;
const PinName M2_CURRENT = D10;
const PinName M3_CURRENT = A2;

// Hall sensor pins
const PinName M1_SPEED = A4;
const PinName M2_SPEED = D3;
const PinName M3_SPEED = A1;

// IMU pins
const PinName IMU_SDA = D4;
const PinName IMU_SCL = D5;

#endif
```

B.4 Main program

The `cubli.h` file includes the modules, drivers and utility files, in order that, when this single file is included, all others are referenced.

```
// Include utilities
#include "src/utils/parameters.h"
#include "src/utils/pin_names.h"

// Include drivers
#include "src/drivers/hall.h"
#include "src/drivers/lsm9ds1.h"
#include "src/drivers/motor.h"

// Include modules
#include "src/modules/estimator_wheel.h"
#include "src/modules/estimator_attitude.h"
#include "src/modules/controller_attitude_wheel.h"
#include "src/modules/trajectory_attitude.h"
```

The main program file `main.cpp` includes the `cubli.h` file and interfaces all classes, as shown in Fig. 85.

```
#include "mbed.h"
#include "cubli.h"

// Objects
Motor motor_1(M1_ENABLE,M1_CURRENT), motor_2(M2_ENABLE,M2_CURRENT), motor_3(M3_ENABLE,M3_CURRENT);
WheelEstimator whe_est_1(M1_SPEED), whe_est_2(M2_SPEED), whe_est_3(M3_SPEED);
AttitudeEstimator att_est(IMU_SDA,IMU_SCL);
AttitudeWheelController cont;
AttitudeTrajectory att_tra;
Ticker tic;

// Interrupt flags and callback functions
bool flag = false;
void callback() { flag = true; }

// Quaternion error
float phi;
float phi_lim = phi_min;

// Trajectory flag
bool flag_tra = true;

// Security flags
bool flag_arm = false;
bool flag_terminate = false;

// Torques
float tau_1, tau_2, tau_3;

// Main program
int main()
{
    // Initializations
    whe_est_1.init();
    whe_est_2.init();
    whe_est_3.init();
    att_est.init();
    tic.attach_us(&callback, dt_us);
    // Endless loop
    while (true)
    {
        if (flag)
        {
            flag = false;
            whe_est_1.estimate(tau_1);
            whe_est_2.estimate(tau_2);
            whe_est_3.estimate(tau_3);
            att_est.estimate();
            cont.control(att_tra.qr0,att_tra.qr1,att_tra.qr2,att_tra.qr3,att_est.q0,att_est.q1,att_est.q2,att_est.q3,
                 att_tra.omega_r_x,att_tra.omega_r_y,att_tra.omega_r_z,att_est.omega_x,att_est.omega_y,att_est.omega_z
                 ,att_tra.alpha_r_x,att_tra.alpha_r_y,att_tra.alpha_r_z,whe_est_1.theta_w,whe_est_2.theta_w,whe_est_3.
                 theta_w,whe_est_1.omega_w,whe_est_2.omega_w,whe_est_3.omega_w);
            phi = 2.0*acos(cont.qe0);
            if ((abs(phi) <= phi_lim) && !flag_terminate)
            {
                flag_arm = true;
                phi_lim = phi_max;
                if (flag_tra)
                {
                    flag_tra = false;
                    att_tra.init();
                }
                att_tra.generate();
                tau_1 = cont.tau_1;
                tau_2 = cont.tau_2;
                tau_3 = cont.tau_3;
            }
            else
            {
                tau_1 = 0.0;
                tau_2 = 0.0;
                tau_3 = 0.0;
                motor_1.set_torque(tau_1);
                motor_2.set_torque(tau_2);
                motor_3.set_torque(tau_3);
                if(flag_arm)
                {
                    flag_arm = false;
                    flag_terminate = true;
                }
            }
            motor_1.set_torque(tau_1);
            motor_2.set_torque(tau_2);
            motor_3.set_torque(tau_3);
        }
    }
}
```