OrigaBot: Origami-based Reconfigurable Robots for Multi-modal Locomotion

Evandro Bernardes

Aix*Marseille université
Socialement engagée

PHD THESIS

Defended at Aix-Marseille University
the 12th of April 2023

Evandro BERNARDES

OrigaBot: Origami-based Reconfigurable Robots for Multi-modal Locomotion

Discipline
Movement sciences

Doctoral school
ED 463 Sciences du Mouvement Humain

Laboratory/Research Partners
Institut des Sciences du Mouvement
Biorobotics team

Thesis Committee

Julien FAVIER M2P2, Marseille, France	Jury president
Isabelle FANTONI L2SN, Nantes, France	Reviewer
Luc JAULIN ENSTA Bretagne, Brest, France	Reviewer
Frédéric BOYER IMT Atlantique, LS2N, Nantes, France	Jury member
Coen DE VISSER TU Delft, Delft, The Netherlands	Jury member
Stéphane VIOLLET ISM, Marseille, France	Thesis supervisor

INSTITUT DES SCIENCES DU MOUVEMENT ETIENNE JULES MAREY
cnrs
AGENCE NATIONALE DE LA RECHERCHE ANR
origabot

Affidavit

I, undersigned, Evandro Bernardes, hereby declare that the work presented in this manuscript is my own work, carried out under the scientific direction of Stéphane Viollet, in accordance with the principles of honesty, integrity and responsibility inherent to the research mission. The research work and the writing of this manuscript have been carried out in compliance with both the French national charter for Research Integrity and the Aix-Marseille University charter on the fight against plagiarism.

This work has not been submitted previously either in this country or in another country in the same or in a similar version to any other examination body.

Marseille, the 1st of February 2023

List of publications and conferences

List of publications produced during the thesis project:

1. E. Bernardes, S. Viollet and T. Raharijaona, "A Three-Photo-Detector Optical Sensor Accurately Localizes a Mobile Robot Indoors by Using Two Infrared Light-Emitting Diodes." IEEE Access, vol. 8, pp. 87490-87503, 2020, doi: 10.1109/ACCESS.2020.2992996.
2. E. Bernardes and S. Viollet, "Design of an Origami Bendy Straw for Robotic Multistable Structures." ASME. J. Mech. Des. March 2022, 144(3): 033301. https://doi.org/10.1115/1.4052222
3. E. Bernardes and S. Viollet, "Quaternion to Euler angles conversion: a direct, general and computationally efficient method,". PLoS ONE 17(11): e0276302. https://doi.org/10.1371/journal.pone.0276302

List of submissions awaiting approval:

1. Bernardes, E, Boyer, F., and Viollet, S. Modelling, control and simulation of a single rotor UAV with swashplateless torque modulation. Elsevier Aerospace Science and Technology. Submitted: January 2023.

Summer schools and conferences attended during the thesis:

1. Journée des Jeunes Chercheurs en Robotique (JJCR) 2021, 12th October 2021, Paris, France.
2. Journée de l'École Doctorale (JED) Sciences du Mouvement Humain 2022. poster presentation, 3rd June 2022, Nice, France.

Résumé

Ce travail de thèse est organisé autour de deux axes bien distincts : la conception et la réalisation d'une structure 3D multistable et la conception et la réalisation d'un drone monorotor. Le point de rencontre de ces deux axes est l'origami qui permet de mettre en œuvre des structures mécaniques à la cinématique complexe tout en conservant une simplicité de mise en œuvre et une grande légèreté. Les mécanismes souples ont une grande variété d'applications pour la robotique : du biomimétisme à l'assistance chirurgicale, de nombreux systèmes peuvent bénéficier de l'utilisation de mécanismes souples. Les structures en origamis sont une sous-classe des mécanismes souples qui peuvent être construits à partir d'une fine plaque de matériau. Le projet OrigaBot a été lancé dans le but d'étudier des structures en origamis et leurs applications en robotique. En particulier, pour la création de robots volants autonomes. Dans cette thèse, j'ai d'abord réalisé une étude exhaustive de différents origamis. En particulier, la "Magic Ball" a été étudiée pour sa capacité à changer de forme, d'un sphéroïde à un cylindre, et la "tour de Kresling" pour ses propriétés bistables. Le manque d'une structure origami à flexion bistable dans la littérature nous a conduits à la conception de "l'Origami Bendy Straw". Cet origami présentant des propriétés de multistabilité unique pourra trouver de nombreuses applications robotiques, notamment pour des grippers.

Dans un but de concevoir un nouveau type de robot volant autonome, nous avons décidé de concevoir un drone monorotor et donc extrêmement minimaliste. L'absence d'un second rotor fournissant un contre-couple fait tourner le robot constamment dans une direction, ce qui constitue un vrai défi en termes de contrôlabilité. Cependant, l'utilisation d'ailettes qui profitent du flux d'air du rotor pour ralentir la rotation a été étudiée. La structure origami appelée "tour de Kresling" a été choisie pour orienter ces ailettes. Nous avons étudié un système sans pas cyclique basé sur un rotor qui utilise une vitesse sinusoïdale pour contrôler la direction des forces appliquées au robot.

Nous avons fait une étude théorique approfondie du monorotor et analysé les équations du mouvement du drone avec l'équation de Poincaré, une alternative à la méthode d'Euler-Lagrange. Nous avons aussi proposé une décomposition de l'orientation du robot qui dissocie la rotation incontrôlable de la composante contrôlable d'attitude réduite. Un contrôleur non linéaire agissant sur l'attitude réduite est dérivé et démontré avec une fonction de Lyapunov. Enfin, j'ai développé une simulation du système complet. Nous avons analysé des simulations dans

lesquelles l'estimation du lacet est imprécise, un problème réel avec des gyroscopes et compas peu fiables. Nous avons aussi proposé un observateur qui utilise la composante d'orientation réduite contrôlable pour estimer l'erreur dans la composante de rotation incontrôlable. Enfin, nous avons montré l'utilité d'un tel observateur dans le but de détecter une instabilité introduite par le manque de fiabilité de l'estimation de l'angle de lacet.

Mots clés : origami, tour de Kresling, Waterbomb, Bendy Straw, swashplateless, quaternion, Équations de Mouvement, Poincaré, Euler-Lagrange, contrôle non-linéaire, monorotor, contrôle sous-actionné

Abstract

This work focuses on two distinct problems: the design of a multistable 3D structure and the design of a single-rotor drone. The intersection of these two very distinct ideas is origami, which makes it possible to implement mechanical structures with complex kinematics while being lightweight and maintaining ease of implementation. Soft mechanisms have a wide variety of applications for robotics: from biomimicry to surgical assistance, many systems can benefit from the use of soft mechanisms. Origami-based structures can be seen as a subclass of compliant mechanisms: they are soft structures that can be assembled by folding a thin sheet of material. The OrigaBot project was born with the goal of studying origami-based structures and their applications in robotics. In particular, unmanned aerial vehicles (UAVs). In this thesis, first, an extensive study of the different kinds of origami has been carried on. In particular, the "Magic Ball" was studied for its ability to change its shape from a spheroid to a cylinder, and the "Kresling Tower" for its bistable properties. The lack of a bistable bending origami structure in the literature led to the design of the "Origami Bendy Straw". This origami structure with unique multistable properties could find many robotic applications, especially for grippers.

In order to design a new type of autonomous flying robot, we decided to design a monorotor and thus extremely minimalist drone. The lack of a second rotor providing counter-torque makes the robot constantly spin in one direction, which is a real challenge in terms of controllability. However, the use of passive "fins" that take advantage of the rotor's airflow to slow down the rotation has been investigated. The origami structure called "Kresling tower" was chosen to control the orientation of these fins. We have studied a swashplateless rotor system based on the use of a sinusoidal speed to control the direction of the thrust forces applied to the robot.

We made a thorough theoretical analysis of the monorotor and analyzed the Equations of Motion of the UAV with the Euler-Poincaré equation, an alternative to the Euler-Lagrange method. We also proposed a novel decomposition of the robot's orientation that uncouples the uncontrollable spin from the reduced controllable attitude component. A non-linear controller acting on the reduced attitude is derived and demonstrated with a family of Lyapunov functions. A closed-loop simulation of the complete system is implemented. We analyzed simulations in which the yaw estimation is inaccurate, a real problem with unreliable gyroscopes and compasses. An observer that uses the controllable reduced orientation component

to estimate the error in the uncontrollable spin component was studied. We have demonstrated the usefulness of such an observer for the purpose of detecting an instability introduced by the unreliability of the yaw angle estimate.

Keywords: origami, Kresling tower, Waterbomb, Bendy Straw, swashplateless, quaternion, Equations of Motion, Poincaré, Euler-Lagrange, non-linear control, monorotor, under-actuated control

Acknowledgments

This thesis represents the culmination of several years of work, and even more years of personal preparation, and it would not have been possible without the influence, support, and guidance of many have individuals. Therefore, before we start, I would like to express my appreciation to them.

I would like to thank my thesis advisor, Stéphane, for his guidance, ideas, and opportunity to pursue this research. His support and encouragement throughout this process has been invaluable.

I would like to express my deepest gratitude to Margarete Domingues for first sparking my interest in research many years ago. If I am finishing a PhD thesis today, it is likely the result of her influence, and it is, therefore, arguably her responsibility.

I would like to thank Julien, Marc and Jean-Marc, for saving me more than once with their technical help. In particular, I would also like to thank Julien for teaching me basically all I know about CAD 3D design.

I would also like to thank the whole OrigaBot consortium team, including the iCube team and, in particular, Léo Wurtz, who provided invaluable help with the drone design, prototyping and test bench.

I must also thank Fred Boyer for teaching me his way of dealing with dynamical modelling. If I managed to maintain (at least part of) my sanity during this thesis, I probably owe this to him and his method.

I would like to extend my gratitude to Marie-Eve, Nathalie, and the rest of the ISM administration for always being incredibly helpful whenever I needed.

Lastly, I would like to express my gratitude to my team and the rest of the PhD students, interns, and professors with whom I had the pleasure of sharing the lab (and who have fed me croissants in many occasions) throughout this journey.

Thank you all for your support and encouragement.

This research was supported by CNRS, Aix-Marseille University and the French National Research Agency (ANR) via the OrigaBot project (ANR-18-CE33-0008-01).

Contents

List of Figures

List of Tables

List of Acronyms

ABS

Acrylonitrile Butadiene Styrene. 81

ANR

French National Agency for Research. 26

ENU

East-North-Up. 87

EOM

Equations Of Motion. 20, 28, 110, 232

FEMTO-ST

Franche-Comté Électronique Mécanique Thermique et Optique - Sciences et Technologies. 27

FOLD

Flexible Origami List Datastructure. 33

ICube

Laboratory of Engineering, Computer Science and Imagery. 27

IMU

Inertial Measurement Unit. 83

ISM

Institute of Movement Sciences. 27

JSON

JavaScript Object Notation file. 33

NED

North-East-Down. 87

PID

Proportional-Integral-Differential controller. 75, 95

PLA

Polylactic Acid. 81

PTD

Pop-Through Defect. 50

REBO

Reconfigurable Expanding Bistable Origami. 51

SVG

Scalable Vector Graphics. 33

TPU

Thermoplastic Polyurethane. 53

UART

Universal Asynchronous Receiver Transmitter. 80

UAV

Unmanned Aerial Vehicle. 21, 68, 77

Glossary

bendy straw

Soft multistable structure based on the commercial drinking straw. 50

biomimetics

Field in which nature is used as an inspiration for the development of new technological solutions. 25

bistable

Having 2 distinct stable configurations. 21, 39, 51

circumradius

Radius of the circle inside which a polygon can be inscribed. 39

compliant mechanism

Flexible mechanism that achieves motion through body deformation, as opposed to a rigid component. 23, 25

Davenport angles

Generalization of Euler angles. 173

Elle0

Paparazzi compatible autopilot, discontinued. 216

Euler angles

Set of three angles introduced by Leonhard Euler to describe the orientation of a rigid body. 20, 22, 23, 161, 243

Euler-Lagrange

Classical method of obtaining the EOM of a system using the expression of its energies. 21, 28, 232

Euler-Poincaré

Alternative formulation of the Euler-Lagrange method, possible when the configuration space of the system is a Lie group. 28, 110

flat-foldable

Origami patterns that can be theoretically completely folded flat. 37

Formlabs

Resin 3D-printer manufacturing company. 80

frusta

Plural of frustum. 50

frustum

Cone / pyramid whose tip has been truncated by a plane parallel to its base. 21

Inkscape

Open source vector graphics software. 22, 33

Kresling

A bistable, non rigid-foldable origami pattern. 38

Lie algebra

Tangent space of an associated Lie group, useful for representing their derivatives. 75

Lie group

A group that is also a smooth manifold, useful for representing configuration. 21, 75

Magic Ball

An origami structure based on the waterbomb pattern. 25, 36

monorotor

In this work, synonym of monospinner. 28, 68

monospinner

Flying UAV that spins, because of a lack of counter-torque. 21, 69, 96

mountain fold

Folds with flaps away from the folder. 31

multi-modal

Having 2 or more distinct modes of activity. 26

multistable

Having 2 or more distinct stable configurations. 20, 50

origami

Art of folding (and sometimes, also sculpting) paper to create structures or sculptures. 30

origami crease pattern

Origami diagram that consists of all or most of the creases in the final model, rendered into one image. 22, 30, 31

OrigamiPatterns

Inkscape extension for designing origami crease patterns. 33

OrigamiSimulator

Open source origami simulator. 33

Paparazzi

Open source flight control software for drones. 216

Phantom

Manufacturer of high-speed video cameras. 91

Pixracer

PX4 compatible autopilot. 216

Proper Euler angles

Set of three Euler angles in symmetric sequences (ZYZ, XYX, etc). 125, 161, 243

PX4

Open source flight control software for drones. 132, 216

quadcopter

Drone or helicopter with four rotors. 23, 68

quadrotor

Synonym of quadcopter. 77

quaternion

Hypercomplex number that can be used to represent a body's orientation. 28, 69, 116, 232

rigid component

Component whose body is supposed rigid, as opposed to a compliant mechanism. 20, 25

rigid-foldable

Origami patterns that can be folded and unfolded even if each facet is made of an infinitely rigid material. 21, 30, 37, 50

SciPy

Widely used Python scientific library. 161, 163, 167, 172, 178

SE(3)

The Special Euclidean group in 3 dimensions: 3 dimensions of position and a full 3D rotation. 111

SO(3)

The Special Orthogonal group in 3 dimensions: a full 3D rotation. 161

soft robotics

Field of robotics that studies the design and control of robots composed of compliant mechanisms. 25

swashplate

Mechanical device that translates helicopter flight controls into rotor blades motion. 23, 68, 78

swashplateless

System that emulates a swashplate. 28, 69, 71, 77, 80

SymPy

Pure Python symbolic mathematics library. 178, 225

Tait-Bryan angles

Set of three Euler angles in asymmetric sequences (ZYX, XYZ, etc). 161

tessellation

Tiling of a surface, in our case, a plane, using one or more repeated geometric shapes with no overlaps and no gaps. 36

Tracker

Free video analysis tool "*built on the Open Source Physics (OSP) Java framework*". 92

Tyto Robotics

Manufacturer of testing tools for drone characterization, formerly RC Benchmark (https://www.tytorobotics.com/). 84

Ultimaker

Filament 3D printer-manufacturing company. 53

underactuated

Not having enough actuators to be able to follow arbitrary trajectories in the configuration space. 68

valley fold

Folds with flaps towards the folder. 31

Vertiq

Manufacturer of motors with swashplateless firmware, formerly IQControl. 80

waterbomb pattern

A rigid-foldable origami pattern made of a waterbomb tessellation. 21, 24, 36

waterbomb tessellation

A tessellation pattern used for the construction of the waterbomb pattern. 24, 25

Introduction

Classical robotic systems are designed using rigid components, and most of their control strategies are facilitated by the fact that the movements of each part in an assembly can be analyzed separately. Soft robotics, on the other hand, focuses on the design and control of robotic systems using soft / compliant components. Designing robots with compliant mechanisms can be considerably harder: most of the methods developed for the finite degrees of freedom of classical robots fail to generalize to the continuous motion of a soft component. Larry Howell's book on Compliant Mechanisms [27] is a great resource that I would recommend to the interested reader.

Despite the difficulties, there are numerous advantages that explain why compliant robots are gaining such interest: for example, not only a flexible arm is capable of gently manipulating objects, it can also be considerably less dangerous in case of accidental contact when working in close contact with humans. For example, [19] considers a particular compliant mechanism for a surgery-related application. Robots made predominantly with soft material can be considerably more flexible and impact-resistant. Furthermore, soft robotics is of great interest for biomimetics: soft designs can take advantage of the millions of years nature has spent into finding optimal solutions for a multitude of problems. Moreover, as it is usually the case with biomimetics applications, the design and control of robots that mimic nature can also give new insights into understanding how these natural solutions work in nature.

Origami-based robotics can be seen, in many ways, as a sub-field of soft robotics. Origami structures created by folding sheets of material can still present many of the flexibility properties necessary for compliant mechanisms, while being usually simpler to fabricate and control. Actuating only some necessary folds can be enough to achieve many of the compliant and/or biologically-inspired properties of interest. *Rus and Sung* predict that origami-based robotics will be used for more customizability and adaptability on [53]. For example, [18] describes how a simple vacuum system together with an origami pattern known as the waterbomb tessellation[1] can be used to design a surprisingly simple and efficient soft gripper.

[1] A famous origami based on the waterbomb tessellation is called the Magic Ball. Since this particular origami is so well known, sometimes "waterbomb tessellation" and "magic ball" are used interchangeably.

The current research on origami-based structures includes very different and sometimes even surprising subjects. The lack of joints assembling multiple single structures makes it possible, in theory, to create more precise movements all while taking less space. For this reason, many researchers are interested on the use of origami-based structures for the design of microscopic robots. This same property, together with the fact that origami structures can be easily built, inspired many studies on the use of origami for medical applications [28]. For example, the oriceps: an origami-inspired forceps for surgery robots [20]. In a completely different context,

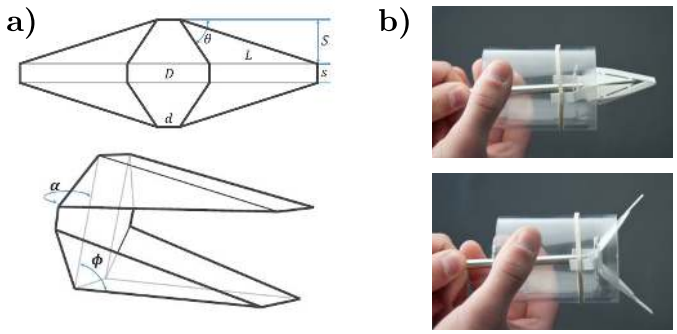


Figure 1.: Origami forceps illustrations. **a)** pattern and 3D illustration of folded structure. **b)** pictures of prototype. Source: [20].

a disc-shaped origami called "flasher" which can be easily opened and closed[2], has been studied as an alternative for solar panel assemblies in satellites. Its ease of actuation makes it a good candidate, since the solar panels must be deployed only after the satellite is already in orbit, and not before or during launch.

The shape-shifting capabilities of origami-based structures are also central to many projects aiming at constructing multi-modal flying and terrestrial robots. For example, [36] shows yet another application of the magic ball / waterbomb tessellation: a system of wheels that can change their shape and size to better adapt to certain terrain. Multi-modal locomotion is the key idea that kick-started the OrigaBot project: how can we take advantage of origami-based structures to create a shape-shifting drone that can move in different ways, according to its need?

The OrigaBot consortium, funded by the French National Agency for Research (ANR), was created to study this exact problem. As stated in the original funding proposal, "*The aim of the OrigaBot project is to develop a brand-new class of actuated origami-based structures*". The project also aimed at introducing the origami-based design expertise into France, while developing a robotic unmanned vehicle showing multi-modal locomotion, which means it can move in different ways according to need, with the use of shape-shifting origami. The initial idea was to make a flying drone that can also change into a terrestrial robot, according to

[2]And sometimes used by hobbyists to create an origami hat!

need, as illustrated in the Figure 2. To account for different domains of expertise

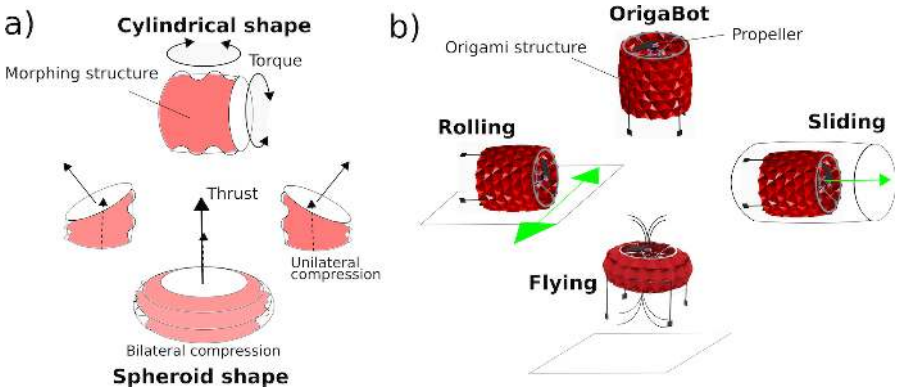


Figure 2.: Illustration from the original OrigaBot proposal.

required for making such a project possible, the OrigaBot consortium was created. Its key researchers are:

- **Stéphane Viollet** (Coordinator, ISM Biorobotics, Marseille): Control, autonomous robots design and embedded systems;
- **Pierre Renaud** (ICube, Strasbourg): Mechatronics and mechanical design:
- **Kanty Rabenorosoa** (FEMTO-ST, Besançon): Micro-mechatronics and kinematics.

Each one of them supervising a PhD student:

- Me, **Evandro Bernardes**, supervised by Stéphane Viollet: Preliminary origami design, dynamical modelling and flight control;
- **John Berre**, supervised by Pierre Renaud: Origami design and mechanical integration;
- **Kejun Hu**, supervised by Kanty Rabenorosoa: Micro actuators for origami folds.

In this manuscript, I will document my work for the OrigaBot project.

Since I worked on two very different aspects of the project, that will eventually

converge to a final project during the last year of the OrigaBot project, this document will be roughly divided into two parts.

The first part will document my work and findings during my first year working for the OrigaBot project, in which I was mostly implicated in the origami aspects of the project, comprising Chapters 1 and 2. Chapter 1 will document my first steps into origami design, the tools and processes I developed for designing origami, and the particular origami designs of interest that were studied. Chapter 2 talks about the origami bendy straw, a new origami design based on the bendy straw that was developed at ISM Biorobotics and published at ASME Mechanical Design.

The second part of the manuscript will not discuss origami design. At some point during the project, the choice for an extremely minimalist flying drone was chosen instead of the original multi-modal terrestrial/aerial drone idea. We decided to pursue the design of a monorotor flying drone which uses an origami structure to control its rotation speed. Chapter 4 discuss the swashplateless rotor system used for the motor. Chapter 5 gives an overview of each part of the drone, as conceived during the project. Chapter 6 gives the modeling of the Equations Of Motion (EOM) of the flying drone using the Euler-Poincaré equation, an alternative to the Euler-Lagrange equation. Chapter 7 details how to decompose the rotation quaternion of the robot into two parts: a controllable reduced part and an uncontrollable spinning component. Chapter 8 derives the non-linear control law we plan on using to stabilize the spinning robot. Chapter 9 assembles everything into a simulation of the drone, and uses it to derive a possible estimator of the robot's spin rotation.

Part I.

Origami studies

1. Preliminary studies

Even though origami was, initially, a big part of the project, I ended up not working much with it after the first few months, since this task was ultimately taken over by the John Berre and our other partners from the ICube team. In this chapter, I will discuss all the preliminary origami studies that I did during the first steps of my thesis project, which culminated on the publication of my first article in this project. If the reader is interested on the monorotor drone project, which ended up being the focus of most of my thesis, he/she can safely jump to Chapter 4 and start from there.

1.1. What even is origami?

In this section, I will define what the word "origami" means in the context of this work, and some important basic origami definitions. Origami [1] is the name given to the art of folding (and sometimes, also sculpting) paper to create structures or sculptures. The modern definition of origami varies slightly according to different origami artists. Some, more on the purist side, prefer to use the word "origami" strictly for sculptures created from a **square**, **uncut** sheet of paper and without the use of any kind of **glue**. According to the purist definition, even a single cut on the pattern is enough to call the structure a "kirigami" instead of an "origami", even though kirigami is a very different form of art, albeit a related one. However, as explained by Robert Lang on the first page of his excellent book on mathematical origami [33], "*... there are no fixed rules about paper, glue, or use of cuts: traditional Japanese designs, many of which can be reliably dated to be hundreds of years old, used various sizes and shapes of paper, sometimes multiples sheets, and often used cuts.*" Figure 1.1 shows a simple example of an origami origami crease pattern, and Fig. 1.2 shows some very different styles of origami used in art.

In this work, I consider origami as structures that can be created by folding a flat sheet of material regardless of shape, glued or not. Moreover, if it has cuts, I will still call it an origami and not a kirigami as long as the properties of interest of the structure come from the folding itself (or from the folding-like assemblies, in case of rigid-foldable origami).

[1] Literally "paper folding" in Japanese

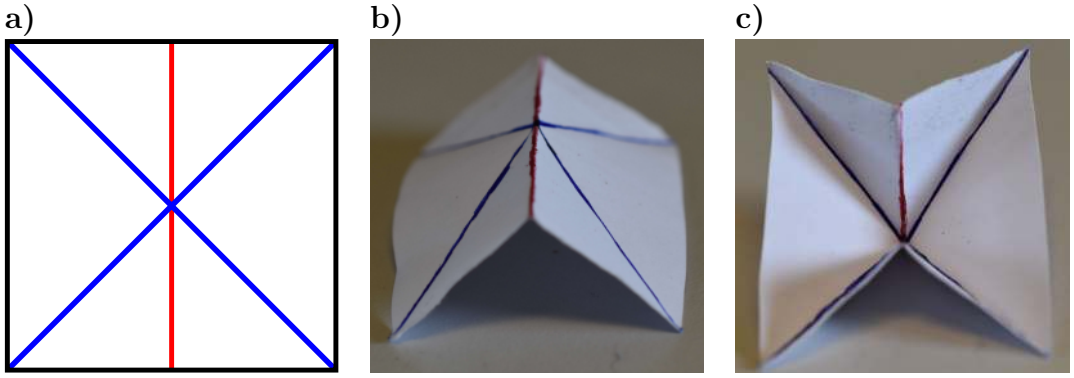


Figure 1.1.: Waterbomb base crease pattern. **a)** The original origami crease pattern. **b)** Waterbomb base after folding only its mountain fold. **c)** Same origami after folding every fold.

In particular, for this work, I am interested in origami structures that can be well defined using an origami origami crease pattern. Origami crease patterns are diagrams that can be thought of as instructions on how to fold an origami structure. In this work, **blue** lines represent valley folds (the flaps fold towards you) and **red** lines represent mountain folds (the flaps fold away from you). **Black** lines represent the edges of the paper, and occasionally, **green** colors represent cuts on the paper. Intersection points between two or more crease patterns are called *vertices*.

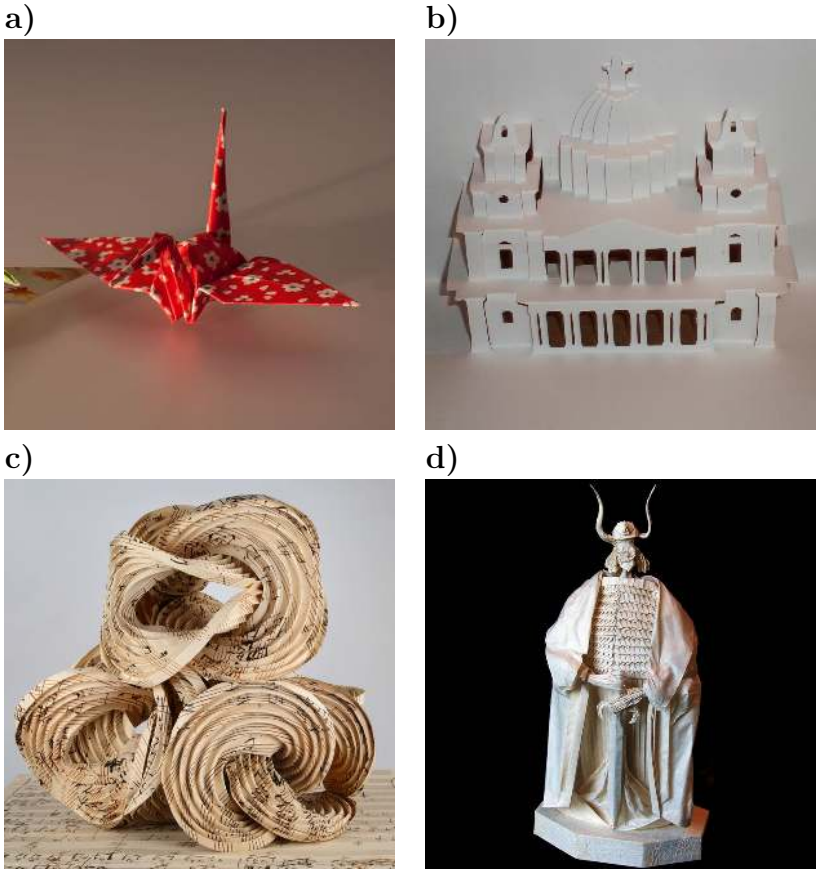


Figure 1.2.: Examples of different origami styles. **a)** The most classic origami structure, the crane. Pure enough even for the most purist origami artist! **b)** St Paul's Cathedral kirigami model, showcasing how kirigami can be used for architectural reproduction. **c)** Erik Demaine's "Fugal Form" from his Beethoven Series, showcasing the use of curved folds. **d)** A figurine by the late Éric Joisel. Adept of the wet folding method, he called his art "jazz origami", for his use of artistic improvisation.

1.2. Useful software

In this section, I will briefly talk about two tools that greatly helped the study of different origami structures.

1.2.1. OrigamiSimulator

OrigamiSimulator is a web app was built by Amanda Ghassaei and is based on the simulation method[2] published by herself and Erik Demaine on [24]. The web app can be accessed on [23]. While this simulator lacks the ability of setting the properties, thickness, etc. of the material used for the simulation, it compensates with its ease of use. It takes either a FOLD file or an SVG vector image file in order to define its patterns.

A FOLD file consists of a specialized JSON file that defines all the nodes, folds, facets, etc. of a geometrical folded structure.

Using an SVG vector image file as an input is the simplest way of using it. A simple drawing of the patterns can be used for starting the simulation. As stated on the website, the most important instructions are:

- Mountain folds have **red** stroke - rgb(255, 0, 0), hex #ff0000.
- Valley folds have **blue** stroke - rgb(0, 0, 255), hex #0000ff.
- Boundary edges have **black** stroke - rgb(0, 0, 0), hex #000000 - use this edge type for both the outline of the pattern, and any internal holes.
- Undriven (free) creases have **magenta** stroke - rgb(255, 0, 255), hex #ff00ff.
- The final fold angle of a mountain or valley fold is set by its opacity.

Figure 1.3 shows an example of simulation using an SVG file.

1.2.2. Inkscape and OrigamiPatterns extension

In order to quickly design different origami structures based on geometric patterns, I developed an extension for the free vector graphical software Inkscape, named OrigamiPatterns. This extension define classes that are used to easily implement different origami patterns, with a common system for defining settings separately for every kind of stroke (Mountains, Valleys, etc.)

Figure 1.4 illustrates a simple use for the extension. Figure 1.5 shows some complex origami structures designed with OrigamiPatterns. The process used for creating the thick origami of Figure 1.5 b), in particular, is described in Appendix B.

[2]This method considers a purely elastic model.

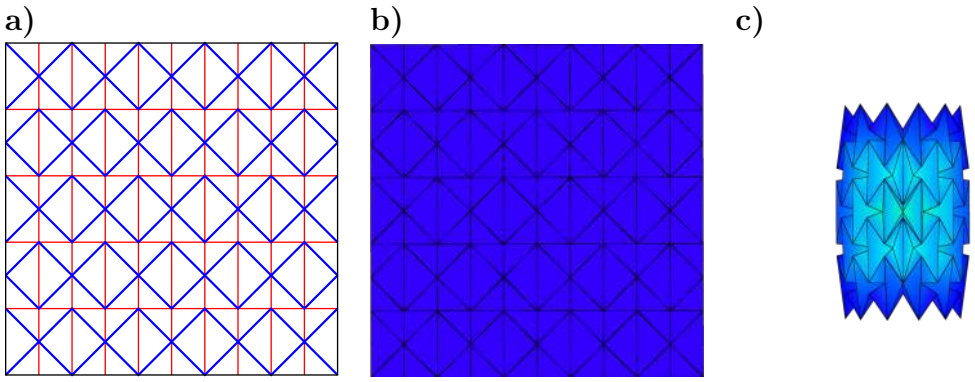


Figure 1.3.: OrigamiSimulator example. **a)** Origami pattern, saved as an SVG file. **b)** Pattern loaded on OrigamiSimulator, 0% folded. **c)** 75% folded.

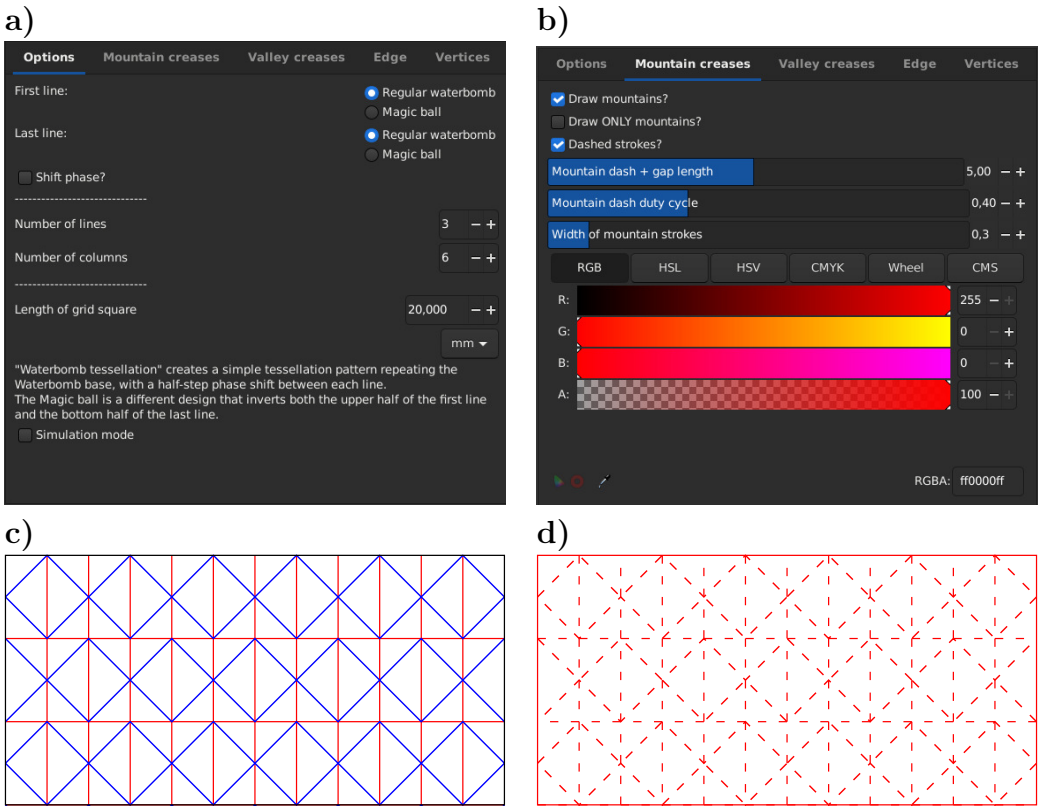


Figure 1.4.: OrigamiPatterns implementation of the Waterbomb tessellation. **a)** Specific options for Waterbomb pattern. **b)** mountain fold options. **c)** Pattern generated for use with OrigamiSimulator. **d)** Pattern generated for laser-cutting.

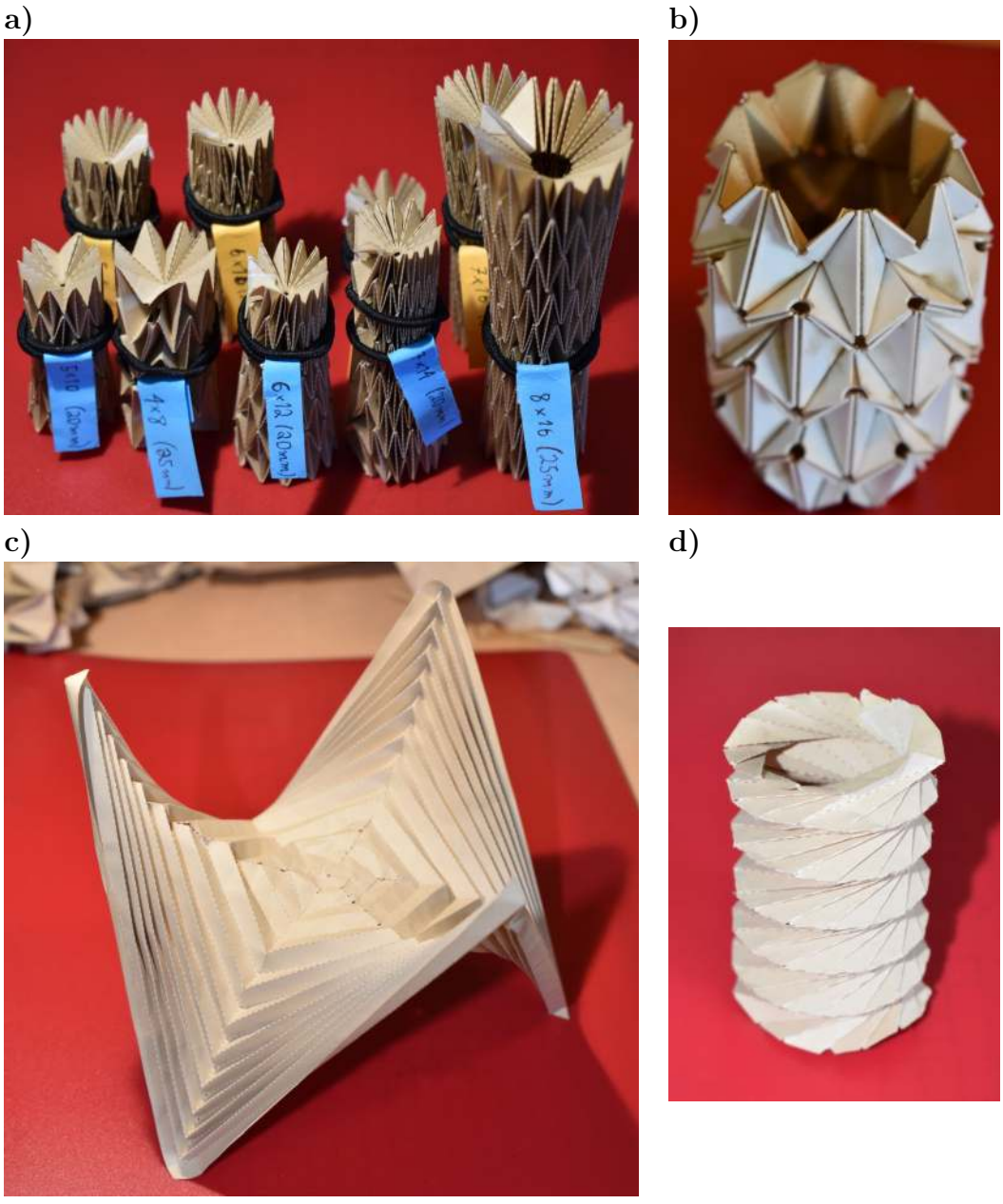


Figure 1.5.: Laser-cut origami structures designed with the help of OrigamiPatterns. **a)** Multiple magic balls. **b)** Thick and rigid magic ball. **c)** Hexagonal hypar. **d)** Kresling tower. Photos: E. Bernardes.

1.3. Drone shapes and existing origami structures

One of the first steps of the OrigaBot project was a preliminary study of the possible shapes for the robot, and which kind of existing origami structures can be used for achieving the desired shapes. A big part of this step was done by John Berre (first, as an intern for his master's thesis and then again as a PhD student). Figure 1.6 shows some possible shape-changing origami that exist in the literature. We will discuss with more detail the properties of some of them.

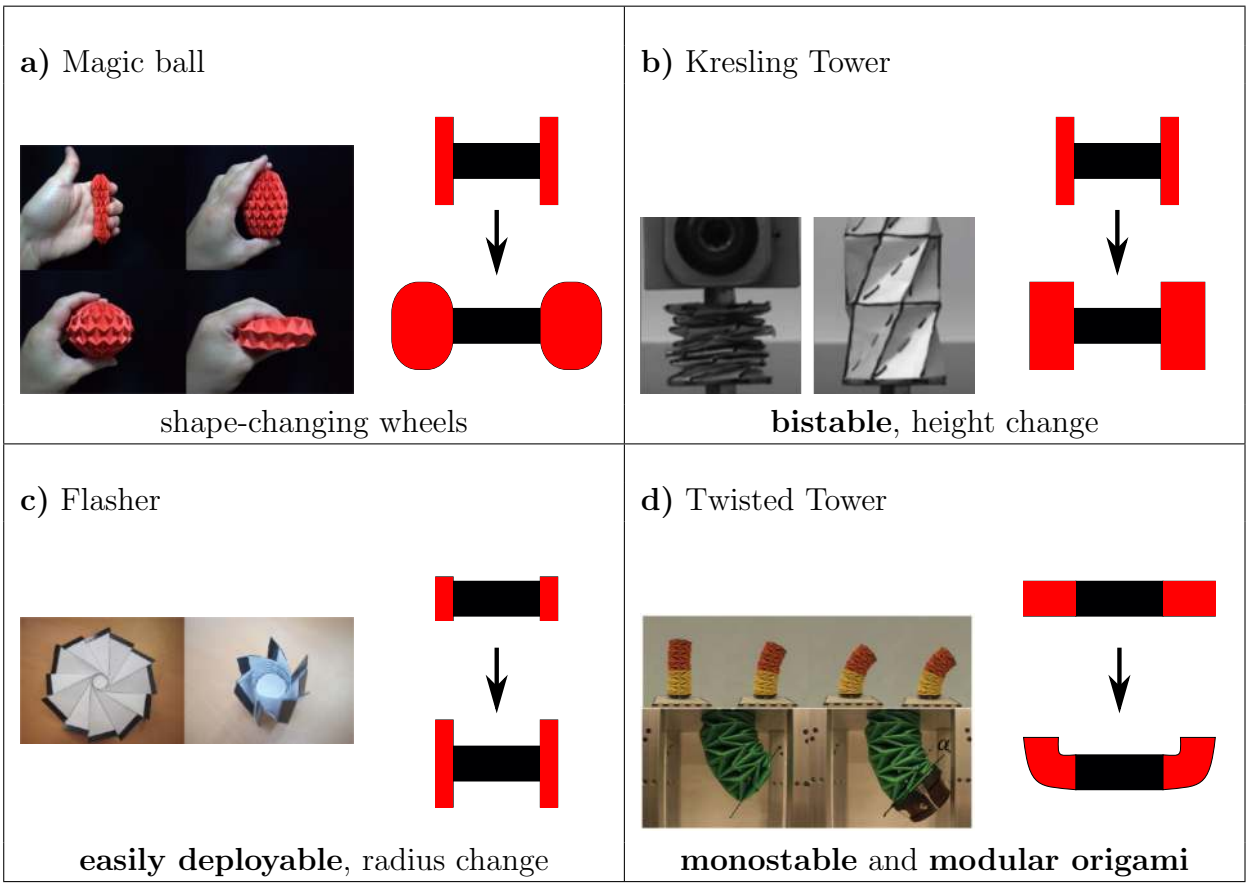


Figure 1.6.: Different shape-changing origami structures. **a)** Illustration of a magic ball origami, changing its shape from a cylinder into a spheroid, and finally into a disc. Source: flickr.com/photos/26201012@N02/5411813561. **b)** Kresling tower, converts a rotation into a height change and has two equilibrium points. **c)** Flasher, change of radius. It was studied by John Berre for the OrigaBot project. **d)** Twisted tower, which has a flexion but is a modular origami (which is harder to make) and is not bistable.

1.3.1. Change of shape with the "Magic Ball"

One of the main origami structures studied in the beginning of the project is the Magic Ball. It consists on a tessellation of the waterbomb pattern, usually closed

in order to create a cylinder, which can be seen in Fig. 1.6 **a**.

The magic ball is a good example of rigid and flat-foldable origami. A flat-foldable origami crease pattern is a pattern that can be folded completely (every fold is folded to 180°) and creates a flat structure. In order for a pattern to be flat-foldable, it must respect two rules:

- *Maekawa-Justin Theorem*: the number of mountains and valleys connected at any vertex always differ by two. Figure 1.7 illustrates it on the two types of vertices found on the magic ball's crease pattern.

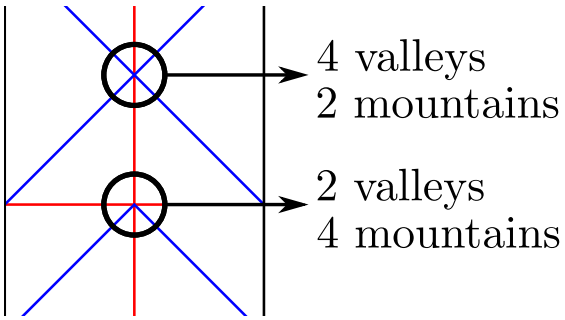


Figure 1.7.: Illustration of the Maekawa-Justin theorem.

- *Kawasaki-Justin Theorem*: at each vertex, the sum of all odd sector angles must be equal to the sum of all even sector angles. Figure 1.8 illustrates it on one of the vertex types of magic ball.

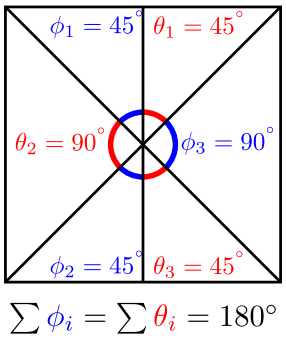


Figure 1.8.: Illustration of the Kawasaki-Justin theorem.

Moreover, the waterbomb tessellation is also rigid-foldable: it can be easily constructed with sheets of rigid material, and if each facet of the paper between its folds is infinitely rigid (thus, they do not bent), the origami can still be folded. This also means that the folds can be simulated in the form of rotational hinges [17]. Rigid origami structures are, in a sense, easier to study: they can be analyzed as a simple assembly of rigid elements. Figure 1.9 illustrates an example of a rigid waterbomb tessellation constructed with the membrane technique. There are many techniques for accommodating sheets of thick material [65, 80, 34], and the example

on Fig. 1.9 uses a "membrane" on which the rigid facets are glued. Note how the facets do not have all the same thickness: this make the completely folded structure have a constant thickness by compensating the fact that, when completely folded, not every part of the structure has the same amount of layers. Also, note how some space is left in each valley fold according to the thickness of its neighboring facets.

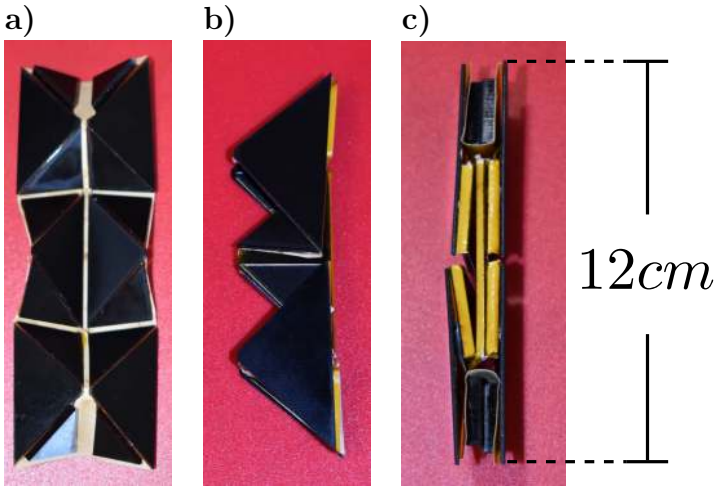


Figure 1.9.: Rigid 3×1 waterbomb tessellation. **a)** Flat origami before folding. **b)** Completely folded origami, showing how the waterbomb tessellation is flat-foldable: every fold is folded at 180°. **c)** Side view of the completely folded origami, showing its layers of different thickness. Photos: E. Bernardes.

These properties, together, make the magic ball a very flexible structure. It can be easily actuated, and its number of degrees of freedom is proportional to the number of cells. A magic ball with many cells (and a high number of degrees of freedom) was used to create a soft gripper [18], for example (see Fig. 1.10). This gripper is actuated by a vacuum system and. Even though it closes when the vacuum system is activated, it is still relatively soft in order to fit the shape of the object.

Another use for the magic ball is the origami wheel [36]. In this design, a relatively stiffer magic ball with fewer cells was used to create a wheel that changes its diameter, in order to better adapt to different types of terrain.

One of the initial ideas for the OrigaBot project was to study the possible use of a magic ball structure for a shape-shifting drone. In the end, the flexibility of the magic ball turned out not to be useful on the project.

1.3.2. Bi-stability with the Kresling tower

The second origami-based structure of interest I studied was the Kresling tower. Arguably simpler to design than the magic ball, it is a structure that converts a rotating movement into a linear movement.

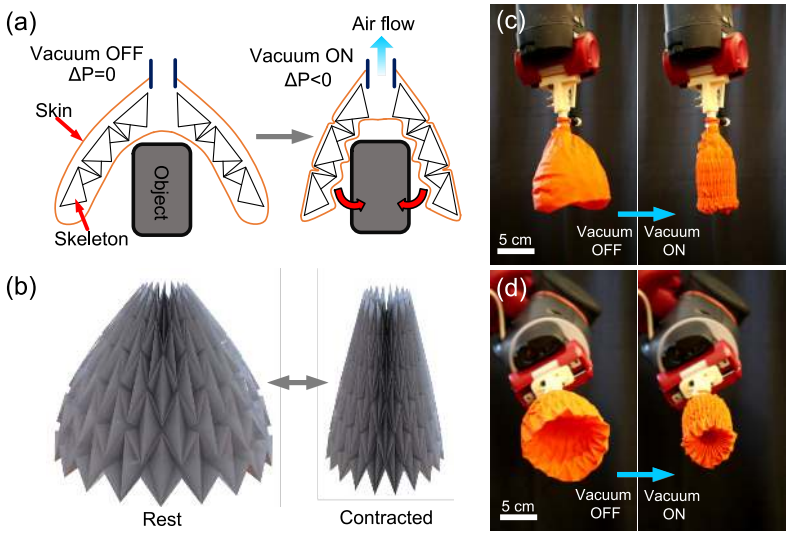


Figure 1.10.: Gripper system using Waterbomb tessellation origami. (a) The working principle of the gripper. (b) Origami "magic-ball" skeleton. (c)-(d) Prototype of the gripper from different perspectives. Source: [18].

According to [43], we can fully define a general Kresling tower pattern by three parameters: n, the number of sides of the polygon defining the cylinder, R, the circumradius of the polygon, and λ, the helical angle ratio. These parameters are enough to define all the other variables shown in Figure 1.11.

$$\begin{aligned} a &= 2\,R\,\sin\left(\frac{\pi}{n}\right) \\ b &= \left(l^2 + a^2 - 2la\cos(\lambda\theta)\right)^{1/2} \\ \theta &= \frac{\pi(n-2)}{2n} \\ l &= 2\,R\,\cos\left(\theta\,(1-\lambda)\right) \\ r_i &= \quad R\,\sin\left(\theta\,(1-\lambda)\right) \end{aligned} \tag{1.1}$$

This structure also appears naturally when gluing paper into a cylinder and applying a torque. This is illustrated on Fig. 1.12. Appendix C shows an analysis of the spatial constraints when closing the structure.

A very important aspect of the Kresling tower, that separates it from the magic ball, is the fact that **the Kresling tower is not rigid-foldable** (it cannot be constructed from an infinitely rigid material). In order to make a foldable Kresling tower, the material has to be flexible and/or elastic enough for deformations to be possible during the folding process. This may appear as a shortcoming, but it is directly responsible for making the Kresling tower bistable, which is key property of which we plan on taking advantage for this project. This happens because when the tower is actuated from one state to the other, its fold lines are briefly distorted

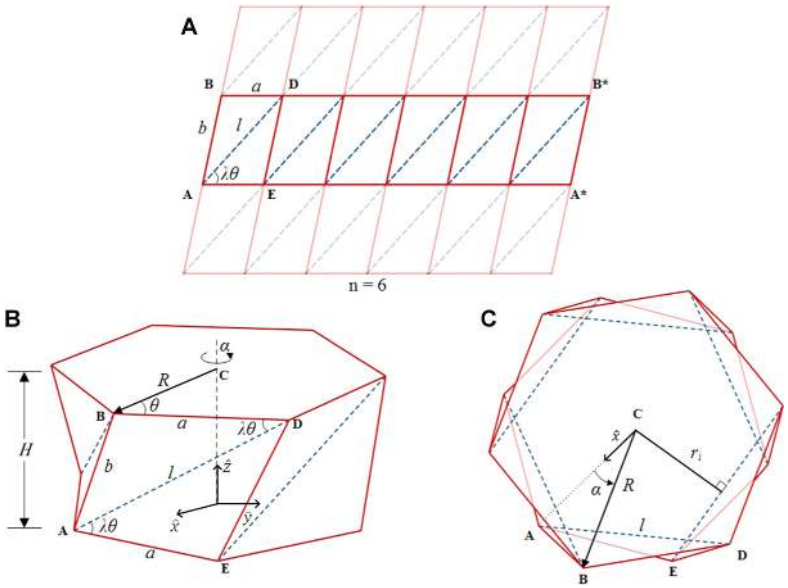


Figure 1.11.: Geometry of the Kresling tower structure [43]. **A** The origami crease pattern for the polygonal origami structure shown in **B** which is derived from the Kresling pattern. Mountain folds are shown by solid red lines and valley folds are shown by dashed blue lines. The parameters are: n, the number of sides, a is the polygon side length, l is the diagonal valley-crease length, b is the side panel length, the angle θ is half the internal angle of the basal polygon, and λ, the angle ratio, is a metric of transformation between open and closed states and can vary between 0.5 and 1. Also note the rotation angle α of the top polygon about the axis shown in **B**. The internal radius, r_i defined with respect to the cavity formed by the valley-folds while in the closed position is also shown in **C**.

and the facets are bent, which creates a maximum of stored energy. When this energy curve is sufficiently steep, some energy must be applied in order to make the tower go from one state to the other, creating a structure that has two equilibrium positions: One low energy stable position, the fully open tower, and one high energy unstable equilibrium. When this unstable equilibrium is sufficiently stable, we managed to effectively create a bistable structure. Various assemblies of Kresling cells can also be multistable [42, 43]: the authors of several studies have described how to control the bistability of Kresling cells [29, 77]. Mathematically, there are only two possible positions which respect the relationships found in Eq. 1.1.

Figure 1.13 shows an example of a 5 sided Kresling tower with 3 cells. If height bi-stability is desired without rotation between the two extremities of a tower, it is possible to achieve this by inverting the chirality of half of the cells, as shown in Fig. 1.14.

It can be hard to predict when a Kresling tower will be flexible enough to fold without tearing and, at the same time, be rigid enough to be bistable. The hardest parameters to study, are it usually happens when studying origami-based structures,

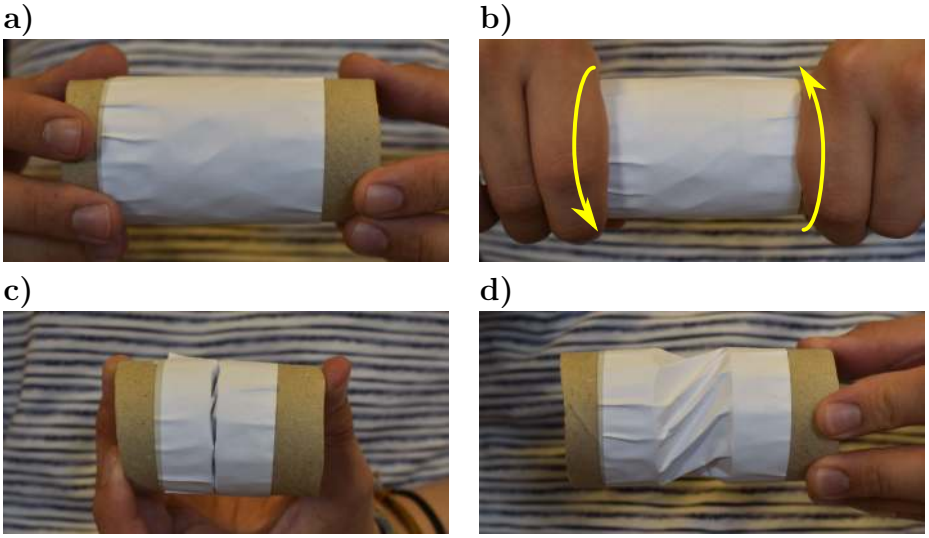


Figure 1.12.: Natural occurrence of Kresling pattern. **a)** Paper cylinder glued into rigid bases. **b)** Torque applied to both sides. **c)** Cylinder closed after application of torque. **d)** Opening of structure after having closed it on purpose. We can observe that the folds created on the paper resemble those of a Kresling tower. Photos: E. Bernardes.

are the thickness and properties of the material, since they are both not taken into account in the geometrical design of the origami pattern. The geometric parameters are more easily analyzed, though, and also take part in it. We can summarize with the following:

- The thicker the material, the stiffer the tower will be.
- The bigger the radius R, the bigger the tower, and the less stiff the tower will be (since the ratio between thickness / scale will be smaller.)
- The bigger the number of sides N, the less stiff the tower will be, since it will have more folds.
- The bigger the angle ratio λ, the stiffer the tower will be, since the fold lines will be almost perpendicular as λ approaches the limit of $\lambda = 1$.

As it was not defined in the literature, a purely geometrical analysis was done (which can be seen with more detail in Appendix D) to show that there are two possible solutions for the stable heights:

$$\begin{aligned} z^+ &= \sqrt{(l^2 + b^2)^2 - 4R^2} \\ z^- &= 0 \end{aligned} \tag{1.2}$$

The solution $z = z^+$ is accurate according to height measurements of many paper Kresling towers. The solution $z = z^- = 0$, though, is very inaccurate, as shown

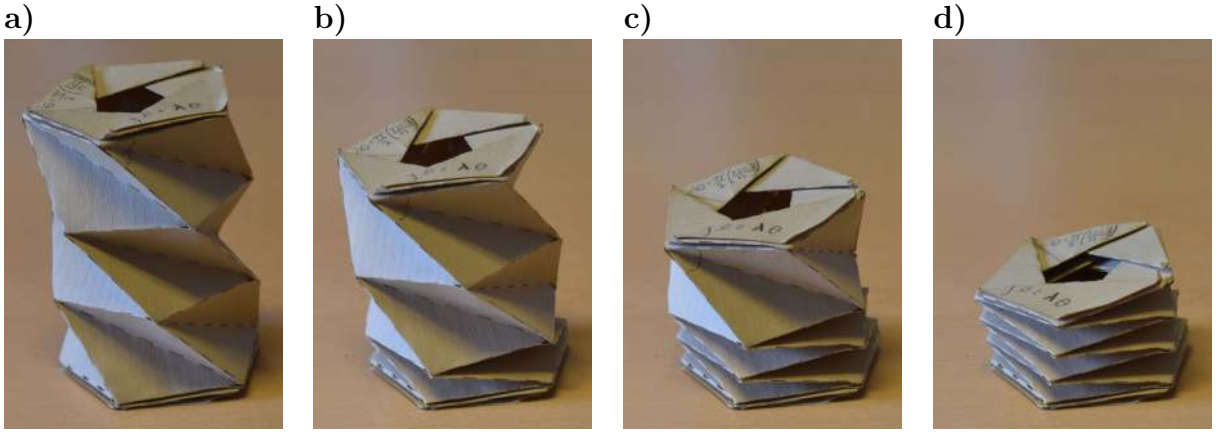


Figure 1.13.: 5-sided Kresling tower with 3 cells. From **a)**, with 0 cells collapsed, to **d)** with all cells collapsed. Photos: E. Bernardes.

from all the pictures. This is because the properties not taken into account on the geometric parameters (like thickness) have a strong influence on the structure. In fact, the solution of $z = 0$ would implicate an absolute compression of the layers of material against one another! The relationship in Eq. 1.2 can still be used to somehow predict the height of the uncollapsed tower, though. Figure 1.15 shows the comparison of the theoretical height of different towers and the comparison with the actual heights from different sets of towers built with different materials.

Usually, R is fixed by some kind of constraint given by the project. We can also choose N according to practical considerations. For example, for higher values of N, we have more folds which can be hard and unpractical to fold, specially with thicker material. Figure 1.5 **d)** shows a Kresling tower with multiple cells and $N = 16$. It was a very hard model to fold properly! Low values of N can also be bad, though, since they lead to very small values for r_i. In my tests, I usually fixed $N = 8$ as a practical number that is not very high (easy to fold) but sufficiently large so that the polygon defining the geometry of the tower resembles a circle sufficiently well so that its interior can be well represented by the radius. Finally, the remaining parameter is λ, which can be set as a stiffness parameter. This parameter can be set in order to find the optimal stiffness and consequential bi-stability for the tower. Figure 1.16 shows a comparison for λ varying from 0.5 to 1.0. We observe that the towers get taller with increasing values of λ, and their internal radius at the collapsed stage get smaller. Moreover, we note that the towers with $\lambda = 0.5$ and 0.6 are not bistable, so they only have the open, tall state. The tower with $\lambda = 0.7$ is bistable, but barely: simply touching it can cause it to change back to its tall, low energy equilibrium. The tower with $\lambda = 0.8$ is very stable in both states and easy to fold, while the tower with $\lambda = 0.9$ (also very stable) starts to be very stiff: much force is needed to make it collapse, and we can see that the paper gets very deformed after a single collapsing movement. This particular tower with $\lambda = 1.0$

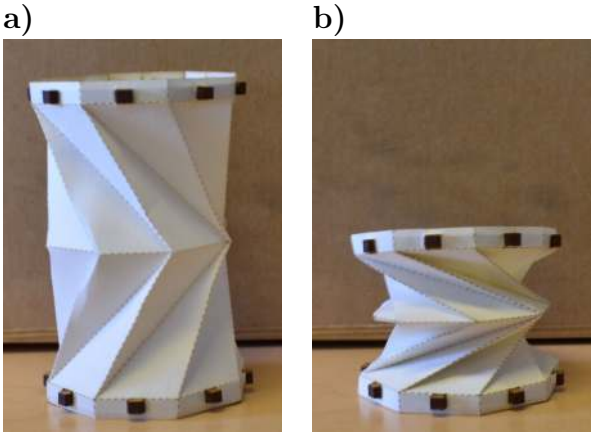


Figure 1.14.: 8-sided Kresling tower with 2 cells of inverse chirality. They can be opened (**a**) or closed (**b**) without rotating the top base with respect to the bottom base, at the expense of having both cells actuated at the same time in opposite directions. Photos: E. Bernardes.

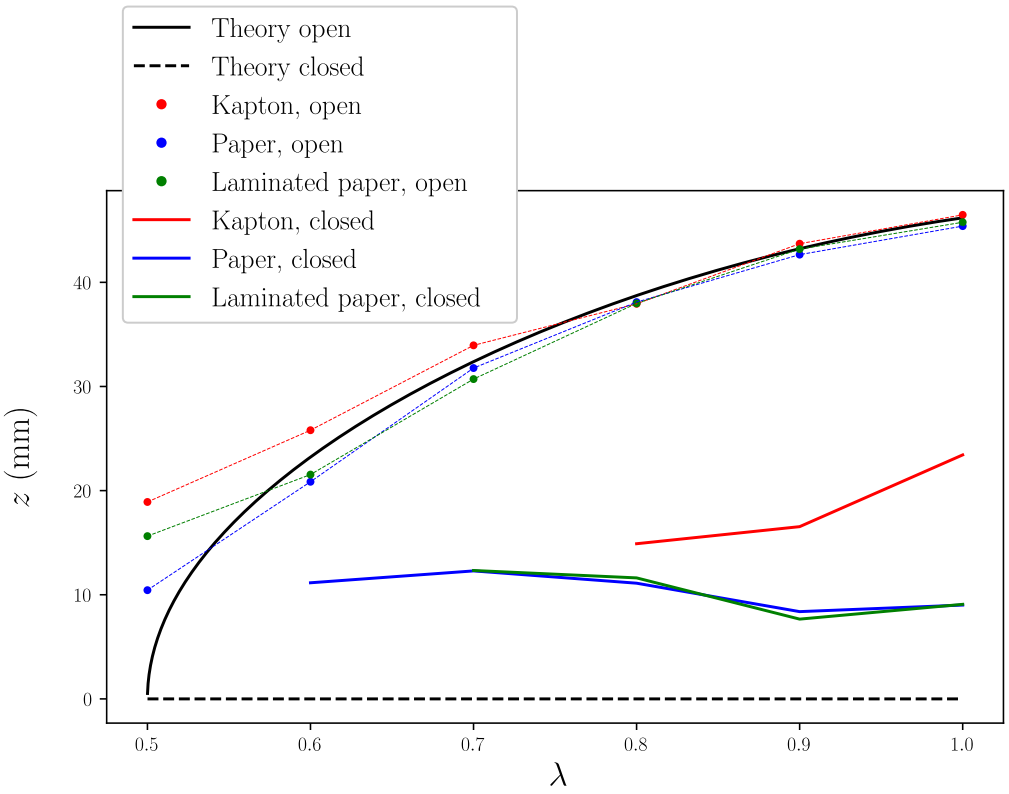


Figure 1.15.: Analytical solution of the Kresling tower heights, compared to real tests. Ratio $\frac{h}{2R}$ in function of λ, $R = 25$mm and $n = 8$.

was too stiff to be folded. The paper got very deformed, and it was not possible to keep folding it. We can notice on Fig. 1.16**a** how the towers with $\lambda = 1.0$ and $\lambda = 0.9$ are deformed. This hints that we must perform a proper study of the life

cycle of such a structure.

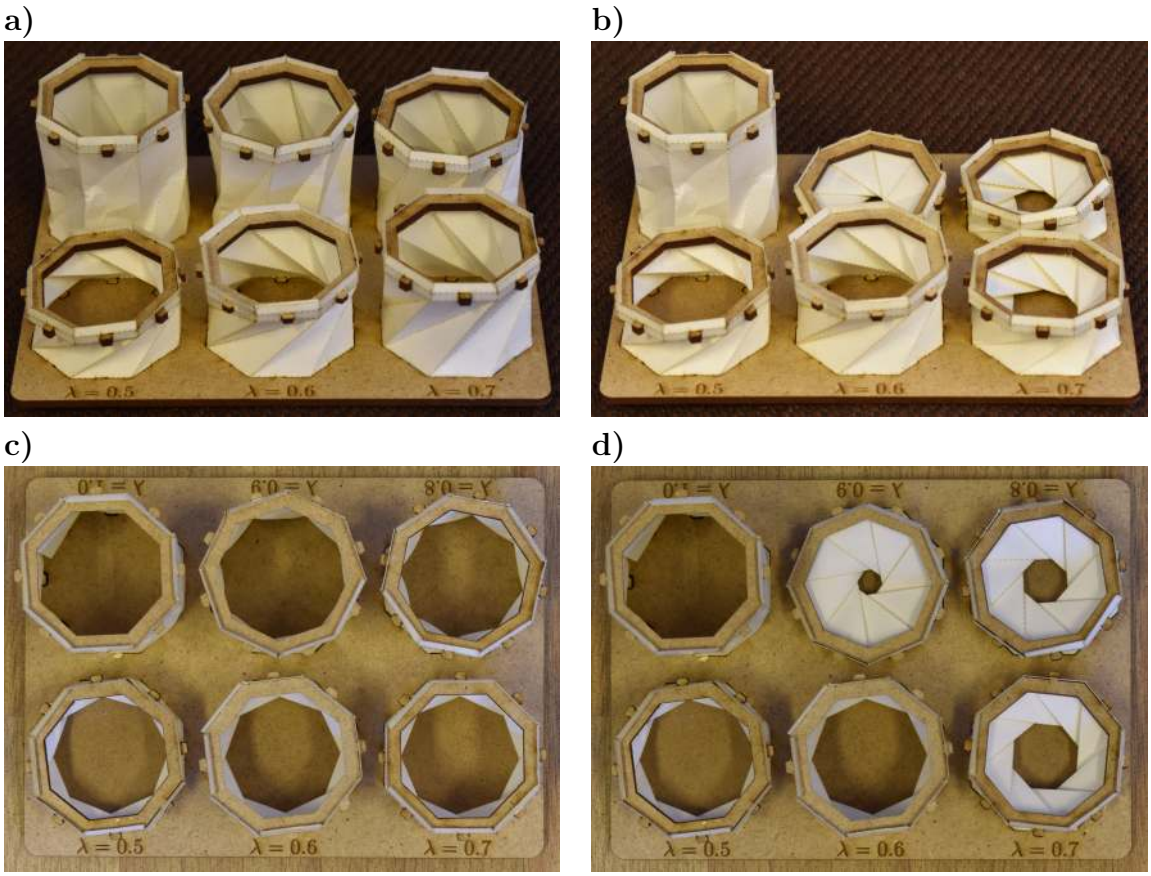


Figure 1.16.: Comparison of Kresling towers. All towers have $R = 25mm$, $N = 8$ and are made of bi-laminated paper with thickenss $= 0.2mm$. The values for λ are, counterclockwise starting from bottom left, $0.5, 0.6, 0.7, 0.8, 0.9$ and 1.0. **a)** Side view, all towers open. **b)** Side view, all collapsed. **c)** Top view, open. **d)** Top view, collapsed. Photos: E. Bernardes.

A more in depth study of the control of the material deformation of the Kresling tower, apart from its geometric considerations, was done by John Berre. His work can be seen in [7, 8].

1.3.3. Prototype 1: Pendulum-driven robot with Kresling tower

The Kresling tower was studied for the OrigaBot project because of its ease of construction, simple shape and bistable property. Moreover, its cylindrical shape with variable height makes it a great option for a shape-shifting drone. The first briefly considered shape was based on a single wheel, using the internal mass and inertia of the non rolling parts to roll the outer wheel without rolling the internal components (a full work on this type of robots can be seen in [11]). Being able to

control such a prototype is important: when a twin-wheel robot is moving straight ahead without turning, controlling it is essentially the same problem. We made a very simple prototype consisting of a single origami wheel structure, that can be seen on Fig. 1.17. Note that both sides of the structure are connected in this experiment, while each of them is connected to a different motor. The actual connection between the motor and the structure was made to be very thin: this way, if there is some speed difference between both motors, the structure will break (instead of the motors). The internal structure of the electronics is made in such

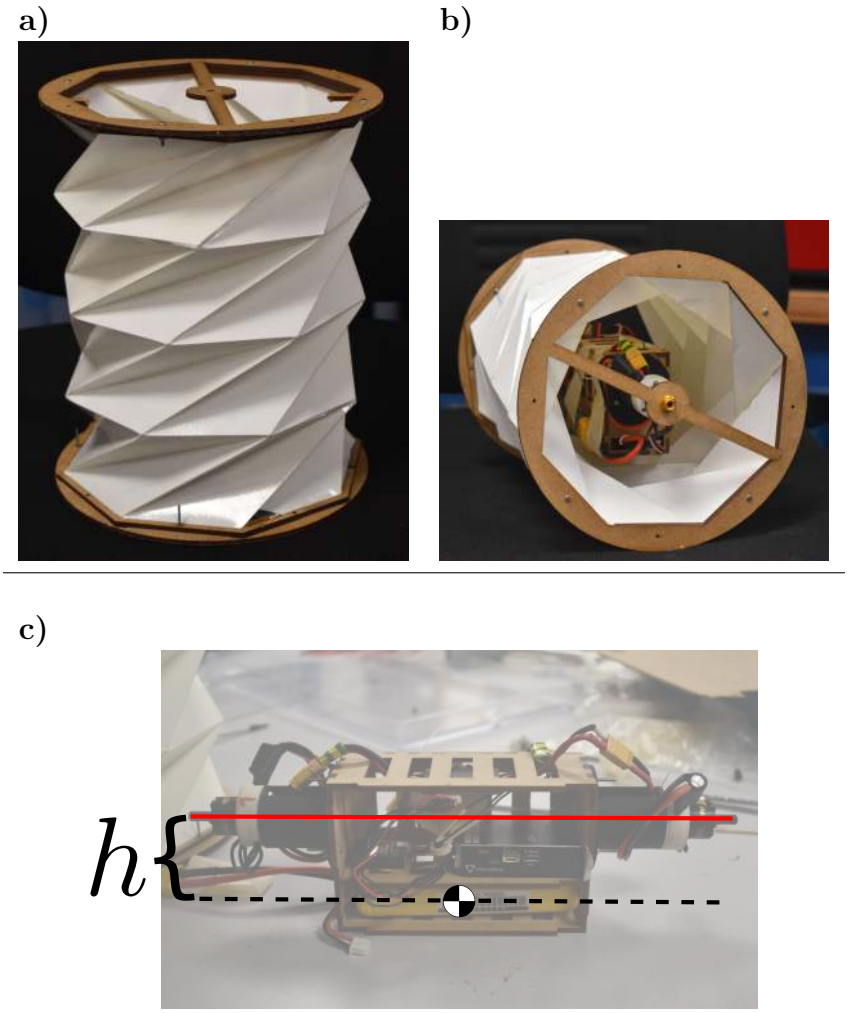


Figure 1.17.: **a)** Kresling tower serving as the whole external body and wheel of the robot. **b)** Side picture showing the interior after installing the electronics. **c)** Schematic showing the distance between the axis of rotation and the center of mass of the electronic components. Photos: E. Bernardes.

a way that its center of mass is not centered along the rotation axis. This way, we effectively create a pendulum-driven robot: actuating both motors creates an angle α that forces the robot to move by directly increasing the rolling speed $\dot{\varphi}$.

Assuming a perfect rolling movement without slipping, the linear velocity of the robot can be given by $\dot{x} = R\dot{\varphi}$, where R is the wheel's radius, as shown in Fig. 1.18.

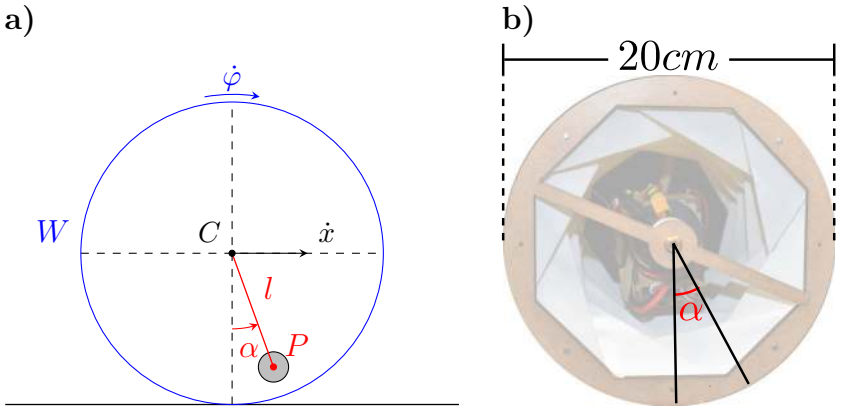


Figure 1.18.: **a)** Free-body diagram for the monowheel. Changing the pendulum angle α has a direct effect on the rolling speed $\dot{\varphi}$. **b)** Illustration of the angle α created by the pendulum during movement.

A mathematical analysis of this kind of pendulum-driven robots can be seen in [75]. One surprising fact of this prototype is that while we planned on doing a controller that acted on setting a different angle α for a given desired rolling speed $\dot{\varphi}$, the prototype worked well enough without any need of feedback control. This was a good indication that this prototype could be made without many complications. Figure 1.19 shows pictures taken during the tests. The monowheel prototype was

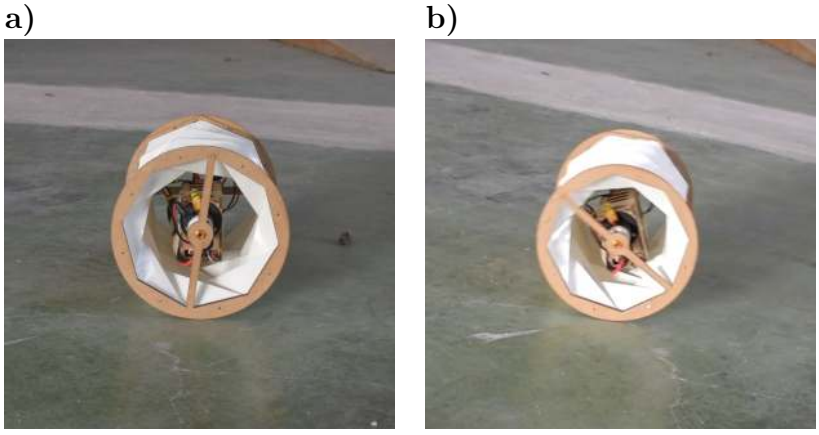


Figure 1.19.: **a)** Picture taken from the monowheel moments before the test. **b)** Frame taken from a video of the monowheel. Note that we can see the pendulum angle. Photos: E. Bernardes.

a single test to see if the configuration of the monowheel moving straight ahead would be a problem. We decided to proceed to the next prototypes before working on an actual controller for this terrestrial robot.

1.3.4. Prototype 2: Kresling-based origami twin-wheel

While the monowheel is simple to build, its movements are very limited: it cannot turn while rolling. For this reason, a different prototype separating the monowheel into two different origami wheels was studied, as shown in Fig. 1.20. Each wheel

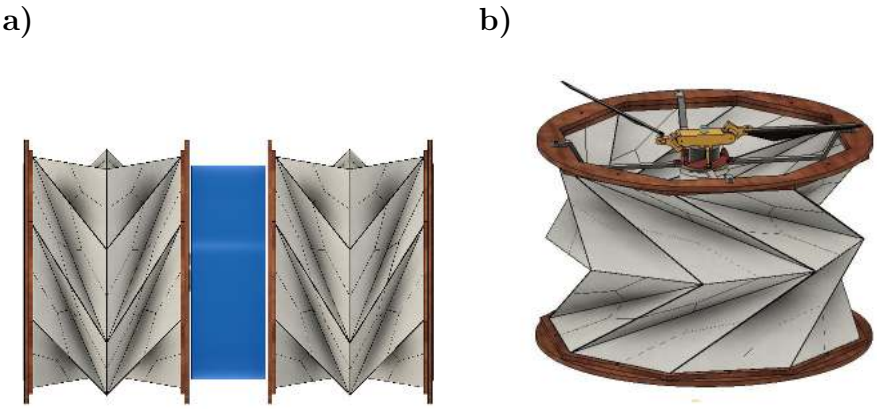


Figure 1.20.: **a)** Simple CAD of Kresling-based twin-wheel robot. **b)** Separated full wheel.

was conceptualized in order to have a latch system that would make it possible for the robot to pass from rolling to flying mode. This latch system can be seen in Fig. 1.21. The motor's stator is fixed to the center of the robot's main body via an actuated sliding joint, as seen in Fig. 1.21 **a**. This body is then connected to the Kresling wheel's bottom part via a loose ball joint, which can also be seen in Fig. 1.21 **a**. While the robot is in flight mode, the motor's rotor is disconnected to any other part of the robot and free to rotate. When the robot must enter the rolling mode, the stator's slide joint is actuated, putting the motor in the direction of the robot's center. This connects the rotor's latch hook (as seen in Fig. 1.21 **b**) into the Kresling wheel's latch socket, rigidly connecting them (Fig. 1.21 **c**). The wheel's latch socket was conceptualized to have a particular shape in order to guide the latch pin during this movement. Note that the rotor's structure is a very particular one, consisting of many components. This is the swashplateless system that will be explained with more detail in another chapter. In practical terms, for now, this system makes it possible to control the direction of the rotor's airflow.

This possible configuration would have two different regular modes of operation, and an added transition mode:

- **Flying mode**: The robot is in vertical position, with two sets of motors+helices rotating in opposite directions.
- **Rolling mode**: The robot is in horizontal position, and each rotor is used to perform differential control on the wheels.
- **Transition mode**: The robot uses the motors to change from a horizontal

to a vertical position, or vice-versa.

The transition mode should be made possible with the use of the thrust vector controllability given by the swashplateless mechanism (which will be explained in Chapter 4).

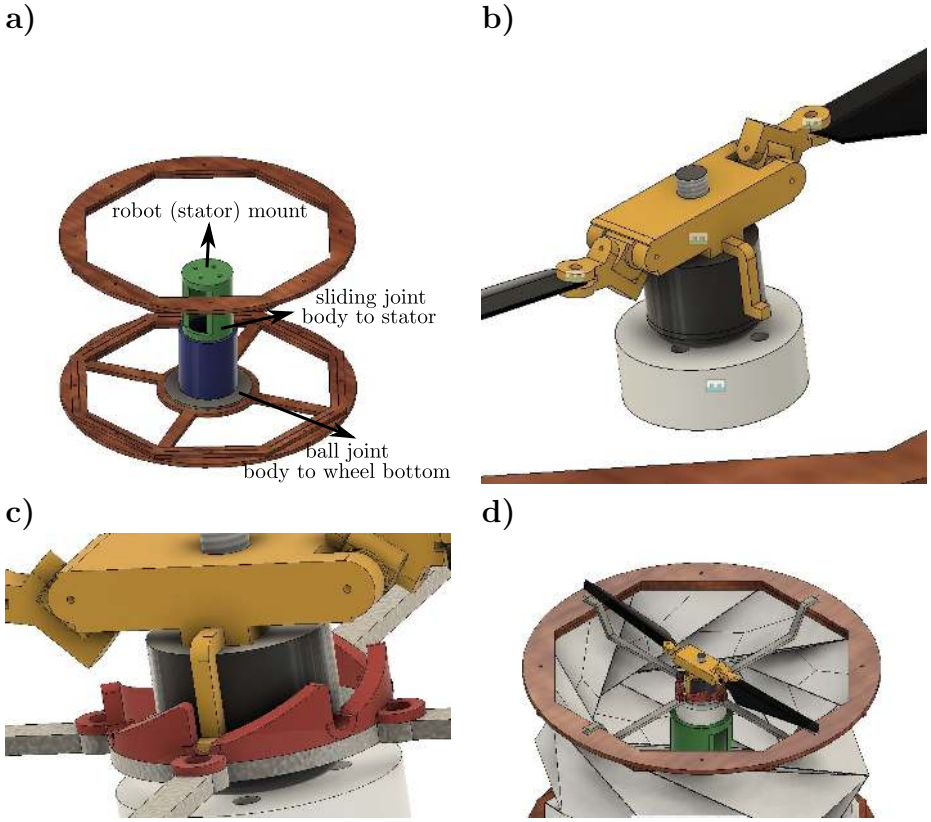


Figure 1.21.: CAD details for the latch system. **a)** Joint schematics. **b)** Rotor latch hook. **c)** Rotor latch hook connected to the wheel's latch socket. **d)** CAD of the latch connection, showing all components.

1.3.5. Other shapes and final considerations

One problem, found in this particular twin-wheel prototype, is that the swashplateless system might not be powerful enough to make the robot transform from the horizontal to vertical mode consistently. This is one of the reasons that led us to think of other possibilities. In the end, the robot shape of choice was a completely different one: the flying monorotor (that will be explained from Chapter 5 until the rest of this thesis). This is ultimately a complete change from the initial goals of the OrigaBot project.

A last consideration though, considered before switching to the monorotor idea, was a flexion based rotor. Instead of Kresling towers, a structure that adjusts the

direction in which the motors are facing would greatly simplify one of the problems of our twin-wheel idea, namely the change from horizontal to vertical position. This is because the robot would be always be horizontal, and only the motors would change direction. Figure 1.22 shows an example of this: both the twin-wheel robot configuration is well understood and easy to implement, and the bi-rotor configuration has also already been studied in the literature.

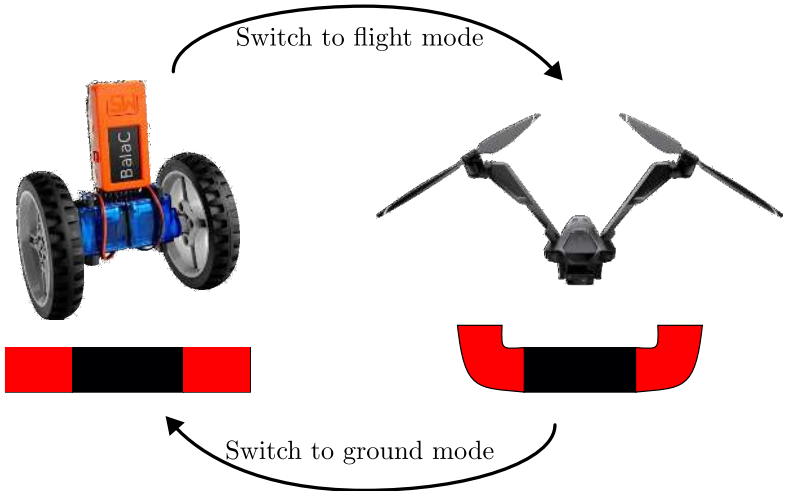


Figure 1.22.: Ground mode with a twin-wheel type robot and flight mode with a bi-rotor. On the left, a BALA-C PLUS balancing robot: an example of a twin-wheel robot, with the normal vector of the rotors parallel to the ground. On the right, the V-Coptr Falcon UAV, a flying bi-rotor with the normal vector of the rotors pointing up. Is there an origami structure that would help go from one configuration to the other effortlessly?

A survey of literature quickly showed that there are no satisfactory bistable origami-based structure able to do this movement. For example, Fig. 1.6 **d** illustrates the twisted tower, but this structure is hard to create (since it is a modular origami) and is not bistable. The structure that would fit our requirements the best is the bendy straw, which is not an origami structure. Seeing this opportunity, I succeeded in creating a new class of origami structures that do exactly this: the origami bendy straw. The next chapter will describe this structure.

2. Design of an origami bendy straw for robotic multistable structures

This section presents a soft cylindrical multistable origami structure based on "bendy straws", consisting of multiple conical frusta mimicking the structure of a flexible drinking straw. These frusta are connected in such a way that the whole structure is axially multistable, having a stable compressed state in which its smallest frustum is collapsed. The bendy straw structure can also be modified so that the smallest frustum collapses only partially, keeping the structure in a bent state. We studied the geometry of a similar structure consisting of polygonal frusta instead of conical ones, and used this geometry to design a non-rigid-foldable origami pattern folding into a similar origami bendy straw structure. Most of the origami structures presented so far have been modeled from rigid-foldable origami patterns: these origami structures do not rely on local deformations of the sheet, and cannot use it to their advantage; whereas the non-rigid origami structure presented here features multi-stability. We have established that this origami structure is not only axially multistable, but that it can also be kept in a bent state, thanks to the use of a Pop-Through Defect (PTD). The origami bendy straws studied here were made from paper (with a density of $90g/m^2$) bi-laminated with a 42.5-micron thick plastic film. A digital dynamometer was used to study the forces required to compress and expand a single origami bendy straw, create and reverse a PTD, and bend an origami bendy straw using PTDs. This chapter is largely based on the results published in [5].

2.1. Introduction

Most of the origami patterns used so far have been based on rigid-foldable designs, as in the case of the waterbomb tessellation pattern introducted in Chapter 1 (see [36, 35]). Tessellation origami patterns, whether they are rigid-foldable or not, can have useful properties, and existing patterns can sometimes be modified to endow them with even more desirable properties [55, 56]. Non-rigid-foldable origami structures have some useful properties, and can adopt various mechanically stable states. Origami tubes are one particular class of 3D origami structures that hold

great promise in fields such as medical robotics [15] and robotic manipulation, where they can serve as grippers [18]. Origami tubes have been thoroughly studied and modeled in recent studies [22, 21]. Among these tubular structures, the Kresling tower is an origami-based structure of particular interest.

It is worth noting that bi-stability and even multi-stability can also be observed in soft structures such as the common bendy straw, which can be either bistable or multistable as required [2]. This was the starting point of our study on the design and construction of a multistable origami tube. The present bendy straw structure consisting of non-identical conical frusta can easily be made to become axially bistable, which means that it can adopt two different states: a tall stable state and a compressed state in which its smallest frustum is inverted. It is also possible to add another stable state: when the structure has some built-in stress, its frustum can become partially inverted, giving the structure a bent shape, the direction of which can be continuously varied while maintaining the bent state [2].

One of the advantages of origami structures over those based on conventional bendy straws is that they can be constructed from folded sheets of thin material. Origami structures can be quickly and easily prototyped with paper (or laminated paper to obtain greater rigidity while still keeping it thin and flexible, as described in [14]), and subsequently replacing paper with other materials gives these structures a durable life cycle, better shock absorption properties [57] and a low mass, while maintaining their flexibility without any need for conventional joints. Origami structures can also be easily produced in large numbers: laser-cutting is a much faster means of creating an origami pattern than 3D printing or casting, as well as being more precise than 3D printing.

In this study, we analyzed the geometry of both the original bendy straw, a polygonal approximation of the bendy straw, and study for the first time an origami structure we have called the origami bendy straw. It is worth noting that this origami pattern is very similar to the previously presented Reconfigurable Expanding Bistable Origami (REBO) pattern [12], but we have described here how to introduce Pop-Through Defects (PTDs) in order to obtain the original bendy straw's bent position, which has never been applied so far to a REBO-like pattern. We have also established how to calculate the lengths and angles of each crease in the origami pattern in order to replicate a given bendy straw.

This paper focuses on the geometrical and design parameters required to construct an origami bendy straw structure with any given height, radius and bending angle, while analyzing the conditions under which the structure will be multistable. In the Section 2.2, we describe the geometry of the bendy straw and introduce the origami bendy straw pattern. In the Section 2.3, the force required to compress and expand an origami bendy straw cell is analyzed. In the Section 2.4, we describe how to modify the creases of the origami in order to replicate the bent state of the bendy straw, while also analyzing the force required to bend it.

2.2. Geometry and characterization

The circular bendy straw cell (as described in [2]) consists of a pair of conical frusta. These bendy straws can be described in terms of the following parameters: R, the external radius of the cell, the ratio $\rho = \frac{r}{R}$ between the internal and external radii, and α_1 and α_2, the angles between the XY plane and the top and bottom conical frusta, respectively. As shown in Fig. 2.1a, R, ρ, α_1 and α_2 can be used to

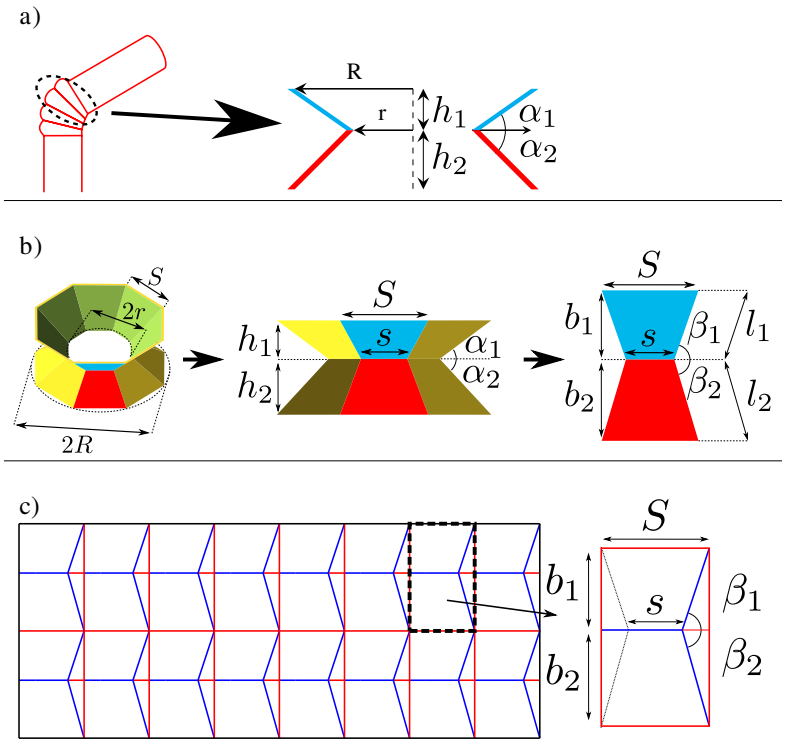


Figure 2.1.: **a)** Picture of commercial flexible straw, with one of its cells in the detail, and cross-section of a bendy straw structure, designed to replicate a flexible straw. The angles α_1 and α_2 are the angles between the XY plane and the surfaces of the top and bottom frusta, respectively. **b)** 3D model of an octagonal origami bendy straw, showing the geometry of the top facets (in blue) and bottom facets (in red). Left: Perspective view of a polygonal bendy straw, showing its external and internal radii. Middle: side view showing the cell heights h_1 and h_2, the side lengths S and s and the cell angles α_1 and α_2. Right: the facets shown in red and blue are flattened, and the lengths and angles of the lines are shown. Note that when $n \to \infty$: $S, s \to 0$, $b_1 \to l_1$, $b_2 \to l_2$ and $\beta_1, \beta_2 \to \pi/2$. **c)** Origami bendy straw pattern used to produce the same structure. Valley folds are in blue, mountain folds are in red, and the dashed black line stands for a virtual fold that was removed ($\rho = 0.5$, $n = 8$, $\alpha_1 = 35°$, $\alpha_2 = 45°$, 2 stages).

calculate r, the internal radius, h_1 and h_2, the heights of the two frusta as follows:

$$\begin{cases} r &= \rho R \\ h_1 &= (R - r)\tan\alpha_1 \\ h_2 &= (R - r)\tan\alpha_2 \end{cases} \tag{2.1}$$

A polygonal bendy straw structure can also be defined as a similar structure, the conical frusta of which are replaced by pyramidal frusta (truncated pyramids) defined by n-sided regular polygons. To test whether the polygonal version had similar properties to those of the conventional bendy straw, a 3D printer (Ultimaker White TPU) was used to obtain a 3-stages polygonal bendy straw (see Fig. 2.2 a), with $R = 25mm$, $\rho = 0.4$, $\alpha_1 = 45°$, $\alpha_2 = 35°$ (setting $\alpha_1 - \alpha_2 = 10°$), $n = 8$, and thickness $t = 2mm$. The polygonal bendy straw structure was axially bistable

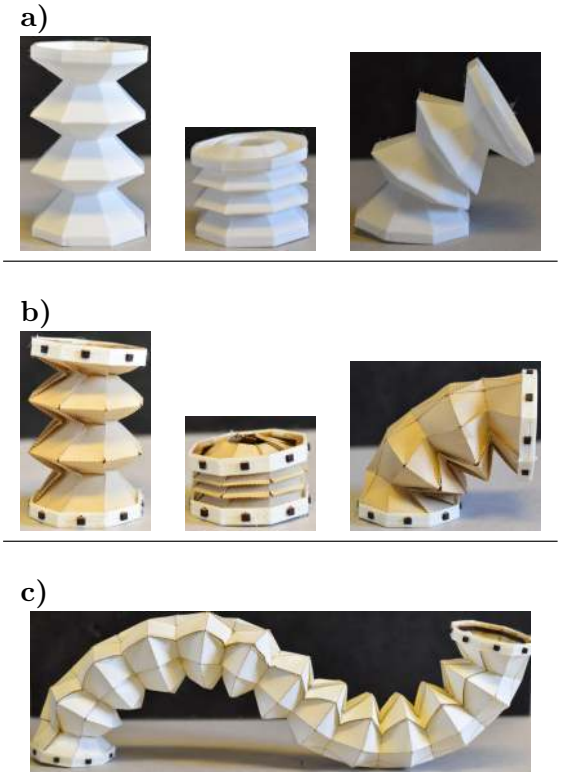


Figure 2.2.: **a)** Polygonal bendy straws 3D printed with Ultimaker White TPU. We observed that 3D printed polygonal bendy straws seem to have similar properties to those of the circular bendy straws. These polygonal bendy straws were 3D printed for the sake of comparison, whereas all the other bendy straws used in this study were origami bendy straws made of bi-laminated paper. **b)** Origami bendy straw version with a similar structure. **c)** 12-stage origami bendy straw with an "S" shape. Half of the cells bend in one direction, and the other half in the opposite direction. Parameters: $R = 25mm$, $\rho = 0.4$, $n = 8$, $\alpha_1 = 45°$, $\alpha_2 = 35°$.

right after being printed. After undergoing pressure in its compressed state for

about 5 minutes (as explained in [2]), it continued to be stable in the bent position. As shown in the Fig. 2.1 b, each frustum constituting the polygonal bendy straw structure was composed of multiple isosceles trapezoids. The angles and dimensions of each trapezoid were calculated as follows:

$$\begin{cases} S & = 2R\sin\frac{\pi}{n} \\ s & = \rho S \\ \beta_i & = \cos^{-1}\left(\cos\alpha_i \sin\frac{\pi}{n}\right) \\ l_i & = (R-r)\frac{1}{\cos\alpha_i} \\ b_i & = l_i \sin\beta_i \end{cases} \quad \text{for } i = 1, 2 \tag{2.2}$$

The origami pattern shown in Fig. 2.1c was constructed using the parameters defined in the Eq. 2.3. The vertices of the origami bendy straw pattern meet for flat foldability conditions only when $\beta_1 = \beta_2$, which is equivalent to $\alpha_1 = \alpha_2$. For the fully compressed and collapsed stable state to exist, $\alpha_1 \neq \alpha_2$ is therefore a necessary condition. Otherwise, compressing the origami bendy straw will flatten it instead of collapsing one of the frusta.

Note that the parameters in Eq. 2.3 are defined in such a way that, for a given set (S, s, n, β_1, β_2), the folded origami bendy straw will be similar to a circular bendy straw with the parameters (R, ρ, α_1, α_2). The parameters S and s are the side lengths of the outer and inner polygons, respectively, defining the pyramidal frusta. The perimeter of the outer polygon is given by $P = nS = n2R\sin\frac{\pi}{n}$. For sufficiently large n ($n \to \infty$), $\frac{\pi}{n}$ approaches 0. Which leads to the circumference of a circle with radius R as follows:

$$\begin{aligned} P &= 2nR\sin\left(\frac{\pi}{n}\right) \\ &= 2nR\left(\frac{\pi}{n}\right) \\ &= 2\pi R \end{aligned} \tag{2.3}$$

When $n \to \infty$: $\beta_1 = \beta_2 = \frac{\pi}{2}$, which is also expected to be the case, since all the folds become increasingly similar to vertical lines when at large values of n.

The behavior of a single origami bendy straw cell with the parameters ($R = 25mm$, $\rho = 0.4$, $n = 8$, $\alpha_1 = 45°$, $\alpha_2 = 35°$) was found to be similar to that of the original bendy straw. This behavior was tested both by using an Origami Simulator [24, 62] and by hand-folding an origami bendy straw using normal ($90g/m^2$) paper bi-laminated with a 42.5-micron plastic film, on which the pattern was engraved by means of a laser-cut machine (Trotec Speedy 100). Examples of both procedures are given in Fig. 2.3. The extremities of the cell were fixed to a rigid polygonal

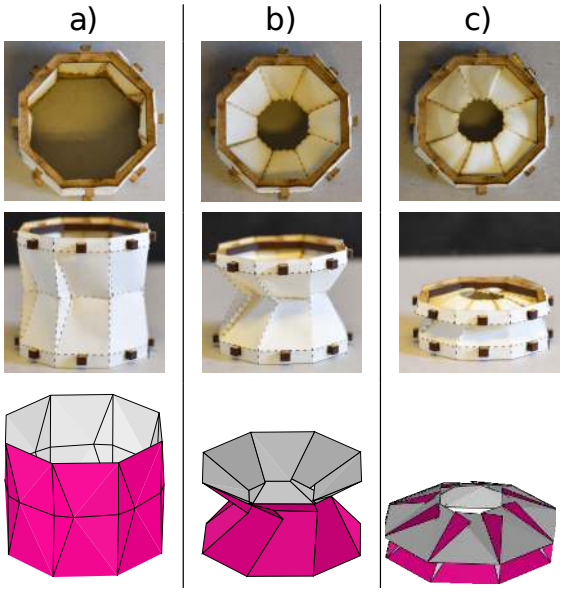


Figure 2.3.: Top view, side view and the simulated model for an origami bendy straw cell ($R = 25mm$, $\rho = 0.4$, $n = 8$, $\alpha_1 = 45°$, $\alpha_2 = 35°$). **a)** Completely deployed. **b)** Middle position. **c)** Completely compressed, with its smaller frustum reversed. The completely deployed and middle position images of the simulation were vectorized for this image. We did not vectorize the image of the completely compressed bendy straw because we note that the simulation breaks at the fully compressed state: some facets pierce through others.

support made of wood (keeping the boundaries rigid in order to keep the shape of the bendy straw). This structure was also found to be axially bistable, which indicates that the bi-stability of the origami bendy straw does not depend on the use of elastic material. The height in each state can be easily calculated. In the middle position, the height h_{middle} was defined as follows:

$$h_{\text{middle}} = h_1 + h_2 \tag{2.4}$$

In the case of axially bistable origami bendy straws, it was observed that the smallest frustum was fully reversed in the compressed position, as expected to occur in the case of the original bendy straw. This assumption does not always hold true, however, as can be seen from the insert in Fig. 2.4c, where only part of the frustum is turned inside out, but when it does hold, the compressed state $h_{\text{compressed}}$ can be calculated as follows:

$$h_{\text{compressed}} = |h_1 - h_2| \tag{2.5}$$

Contrary to what occurs with the original bendy straw, there is also a third fully unfolded state, the height of which was denoted h_{expanded} which can be defined as

follows:

$$h_{\text{expanded}} = b_1 + b_2 \tag{2.6}$$

The measurements of the heights of several origami bendy straws were compared with the expected heights given by the Eqs. 2.4, 2.5 and 2.6, as shown in the Fig. 2.4. Only small errors were observed between the predicted and measured heights of the origami bendy straws in the unfolded state (mean error $\mu = -0.61mm$, standard deviation $\sigma = 0.17mm$) and the middle state ($\mu = 2.15mm$, $\sigma = 2.29m$). The largest error was obtained on the compressed state ($\mu = 6.71mm$, $\sigma = 7.23mm$), especially with the tallest origami bendy straw (see Fig. 2.4c). The bigger error values for the origami bendy straws with $\alpha_1 = 60°$ and $\alpha_2 = 50°$ at the fully compressed state were due to the partial reversal of the smaller frusta. This was observed with the largest values of α_1 and α_2 (which are both greater than 50°), which resulted in larger measurement errors in the height of the structure in the compressed state.

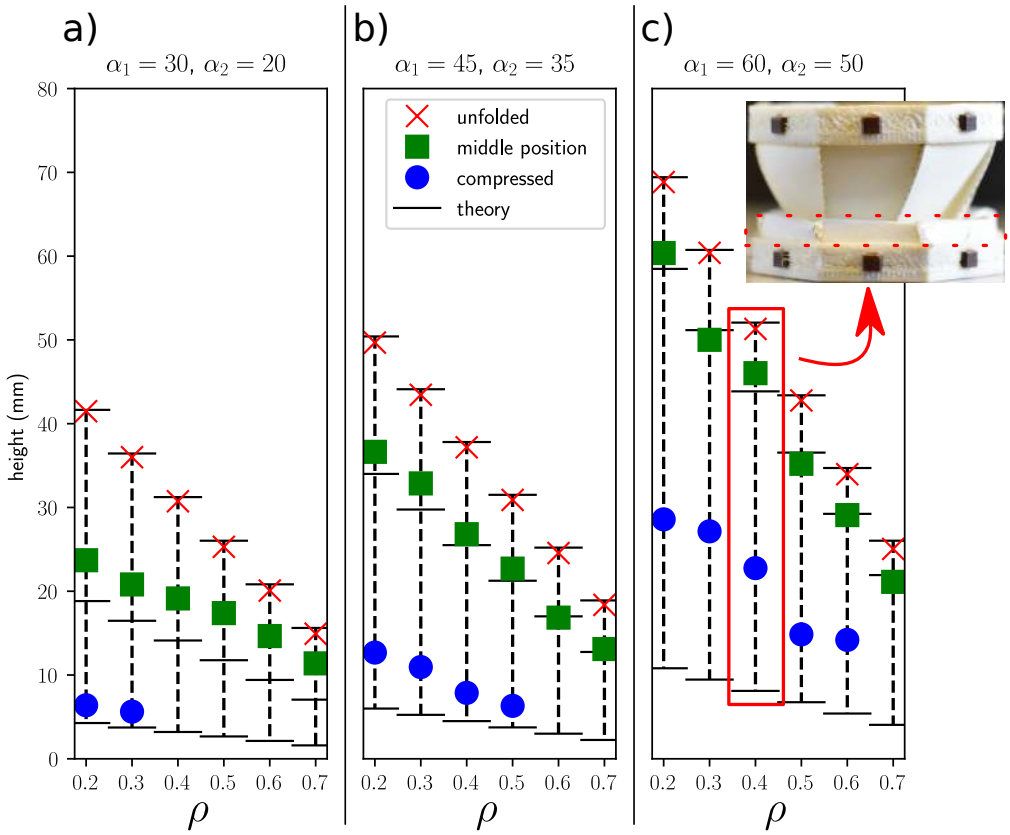


Figure 2.4.: Comparison between predicted and measured heights of various sets of origami bendy straws with $n = 8$, $R = 25mm$, $\Delta\alpha = 10°$ and ρ ranging from 0.2 to 0.7, having the following parameters: **a)** $\alpha_1 = 30°$, **b)** $\alpha_1 = 45°$ and **c)** $\alpha_1 = 60°$. It is worth noting that excessively small heights make it impossible for the compressed stable state to exist (the absence of some markers indicates that these origami bendy straws were not in a stably compressed state). We can also note the large difference between height measurements of the origami bendy straws in the compressed state with $\alpha_1 = 60°$ and $\alpha_2 = 50°$. This occurs because the smallest frustum is not completely inverted, as it can be seen in the insert.

2.3. Force study

A test bench was set up in order to analyze the forces exerted on the origami structure during its compression and expansion (Fig. 2.5). Figure 2.6 shows the

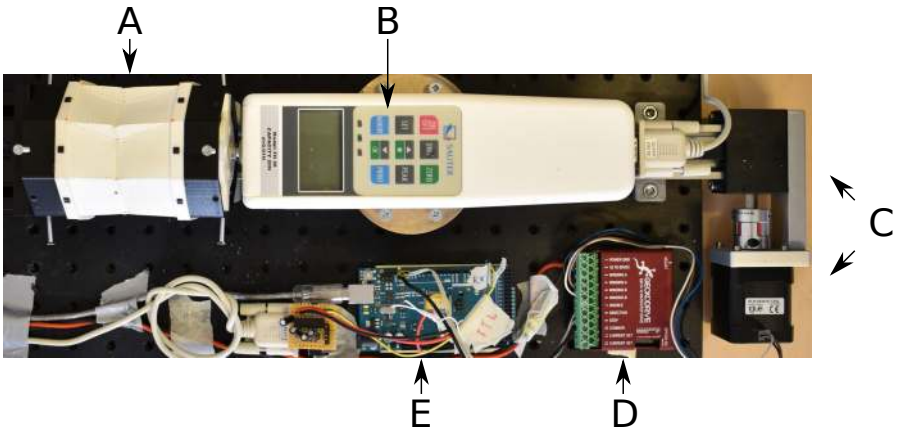


Figure 2.5.: The low-cost custom-made test bench used to measure the forces applied to the origami structure in (**A**). The force bench consisted of: a Sauter FH-20 digital force gauge (**B**), an Igus DLE-SA-0001 linear actuator (**C**) driven by a Gecko micro-step driver (**D**), and an Arduino Mega (**E**) controlling the motor driver and delivering the measurements from the force gauge to a computer. With this force bench, we expanded and compressed the origami structure in 0.027mm steps while measuring the forces with a resolution of 0.01N.

expansion/compression force curves recorded with several origami bendy straws and the heights predicted based on the equations (2.6), (2.4) and (2.5). Both curves had very similar shapes featuring a hysteresis, but upon analyzing the absolute values of the forces, the expansion and compression behaviors of the origami bendy straws were found to differ considerably.

Figures 2.4 and 2.6 were compared in order to determine why the origami bendy straws did not all adopt a stable, compressed state. The origami bendy straw cell with the parameters ($\rho = 0.7$, $n = 8$, $\alpha_1 = 60°$, $\alpha_2 = 50°$) shown in Fig. 2.4c did not reach a stable compressed state, and it can be seen from Fig. 2.6d that the first force peak recorded with all the similar origami bendy straw was always negative. The origami bendy straw with the parameters ($R = 25mm$, $\rho = 0.5$, $n = 8$, $\alpha_1 = 30°$, $\alpha_2 = 20°$) shown in Fig. 2.4 did not remain stable either in the compressed state. The largest version of the origami bendy straw, having the parameters $R = 40mm$, $\rho = 0.5$, $n = 8$, $\alpha_1 = 30°$, $\alpha_2 = 20°$), shown in Fig. 2.6b reached this state, but it was not very stable: simply touching the structure could make it "pop" into another more stable state. The Fig. 2.6b clearly shows that the force corresponding to the first peak remained positive only very briefly.

When a fully compressed origami bendy straw was expanded by hand beyond its first force peak, (between the first two red dots in Fig. 2.6), the origami bendy

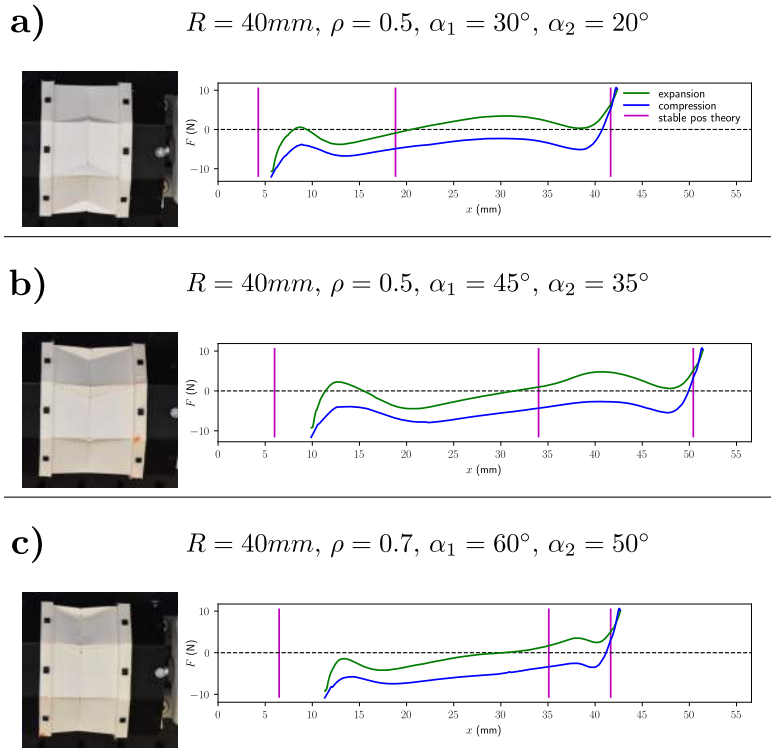


Figure 2.6.: Force curves corresponding to the expansion (green) and compression (blue) of various origami bendy straws. Purple lines give the theoretical heights of the structure in the stable position predicted by Eqs. 2.5, 2.4 and 2.6. The middle stable position predicted by Eq. 2.4 lay close to a point at which the force along the expansion curve was zero. The actual point of minimum energy located near the fully extended state was always less than that predicted by Eq. 2.6, and the force necessary to expand the structure any further increased sharply. The fully compressed height predicted by Eq. 2.5 was always the farthest away from the actual stable position, as was to be expected, since this equation does not take into account the non-zero thickness of the material.

straw quickly "snapped" into the next stable position, releasing the energy stored in the inverted frustum. We then kept on expanding the origami bendy straw further, but it did not show the same sudden response, which means that the origami bendy straw had to be actively expanded until it was fully deployed. Trying to expand the origami bendy straw even further would only have damaged the origami structure.

This behavior was not observed when compressing a fully expanded origami bendy straw structure: the fact that the force never changed its direction indicates that a constant compression force was required in order to compress the structure (the "snapping" movement that jumped it into the next state was weaker in this direction).

2.4. Multi-stability with pop-through defects

The origami bendy straw cell is axially stable, but does not naturally stay in a bent position. The original bendy straw is easily bistable, and can become multistable (having multiple equilibrium positions) when built-in stress is introduced into the structure, as described by [2] (a new "bent" state is added, so the bendy straw now has three different stable states, and thus becomes multistable). One way of putting the origami bendy straw into a new stable position is to change the folding of some creases. In order to introduce a similar bent position into the origami bendy straw, while avoiding the need to make them out of elastic or stretchable material, we studied the effects of inverting the folds of some creases.

A pop-through defect (PTD) [61] was used here: pressure was applied to some vertices of the structure until it inverted its folds, popping into a different mechanically stable state. By using a PTD to "program" the state of the material, some parts of the structure can be made more rigid. Adding a PTD to one side of an origami bendy straw (as in the Fig. 2.9) makes this side stiffer and keeps it in an unfolded position. Instead of being compressed, the origami bendy straw bends towards the opposite side. By choosing where to put the PTDs, more complex structures can be created (see the Fig. 2.2c).

Our radially oriented paper origami bendy straw structure was then installed on the force bench, and a force of about $1.5N$ was found to be required to either make the vertex pop into a PTD or reverse it back into its original position. In the Figures 2.9 a and b, comparison are made between the crease assignments observed before and after creating a PTD: the Fig. 2.9d shows the force produced during this process.

2.4.1. Bending angle

It was tricky to find an analytical expression for θ_{stable}, the angle of the stable bent position, but the maximum bending angle $\theta_{\max}$ could be calculated (as in the Fig. 2.7) as follows:

$$\cos(\theta_{\max}) = 1 - \frac{(b_1 + b_2)^2 - \Delta L^2}{2L(L + \Delta L)} \tag{2.7}$$

In which the exact expressions for both L and ΔL depend on whether n is an even or odd number. With an even-numbered n, the bending movement occurs from one side to the opposite side of the polygon. L is the diameter of the inscribed circle of the polygon, as shown in Figure 2.7 c. The radius of this inscribed circle is equal to the apothem of the polygon, which is $A = R\cos\frac{\pi}{n}$. In addition, the facets

of height b_1 and b_2 touch when fully bent, giving the following expression for ΔL:

$$\begin{cases} \Delta L &= |b_1 - b_2| \\ L &= 2R\cos(\pi/n) \end{cases} \tag{2.8}$$

With an odd-numbered n, the bending angle occurs from one side of the polygon to the opposite vertex. L is the sum of the radius of the inscribed circle (A) and the radius of the circumscribed circle (R), as shown in Figure 2.7 d. Moreover, this time, the lines of length l_1 and l_2 are touching, giving a different expression for ΔL:

$$\begin{cases} \Delta L &= |l_1 - l_2| \\ L &= R(1 + \cos(\pi/n)) \end{cases} \tag{2.9}$$

It is worth noting that the Eq. 2.7 is always independent of R (both the numerator and the denominator are proportional to R^2). In addition, ΔL depends on $\alpha_1 - \alpha_2$. As this difference becomes smaller, ΔL becomes negligible.

The Fig. 2.8 shows the measured stable and maximum θ recorded in the case of several origami bendy straw cells. The measurements of the $\theta_{\max}$ matched the predictions quite closely. In addition, the relationship between θ_{stable} and $\theta_{\max}$ was found to depend on α_1 and α_2. With the samples made of bi-laminated paper, we observed, as shown in Fig. 2.8, that with $\alpha_1 = 45°$ and $\alpha_2 = 35°$, we have $\theta_{\max} \approx 2\,\theta_{\text{stable}}$. To design a mechanism using a different material (with different mechanical properties, thickness, etc.), a similar study can be performed in order to obtain the exact parameters required.

2.4.2. Using two equal frusta

In the case of applications that do not require the compressed state, we can set $\alpha = \alpha_1 = \alpha_2$. The origami bendy straw will not have a stable compressed state in this case, since instead of having an inverted frustum, the origami bendy straw cell will become flat-foldable. In this case, $\Delta L = 0$ in both the even and odd cases, and it can be established that:

$$\sin(\theta_{\max}/2) = \frac{1-\rho}{2\cos(\pi/n)}\sqrt{\tan^2(\alpha) + \cos^2(\pi/n)}, \text{ for even } n \tag{2.10}$$

$$\sin(\theta_{\max}/2) = \frac{1-\rho}{1+\cos(\pi/n)}\sqrt{\tan^2(\alpha) + \cos^2(\pi/n)}, \text{ for odd } n \tag{2.11}$$

This equation was used to find the ρ required to obtain a given set of α, θ and n (Figure 2.7 e, f).

To study the force required to bend an origami bendy straw cell, we built a mechanical support consisting of a sliding joint and a ball joint. This support was

connected to the tip of the force gauge and to one of the ends of an origami bendy straw, and the latter was then installed $2.5cm$ to the right so that the force was applied off-center. While measuring the force, a camera mounted on top of the test bench recorded the whole test. This video was then analyzed with Physlets Tracker [67] in order to measure the bending angle of the origami bendy straw (Fig. 2.9). The stable position was found to match one of the zeroes of the expansion curve, and the maximum bending angle was found to be $35°$, as predicted by the Eq. 2.7.

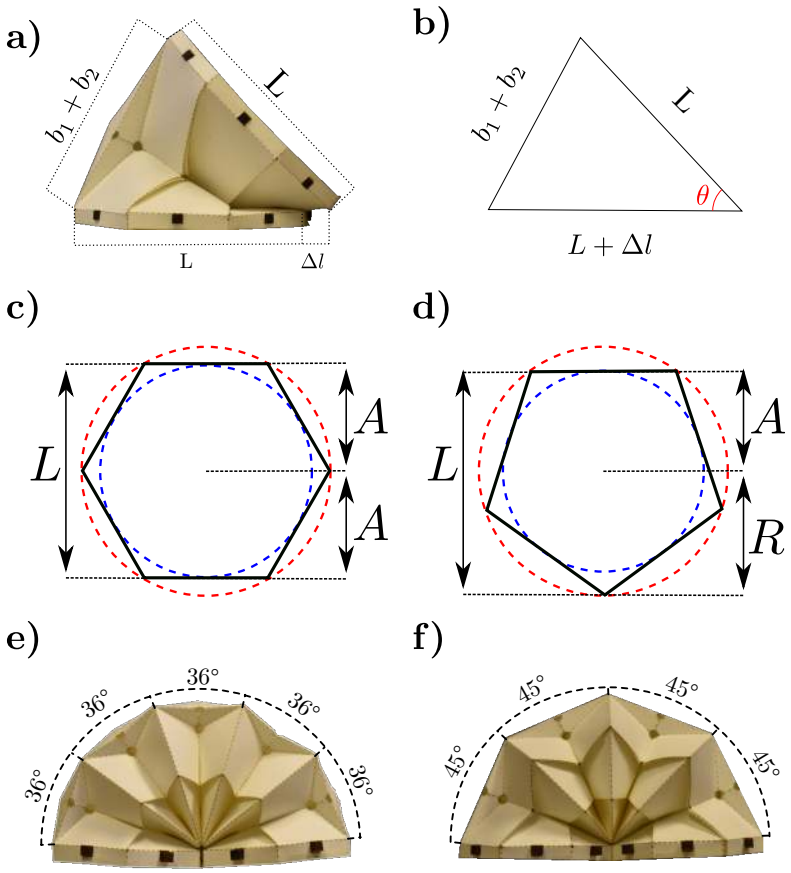


Figure 2.7.: Maximum banding angle of origami bendy straw. **a)** Origami bendy straw with the parameters $R = 40mm$, $\rho = 0.4$, $\alpha_1 = 50°$ and $\alpha_2 = 35°$ forced to reach the maximum bending angle. **b)** Simplified geometry of the maximum bending angle. **c)** An even-numbered polygon, from which we can deduce that $L = 2A = 2R\cos(\pi/n)$ (A is the apothem of the polygon, which is the equal to the radius of the inscribed circle, drawn here in blue). **d)** An odd-numbered polygon, from which we can derive that $L = R + A = R\,(1 + cos(\pi/n))$. **e-f)** Two different origami bendy straws produced with $R = 25mm$, showing how many cells are required to obtain a 180° bent origami bendy straw. **e)** $n = 6$, $\rho = 0.4558$, $\alpha_1 = \alpha_2 = 25°$, producing cells with $\theta = 36°$. **f)** $n = 7$, $\rho = 0.3624$, $\alpha_1 = \alpha_2 = 35°$, producing cells with $\theta_{\max} = 45°$.

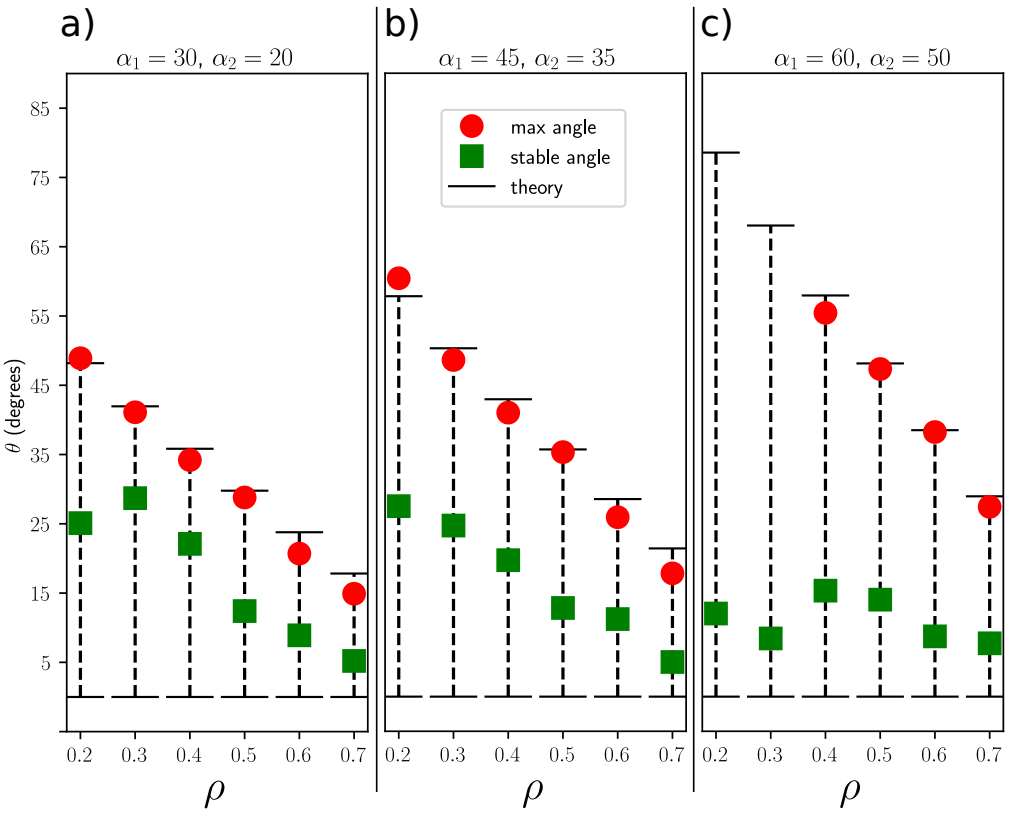


Figure 2.8.: Measured stable and maximum bending angles in various sets of origami bendy straws with $n = 8$, $R = 25mm$, $\Delta\alpha = 10°$ and ρ ranging from 0.2 to 0.7, with the following parameters: **a)** $\alpha_1 = 30°$, **b)** $\alpha_1 = 45°$ and **c)** $\alpha_1 = 60°$. Note that in this case ($\alpha_1 = 60°$ and $\alpha_2 = 50°$), the two bendy straws with $\rho = 0.2$ and $\rho = 0.3$ could not be bent all the way to the theoretical max bending angle without tearing it.

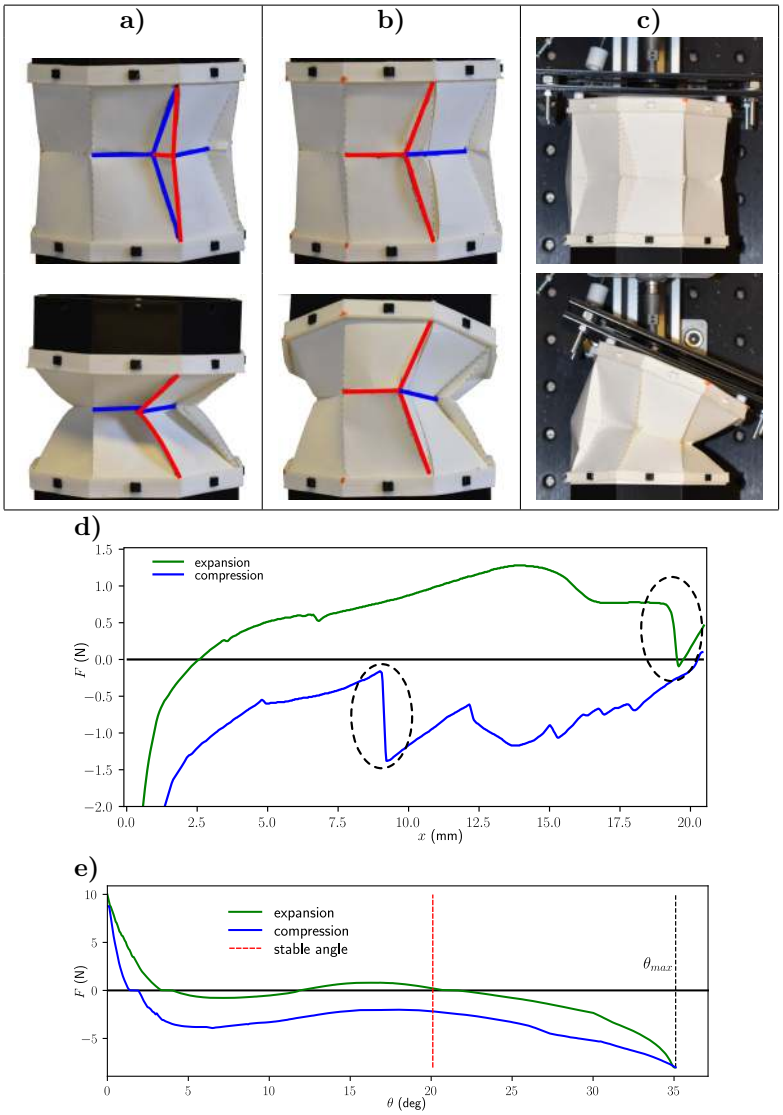


Figure 2.9.: Force test of origami bendy straw during bending movement. **a)** Original origami bendy straw before PTD. **b)** Same origami bendy straw after creating a PTD. **c)** Use of the test bench with the custom-made sliding mechanical support, allowing the origami structure to bend while the force is being measured. **d)** Force necessary to compress/expand the vertex of the laminated paper origami bendy straw and to pop it from one state into another. The dashed ellipses on the image indicate the moment when the vertex popped into a different state, showing how the force suddenly changed at that moment. **e)** Force curves measured while the origami bendy straw was performing a bending motion: expansion (in green) and compression/bending (in blue). We noted that the stable bending angle θ_{stable} measured (about 20°), which is indicated by the dotted red line, was close to the angle at which the force measured on the expansion curve was equal to zero. Parameters: $R = 40mm$, $\rho = 0.4$, $\alpha_1 = 45°$ and $\alpha_2 = 35°$.

2.5. Conclusion

Non-rigid-foldable origami patterns are rather complicated to analyze, but they also promise to have some useful properties that are worth exploring. The origami pattern presented here, which replicates the bendy straw structure, could be used for many robotic applications. An origami bendy straw in which one of the sides of each cell is rigidified with a PTD could be used for example to design robotic arms [30]. Since a structure in this configuration naturally bends to the side opposite to the PTD, it could be easily actuated, for example, using shape-memory alloy [31]. The possibility of easily constructing several bendy straw structures is liable to be useful for prototyping future applications, since the large number of parameters involved makes it difficult to predict all the properties of these structures without testing them. The present origami version of the bendy straw takes much less time to produce than large numbers of 3D printed bendy straws or a single large bendy straw. In addition, PTDs provide a useful means of putting an origami bendy straw structure in a new bent state that would otherwise not be stable. Thanks to the use of PTDs, multi-stability and adjustable stiffness can be obtained much more easily than with the built-in stress method based on the use of TPU material to make soft bendy straws. In future studies, it is planned to investigate how PTDs may serve to develop actuators such as shape-memory alloy actuators embedded in a structure, making it possible to change its state dynamically as required.

Acknowledgments

We would like to thank J. Diperi and J-M Ingargiola for their technical assistance. This research was supported by CNRS, Aix-Marseille University and the French National Research Agency (ANR) via the OrigaBot project (ANR-18-CE33-0008-01).

Part II.

Flight, rotation and the monorotor project

3. Introduction

Unmanned Aerial Vehicles (UAVs) are becoming ubiquitous, with their possible uses ranging from professional, to personal or even rescue applications [1]. Fully actuated multi-copters (and specially, quadcopters) are by far the most common type of UAVs: they are cheap to build, mechanically simple, easily controllable and the use of multiple rotors allows the thrust force and the torques to vary within a broad range.

Nevertheless, different UAV configurations exist with different sets of advantages and disadvantages. The Gemini [51] uses a bi-rotor configuration based on tandem helicopters that is arguably more energetically efficient and easier to scale while maintaining the same width for indoors applications. However, Gemini features a relatively important mechanical complexity due to the use two additional servo-motors in order to produce a roll motion. The Agile [52] is a quadcopter that reduces its width temporarily by folding itself, maintaining at most times the easy controllability of a quadcopter, at the cost of one servo motor and the need of producing aggressive movements while folded.

Highly underactuated UAVs using a single rotor are also of great interest: they can be easier to miniaturize, like the Piccolissimo [48], and even be designed in such a way that an active control is not needed for achieving stable hovering flight [47]. An interesting subclass of these monorotor drones are mono-wing drones, like the ones proposed in [74, 73]. These are usually inspired by the flight of the samara winged fruit/seed [41]. Moreover, their power efficiency is also of great interest [25].

In [38] the authors also studied solutions for a relaxed definition of hover for different configurations of motors pointing in the same direction. The same authors also studied the related problem of controlling a quadcopter under the failure of one or several motors [39]. In [78] and [79] an example of how to maintain position control using a single motor is shown.

There are also different efforts for reducing the mechanical complexity while maintaining full controllability. For example, [66] studies the design of a gyroscopically controlled micro air vehicle. Another elegant effort to achieve the same controllability is cyclic flapping by torque modulation [45, 46]: By attaching the tip of each blade of the rotor on a specialized kind of passive hinge assembly, and then cyclically accelerating and decelerating the rotation speed of the rotor, a vector thrust control is achieved without the need of additional servomotors or complicated swashplate

mechanisms. A coaxial bi-rotor UAV based on this technology was demonstrated [44], and more recently, the Gemini-II [50], a follow-up to [51] that replaces both servomotors by a swashplateless cyclical system was also demonstrated successfully.

Most of the work in this study aims at developing a monospinner, i.e., a monorotor UAV, capable of maintaining both reduced attitude and position control with a single rotor. The following chapters in this work analyze different aspects of the monospinner project:[1]

- Chapter 4 describes the swashplateless mechanisms and documents the process of building it for the OrigaBot project.
- Chapter 5 describes the different parts of the monospinner vehicle, its general working principles and properties.
- Chapter 6 develops the Equations of Motion used for the simulation and describes the methodology used to derive them by means of the Euler-Poincaré equation, a special form of the Euler-Lagrange equation that can be used when the space of configurations is a member of a Lie group [49, 76] of transformations.
- Chapter 7 a novel kind of quaternion decomposition intro uncontrollable spin and controllable attitude components.
- Chapter 8 develops a non-linear family of control laws using the decomposition developed in Chapter 7.
- Chapter 9 develops a closed-loop simulation of the complete monospinner system submitted to disturbances.

As an extra, Chapter 10 describes a new formula for the direct conversion of a quaternion variable to a set of Euler angles in any sequence.

Note that, in this work, we will not present any experimental results with a flying monorotor: we focus on doing a thorough theoretical analysis of the problem of modeling and controlling this proposed monorotor UAV.

[1] Chapters 5, 6, 7, 8 and 9 have been submitted for publication in [4].

Nomenclature

$\boldsymbol{e}_x$

Unit vector aligned with the x-axis. 188

$\boldsymbol{e}_y$

Unit vector aligned with the y-axis. 188

$\boldsymbol{e}_z$

Unit vector aligned with the z-axis. 188

f_p

Thrust force from rotor. 84

τ_p

Thrust torque from rotor. 84

$\boldsymbol{B}_\tau$

Matrix modeling the relationship between thrust force and total torque. 107, 235

k_f

Constant ratio between thrust force from rotor and the square of the rotor angular speed. 70, 84

k_τ

Constant ratio between thrust torque from rotor and the square of the rotor angular speed. 70, 84

k

Ratio between k_τ and k_f. 84

γ

Instantaneous angular position (revolution) of rotor blade. 71, 78, 99

$\dot{\gamma}$

Derivative of γ, angular velocity of the rotor around the z-axis in its own frame. 100

β

Precession angle of rotor blade, and phase for swashplateless system. 78, 99

α

Nutation angle of rotor blade, related to the amplitude voltage for the swashplateless system. 99, 100

V

Constant voltage component, in Volts. 78

$\tilde{V}$

Sinusoidal voltage term, in Volts. 78

$\tilde{V}_a$

Input polar amplitude, in Volts. 78

$\mathcal{B}_b$

Main drone body. 71, 96

$\mathcal{B}_s$

Battery body. 71, 96

$\mathcal{B}_p$

Rotor system body. 71, 96

$\mathcal{F}_S$

Frame attached to the battery body $\mathcal{B}_s$. 96

$\mathcal{F}_B$

Frame attached to main drone body $\mathcal{B}_b$. 96

$\mathcal{F}_P$

Virtual frame attached to rotor system body $\mathcal{B}_p$. 96

$\mathcal{F}_E$

Inertial frame. 98

h_S

Distance between centers of mass of main body and battery. 96

h_P

Distance between centers of mass of main body and rotor. 99

m_b

Mass of main body. 96

m_s

Mass of battery. 96

m_p

Mass of rotor. 96

m

Total mass of drone. 103

Δm

Quantity modeling the pendulum-like effect of the masses away from the center of mass of the main body: $\Delta m = m_s h_S - m_p h_P$. It's important to note that $m_s h_S \gg m_p h_P$. 104

J_b^B

Inertia matrix of main body in body frame. 96

J_s^S

Inertia matrix of battery in battery frame. 96

J_p^P

Inertia matrix of rotor in virtual rotor frame. 96

J_p^B

Inertia matrix of rotor in body frame. 102

$\boldsymbol{J}$

Total inertia matrix body frame in body frame. 103

R_B^E

Rotation matrix from main body's frame of reference to inertial frame, also written as R for simplicity. 73, 98, 116

R

Simplified notation for R_B^E. 72, 116

R_P^B

Rotation matrix from rotor's virtual frame of reference to main body's frame of reference. 99

g_B^E

Full transformation from main body's frame of reference to inertial frame, also written g for simplicity. 73, 98, 116

g

Simplified notation for g_B^E. 73, 116

g_P^B

Full transformation from rotor's frame of reference to main body's frame of reference. 99

q_B^E

Quaternion representing rotation from main body's frame of reference to inertial frame, also written as q for simplicity. 73, 116

q

Simplified notation for q_B^E. 73, 116

q_a

Reduced attitude quaternion. 120

q_s

Spin quaternion. 120

θ_s

Spin angle. 124

$\boldsymbol{\omega}_a$

Angular velocity of reduced attitude quaternion. 120

$\boldsymbol{\omega}_s$

Angular velocity of spin quaternion. 120

$\boldsymbol{s}_B^E$

Position of main body's frame of reference in the inertial frame, also written as $\boldsymbol{s}$ for simplicity. 74, 98, 116

$\boldsymbol{s}$

Simplified notation for $\boldsymbol{s}_B^E$. 74, 116

$\boldsymbol{\Omega}_{P/B}^P$

Full angular velocity between the body and the propeller in the propeller frame, also written as $\boldsymbol{\Omega}$ for simplicity. 74, 100, 111

$\boldsymbol{\Omega}$

Simplified notation for $\boldsymbol{\Omega}_{P/B}^P$. 74, 103, 111

$\boldsymbol{\omega}_{P/B}^P$

Angular velocity between of the virtual rotor frame w.r.t. the main body frame, usually considered $\approx \mathbf{0}$. 100

$\boldsymbol{\omega}_B^B$

Total angular velocity of the main body frame of reference, also written as $\boldsymbol{\omega}$ for simplicity. 74, 98, 101, 111

$\boldsymbol{\omega}$

Simplified notation for $\boldsymbol{\omega}_B^B$. 74, 111

$\boldsymbol{v}_B^B$

Total linear velocity of the main body frame of reference, also written as $\boldsymbol{v}$ for simplicity. 74, 98, 101, 111

$\boldsymbol{v}$

Simplified notation for $\boldsymbol{v}_B^B$. 74, 111

η_B^B

Inertial twist of the body: $\eta_B^B = \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix}$, also written as η for simplicity. 74, 103

η

Simplified notation for η_B^B: $\eta = \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix}$. 74, 103

$\boldsymbol{\Omega}_P^P$

Total angular velocity of the virtual rotor frame of reference w.r.t. the inertial frame. 100

$\boldsymbol{v}_P^P$

Total linear velocity of the virtual rotor frame of reference w.r.t. the inertial frame. 100

$\boldsymbol{P}$

Nonlinear attitude controller term. 136

$\boldsymbol{K}_p$

Proportional term of a PID controller. 137

$\boldsymbol{K}_d$

Differential term of a PID controller. 137

$\boldsymbol{K}_i$

Integral term of a PID controller. 137

Ad

Adjoint representation of a Lie group. 101

ad

Adjoint representation of a Lie algebra. 112

$\boldsymbol{r}$

Thrust direction vector. 105

T

Total kinetic energy. 103

$F_{\textbf{ext}}^B$

Sum of external forces acting on the main body. 104

$\boldsymbol{n}_b$

Normal vector. 76, 132

$\boldsymbol{m}$

Middle (half-way) normal vector. 76, 122, 133

$\boldsymbol{\omega}_r$

Desired angular velocity. 137

$\bar{q}$

Desired orientation quaternion. 134

$\bar{\boldsymbol{n}}$

Desired normal vector $\boldsymbol{n}_b$. 134

$\bar{\boldsymbol{m}}$

Desired middle normal vector $\boldsymbol{m}$. 134

$\boldsymbol{\tau}_c$

Attitude correction torque. 137

$\boldsymbol{f}_c$

Attitude correction force. 137

ψ

Spin angle estimation error. 150

4. Motor and vector control with swashplateless system

Reducing the number of motors in a flying drone can drastically reduce the vehicle's controllability, and a monorotor is an extreme example of that. In order to maintain some level of controllability while still using a single motor, a specialized swashplateless system based on [45, 46, 44] was used in order to help us achieve vector control of the thrust vector.

This type of swashplateless mechanisms can be very efficient in simplifying the mechanical complexity of already existing systems. A good example of that can be seen in the works of Qin: [51] describes the Gemini I, a bi-copter that relies on a central servomotor system to tilt both rotors in order to have full controllability. On the other hand, [50] describes the Gemini II: a revised bi-copter drone published by the same team, in which they greatly simplified the construction of their drone by using swashplateless mechanisms and removing the need of a servomotor to tilt the motors.

In this chapter, we will describe the steps done to reproduce this swashplateless system for the OrigaBot project, which was a crucial part of the monorotor project.

4.1. Working principle

In order for a pilot to be able to control the flight of a helicopter, their commands must be translated into movements on the rotor blades. The swashplate mechanism is a crucial mechanical device that enables this control of the helicopter. This device is a rather complicated mechanical assembly, consisting of many pieces (see Fig. 4.1). But when properly set up, it allows the pilot to adjust the radial orientation of one or more blades to create various pitching and rolling movements.

In contrast with traditional helicopter design, most small indoor UAVs are quadrotors. These drones use multiple identical rotors to create both thrust and torques necessary for position and attitude control. The OrigaBot project was mainly interested in developing new drone designs that take advantage of the shape-shifting capabilities of origami structures and their easy fabrication in order to lower both the number of actuators (using fewer rotors than a conventional quadrotor) and

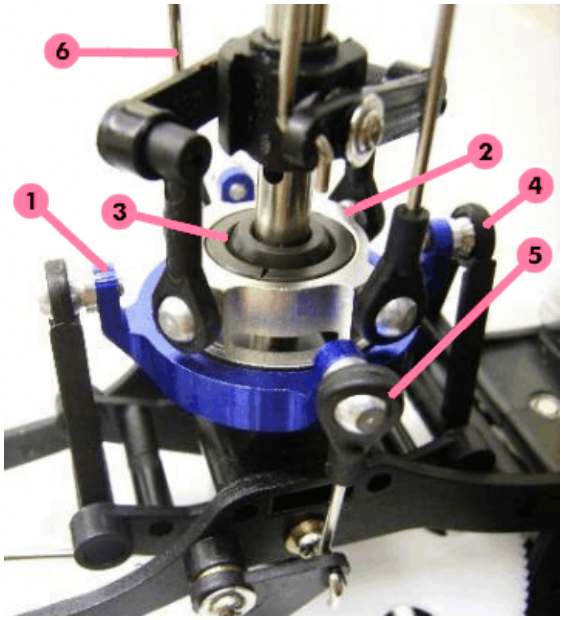


Figure 4.1.: Swashplate on a radio-controlled helicopter. 1 Non-rotating outer ring (blue); 2 Turning inner ring (silver); 3 Ball joint; 4 Control (pitch) preventing turning of outer ring; 5 Control (roll); 6 Linkages (silver) to the rotor blade. Source: Wikipedia.

complex mechanisms (like the swashplate). This is the reasons why a big part of my work was focused on the reproduction of a swashplateless mechanism as first published by James Paulos and Mark Yim in [45].

This mechanism works by replacing all the complicated mechanical assemblies of a swashplate (and their respective actuators) and replacing them by a simple passive hinge assembly, as seen in Fig. 4.2. The key to make this work is a specialized motor control: instead of controlling the rotor's rotation to a constant speed, a sinusoidal term is added to it, with a frequency equal to 1/rev. As seen in [44], the sinusoidal voltage $\tilde{V}$ term added to the rotor controller can be expressed as:

$$\tilde{V} = \tilde{V}_a \cos(\gamma - \beta - \beta_0) \tag{4.1}$$

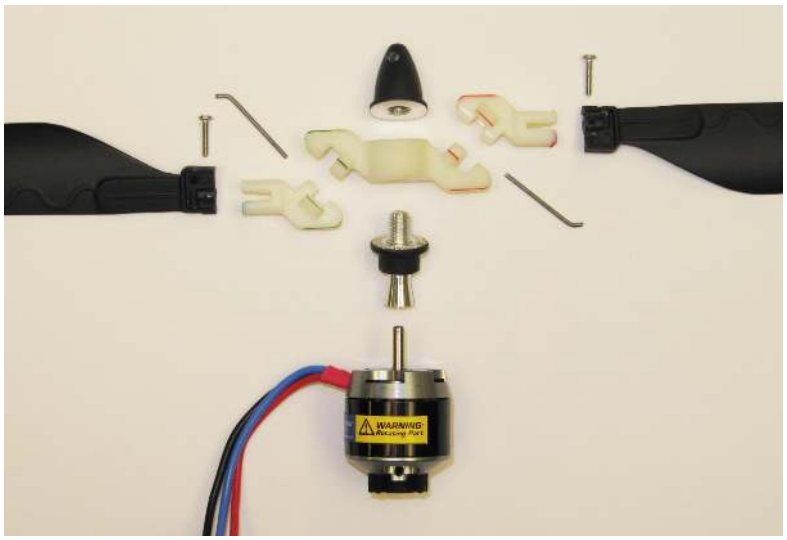
And the total voltage is:

$$V_{\text{total}} = V + \tilde{V} \tag{4.2}$$

Where V is the constant voltage component, $\tilde{V}_a$ is the input polar amplitude, γ is the instantaneous rotor angle, β is the derived input phase and β_0 is a constant angle accounting for the orientation of the motor with respect to the drone. We can clearly note from Eq. 4.1 that the exact position of the rotor must be measured, which is achieved with a magnetic encoder.

By having an oscillating rotation speed for the motor, the inertia of the blades will

a)



b)

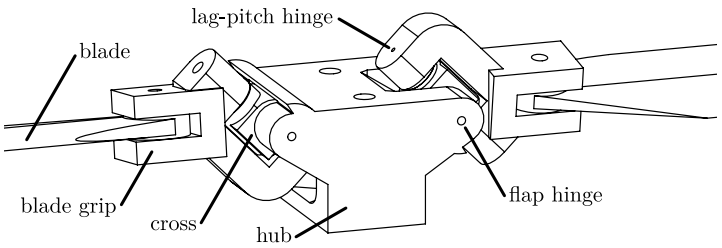


Figure 4.2.: **a)** Hinged propeller components for swashplateless system. Source: [45]. **b)** Schematic showing the hinges. Adapted from [46].

naturally actuate the lag-pitch hinge (as shown in Fig. 4.2**b**): each blade will go to a maximum of inclination in one direction when $\gamma - \beta - \beta_0 = 0$ and again in the opposite direction when $\gamma - \beta - \beta_0 = \pi$. The lag-pitch is inclined, thus creating the flapping motion as desired. Note that the flap hinges also have an important job: to relieve the kinematic constraints on the assembly and allowing for a smoother sinusoidal flapping motion [46].

4.2. Reproduction of full rotor system

To reproduce a fully working swashplateless system, two distinct parts must be manufactured:

1. The motor, with its controller and necessary encoders.
2. The swashplateless assembly.

Instead of reproducing the motor with its necessary magnetic encoder and controller, we instead ordered a set of motors from Vertiq[1]. The company, started by the researchers behind the original swashplateless publications, sell motors with everything already integrated, including an easy-to-use firmware and the necessary connections for UART messaging with an autopilot. This **greatly** simplified the complexity of the project, making it possible for us to spend more time working on other aspects of the drone. Figure 4.3 shows both motor modules used during the project. First, we used a F20 II motor with added custom controller and firmware

Figure 4.3.: Motors provided by Vertiq (former IQ-Motion). On the left, a F 20 II motor with a custom magnetic encoder, controller, and firmware installed by Vertiq. On the right, a Vertiq module.

from Vertiq, smaller than their regular products (on the left in Fig. 4.3). Then we decided to stick to their slightly bigger regular module (on the right in Fig. 4.3).

The swashplateless assembly, however, was not yet available to buy from Vertiq[2], so we had to make our own. The reproduction of this device was one of our main tasks. The first versions built for the product were made by Yahia Mallem during his internship at the ISM Biorobotics team in 2019. As seen in Figure 4.4, these first versions were rather small (30mm total length), and were created using a Formlabs Form 2[3] resin printer.

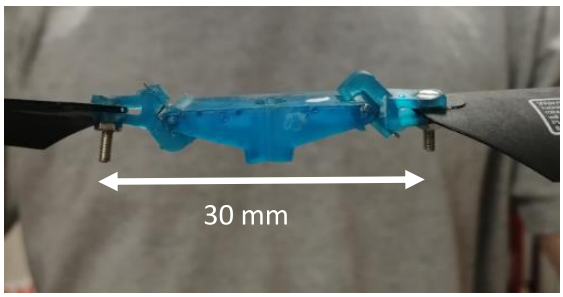


Figure 4.4.: First swashplateless assemblies produced at ISM Biorobotics.

[1]Formerly known as IQ-Control: https://www.vertiq.co
[2]According to their website (last accessed on November 23rd 2022), a first commercial version is planned to be released in Q1 2023.
[3]https://formlabs.com/

These first versions were too fragile to be used in an actual drone, even a small one: they only lasted for some minutes of testing. But they were resistant enough to prove, during Mr. Mallem's master thesis, that this system worked as expected. Many new and different swashplateless devices were 3D printed and assembled after Mr Mallem's work was done, mostly using more standard filament printers. Some of them can be seen on Fig. 4.5. A key difference we adopted is to follow the design used in [44]: instead of two separate flap hinges, we use a single teetering hinge. We assembled a simple gimbal system for preliminary experiments with

Figure 4.5.: Some 3D printed swashplateless assemblies. The one on the left was printed on an Ultimaker S5 using ABS plastic. The assembly on the right was printed on a Volumic MKII printer, using PLA plastic.

these swashplateless assemblies (see Fig. 4.6).

These new swashplateless assemblies needed to be bigger and more resistant, in order to assess the problem of longevity of the first smaller versions. When the decision to make a monorotor drone was made, it was clear that the pieces would have to be even tougher: a single rotor would have to be able to produce enough thrust for the flying vehicle, which is only possible by greatly increasing the rotor mean velocity. To illustrate this, supposing the rotor follows the usual thrust model:

$$f_p = k_f |\mathbf{\Omega}|^2 \tag{4.3}$$

And supposing the hover condition:

$$\sum f_p = mg \tag{4.4}$$

Then, with two motors, supposing $m = 0.4kg$, $g = 9.8m/s^2$ and assuming a measured value of $k_f = 2{,}48 \cdot 10^{-5} N/(\text{rad}/s)$:

$$\begin{aligned} 2f_p &= mg \\ 2k_f|\mathbf{\Omega}|^2 &= mg \\ |\mathbf{\Omega}| &= \sqrt{\frac{mg}{2k_f}} \\ |\mathbf{\Omega}| &\approx 281\text{rad}/s \end{aligned} \tag{4.5}$$

a)

b)

Figure 4.6.: First gimbal system used to test the swashplateless device. **a)** Unloaded. **b)** Loaded with a Pixhawk Mini 4 card controlling the Vertiq module.

While with a single motor:

$$\begin{aligned} f_p &= mg \\ k_f|\mathbf{\Omega}|^2 &= mg \\ |\mathbf{\Omega}| &= \sqrt{\frac{mg}{k_f}} \\ |\mathbf{\Omega}| &\approx 397\text{rad}/s \end{aligned} \tag{4.6}$$

This larger velocity greatly affects the longevity of the swashplateless device. After some months of testing, we decided that it would be easier and safer to have the new pieces made of aluminum instead of plastic. On Fig 4.7 we can see some swashplateless devices made of metallic parts built by our colleges at the ICube

lab in Strasbourg[4]. These swashplateless parts were very resistant: even after 4 months of intensive tests, they were still holding, albeit with clear marks of use around the axes[5].

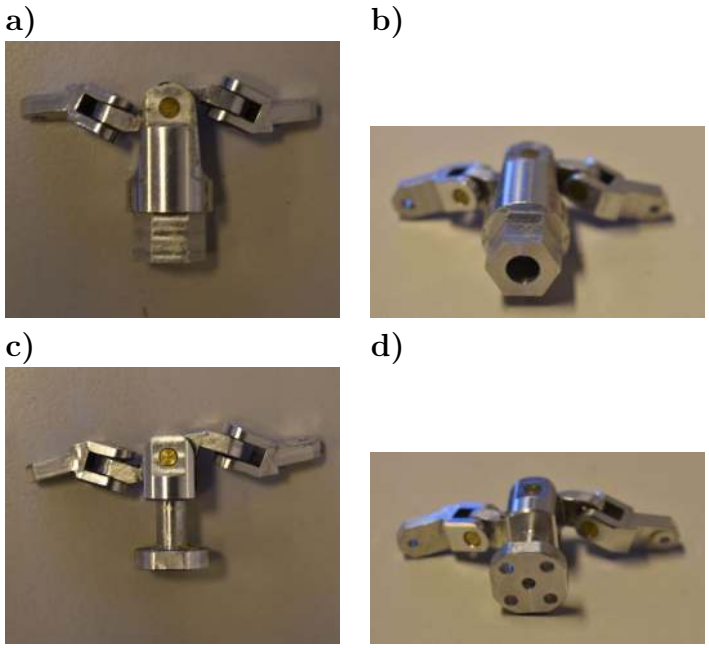


Figure 4.7.: Aluminum swashplateless devices made and assembled at ICube. **a)** First version, side view. **b)** First version, bottom view. **c)** Second version, side view. **d)** Second version, bottom view (note the use of 4 small screws).

A last problem with these pieces from Fig. 4.7 **a** and **b** is that they are mounted on the motors using a single central screw. This has one big disadvantage: every time the piece is mounted on the motor, the exact angle with respect to the motor's stator must be calculated. Otherwise, it is impossible to correctly produce thrust and torques in the required direction. In technical terms, we need to recalculate the parameter β_0 as shown in Eq. 4.1. I have tried, at first, to create a software solution[6] to this, by using the internal IMU's rotation estimation to calculate it. A much better solution to this was to ask for our ICube colleges a new modified piece which is mounted with 4 screws instead. This greatly simplifies the problem, and the only possible values for the angle are: $\beta_0 \in [0°, 90°, 180°, 270°]$. This new modified piece can be seen in Fig. 4.7 **c** and **d**.

[4]In particular, I would like to thank Mr Léo Wurtz for his precious help with this task
[5]Our axes were simple thin aluminum rods. A real ball joint could be used for more longevity.
[6]And it was a frustratingly bad idea.

4.3. Motor characterization

The firmware provided by Vertiq/IQ Motion Control can be controlled with any of its 4 available APIs: Python, MATLAB, Arduino, and C++[7]. In order to simplify the future integration with the drone, I decided to make all of my tests using the C++ API[8]. Details concerning the integration with the autopilot can be read on Appendix E. Both the rotor's thrust and torque are usually modeled with a square relationship on the rotor mean rotation speed:

$$\begin{aligned} f_p &\approx k_f |\mathbf{\Omega}|^2 \\ \tau_p &\approx k_\tau |\mathbf{\Omega}|^2 \end{aligned} \tag{4.7}$$

Where k_f and k_τ are constants. We also define their ratio k:

$$k = \frac{k_\tau}{k_f} \tag{4.8}$$

In order to estimate the parameters f_p and τ_p, which mostly depend on the wings, we used a Series 1580 Test Stand from Tyto Robotics. This particular test bench can "*Measure up to 5 kgf of thrust and 2 Nm of torque...*", which was more than enough for our needs. Figure 4.8 shows one of our Vertiq modules mounted on our Series 1580 Test Stand. Figure 4.9 shows the curves for both the thrust force and torque produced by our motor and blades.

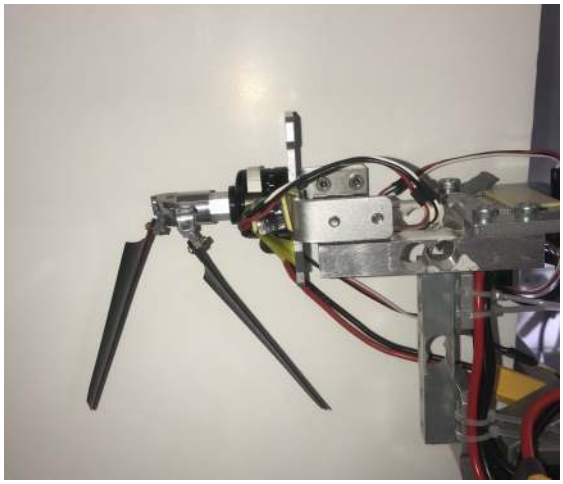


Figure 4.8.: Vertiq module on Tyto Robotics Series 1580 Test Stand.

A last useful characterization made with the test stand was the voltage / speed ratio. The Vertiq firmware accepts both a required mean rotation speed $|\mathbf{\Omega}|$ or the

[7]The C++ API can be found here: https://github.com/iq-motion-control/iq-module-communication-cpp

[8]The code I used can be found here: https://github.com/evbernardes/IQ_Motor_CPP/

mean voltage V as an input. For consistency with the pulsating voltage $\tilde{V}$, we decided to use the voltage mode. We compared both values using an optical probe also provided by Tyto Robotics[9]. Figure 4.10 shows the curves. We notice that the relationship is linear:

$$|\mathbf{\Omega}| \approx c \cdot V \tag{4.9}$$

For our particular system (rotor + blades), the coefficient is: $c \approx 134\,(\mathrm{rad}/s)/V$. In comparison, the same test with the unloaded Vertiq module (no swashplateless device or blades) gives $c \approx 202\,(\mathrm{rad}/s)/V$.

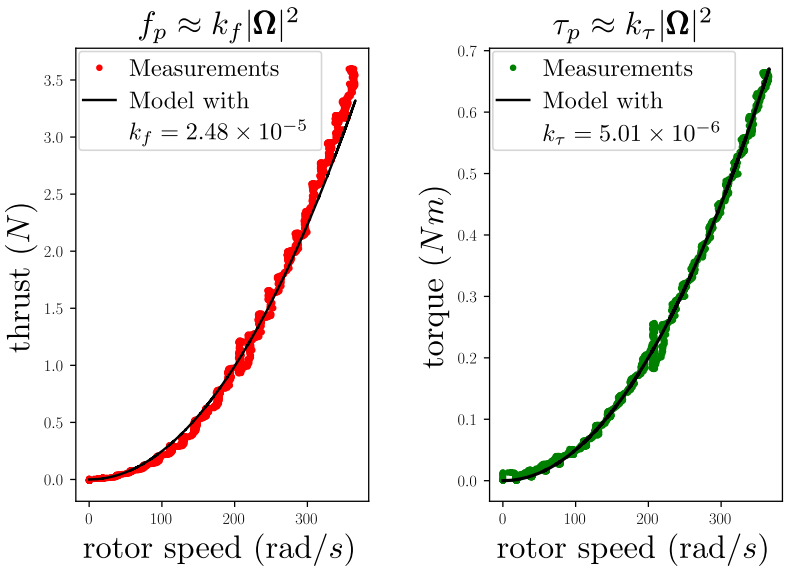


Figure 4.9.: Thrust force and torque curves measured with test stand. In this particular test, we see that we can estimate $k_f = 2{,}48 \cdot 10^{-5}$ and $k_\tau = 5{,}01 \cdot 10^{-6}$, which give $k \approx 0.2$.

[9] https://www.tytorobotics.com/products/optical-rpm-probe-v2-2-for-series-1580

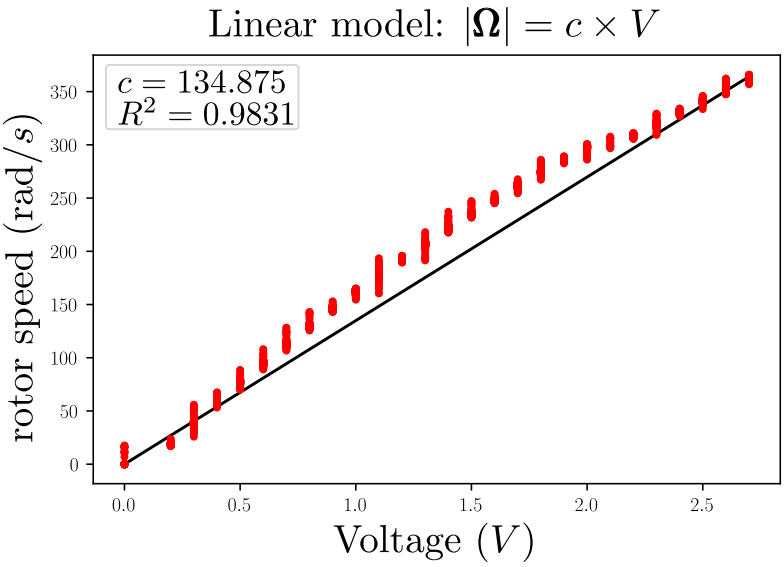


Figure 4.10.: Rotor speed / Voltage curve. In this particular test, the relationship was practically linear and the ratio was about $c = 134.875$, with a coefficient of determination $R^2 = 0.9831$ and a maximum error of 30rad.

4.4. Vertiq module command method

This section discusses how to map the controller output to desired rotor commands, namely, the rotor mean velocity $\dot{\gamma}$, its corresponding mean voltage V, the pulse voltage amplitude $\tilde{V}_a$ and the pulse phase β.

To control the motor and swashplateless mechanism, the following parameters were set on our custom PX4 fork:

- `PROP_MAX_SPEED`: the maximum rotation speed of the rotor, in rad/s. Typically, around 1000rad/s.
- `VOLTAGE_COEF`: the measured ratio between the resulting motor speed $\dot{\gamma}$ and applied mean voltage V, given in (rad/s)/V. Typicality between 130 and 200.
- `PROP_MIN_PULSE`: the minimum pulse voltage $\tilde{V}_{\text{min}}$, in volts. Typically, around 0.2 volts.
- `PROP_MAX_VOLTAGE`: the maximum allowed voltage $V + \tilde{V}_a$, in volts. This security parameter is used to limit the total peak voltage, acts by limiting $\tilde{V}$ if their sum is bigger than the max voltage. Typically, around 8 volts (must be bigger or equal to `PROP_MAX_SPEED`/`VOLTAGE_COEF`).

Internally, PX4 passes the desired control output as a 4 valued vector called `actuator_control`, in which:

- `actuator_control`[0]: roll, manually, or calculated from the controller. Values in range: $[-1, 1]$.
- `actuator_control`[1]: pitch, manually, or calculated from the controller. Values in range: $[-1, 1]$.
- `actuator_control`[2]: yaw, not used in this project, since we cannot directly control the yaw.
- `actuator_control`[3]: thrust, passed directly from the manual control. Values in range: $[0, 1]$.

Calculating the mean voltage is simple:

$$\begin{aligned} \dot{\gamma} &\leftarrow \texttt{actuator\_control}[3] * \texttt{PROP\_MAX\_SPEED} \\ V &\leftarrow \min\left[\texttt{PROP\_MAX\_VOLTAGE}, \dot{\gamma}/\texttt{VOLTAGE\_COEF}\right] \end{aligned} \tag{4.10}$$

The pulse phase is also very simple to calculate, but we must make some special care to account for frame transformations:[10]

$$\beta \leftarrow \text{atan2}\left(-\texttt{actuator\_control}[1], \texttt{actuator\_control}[0]\right) \tag{4.11}$$

[10] PX4 uses the North-East-Down (NED) frame convention instead of East-North-Up (ENU).

The pulse voltage $\tilde{V}_a$, though, is slightly harder. We know it is proportional to the following amplitude variable:

$$a \leftarrow \sqrt{\frac{\texttt{actuator\_control}[0]^2 + \texttt{actuator\_control}[1]^2}{2}} \tag{4.12}$$

In which $a \in [0, 1]$. We then tested two methods:

4.4.1. Direct voltage amplitude input

In this first method, the variable a from Eq. 4.12 is used directly. First, we set the following new parameter:

- `PROP_MAX_PULSE`: Maximum value for pulse voltage, usually around $3V$.

Then we calculate the pulse directly, set it to zero if smaller than lower bound, and limit it if bigger than total accepted voltage:

$$\begin{aligned}
\tilde{V}_a &\leftarrow a * \texttt{PROP_MAX_PULSE} \\
\tilde{V}_a &\leftarrow 0 \text{ , if } \tilde{V}_a < \texttt{PROP_MIN_PULSE} \\
\tilde{V}_a &\leftarrow \min\left[\tilde{V}_a,\ V\right] \\
\tilde{V}_a &\leftarrow \min\left[\tilde{V}_a,\ V - \texttt{PROP_MAX_VOLTAGE}\right]
\end{aligned} \tag{4.13}$$

This method is direct and simple, but has a fundamental problem: The pulse voltage has no direct relationship with the mean voltage V or, equivalently, the motor mean velocity $\dot{\gamma}$. For example, when controlling the motor with $\dot{\gamma} = 200\text{rad}/s$ at a velocity-voltage ratio of 130, the mean voltage is around 2 volts. If the maximum pulse voltage `PROP_MAX_PULSE` is around 3.2 volts, then even at half the max voltage, the pulse is already as big as the mean voltage, creating a pulse that is so big that the instantaneous velocity of the motor reaches 0. This creates an unstable flapping movement on the swashplateless device. Conversely, when turning the motor at higher mean velocities, the pulse voltage to mean voltage ratio might be too small, creating a flapping motion that is not strong enough. To solve this, the following alternative strategy was implemented:

4.4.2. Proportional voltage amplitude input

In this strategy, the following parameters are introduced:

- `PROP_MAX_COEF`: Maximum value for the ratio between the pulse voltage and the mean voltage, maximum value is 1. Usually around 0.3.

- `PROP_POWER_PULSE`: Exponent of power, 1, 2 or 3 (linear, quadratic or cubic relationship). More on this later.

This time, first the ratio V_{ratio} is calculated from the input:

$$V_{\text{ratio}} \leftarrow a^{(\texttt{PROP\_POWER\_PULSE})} \tag{4.14}$$

And then it is used to calculate the amplitude:

$$\tilde{V_a} \leftarrow V * \min\left[V_{\text{ratio}},\ \texttt{PROP\_MAX\_COEF}\right] \tag{4.15}$$

And then the same bound checks are performed:

$$\begin{aligned} &\tilde{V_a} \leftarrow 0 \text{ , if } \tilde{V_a} < \texttt{PROP\_MIN\_PULSE} \\ &\tilde{V_a} \leftarrow \min\left[\tilde{V_a},\ V\right] \\ &\tilde{V_a} \leftarrow \min\left[\tilde{V_a},\ V - \texttt{PROP\_MAX\_VOLTAGE}\right] \end{aligned} \tag{4.16}$$

This assures that the pulse will be automatically set according to the input rotor velocity. The parameter `PROP_POWER_PULSE` was estimated using the experiments from the next section.

4.5. Swashplateless inclination characterization

In this final section of the study of the swashplateless device, we will describe how we tried to quantify its ability to control the direction of the air flux.

4.5.1. Test bench

All the subsequent tests were made using a new 3-axes test bench designed by the ICube team[11] and reproduced by us at ISM Biorobotics. This test bench can be seen in Fig. 4.11. The reasons behind this particular test bench are:

- Putting the swashplateless system at a **larger distance from the ground** helps to remove ground effect, better simulating the properties of the system mid-flight.
- Having a **3-axes** system was very helpful in letting us study both the isolated swashplateless system or the full spinning robot (by blocking or not the yaw axis from spinning, as seen in Fig. 4.11 **c**).

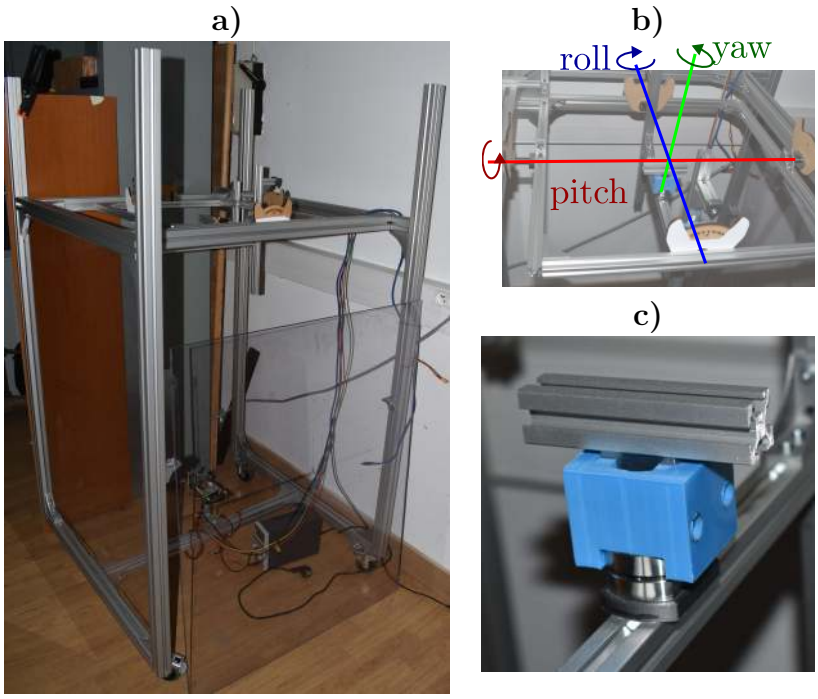


Figure 4.11.: 3-axes test bench designed by ICube and constructed at ISM. **a)** The test bench. Note its height, useful for reducing aerodynamic ground effect. **b)** The 3 axes of rotation allowed by the test bench. **c)** A 3D printed blocker used to fix any of the axes if should not be free for a test. It was mostly used to block the spin rotation during tests for the swashplateless device, as shown in the image.

[11] Once again, I'd like to thank Mr Wurtz for this.

4.5.2. High-speed camera tests

In order to accurately test the torque caused by actuating the swashplateless device and account for the loss in thrust power, we would need a proper test bench that could accurately measure forces in any direction. Sadly, we did not have at ISM Biorobotics any kind of measuring device that could be used for this. We decided to do a study on the resulting inclination angle of the rotation plane, and we suppose in our model that the swashplateless device's effect is to simply rotate the thrust vector[12].

In order to test this, a Phantom high speed camera[13] was used. Using this camera, we can closely follow each rotation of the motor, even when it is being actuated on high speed. Figure 4.12 shows two frames of a test of the motor + swashplateless setup. Note that for these experiments, **every axis of the test bench was blocked**. With this camera, the following test strategy was used:

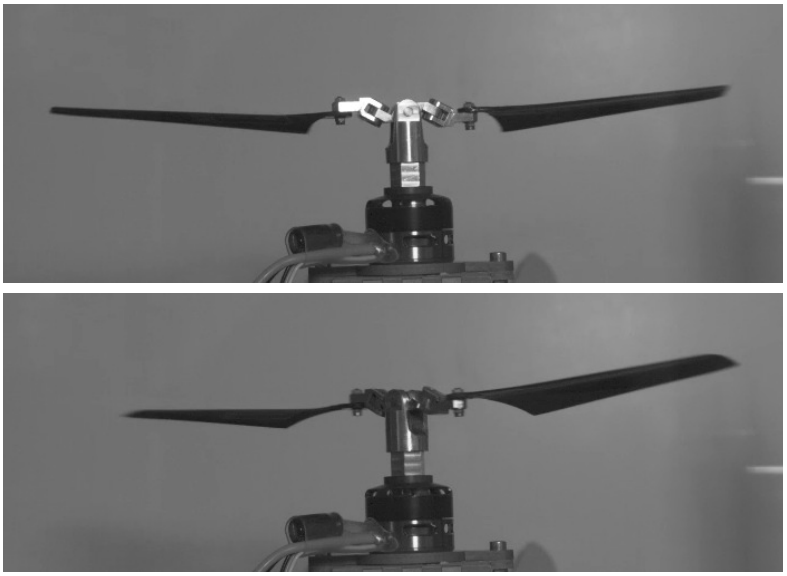


Figure 4.12.: Snapshots from Phantom high-speed camera during swashplateless tests.

a) White tape was glued on one of the sides of the motor, in order to easily detect when the same side is facing the camera.

b) More white tape was glued on the tips of both blades.

c) A black background was put to maximize contrast between the white tape and the rest of the image.

d) The settings of the camera were changed to have finer time resolution instead of high image quality (which, for us, was not very important).

[12]Which is, most definitely, a very crude approximation...
[13]https://www.phantomhighspeed.com/

e) Each experiment with this setup was performed with a constant mean voltage and increasing pulse voltage amplitude.

f) A Python script was used to calculate the luminosity of the square defining the expected position of the white tape from step a) in each frame, and save a new video containing only the snapshots corresponding to peaks in the luminosity.

These steps are illustrated on Fig. 4.13. Frames of the new video containing only the desired snapshots can be seen on Fig. 4.14.

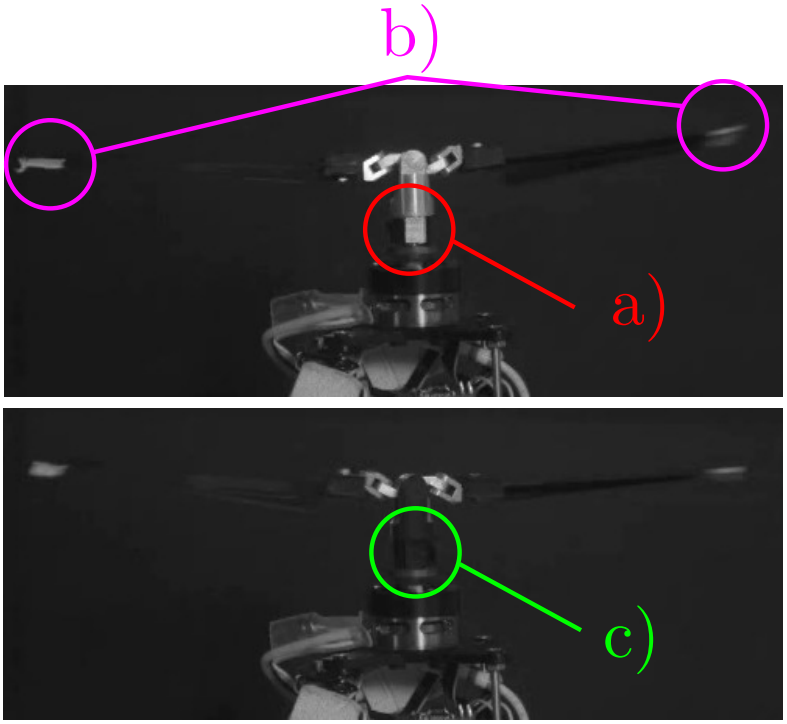


Figure 4.13.: Frames from swashplateless tests with white tape. We can note that the low quality of the image is not a problem for this test strategy. **a)** White tape on the motor, to easily take a snapshot when the motor is facing the camera. **b)** White tape on tips of blades. **c)** Back side of the motor, without tape.

4.5.3. Angle measurements with Physlets Tracker

The final videos corresponding to these snapshots were then analyzed with Tracker[14], a free video analysis tool "*built on the Open Source Physics (OSP) Java framework*". The following steps were performed:

A) The image is imported into Tracker.

[14]https://physlets.org/tracker/

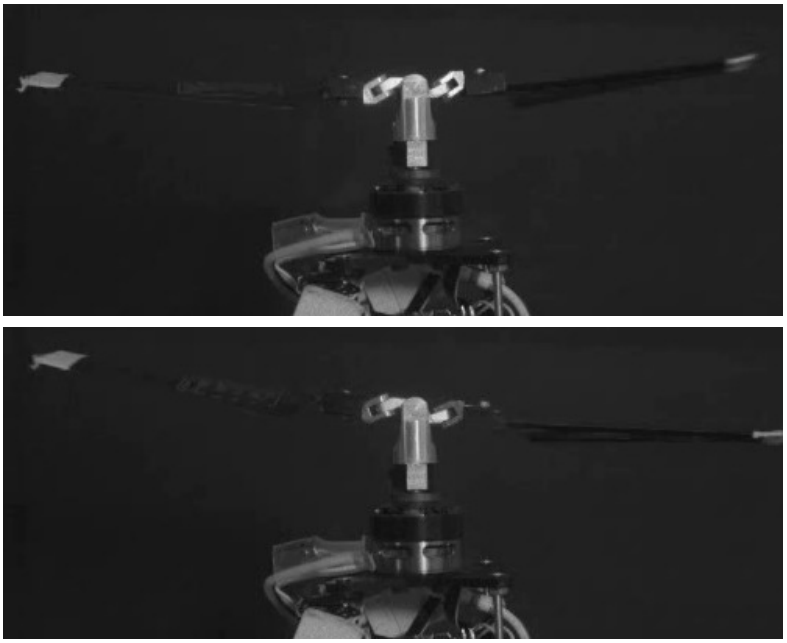


Figure 4.14.: Saved snapshots from swashplateless tests with white tape.

B) A *Brightness* filter is created. The contrast glider is set to 100%, creating a binary image. The brightness glider is then manually set so that both blade tips are very visible, and no noise is around them.

C) Two *mass* objects are created and, on the first frame, their positions are manually set to coincide with the initial positions of the blade tips.

D) The tracking process is started. If everything goes as planned, the position of each mass is measured for the whole video. Some twitching might be necessary for the tracking algorithm, and if everything fails, the position can be manually set at each point when the algorithm fails to measure the next step. This did not happen often, thanks to the black background and white tape on blades.

E) Finally, a *tape* object is created, and its ends are attached to each mass. This measurement object directly gives the distance between the two mass objects and the angle of the straight line passing by both points.

Figure 4.15 shows some elements of the Graphical User Interface of Tracker.

4.5.4. Angle / pulse characterization

The last step was to set up a Python script to assemble these angle measurements and synchronize them with the known voltage amplitude $\tilde{V}_a$. Each experiment had a constant mean rotor velocity $\dot{\gamma}$ and a corresponding mean voltage V. A

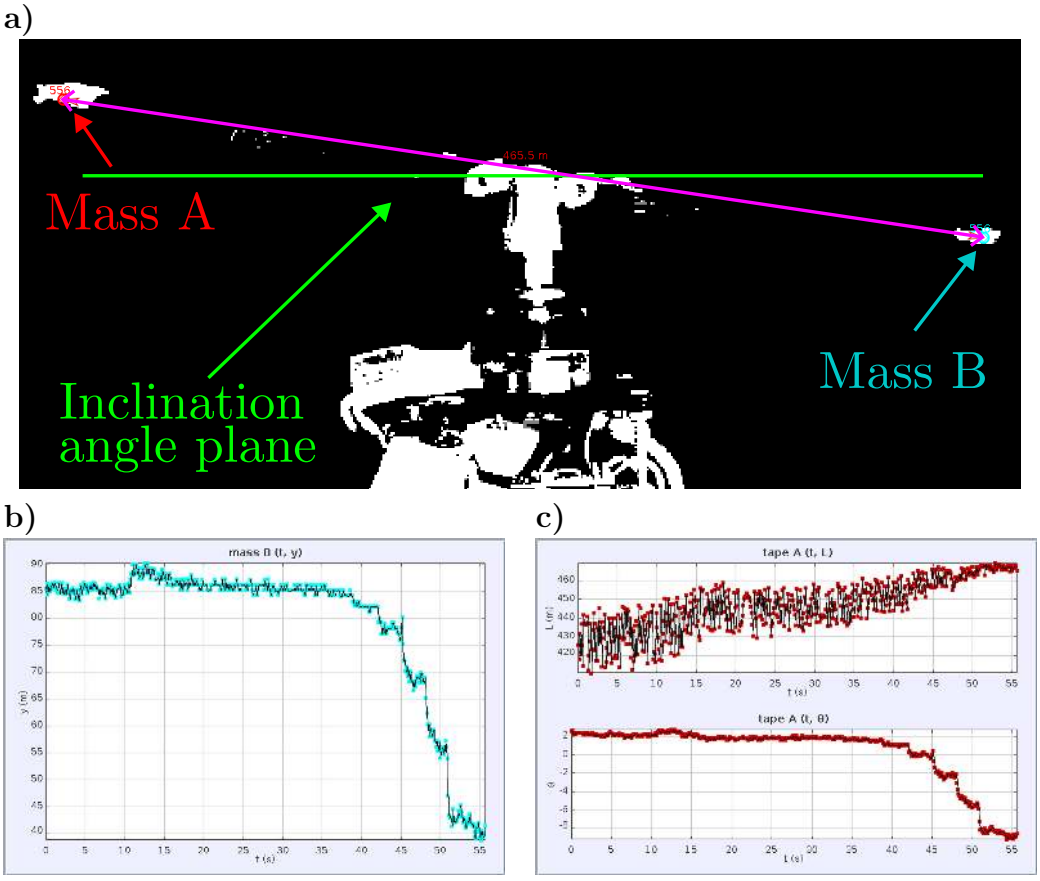


Figure 4.15.: Elements used in Tracker for angle measurement. **a)** Two masses are created on the first frame, and their positions are automatically tracked until the end of the video. A tape object is then created between the two masses, to calculate their angle. **b)** The y component of the position of mass A. **c)** The distance and angle measured by the tape object. We only need the angle.

polynomial fit of order equal to the parameter `PROP_POWER_PULSE` was used to fit the data, with different values for `PROP_POWER_PULSE`. The order that better fit the data was a polynomial of order 3:

$$\begin{aligned}\alpha &\approx f\left(\frac{\tilde{V}_a}{V}\right) \\ &\approx k_0 + k_1\left(\frac{\tilde{V}_a}{V}\right) + k_2\left(\frac{\tilde{V}_a}{V}\right)^2 + k_3\left(\frac{\tilde{V}_a}{V}\right)^3\end{aligned} \tag{4.17}$$

Nevertheless, for each different mean velocity, the fit function f was a different polynomial. This is not a problem for us for the following reasons:

- Usually, only one appropriate mean velocity will be used for the drone most

of the time.

- These tests were not meant to be a precise measurement of the angle, mostly I was interested in measuring the kind of relationship between linear, quadratic, cubic, etc.

Ultimately, the following much simpler fit was implemented:

$$\alpha \approx k_3 \left(\frac{\tilde{V}_a}{V} \right)^3 \tag{4.18}$$

Figure 4.16 shows the measurements and fit for two different mean velocity values. Note that for a single term, the constant k_3 is irrelevant, since this will be directly multiplied by the PID parameters.

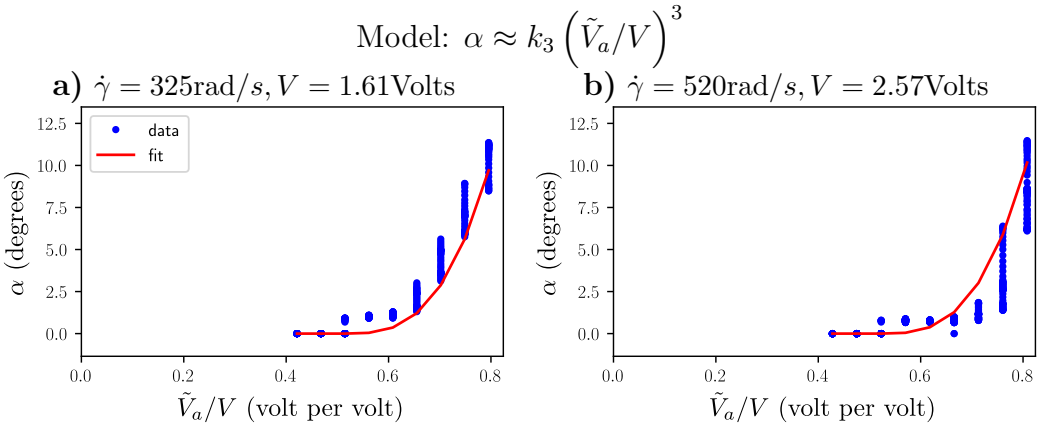


Figure 4.16.: Relationship between inclination angle of Swashplateless and the voltage ratio, and fit assuming a third-order model with a single term. Two mean rotor speeds: **a)** 356rad/s and **b)** 520rad/s.

4.6. Final thoughts

In this chapter, we described the process of reproducing a version of the swashplateless mechanism that was well-suited for our needs in the OrigaBot project. As a direct consequence of the successful reproduction of this system for our project, we decided on proposing a monorotor UAV that will be described in the next chapter.

5. Flying monorotor overview

Following the successful reproduction of the swashplateless system described in the last chapter, we decided on pursuing the creation of an extremely minimalistic drone: a monorotor drone. In this chapter, the monospinner vehicle will be described. Section 5.1 presents the different bodies of the monospinner, along with their masses and inertia matrices. Section 5.2 will describe the dynamics of the monospinner, developing the linear and angular velocities of each part and using them to develop the complete expression of the kinetic energy. Section 5.3 analyzes the external forces acting on the monospinner. This chapter has been submitted as a part of [4].

5.1. System overview

The robot is composed of 3 separated rigid bodies, as seen on Figure 5.1:

- The main body $\mathcal{B}_b$, defining a frame of reference $\mathcal{F}_B$ at its center of mass. It comprises most of the body and all the electronics including the stator;
- The power supply $\mathcal{B}_s$, defining a frame of reference $\mathcal{F}_S$. It is located at the bottom and consists of the battery, which is a big part of the overall mass;
- The propeller $\mathcal{B}_p$, defining a frame $\mathcal{F}_P$ that can rotate with 3 degrees of freedom with respect to the main body frame $\mathcal{F}_B$. It is composed of only the parts of the rotor that rotate freely with 3 DoF w.r.t. the main body frame.

There are two main reasons for choosing to analyze the battery mass at the bottom as a separate body: first, it keeps the main center of mass close to the electronics and inboard IMUs, and second, to help stabilize the robot's attitude passively using gravity. Analyzing it as a separate body also makes it easier to study the effect of the distance h_S between the centers of mass of the main body and the battery.

The bodies $\mathcal{B}_b$, $\mathcal{B}_s$ and $\mathcal{B}_p$ have masses m_b, m_s and m_p and inertia matrices J_b^B, J_s^S and J_p^P, respectively. In their own reference frames, these inertia matrices are constant and approximately diagonal. For $\mathcal{B}_b$ and $\mathcal{B}_s$:

$$\begin{aligned} J_b^B &= \text{diag}(J_{bx}, J_{by}, J_{bz}) \\ J_s^B = J_s^S &= \text{diag}(J_{sx}, J_{sy}, J_{sz}) \end{aligned} \tag{5.1}$$

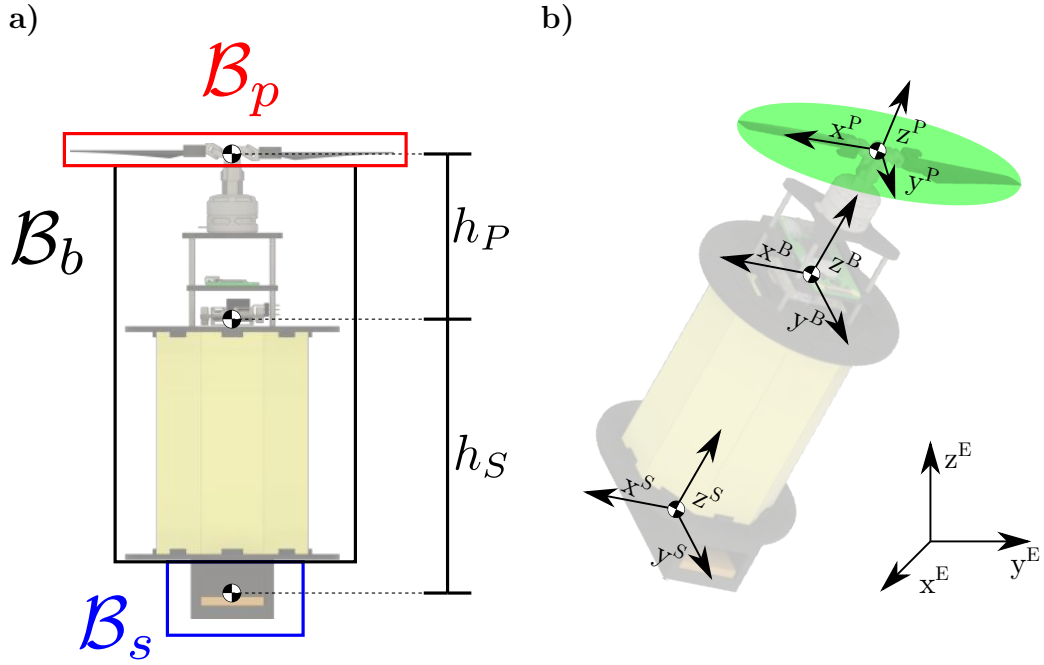


Figure 5.1.: **a)** Illustration of the 3 separated bodies that compose the monorotor, and the distances between their centers of mass. $\mathcal{B}_p$ is the propeller's body, $\mathcal{B}_b$ is the main body and $\mathcal{B}_s$ is the power supply body. **b)** Diagram showing the inertial coordinate frame $\mathcal{F}_E$, the main body frame $\mathcal{F}_B$, the power supply frame $\mathcal{F}_S$ and the propeller frame $\mathcal{F}_P$. The green circle defines the virtual propeller frame.

For the propeller's inertia matrix, since the angular speed of the rotor is large compared to all the other angular velocities, we approximated the propeller by averaging its inertia matrix around one rotation around the z-axis. For a tedious and over-complicated reasoning for this, see Appendix G.

The real inertia matrix of the propeller is:

$$J_{p*}^{P*} = \text{diag}(J_{px}, J_{py}, J_{pz}) \tag{5.2}$$

Where, in general, $J_{px} \neq J_{py}$. We use calculate instead the mean of J_{p*}^{P*} in a full revolution:

$$J_p^P = \frac{1}{2\pi} \int_0^{2\pi} R_z(\gamma) \left(J_{p*}^{P*}\right) R_z(\gamma)^T d\gamma \tag{5.3}$$

which gives:

$$J_p^P = \text{diag}(J_{pd}, J_{pd}, J_{pz}) \tag{5.4}$$

With $J_{pd} = (J_{px} + J_{py})/2$. Moreover, the propeller inertia matrix can be represented in the body frame $\mathcal{F}_B$ with the following transformation:

$$J_p^B = R_P^B \, J_p^P \, (R_P^B)^T \tag{5.5}$$

5.2. Pose, velocities, and kinetic energies

In this section, the expression of the linear and angular velocities of each body will be analyzed and the kinetic energy of the whole system will be calculated.

5.2.1. Body pose and velocities

The inertial pose g_B^E of the body is defined as the homogeneous transformation from frame $\mathcal{F}_B$ to the inertial frame $\mathcal{F}_E$:

$$g_B^E = \begin{bmatrix} R_B^E & \boldsymbol{s}_B^E \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{5.6}$$

Where R_B^E is the rotation matrix between the body frame $\mathcal{F}_B$ and the inertial frame $\mathcal{F}_E$, and $\boldsymbol{s}_B^E$ is the position of the body in the inertial frame. Defining $\boldsymbol{\omega}_B^B$ and $\boldsymbol{v}_B^B$ the angular and linear velocities of $\mathcal{B}_b$ in $\mathcal{F}_B$, we can define $\mathcal{B}_b$'s inertial twist η_B^B as:

$$\eta_B^B = \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \end{bmatrix} \tag{5.7}$$

And the body's kinetic energy is:

$$\begin{aligned} T_B &= \frac{m_b}{2}(\boldsymbol{v}_B^B)^T \boldsymbol{v}_B^B + \frac{1}{2}\left(\boldsymbol{\omega}_B^B\right)^T J_b^B \boldsymbol{\omega}_B^B \\ &= \frac{1}{2}\begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \end{bmatrix}^T \begin{bmatrix} J_b^B & \mathbf{0} \\ \mathbf{0} & m_b\mathbb{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \end{bmatrix} \end{aligned} \tag{5.8}$$

For sake of clarity, we expand Eq. 5.8 in the following form:

$$T_B = \frac{1}{2}\begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \\ \boldsymbol{\Omega}_{P/B}^P \end{bmatrix}^T \begin{bmatrix} J_b^B & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & m_b\mathbb{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \\ \boldsymbol{\Omega}_{P/B}^P \end{bmatrix} \tag{5.9}$$

5.2.2. Rotor pose and velocities

In order to generate vector thrust control with a single motor, we can use a swashplateless system that works by producing cyclic flapping by torque modulation [45, 46], as explained in Chapter 4. This technology consists of attaching the blades on passive asymmetric lag-pitch hinges (as seen in Fig. 5.2), that are excited by using a motor speed controller that adds a sinusoidal speed component of frequency equals to 1/rev.

The passive hinges oscillate by the inertia of the blade assembly, giving an inclination to the rotor disc. By raising the amplitude of this sinusoidal term, the rotor disc

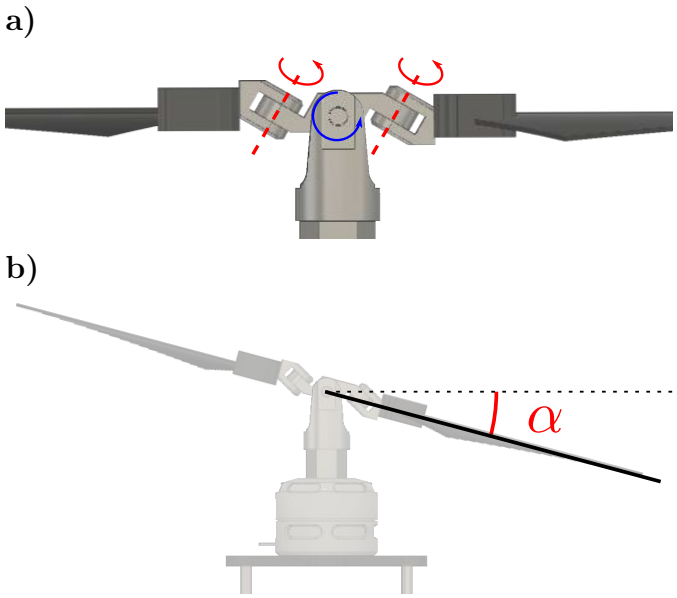


Figure 5.2.: 3D model of Swashplateless assembly. **a)** The lag-pitch hinge rotations (detailed in red) are responsible for creating the oscillatory flapping due to their inclination angle. The center teetering hinge helps to reduce higher order components on the oscillation. **b)** Angle α between the plane created by the wings and the horizontal plane in our approximate model (note that this plane is horizontal w.r.t. to $\mathcal{F}_B$, and not the inertial frame). This model considers the teetering hinge as the center of the controlled inclination. This approximates well the behavior of the system.

inclination is more pronounced. By changing the phase of this term, we can effectively control the direction of this inclination. In this work, we modeled the rotor by considering it as a simple motor connected to two revolute joints.

The rotor is fixed at a distance h_P from the center of $\mathcal{F}_B$. The rotation matrix R_P^B from the propeller frame $\mathcal{F}_P$ to the body frame $\mathcal{F}_B$ is:

$$R_P^B \equiv R_z(\beta)\, R_y(\alpha) = \begin{bmatrix} c_\alpha c_\beta & -s_\beta & s_\alpha c_\beta \\ c_\alpha s_\beta & c_\beta & s_\alpha s_\beta \\ -s_\alpha & 0 & c_\alpha \end{bmatrix} \tag{5.10}$$

Where α is the inclination angle, β is the direction angle, $s_x = \sin(x)$ and $c_x = \cos(x)$. We also define γ as the angle around the z-axis in which the rotor wings are continuously rotated. We can then define the inertial pose g_P^B of the frame $\mathcal{F}_P$ with respect to $\mathcal{F}_B$ as an element of SE(3) as follows:

$$g_P^B = \begin{bmatrix} R_P^B & h_P \boldsymbol{e}_z \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{5.11}$$

Where $\boldsymbol{e}_z = [0,\ 0,\ 1]^T$. For $\boldsymbol{a} = [a_x,\ a_y,\ a_z]^T$, we define the hat operator $\widehat{(\cdot)}$ giving

the skew-symmetric matrix form of the cross product (see Appendix A):

$$\widehat{\boldsymbol{a}} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{5.12}$$

Such that $\boldsymbol{a}_1 \times \boldsymbol{a}_2 = \widehat{\boldsymbol{a}}_1 \boldsymbol{a}_2$ for any $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ in $\mathbb{R}^3$. In particular:

$$\widehat{\boldsymbol{e}}_z = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.13}$$

We define $\boldsymbol{\Omega}^P_{P/B}$[1] as the full angular velocity between the propeller and the body in the rotor frame $\mathcal{F}_P$[2]:

$$\boldsymbol{\Omega}^P_{P/B} = \boldsymbol{\omega}^P_{P/B} + \dot{\gamma}\boldsymbol{e}_z \tag{5.14}$$

Where $\dot{\gamma}$ is the mean velocity of the rotor around the z-axis, and $\boldsymbol{\omega}^P_{P/B}$ is the angular velocity of the virtual rotor frame $\mathcal{F}_B$ (the rotor's disc) w.r.t. the body frame $\mathcal{F}_B$:

$$\begin{aligned} \boldsymbol{\omega}^P_{P/B} &= \begin{bmatrix} -s_\alpha \dot{\beta} \\ \dot{\alpha} \\ c_\alpha \dot{\beta} \end{bmatrix} \\ \dot{R}^B_P &= R^B_P \widehat{\boldsymbol{\omega}}^P_{P/B} \end{aligned} \tag{5.15}$$

Note that α, $\dot{\alpha}$ and $\dot{\beta}$ are very small compared to $\dot{\gamma}$. Moreover, the kinetic energy component generated by the inertia of the rotor blades is negligible. Therefore, we consider $\boldsymbol{\omega}^P_{P/B} \approx 0$ and:

$$\boldsymbol{\Omega}^P_{P/B} \approx \dot{\gamma}\boldsymbol{e}_z \tag{5.16}$$

To find the kinetic energy, we must find the angular velocity $\boldsymbol{\Omega}^P_P$ and the linear velocity $\boldsymbol{v}^P_P$ of $\mathcal{F}_P$ w.r.t. the inertial frame $\mathcal{F}_E$. To transform an inertial twist $\eta = \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix}$ represented in a frame $\mathcal{F}_i$ to a frame $\mathcal{F}_j$, both related by a transformation:

$$g^j_i = \begin{bmatrix} R^j_i & \boldsymbol{s}^j_i \\ \boldsymbol{0}^T & 1 \end{bmatrix} \tag{5.17}$$

Where s^j_i is the position of $\mathcal{F}_i$ in $\mathcal{F}_j$ and R^j_i is the rotation matrix from $\mathcal{F}_i$ to $\mathcal{F}_j$,

[1] As seen in Appendix A, the subscript P/B indicates the angular velocity between the rotor and the body, and the superscript P indicates it is expressed in the propeller $\mathcal{F}_P$ frame

[2] See Appendix F.2 for a derivation of the full angular velocity on the ZYZ sequence, and Appendix G for an argument supporting the virtual frame approximation.

we define the Ad operator (Eq. A.15):

$$\mathrm{Ad}_i^j = \begin{bmatrix} R_i^j & 0 \\ \widehat{\boldsymbol{s}}_i^j R_i^j & R_i^j \end{bmatrix} \tag{5.18}$$

And the inverse operation is:

$$\mathrm{Ad}_j^i = \begin{bmatrix} \left(R_i^j\right)^T & 0 \\ -\left(R_i^j\right)^T \widehat{\boldsymbol{s}}_i^j & \left(R_i^j\right)^T \end{bmatrix} \tag{5.19}$$

Moreover, by composition of velocities (Eq. A.14), we can state that:

$$\eta_j^j = \mathrm{Ad}_i^j \, \eta_i^i + \eta_{j/i}^j \tag{5.20}$$

Considering $\boldsymbol{\omega}_B^B$ and $\boldsymbol{v}_B^B$, respectively, the angular and linear velocities of the main body frame $\mathcal{F}_B$ w.r.t. to the inertial frame $\mathcal{F}_E$, we can calculate the final rotor velocities w.r.t. to the inertial frame as:

$$\begin{aligned} \eta_P^P &= \left(\mathrm{Ad}_B^P\right) \eta_B^B + \eta_{P/B}^P \\ &= \left(\mathrm{Ad}_P^B\right)^{-1} \eta_B^B + \eta_{P/B}^P \\ \begin{bmatrix} \boldsymbol{\Omega}_P^P \\ \boldsymbol{v}_P^P \end{bmatrix} &= \begin{bmatrix} \left(R_P^B\right)^T & 0 \\ -h_P \left(R_P^B\right)^T \widehat{\boldsymbol{e}}_z & \left(R_P^B\right)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Omega}_{P/B}^P \\ \mathbf{0} \end{bmatrix} \end{aligned} \tag{5.21}$$

Which leads to:

$$\begin{aligned} \boldsymbol{\Omega}_P^P &= \left(R_P^B\right)^T \boldsymbol{\omega}_B^B + \boldsymbol{\Omega}_{P/B}^P \\ \boldsymbol{v}_P^P &= \left(R_P^B\right)^T \boldsymbol{v}_B^B - h_P \left(R_P^B\right)^T \widehat{\boldsymbol{e}}_z \boldsymbol{\omega}_B^B \end{aligned} \tag{5.22}$$

And the rotor's kinetic energy is:

$$\begin{aligned} T_P &= \frac{m_p}{2} \left(\boldsymbol{v}_P^P\right)^T \boldsymbol{v}_P^P + \frac{1}{2} \left(\boldsymbol{\Omega}_P^P\right)^T J_p^P \, \boldsymbol{\Omega}_P^P \\ &= \frac{1}{2} \begin{bmatrix} \boldsymbol{\Omega}_P^P \\ \boldsymbol{v}_P^P \end{bmatrix}^T \begin{bmatrix} J_p^P & \mathbf{0} \\ \mathbf{0} & m_p \mathbb{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Omega}_P^P \\ \boldsymbol{v}_P^P \end{bmatrix} \end{aligned} \tag{5.23}$$

Introducing Eq. 5.21 into Eq. 5.23, we can state:

$$T_P = A + B + C + D \tag{5.24}$$

In which:

$$\begin{aligned} A &= \frac{1}{2}\begin{bmatrix}\boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B\end{bmatrix}^T \left(\mathrm{Ad}_B^P\right)^T \begin{bmatrix} J_p^P & \mathbf{0} \\ \mathbf{0} & m_p\mathbb{I}\end{bmatrix} \left(\mathrm{Ad}_B^P\right) \begin{bmatrix}\boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B\end{bmatrix} \\ &= \frac{1}{2}\begin{bmatrix}\boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B\end{bmatrix}^T \begin{bmatrix} R_P^B J_p^P \left(R_P^B\right)^T - m_p h_P^2 \left(\widehat{\boldsymbol{e}}_z\right)^2 & m_p h_P \widehat{\boldsymbol{e}}_z \\ -m_p h_P \widehat{\boldsymbol{e}}_z & m_p\mathbb{I}\end{bmatrix} \begin{bmatrix}\boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B\end{bmatrix} \end{aligned} \tag{5.25}$$

The second term is:

$$\begin{aligned} B &= \frac{1}{2}\begin{bmatrix}\boldsymbol{\Omega}_{P/B}^P \\ \mathbf{0}\end{bmatrix}^T \begin{bmatrix} J_p^P & \mathbf{0} \\ \mathbf{0} & m_p\mathbb{I}\end{bmatrix} \begin{bmatrix} \left(R_P^B\right)^T & \mathbf{0} \\ -h_P \left(R_P^B\right)^T \widehat{\boldsymbol{e}}_z & \left(R_P^B\right)^T\end{bmatrix} \begin{bmatrix}\boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B\end{bmatrix} \\ &= \frac{1}{2}\left(\boldsymbol{\Omega}_{P/B}^P\right)^T \left(R_P^B J_p^P\right)^T \boldsymbol{\omega}_B^B \end{aligned} \tag{5.26}$$

The third term is:

$$\begin{aligned} C &= \frac{1}{2}\begin{bmatrix}\boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B\end{bmatrix}^T \begin{bmatrix} R_P^B & h_P \widehat{\boldsymbol{e}}_z R_P^B \\ \mathbf{0} & R_P^B\end{bmatrix} \begin{bmatrix} J_p^P & \mathbf{0} \\ \mathbf{0} & m_p\mathbb{I}\end{bmatrix} \begin{bmatrix}\boldsymbol{\Omega}_{P/B}^P \\ \mathbf{0}\end{bmatrix} \\ &= \frac{1}{2}\left(\boldsymbol{\omega}_B^B\right)^T \left(R_P^B J_p^P\right) \boldsymbol{\Omega}_{P/B}^P \end{aligned} \tag{5.27}$$

And the fourth and final term is:

$$D = \frac{1}{2}\left(\boldsymbol{\Omega}_{P/B}^P\right)^T J_p^P \boldsymbol{\Omega}_{P/B}^P \tag{5.28}$$

Finally, adding all terms together, we can express the propeller's kinetic energy as:

$$T_P = \frac{1}{2} \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \\ \boldsymbol{\Omega}_{P/B}^P \end{bmatrix}^T \begin{bmatrix} J_p^B - m_p h_P^2 \left(\widehat{\boldsymbol{e}}_z\right)^2 & m_p h_P \widehat{\boldsymbol{e}}_z & R_P^B J_p^P \\ -m_p h_P \widehat{\boldsymbol{e}}_z & m_b\mathbb{I} & \mathbf{0} \\ \left(R_P^B J_p^P\right)^T & \mathbf{0} & J_p^P \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \\ \boldsymbol{\Omega}_{P/B}^P \end{bmatrix} \tag{5.29}$$

Where $J_p^B = R_P^B\, J_p^P\, (R_P^B)^T$.

5.2.3. Power supply velocities

The power supply mass and frame can also be analyzed in the exact same way as the propeller using Eqs. 5.18 and 5.20, with translation $-h_S \boldsymbol{e}_z$, rotation $R_S^B = \mathbb{I}$ and $\eta_{S/B}^S = \mathbf{0}$:

$$\begin{aligned} \eta_S^S &= \left(\mathrm{Ad}_B^S\right)^{-1} \eta_B^B + \eta_{S/B}^P \\ \begin{bmatrix}\boldsymbol{\Omega}_S^S \\ v_S^S\end{bmatrix} &= \begin{bmatrix} \mathbb{I} & 0 \\ h_S \widehat{\boldsymbol{e}}_z & \mathbb{I}\end{bmatrix} \begin{bmatrix}\boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B\end{bmatrix} \end{aligned} \tag{5.30}$$

Which leads to:

$$\begin{aligned} \boldsymbol{\Omega}_S^S &= \boldsymbol{\omega}_B^B \\ v_S^S &= h_S \widehat{\boldsymbol{e}}_z \boldsymbol{\omega}_B^B + \boldsymbol{v}_B^B \end{aligned} \tag{5.31}$$

And the power supply's kinetic energy is:

$$\begin{aligned} T_S &= \frac{m_p}{2} \left(v_S^S\right)^T v_S^S + \frac{1}{2} \left(\boldsymbol{\Omega}_S^S\right)^T J_s^S \boldsymbol{\Omega}_S^S \\ &= \frac{1}{2} \begin{bmatrix} \boldsymbol{\Omega}_S^S \\ v_S^S \end{bmatrix}^T \begin{bmatrix} J_s^S & \mathbf{0} \\ \mathbf{0} & m_s \mathbb{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Omega}_S^S \\ v_S^S \end{bmatrix} \end{aligned} \tag{5.32}$$

Introducing Eq. 5.30 into Eq. 5.32, and expanding it as before, we get:

$$\begin{aligned} T_S &= \frac{1}{2} \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \end{bmatrix}^T \begin{bmatrix} \mathbb{I} & -h_S \widehat{\boldsymbol{e}}_z \\ \mathbf{0} & \mathbb{I} \end{bmatrix} \begin{bmatrix} J_s^S & \mathbf{0} \\ \mathbf{0} & m_s \mathbb{I} \end{bmatrix} \begin{bmatrix} \mathbb{I} & \mathbf{0} \\ h_S \widehat{\boldsymbol{e}}_z & \mathbb{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \end{bmatrix} \\ &= \frac{1}{2} \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \\ \boldsymbol{\Omega}_{P/B}^P \end{bmatrix}^T \begin{bmatrix} J_s^S - m_s h_S^2 \left(\widehat{\boldsymbol{e}}_z\right)^2 & -m_s h_S \widehat{\boldsymbol{e}}_z & \mathbf{0} \\ m_s h_S \widehat{\boldsymbol{e}}_z & m_s \mathbb{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \\ \boldsymbol{\Omega}_{P/B}^P \end{bmatrix}^T \end{aligned} \tag{5.33}$$

5.2.4. Total kinetic energy

The expression for the total kinetic T energy can be given as:

$$T = T_B + T_P + T_S \tag{5.34}$$

Defining $\eta = \eta_B^B$ as the inertial twist of the body such that $\eta = \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \end{bmatrix}$, with $\boldsymbol{\Omega} = \boldsymbol{\Omega}_{P/B}^P$ as the rotor's angular velocity in respect to $\mathcal{F}_B$. The extra inertia component created by the point masses is:

$$\begin{aligned} J_m^B &= \left(m_s h_S^2 + m_p h_P^2\right) \left(-\widehat{\boldsymbol{e}}_z{}^2\right) \\ &= \left(m_s h_S^2 + m_p h_P^2\right) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{5.35}$$

And we also define $\boldsymbol{J} = J_b^B + J_s^B + J_p^B + J_m^B$, the inertia matrix of the whole assembly, $m = m_b + m_s + m_p$ the sum of all masses.

Inserting Eqs. 5.9, 5.29 and 5.33 into Eq. 5.34, and defining $\Delta m = m_s h_S - m_p h_P$[3], we get:

$$T = \frac{1}{2} \begin{bmatrix} \eta \\ \boldsymbol{\Omega} \end{bmatrix}^T \mathbf{M} \begin{bmatrix} \eta \\ \boldsymbol{\Omega} \end{bmatrix} \tag{5.36}$$

Where:

$$\mathbf{M} = \begin{bmatrix} \boldsymbol{J} & -\Delta m \widehat{\boldsymbol{e}}_z & R_P^B J_p^P \\ \Delta m \widehat{\boldsymbol{e}}_z & m\,\mathbb{I} & 0 \\ \left(R_P^B J_p^P\right)^T & 0 & J_p^P \end{bmatrix} \tag{5.37}$$

5.3. External forces

In this section, we will analyze the external forces F_{ext}^B and torques acting on the drone. There are three components:

$$F_{\text{ext}}^B = F_{\text{grav}}^B + F_p^B + F_{\text{aero}}^B \tag{5.38}$$

Where F_{grav}^B represents the external forces and torques generated by gravity, F_p^B represents the forces and torques generated by the rotor and F_{aero}^B represents the aerodynamic torque generated by friction with the air.

5.3.1. Gravity force

The 6×1 vector of the sum of torques and forces caused by gravity in all three bodies is:

$$\begin{aligned} F_{\text{grav}}^B &= \begin{bmatrix} 0 \\ m_b \boldsymbol{g}^B \end{bmatrix} + \begin{bmatrix} R_P^B & h_P \widehat{\boldsymbol{e}}_z R_P^B \\ 0 & R_P^B \end{bmatrix} \begin{bmatrix} 0 \\ m_p \boldsymbol{g}^P \end{bmatrix} \\ &\quad + \begin{bmatrix} R_S^B & -h_S \widehat{\boldsymbol{e}}_z R_S^B \\ 0 & R_S^B \end{bmatrix} \begin{bmatrix} 0 \\ m_s \boldsymbol{g}^S \end{bmatrix} \\ &= \begin{bmatrix} -\Delta m \widehat{\boldsymbol{e}}_z R^T \boldsymbol{g}^E \\ m R^T \boldsymbol{g}^E \end{bmatrix} \end{aligned} \tag{5.39}$$

Assuming $\boldsymbol{g}^E = (0, 0, -g) = -g\boldsymbol{e}_z$ where g is the gravitational constant.

[3]Note that $m_s h_S \gg m_p h_P$.

5.3.2. Motor thrust and torque

The thrust direction vector $\boldsymbol{r}$ in $\mathcal{F}_B$ is controlled using the swashplateless solution. It is given by:

$$\boldsymbol{r} = R_P^B \, \boldsymbol{e}_z = \begin{bmatrix} s_\alpha c_\beta \\ s_\alpha s_\beta \\ c_\alpha \end{bmatrix} \tag{5.40}$$

And the motor thrust force $\boldsymbol{f}_p$ and torque $\boldsymbol{\tau}_p$ are:

$$\begin{aligned} \boldsymbol{f}_p^B &= f_p \boldsymbol{r} \\ \boldsymbol{\tau}_p^B &= \tau_p \boldsymbol{r} \end{aligned} \tag{5.41}$$

The magnitudes of both the motor thrust and torque can be modeled as a quadratic function of the propeller's rotation speed w.r.t. the inertial frame, which is:

$$\left|\boldsymbol{\Omega}_P^P\right|^2 = \left| \left(R_P^B\right)^T \boldsymbol{\omega}_B^B + \boldsymbol{\Omega}_{P/B}^P \right|^2 \tag{5.42}$$

And taking into account that:

$$\left|\boldsymbol{\Omega}_{P/B}^P\right| \gg \left|\boldsymbol{\omega}_B^B\right| \tag{5.43}$$

We can simplify the expression of the rotor velocity as:

$$\left|\boldsymbol{\Omega}_P^P\right| \approx \left|\boldsymbol{\Omega}_{P/B}^P\right| \tag{5.44}$$

We then model the magnitudes as:

$$\begin{aligned} |\boldsymbol{f}_p| = f_p &\approx k_f \left|\boldsymbol{\Omega}_{P/B}^P\right|^2 \\ |\boldsymbol{\tau}_p| = \tau_p &\approx k_\tau \left|\boldsymbol{\Omega}_{P/B}^P\right|^2 \end{aligned} \tag{5.45}$$

As seen in [77], we assume that the propeller torque is linear w.r.t. to the propeller thrust. Defining:

$$k = \frac{k_\tau}{k_f}, \tag{5.46}$$

We can express the torque $\boldsymbol{\tau}_p^B$ as a linear function of the thrust force $\boldsymbol{f}_p^B$:

$$\boldsymbol{\tau}_p^B = k \boldsymbol{f}_p^B \tag{5.47}$$

Moreover, given $h_P \, \boldsymbol{e}_z$ the position vector of the propeller in the body frame $\mathcal{F}_B$, there is also a crossed thrust torque term that is added, as seen in Fig. 5.3:

$$\boldsymbol{\tau}_{\text{cross}}^B = h_P \, \boldsymbol{e}_z \times \boldsymbol{f}_p^B \tag{5.48}$$

Adding both terms of torque, we have:

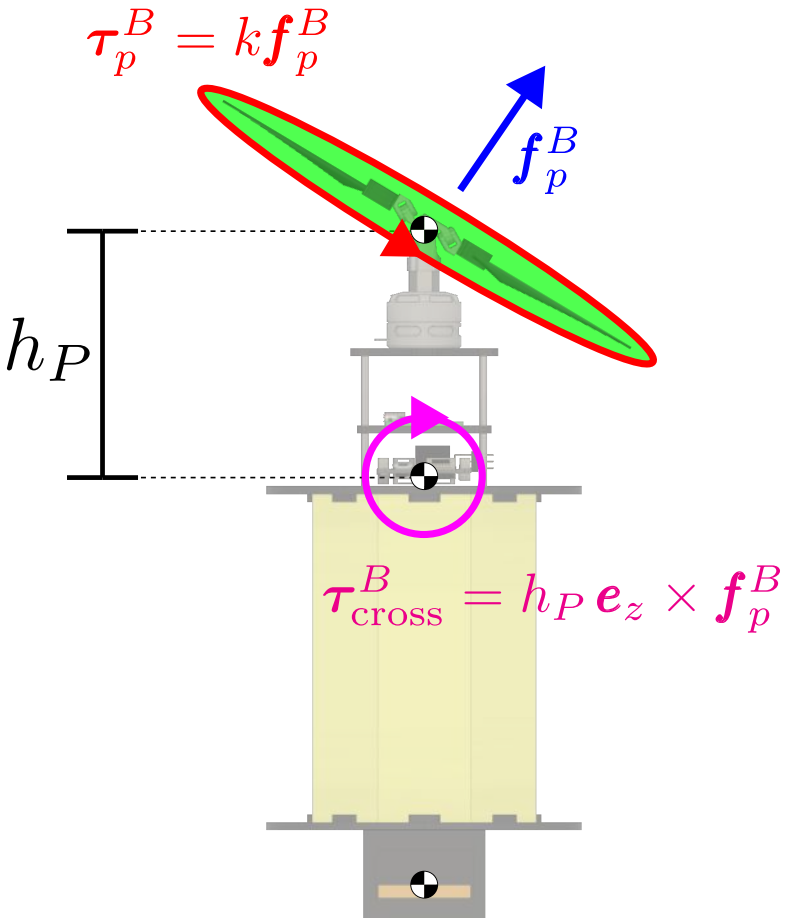


Figure 5.3.: Forces and torques produced by the rotor. Thrust vector (in red) considered to be proportional to the square of the rotor mean angular speed. First torque component (in green) considered linear in the thrust force. Second torque component (in magenta) generated by the distance between the propeller thrust center and the center of mass of the body.

$$\begin{aligned}\boldsymbol{\tau} &= \boldsymbol{\tau}_p^B + \boldsymbol{\tau}_{\text{cross}}^B \\ &= \boldsymbol{B}_\tau \boldsymbol{f}_p\end{aligned} \tag{5.49}$$

With:

$$\boldsymbol{B}_\tau = (k\mathbb{I} + h_P\,\widehat{\boldsymbol{e}}_z) = \begin{bmatrix} k & -h_P & 0 \\ h_P & k & 0 \\ 0 & 0 & k \end{bmatrix} \tag{5.50}$$

It is interesting to note that the matrix $\boldsymbol{B}_\tau$ is invertible, as seen in Appendix I. And finally, the 6×1 vector of all torques and forces produced by the propeller is:

$$F_p^B = \begin{bmatrix} \boldsymbol{B}_\tau f_p \boldsymbol{r} \\ f_p \boldsymbol{r} \end{bmatrix} \tag{5.51}$$

5.3.3. Aerodynamic drag

The aerodynamic drag can be modeled as:

$$F_{\text{aero}}^B = \begin{bmatrix} -\left|\boldsymbol{\omega}_B^B\right| \boldsymbol{K}\, \boldsymbol{\omega}_B^B \\ \mathbf{0} \end{bmatrix} \tag{5.52}$$

Where $\boldsymbol{K}$ is a diagonal positive definite matrix. At the equilibrium, the angular velocity of the body frame $\mathcal{F}_B$ reaches a steady-state velocity with direction opposed to the rotor velocity: $\boldsymbol{\omega}_B^B = \omega_z \boldsymbol{e}_z$, and we can assume $\boldsymbol{K} \approx \text{diag}\,(0, 0, K_z)$, which leads to:

$$\left|\boldsymbol{\omega}_B^B\right| \boldsymbol{K}\, \boldsymbol{\omega}_B^B = \text{sign}(\omega_z) K_z\, \omega_z^2\, \boldsymbol{e}_z \tag{5.53}$$

The rotor torque $\boldsymbol{\tau}_p$ and the aerodynamic drag will cancel each other at equilibrium, and $\boldsymbol{r} = \boldsymbol{e}_z$, leading to:

$$K_z = \text{sign}(\omega_z) k_\tau \left(1 + \frac{\dot{\gamma}}{\omega_z}\right)^2 \tag{5.54}$$

Where $\dot{\gamma}/\omega_z$, the ratio between the steady-state angular velocity of the body and the rotor velocity, can be estimated experimentally. Even though this study does not present a real mono spinner, preliminary tests have shown that the steady-state angular velocity of the monorotor drone's body was higher than the limits of the Pixracer's IMU. In order to overcome this problem, several fins were attached to the main body of a prototype to add drag, and then the rotation speed was measured with its integrated IMUs, as shown in Fig. 5.4. We used our prototype containing a Pixracer card with PX4 firmware to test the steady-state velocity of the monospinner with different numbers of fins, and compared it with measurements from a Phantom high-speed camera. We made the motor's rotation speed vary from 80 to 800 rad/s, and the results can be seen on Fig. 5.5. As expected, we noted that adding more fins drastically reduce the steady-state angular velocity. Moreover, we also noted that two fins were not sufficient to reduce the monorotor's angular velocity to the IMU's gyroscope's range.

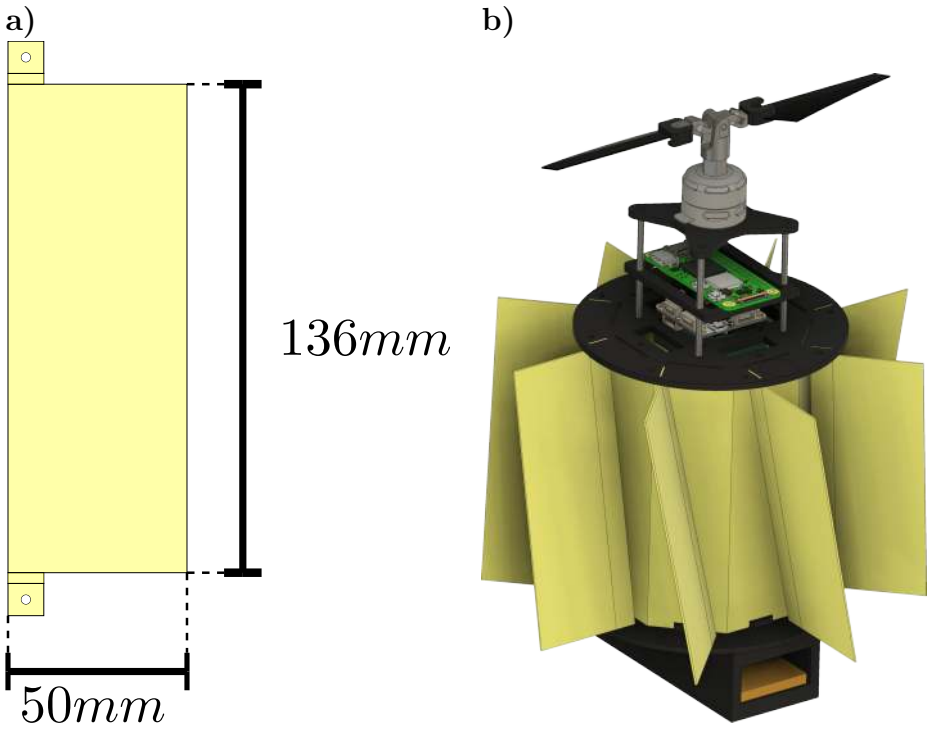


Figure 5.4.: Anti-torque fins. **a)** Dimensions of fins used for tests. **b)** 3D model of prototype with 8 fins installed radially.

5.3.4. Sum of external forces

Introducing Eqs. 5.39, 5.51 and 5.52 into Eq. 5.38, we have the total expression of the external forces:

$$\begin{aligned} F^B_{\text{ext}} &= F^B_{\text{grav}} + F^B_p + F^B_{\text{aero}} \\ &= \begin{bmatrix} f_p \boldsymbol{B}_\tau \boldsymbol{r} + \Delta m\, g \widehat{\boldsymbol{e}}_z R^T \boldsymbol{e}_z - \left|\boldsymbol{\omega}^B_B\right| \boldsymbol{K}\, \boldsymbol{\omega}^B_B \\ f_p \boldsymbol{r} - mg R^T \boldsymbol{e}_z \end{bmatrix} \end{aligned} \tag{5.55}$$

5.4. Final thoughts

In this chapter, a monorotor UAV is proposed, and a complete analysis of its overall shape, kinetic energy and external forces is given. For this model, we took special care of making a minimal set of assumptions and approximations, which makes it possible for us to derive a rather complete expression of the Equations of Motion. These Equations of Motion can, of course, be derived with classical Lagrangian mechanics by using the expressions of the kinetic energies described here. However, as will be seen in Chapter 6, we have decided to use the Euler-Poincaré method to find the solution for our problem.

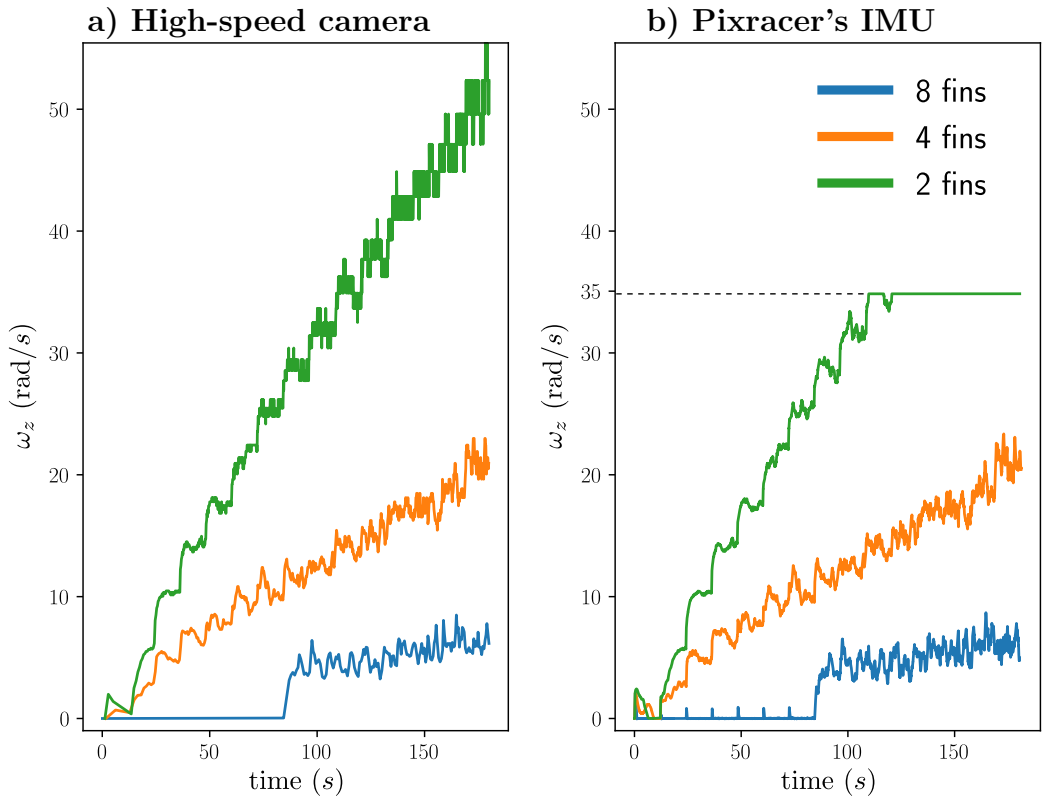


Figure 5.5.: Steady-state angular velocity of monorotor with 2, 4 and 8 fins, while varying the motor's speed from 80 to 800 rad/s. **a)** Angular velocity measured from a Phantom high-speed camera. **b)** Angular velocity measured from the monorotor's IMU's rate gyro. The latter saturates at around 35.68 rad/s.

6. Equations of Motion of the complete system with Euler-Poincaré method

In this chapter, we will apply the Euler-Poincaré equations to the monorotor drone described in Chapter 5 to obtain its Equations Of Motion (EOM). A simplified version of this chapter has been submitted to publication as a part of [4]. The main difference, as will be clear later on, is that this chapter takes into account the movement of the virtual rotor plane. As this is mostly negligible, I have omitted it from the submission in [4].

While the direct application of the Newton-Euler equations is also possible, this approach that makes direct use of vectors and angular pseudo-vectors can become confusing and prone to errors when analyzing a complex multi-body system with additional internal degrees of freedom. This method has, however, the advantage of producing a parameter-invariant solution in terms of the velocities of the system in the moving body-frame. The Newton-Euler equations were used for example for the PULSAR spinner [13]. However it is worth noting that the spinner was considered in this case as a single body. The model of the dynamics did not take into account all the degrees of freedom of the rotor. Lagrangian mechanics, on the other hand, provides an arguably more systematic method by allowing the derivation of the dynamical model of the system from the knowledge of its Lagrangian (which is a scalar) and the model of the external (non-conservative) forces. First, a concise set of "generalized coordinates", usually denoted by $\boldsymbol{q}$, must be defined. Then we can state the Euler-Lagrange equations as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial L}{\partial \boldsymbol{q}} = \boldsymbol{Q} \tag{6.1}$$

Where the Lagrangian $L = L(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)$ is the difference between the kinetic and potential energies, expressed in terms of the generalized coordinates, and $\boldsymbol{Q}$ is the generalized forces vector. One downside of the direct use of the Lagrange equations is that it requires a choice of parameters beforehand. For a simple rotating system, for instance, both Euler angles or a rotation quaternion can be used. However, this is known to introduce artificial nonlinearities and singularities in the case of UAVs whose configuration Lie group is SO(3) or SE(3): the final equations are

coupled and dependent on the parameters chosen. Moreover, the term $\frac{\partial L}{\partial \boldsymbol{q}}$, the direct derivative of the Lagrangian in terms of $\boldsymbol{q}$, can be very cumbersome. A quaternion-based model of the pure rotation of a single-body using Lagrangian mechanics can be seen in [68]. The term $\boldsymbol{Q}$ that models the generalized forces can also be very complicated to calculate. An effort to express the generalization of this term can be seen in [69].

Any solution to Eq. 6.1 is also a solution to the Euler Poincaré equations [9]. Similarly to the Euler-Lagrange equations, the Euler-Poincaré equations allow us to develop the Equations of Motion of a system submitted to conservative forces in a systematic manner. Unlike the Lagrangian equations, however, in the Euler-Poincaré equations, both the energies and the final dynamic equations are directly expressed in terms of the velocities of the system in its moving frame of reference. In practical terms, the Euler-Poincaré equations provide a systematic energy-based approach that is similar to an analysis based on the Euler-Lagrange equations, while producing the same parameter-agnostic formulations usually derived with the direct application of the Newton-Euler equations, which are free from artificial singularities and nonlinearities. We could, naturally, also model the system by using the Newton-Euler equations for the external degrees of freedom of the system together with the Euler-Lagrange equations for the internal degrees of freedom. Using the Euler-Poincaré equations, however, all the degrees of freedom of the system are captured simultaneously in a unified approach. Moreover, note that the product of two Lie groups is also a Lie group. In order to derive a more complex model that takes into account, for instance, a variable battery distance h_S or a more complete swashplateless mechanism model, this generalization would be straightforward without requiring a profound knowledge of Lie group theory. These are some of the reasons why we believe that the Euler-Poincaré equations, although still largely unknown to the robotics community, are the most natural tool for capturing the dynamics of these types of UAVs.

6.1. Poincaré equation

For simplicity, in this chapter, we will denote $\boldsymbol{\omega} = \boldsymbol{\omega}_B^B$, $\boldsymbol{v} = \boldsymbol{v}_B^B$ and $\boldsymbol{\Omega} = \boldsymbol{\Omega}_{P/B}^P$. The Euler-Poincaré equations on the configuration Lie group SE(3) [1] of our system, are:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T}{\partial \eta}\right) - \mathrm{ad}_\eta^T \left(\frac{\partial T}{\partial \eta}\right) = F_{\text{ext}}^B \tag{6.2}$$

Where the ad operator is given by [40]:

$$\mathrm{ad}_\eta = \begin{bmatrix} \widehat{\boldsymbol{\omega}} & 0 \\ \widehat{\boldsymbol{v}} & \widehat{\boldsymbol{\omega}} \end{bmatrix} \tag{6.3}$$

[1] The Special Euclidean group in 3 dimensions: 3 dimensions of position and a full 3D rotation.

And, as a reminder, from Eq. 5.36:

$$T = \frac{1}{2}\begin{bmatrix}\eta \\ \boldsymbol{\Omega}\end{bmatrix}^T \mathbf{M} \begin{bmatrix}\eta \\ \boldsymbol{\Omega}\end{bmatrix} \tag{6.4}$$

Where:

$$\mathbf{M} = \begin{bmatrix} \boldsymbol{J} & -\Delta m\widehat{\boldsymbol{e}}_z & R_P^B J_p^P \\ \Delta m\widehat{\boldsymbol{e}}_z & m\,\mathbb{I} & 0 \\ \left(R_P^B J_p^P\right)^T & 0 & J_p^P \end{bmatrix} \tag{6.5}$$

And:

$$\boldsymbol{m}_\times \equiv \Delta m \begin{bmatrix} 0 & -\widehat{\boldsymbol{e}}_z \\ \widehat{\boldsymbol{e}}_z & 0 \end{bmatrix} \tag{6.6}$$

Applying the partial derivative w.r.t. the inertial twist gives:

$$\frac{\partial T}{\partial \eta} = \mathbf{D}\eta + \begin{bmatrix} R_P^B J_p^P \boldsymbol{\Omega} \\ \mathbf{0} \end{bmatrix} \tag{6.7}$$

Where:

$$\mathbf{D} = \begin{bmatrix} \boldsymbol{J} & -\Delta m\widehat{\boldsymbol{e}}_z \\ \Delta m\widehat{\boldsymbol{e}}_z & m\,\mathbb{I} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J} & 0 \\ 0 & m\,\mathbb{I} \end{bmatrix} + \boldsymbol{m}_\times \tag{6.8}$$

6.2. Time derivatives

Differentiating 6.7 w.r.t. time yields:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T}{\partial \eta}\right) &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\mathbf{D}\eta + \begin{bmatrix} R_P^B J_p^P \boldsymbol{\Omega} \\ \mathbf{0} \end{bmatrix}\right) \\ &= \dot{\mathbf{D}}\eta + \mathbf{D}\dot{\eta} + \begin{bmatrix} R_P^B \left(\widehat{\boldsymbol{\omega}}_{P/B}^P J_p^P \boldsymbol{\Omega} + J_p^P \dot{\boldsymbol{\Omega}}\right) \\ \mathbf{0} \end{bmatrix} \end{aligned} \tag{6.9}$$

Noting that $\dot{\boldsymbol{J}} = \dot{J_p^B}$:

$$\dot{\mathbf{D}} = \frac{\mathrm{d}}{\mathrm{d}t}\left(\begin{bmatrix} \boldsymbol{J} & 0 \\ 0 & m\,\mathbb{I} \end{bmatrix} + \boldsymbol{m}_\times\right) = \begin{bmatrix} \dot{J_p^B} & 0 \\ 0 & 0 \end{bmatrix} \tag{6.10}$$

In which:

$$\dot{J_p^B} = R_P^B \left(\widehat{\boldsymbol{\omega}}_{P/B}^P J_p^P - J_p^P \widehat{\boldsymbol{\omega}}_{P/B}^P\right)(R_P^B)^T \tag{6.11}$$

Inserting Eqs. 6.7 and 6.9 into 6.2, the left-hand side of the equation can be

rewritten as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T}{\partial \eta}\right) - \mathrm{ad}_{\eta}^{T}\left(\frac{\partial T}{\partial \eta}\right) = \mathbf{D}\dot{\eta} + \mathbf{C}\eta + \mathcal{T} \tag{6.12}$$

Where we define:

$$\mathbf{C} = \dot{\mathbf{D}} - \mathrm{ad}_{\eta}^{T}\mathbf{D} \tag{6.13}$$

And the term $\mathcal{T}$ models the full torque generated by the angular velocity of the rotor:

$$\mathcal{T} = \begin{bmatrix} R_P^B\left(\widehat{\boldsymbol{\omega}}_{P/B}^P J_p^P \boldsymbol{\Omega} + J_p^P \dot{\boldsymbol{\Omega}}\right) \\ \mathbf{0} \end{bmatrix} - \mathrm{ad}_{\eta}^{T}\begin{bmatrix} R_P^B J_p^P \boldsymbol{\Omega} \\ \mathbf{0} \end{bmatrix} \tag{6.14}$$

6.3. Analyzing the rotor inertial torque

We can simplify the term $\mathcal{T}$ by noticing that:

$$\begin{aligned} -\mathrm{ad}_{\eta}^{T}\begin{bmatrix} R_P^B J_p^P \boldsymbol{\Omega} \\ \mathbf{0} \end{bmatrix} &= \begin{bmatrix} \widehat{\boldsymbol{\omega}} R_P^B J_p^P \boldsymbol{\Omega} \\ \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} R_P^B (R_P^B)^T \widehat{\boldsymbol{\omega}} R_P^B J_p^P \boldsymbol{\Omega} \\ \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} R_P^B \left(\boldsymbol{\omega}^P \times J_p^P \boldsymbol{\Omega}\right) \\ \mathbf{0} \end{bmatrix} \end{aligned} \tag{6.15}$$

Where $\boldsymbol{\omega}^P = (R_P^B)^T \boldsymbol{\omega}$ is the main body's angular velocity as seen from the rotor frame $\mathcal{F}_P$. Defining $\boldsymbol{\tau}_{\mathcal{F}_P}^B$ as the torque generated in the main body $\mathcal{B}_b$ by the rotation of the virtual rotor frame $\mathcal{F}_P$:

$$\boldsymbol{\tau}_{\mathcal{F}_P}^B \equiv R_P^B \left(\boldsymbol{\omega}_{P/B}^P \times (J_p^P \boldsymbol{\omega}_{P/B}^P) + J_p^P \dot{\boldsymbol{\omega}}_{P/B}^P\right) \tag{6.16}$$

Inserting the fact that $\boldsymbol{\Omega} = \boldsymbol{\omega}_{P/B}^P + \dot{\gamma}\boldsymbol{e}_z$:

$$\begin{aligned} \begin{bmatrix} R_P^B\left(\widehat{\boldsymbol{\omega}}_{P/B}^P J_p^P \boldsymbol{\Omega} + J_p^P \dot{\boldsymbol{\Omega}}\right) \\ \mathbf{0} \end{bmatrix} &= \begin{bmatrix} R_P^B\left(\widehat{\boldsymbol{\omega}}_{P/B}^P J_p^P \left(\boldsymbol{\omega}_{P/B}^P + \dot{\gamma}\boldsymbol{e}_z\right) + J_p^P\left(\dot{\boldsymbol{\omega}}_{P/B}^P + \ddot{\gamma}\boldsymbol{e}_z\right)\right) \\ \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{\tau}_{\mathcal{F}_P}^B + R_P^B\left(\widehat{\boldsymbol{\omega}}_{P/B}^P J_p^P (\dot{\gamma}\boldsymbol{e}_z) + J_p^P \ddot{\gamma}\boldsymbol{e}_z\right) \\ \mathbf{0} \end{bmatrix} \end{aligned} \tag{6.17}$$

Introducing Eqs. 6.15 and 6.17 into Eq. 6.14:

$$\begin{aligned}\mathcal{T} &= \begin{bmatrix} R_P^B \left(\widehat{\boldsymbol{\omega}}_{P/B}^P J_p^P \boldsymbol{\Omega} + J_p^P \dot{\boldsymbol{\Omega}}\right) \\ \mathbf{0} \end{bmatrix} - \mathrm{ad}_\eta^T \begin{bmatrix} R_P^B J_p^P \boldsymbol{\Omega} \\ \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{\tau}_{\mathcal{F}_P}^B + R_P^B \left(\widehat{\boldsymbol{\omega}}_{P/B}^P J_p^P \left(\dot{\gamma}\boldsymbol{e}_z\right) + J_p^P \ddot{\gamma}\boldsymbol{e}_z + \boldsymbol{\omega}^P \times J_p^P \boldsymbol{\Omega}\right) \\ \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{\tau}_{\mathcal{F}_P}^B + R_P^B \left(\widehat{\boldsymbol{\omega}}_{P/B}^P J_p^P \left(\dot{\gamma}\boldsymbol{e}_z\right) + J_p^P \ddot{\gamma}\boldsymbol{e}_z + \boldsymbol{\omega}^P \times J_p^P \boldsymbol{\omega}_{P/B}^P + \boldsymbol{\omega}^P \times J_p^P \dot{\gamma}\boldsymbol{e}_z\right) \\ \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{\tau}_{\mathcal{F}_P}^B + R_P^B \left(\widehat{\boldsymbol{\omega}}_{P/B}^P J_p^P \left(\dot{\gamma}\boldsymbol{e}_z\right) + \boldsymbol{\omega}^P \times J_p^P \dot{\gamma}\boldsymbol{e}_z + J_p^P \ddot{\gamma}\boldsymbol{e}_z\right) + R_P^B \left(\boldsymbol{\omega}^P \times J_p^P \boldsymbol{\omega}_{P/B}^P\right) \\ \mathbf{0} \end{bmatrix}\end{aligned} \tag{6.18}$$

Noticing that $J_p^P \boldsymbol{e}_z = J_{pz}\boldsymbol{e}_z$ and $R_P^B \boldsymbol{e}_z = \boldsymbol{r}$ we can state:

$$R_P^B \left(\widehat{\boldsymbol{\omega}}_{P/B}^P J_p^P \dot{\gamma}\boldsymbol{e}_z + \widehat{\boldsymbol{\omega}}^P J_p^P \dot{\gamma}\boldsymbol{e}_z + J_p^P \ddot{\gamma}\boldsymbol{e}_z\right) = J_{pz} R_P^B \left(\left(\widehat{\boldsymbol{\omega}}_{P/B}^P + \widehat{\boldsymbol{\omega}}^P\right)\dot{\gamma}\boldsymbol{e}_z + \ddot{\gamma}\boldsymbol{e}_z\right) \tag{6.19}$$

In which we can identify two different terms, one given by the acceleration of the rotor mean velocity:

$$\boldsymbol{\tau}_{\text{acc}}^B = J_{pz}\ddot{\gamma}\,\boldsymbol{r} \tag{6.20}$$

And one extra term generated by the rotor mean velocity:

$$\begin{aligned}\boldsymbol{\tau}_{\text{vel}}^B &= J_{pz} R_P^B \left(\widehat{\boldsymbol{\omega}}^P + \widehat{\boldsymbol{\omega}}_{P/B}^P\right)\dot{\gamma}\boldsymbol{e}_z \\ &= J_{pz}\dot{\gamma} R_P^B \left(\widehat{\boldsymbol{\omega}}^P + \widehat{\boldsymbol{\omega}}_{P/B}^P\right)(R_P^B)^T R_P^B \boldsymbol{e}_z \\ &= J_{pz}\dot{\gamma}\left(\boldsymbol{\omega} + \boldsymbol{\omega}_{B/P}^B\right) \times \boldsymbol{r}\end{aligned} \tag{6.21}$$

Where $\boldsymbol{\omega}_{B/P}^B = R_P^B \boldsymbol{\omega}_{P/B}^P$ is the angular velocity of between $\mathcal{F}_P$ and $\mathcal{F}_B$ expressed in the body frame $\mathcal{F}_B$. We can finally express the torque term $\mathcal{T}$ as:

$$\mathcal{T} = \begin{bmatrix} \boldsymbol{\tau}_{\mathcal{F}_P}^B + \boldsymbol{\tau}_{\text{vel}}^B + \boldsymbol{\tau}_{\text{acc}}^B \\ \mathbf{0} \end{bmatrix} \tag{6.22}$$

6.4. Complete Equations of Motion

To summarize, the complete Equations of Motion of the monorotor drone in Eq. 6.2 can finally be stated as:

$$\mathbf{D}\dot{\eta} + \mathbf{C}\eta + \mathcal{T} = F_{\text{ext}}^{B} \tag{6.23}$$

In which:

$$\begin{aligned} \mathbf{D} &= \begin{bmatrix} \boldsymbol{J} & 0 \\ 0 & m\,\mathbb{I} \end{bmatrix} + \boldsymbol{m}_{\times} \\ \mathbf{C} &= \dot{\mathbf{D}} - \mathrm{ad}_{\eta}^{T}\mathbf{D} \\ F_{\text{ext}}^{B} &= \begin{bmatrix} f_p \boldsymbol{B}_{\tau}\boldsymbol{r} + \Delta m\, g\, \boldsymbol{e}_z \times \boldsymbol{n}_b^E - |\boldsymbol{\omega}|\boldsymbol{K}\boldsymbol{\omega} \\ f_p \boldsymbol{r} - mg\, \boldsymbol{n}_b^E \end{bmatrix} \end{aligned} \tag{6.24}$$

Where, as a reminder:

$$\begin{aligned} m &= m_b + m_s + m_p \\ \Delta m &= m_s h_S - m_p h_P \\ \boldsymbol{m}_{\times} &= \Delta m \begin{bmatrix} 0 & -\widehat{\boldsymbol{e}}_z \\ \widehat{\boldsymbol{e}}_z & 0 \end{bmatrix} \\ \mathrm{ad}_{\eta} &= \begin{bmatrix} \widehat{\boldsymbol{\omega}} & 0 \\ \widehat{\boldsymbol{v}} & \widehat{\boldsymbol{\omega}} \end{bmatrix} \end{aligned} \tag{6.25}$$

And the extra torque terms from the rotor are:

$$\mathcal{T} = \begin{bmatrix} \boldsymbol{\tau}_{\mathcal{F}_P}^{B} + \boldsymbol{\tau}_{\text{vel}}^{B} + \boldsymbol{\tau}_{\text{acc}}^{B} \\ \mathbf{0} \end{bmatrix} \tag{6.26}$$

With:

$$\begin{aligned} \boldsymbol{\tau}_{\mathcal{F}_P}^{B} &= R_P^B \left(\boldsymbol{\omega}_{P/B}^{P} \times (J_p^P \boldsymbol{\omega}_{P/B}^{P}) + J_p^P\, \dot{\boldsymbol{\omega}}_{P/B}^{P} \right) \\ \boldsymbol{\tau}_{\text{acc}}^{B} &= \ddot{\gamma} J_{pz}\, \boldsymbol{r} \\ \boldsymbol{\tau}_{\text{vel}}^{B} &= \dot{\gamma} J_{pz} \left(\boldsymbol{\omega} + \boldsymbol{\omega}_{B/P}^{B} \right) \times \boldsymbol{r} \end{aligned} \tag{6.27}$$

6.5. Integration

Isolating the angular accelerations, we have the following equation, which will be useful for the simulation implementation:

$$\begin{aligned} \dot{\eta} &= \mathbf{D}^{-1}\left(F_{\text{ext}}^{B} - \mathbf{C}\eta - \mathcal{T}\right) \\ \begin{bmatrix} \dot{\boldsymbol{\omega}} \\ \dot{\boldsymbol{v}} \end{bmatrix} &= \mathbf{D}^{-1}\left(F_{\text{ext}}^{B} - \mathbf{C}\begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} - \mathcal{T}\right) \end{aligned} \tag{6.28}$$

Which, as expected, expresses the evolution of the system as a function of its velocities. For $\widehat{\eta}$ as defined in Eq. A.12:

$$\widehat{\eta} = \widehat{\begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix}} = \begin{bmatrix} \widehat{\boldsymbol{\omega}} & \boldsymbol{v} \\ 0 & 0 \end{bmatrix} \tag{6.29}$$

For $g = g_B^E$, $\boldsymbol{s} = \boldsymbol{s}_B^E$ and $R = R_B^E$, we can use the reconstruction equations:

$$\begin{aligned} \dot{g}_B^E &= g\,\widehat{\eta} \\ \begin{bmatrix} \dot{R} & \dot{\boldsymbol{s}} \\ \mathbf{0}^T & 0 \end{bmatrix} &= \begin{bmatrix} R & \boldsymbol{s} \\ \mathbf{0}^T & 1 \end{bmatrix}\begin{bmatrix} \widehat{\boldsymbol{\omega}} & \boldsymbol{v} \\ \mathbf{0}^T & 0 \end{bmatrix} \\ \begin{bmatrix} \dot{R} & \dot{\boldsymbol{s}} \\ \mathbf{0}^T & 0 \end{bmatrix} &= \begin{bmatrix} R\widehat{\boldsymbol{\omega}} & R\boldsymbol{v} \\ \mathbf{0}^T & 0 \end{bmatrix} \end{aligned} \tag{6.30}$$

And finally:

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{s}}{\mathrm{d}t} &= R\boldsymbol{v} \\ \frac{\mathrm{d}R}{\mathrm{d}t} &= R\widehat{\boldsymbol{\omega}} \end{aligned} \tag{6.31}$$

Instead of directly using rotation matrices however, we parameterize the orientation of the drone using quaternions (more precisely, classical Hamilton quaternions [63]). Quaternions are an extension of the complex numbers, consisting of 4 components, and form an excellent set of parameters for rotations, being fast to compute, easy to normalize and having no singularity-related problems [32]. A summary of quaternion algebra can be seen in Appendix A. In this work, we will denote quaternions as 4-vectors. Supposing $q = q_B^E$ represents the rotation from $\mathcal{F}_B$ to $\mathcal{F}_E$:

$$q = \begin{bmatrix} q_{\text{r}} \\ \boldsymbol{q} \end{bmatrix} \tag{6.32}$$

Where q_r and $\boldsymbol{q}$ are, respectively, the scalar (or real) and vector (or imaginary) parts of q. Then, for some vector $\boldsymbol{a} \in \mathbb{R}^3$:

$$\begin{bmatrix} 0 \\ \boldsymbol{a}^E \end{bmatrix} = q \odot \begin{bmatrix} 0 \\ \boldsymbol{a}^B \end{bmatrix} \odot q^* \tag{6.33}$$

Where $\odot$ is the Hamilton product (see Appendix A). Finally, to integrate the orientation quaternion, we can use the following rotation reconstruction equation:

$$\dot{q} = \frac{1}{2} q \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \tag{6.34}$$

6.6. Simplified equation for simulation

As expressed in Chapter 5, α, $\dot{\alpha}$ and $\dot{\beta}$ are very small compared to $\dot{\gamma}$. Therefore, we consider $\boldsymbol{\omega}^P_{P/B} \approx 0$, leading to Eq. 5.16:

$$\boldsymbol{\Omega}^P_{P/B} \approx \dot{\gamma} \boldsymbol{e}_z \tag{6.35}$$

Apart from that, we also assume that the mean rotor velocity should not change much from the set rover velocity, leading to $\ddot{\gamma} \approx 0$. Both of these approximations are possible because the inertia of the blades is very small[2].

These assumptions greatly simplifies the simulations, since we assume we can control R^B_P instantaneously. Moreover, this leads to:

$$\begin{aligned} \boldsymbol{\tau}^B_{\mathcal{F}_P} &\approx \mathbf{0} \\ \boldsymbol{\tau}^B_{\mathrm{acc}} &\approx \mathbf{0} \\ \boldsymbol{\tau}^B_{\mathrm{vel}} &\approx \dot{\gamma} J_{pz}\, \boldsymbol{\omega} \times \boldsymbol{r} \\ \mathbf{C} &\approx -\mathrm{ad}^T_\eta \mathbf{D} \end{aligned} \tag{6.36}$$

Leading to the simplified equations used in [4] and in Chapter 9:

$$\mathbf{D} \begin{bmatrix} \dot{\boldsymbol{\omega}} \\ \dot{\boldsymbol{v}} \end{bmatrix} + \mathbf{C} \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} + \begin{bmatrix} \dot{\gamma} J_{pz}\, \boldsymbol{\omega} \times \boldsymbol{r} \\ 0 \end{bmatrix} = F^B_{\mathrm{ext}} \tag{6.37}$$

And isolating for integration:

$$\begin{bmatrix} \dot{\boldsymbol{\omega}} \\ \dot{\boldsymbol{v}} \end{bmatrix} = \mathbf{D}^{-1} \left(F^B_{\mathrm{ext}} + \mathrm{ad}^T_\eta \mathbf{D} \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} - \begin{bmatrix} \dot{\gamma} J_{pz}\, \boldsymbol{\omega} \times \boldsymbol{r} \\ 0 \end{bmatrix} \right) \tag{6.38}$$

[2] Note that the thrust and torque from the rotor, both taken into account as external forces, come from air displacement, not from inertia.

6.7. Final thoughts

In this chapter, we applied the Euler-Poincaré equation to the monorotor presented in Chapter 5, in order to get the final expression of its Equations of Motion. The use of this method greatly simplified the derivation of the EOM, providing us with a simpler and systematic method that produces a complete solution without relying on too many approximations.

The Euler-Poincaré equation is a powerful mathematical tool, albeit largely unknown from the literature. For this reason, we also believe that this analysis contributes with an interesting application of this method, and hope it can have some (small) effect in popularizing it as an alternative to classical Lagrangian mechanics.

7. Quaternion decomposition into normal vector and spin angle

In this chapter, I will study the problem of decomposing a quaternion rotation into two separate quaternions, in order to separate the component representing the uncontrollable spin. This decomposition was, at first, inspired by the decomposition idea used in [70] to implement a complementary filter to estimate a robot's orientation. I will present here a complete generalization of this decomposition, provide a simple closed formula for it and analyze the resulting components.[1]

7.1. Decomposition into two quaternions

Suppose two frames, a body-fixed frame $\mathcal{F}_B$ and another frame $\mathcal{F}_E$ (for example, the inertial frame). We have a body that spins around an axis $\boldsymbol{e}_b$. In the body frame, this vector is equivalent to a constant unit vector $\boldsymbol{e}_b^B \equiv \boldsymbol{e}$. Suppose $q = [q_\mathrm{r},\ \boldsymbol{q}^T]^T$ is the unit quaternion that defines the rotation from the body frame $\mathcal{F}_B$ to the inertial frame $\mathcal{F}_E$. If we note $\boldsymbol{e}_b^E$ as the vector $\boldsymbol{e}_b$ in the inertial frame, we know that:

$$\begin{aligned}\begin{bmatrix}0\\ \boldsymbol{e}_b^E\end{bmatrix} &= q \odot \begin{bmatrix}0\\ \boldsymbol{e}_b^B\end{bmatrix} \odot q^* \\ &= q \odot \begin{bmatrix}0\\ \boldsymbol{e}\end{bmatrix} \odot q^*\end{aligned} \tag{7.1}$$

Or simply: $\boldsymbol{e}_b^E = R\boldsymbol{e}_b^B = R\boldsymbol{e}$, where $R = R(q)$, and as seen in Eq. A.55:

$$\boldsymbol{e}_b^E = \left(\mathbb{I} + 2\left(q_\mathrm{r}\widehat{\boldsymbol{q}} + \widehat{\boldsymbol{q}}^2\right)\right)\boldsymbol{e} \tag{7.2}$$

And, as defined in Eq. A.2:

$$\widehat{\boldsymbol{q}} = \begin{bmatrix}0 & -q_\mathrm{z} & q_\mathrm{y}\\ q_\mathrm{z} & 0 & -q_\mathrm{x}\\ -q_\mathrm{y} & q_\mathrm{x} & 0\end{bmatrix} \tag{7.3}$$

[1] This chapter was originally submitted as an appendix in [4].

Moreover, notating $\boldsymbol{\omega}$ as the angular velocity of the body in the body frame, we know that:

$$\dot{q} = \frac{1}{2} q \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \tag{7.4}$$

We will suppose the existence of two unit quaternions, q_a (called here the "reduced attitude quaternion") and q_s ("spin quaternion") such that:

$$\begin{aligned} q_a &= \begin{bmatrix} a_\mathrm{r} \\ \boldsymbol{a} \end{bmatrix} \\ q_s &= \begin{bmatrix} s_\mathrm{r} \\ \boldsymbol{s} \end{bmatrix} \\ q &= q_a \odot q_s \end{aligned} \tag{7.5}$$

We also define $\boldsymbol{\omega}_a$ and $\boldsymbol{\omega}_s$ as the angular velocities of both q_a and q_s such that:

$$\begin{aligned} \dot{q}_a &= \frac{1}{2} q_a \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega}_a \end{bmatrix} \\ \dot{q}_s &= \frac{1}{2} q_s \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega}_s \end{bmatrix} \end{aligned} \tag{7.6}$$

And we add the constraint that both q_a and q_s are unit quaternions:

$$\begin{aligned} |q_a|^2 &= a_\mathrm{r}^2 + |\boldsymbol{a}|^2 = 1 \\ |q_s|^2 &= s_\mathrm{r}^2 + |\boldsymbol{s}|^2 = 1 \end{aligned} \tag{7.7}$$

7.2. Attitude quaternion

Since the quaternion q_a has four components, but the spin is already represented by q_s, a new constraint must be chosen for q_a, otherwise the system is underdetermined. A natural choice for this constraint, to assure that no rotation around $\boldsymbol{e}$ is done with q_a, is:

$$\boldsymbol{a} \cdot \boldsymbol{e} = 0 \tag{7.8}$$

Using this constraint:

$$\begin{aligned} q_a \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} &= \begin{bmatrix} -\boldsymbol{e} \cdot \boldsymbol{a} \\ a_\mathrm{r} \boldsymbol{e} + \boldsymbol{a} \times \boldsymbol{e} \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ (a_\mathrm{r} \mathbb{I} + \widehat{\boldsymbol{a}})\, \boldsymbol{e} \end{bmatrix} \end{aligned} \tag{7.9}$$

So, defining $\boldsymbol{m} = (a_{\mathrm{r}}\mathbb{I} + \widehat{\boldsymbol{a}})\,\boldsymbol{e}$, we know that q_a has the form:

$$\begin{aligned} q_a &= \begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix}^* \\ q_a &= -\begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \end{aligned} \tag{7.10}$$

Moreover, since $|q_a| = |\boldsymbol{e}| = 1$, we know that $|\boldsymbol{m}| = 1$. Introducing Eq. 7.10 into Eq. 7.1:

$$\begin{aligned} \begin{bmatrix} 0 \\ \boldsymbol{e}_b^E \end{bmatrix} &= q \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \odot q^* \\ &= q_a \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \odot q_a^* \\ &= \begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \\ &= -\begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \end{aligned} \tag{7.11}$$

Multiplying both sides on the right by $-\begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix}$:

$$\begin{aligned} -\begin{bmatrix} 0 \\ \boldsymbol{e}_b^E \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} &= -\begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e}_b^B \end{bmatrix} \\ \begin{bmatrix} \boldsymbol{m} \cdot \boldsymbol{e}_b^B \\ \boldsymbol{m} \times \boldsymbol{e}_b^B \end{bmatrix} &= \begin{bmatrix} \boldsymbol{e}_b^E \cdot \boldsymbol{m} \\ \boldsymbol{e}_b^E \times \boldsymbol{m} \end{bmatrix} \end{aligned} \tag{7.12}$$

From Eq. 7.12 we know that:

$$\begin{aligned} \boldsymbol{m} \times \boldsymbol{e}_b^B &= \boldsymbol{e}_b^E \times \boldsymbol{m} \\ \boldsymbol{m} \times \left(\boldsymbol{e}_b^E + \boldsymbol{e}_b^B\right) &= \boldsymbol{0} \\ \boldsymbol{m} \times \left((R + \mathbb{I})\,\boldsymbol{e}\right) &= \boldsymbol{0} \end{aligned} \tag{7.13}$$

Which means that, for some constant λ, we know that:

$$\boldsymbol{m} = \lambda\left(\boldsymbol{e}_b^E + \boldsymbol{e}_b^B\right) = \lambda\left((R + \mathbb{I})\,\boldsymbol{e}\right) \tag{7.14}$$

Choosing $\lambda > 0$[2] and noting that $|\boldsymbol{m}| = 1$:

$$\lambda = \frac{|\boldsymbol{m}|}{|(R + \mathbb{I})\,\boldsymbol{e}|} = \frac{1}{|(R + \mathbb{I})\,\boldsymbol{e}|} \tag{7.15}$$

[2]Choosing $\lambda < 0$ would lead to the same final solution.

And finally:

$$\boldsymbol{m} = \frac{\boldsymbol{e}_b^E + \boldsymbol{e}_b^B}{|\boldsymbol{e}_b^E + \boldsymbol{e}_b^B|} = \frac{(R + \mathbb{I})\,\boldsymbol{e}}{|(R + \mathbb{I})\,\boldsymbol{e}|} \tag{7.16}$$

We note that $\boldsymbol{m} = \boldsymbol{m}^E$ is the unit vector exactly in the middle of the normal vector before and the normal vector after the rotation, as seen in $\mathcal{F}_E$. We denote $\boldsymbol{m}$ as the "**middle normal vector**". This is directly related to the fact that the rotation quaternion's real and imaginary parts are respectively cosines and sines of **half** of the final rotation angle. Moreover:

$$\begin{aligned} \left|\boldsymbol{e}_b^E + \boldsymbol{e}_b^B\right| &= \sqrt{2\,(\boldsymbol{e}_b^B \cdot \boldsymbol{e}_b^E + 1)} \\ &= \sqrt{2\,(\boldsymbol{e} \cdot R\boldsymbol{e} + 1)} \\ &= 2\sqrt{1 - |\boldsymbol{q} \times \boldsymbol{e}|^2} \\ &= 2\sqrt{q_\mathrm{r}^2 + (\boldsymbol{q} \cdot \boldsymbol{e})^2} \end{aligned} \tag{7.17}$$

Finally, the attitude quaternion q_a can be given as:

$$q_a = -\begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \tag{7.18}$$

Note that this equation has a singularity when $\boldsymbol{e}_b^E = -\boldsymbol{e}_b^B = -\boldsymbol{e}$ ($\boldsymbol{m}$ is undetermined). This happens because there are infinite rotations of 180° that transform $\boldsymbol{e}$ into $-\boldsymbol{e}$. In order to address this in real-time, we can check if:

$$\boldsymbol{m} \cdot \boldsymbol{e} \approx 0 \tag{7.19}$$

In these cases, we can simply update the measurement of $\boldsymbol{m}$ by removing its (small) component in the direction of $\boldsymbol{e}$ and normalizing it:

$$\boldsymbol{m}_{k+1} \leftarrow \frac{\boldsymbol{m}_k - (\boldsymbol{m}_k \cdot \boldsymbol{e})\boldsymbol{e}}{|\boldsymbol{m}_k - (\boldsymbol{m}_k \cdot \boldsymbol{e})\boldsymbol{e}|} \tag{7.20}$$

We can see on Fig. 7.1 the domains of both $\boldsymbol{e}_b^E = R\boldsymbol{e}$ and $\boldsymbol{m}$.

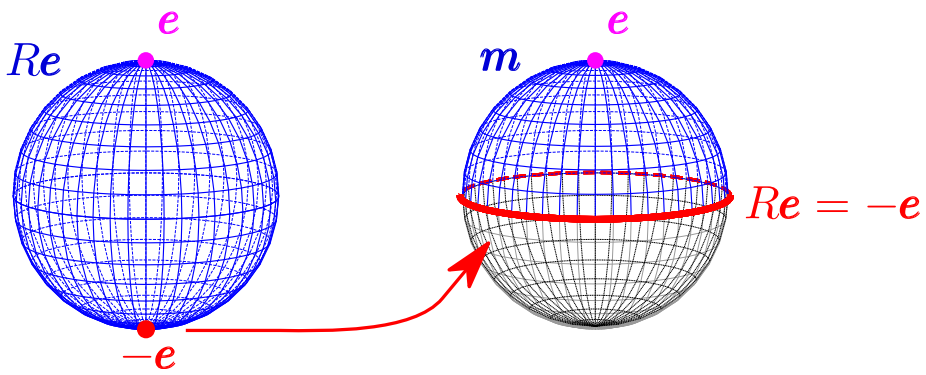


Figure 7.1.: On the left, the domain (in blue) of Re, with a red dot representing the singularity point where $Re = -e$. On the right, the upper hemisphere represents the domain of m. We can see that the whole equator between both hemispheres correspond to the case $Re = -e$, which means that m is undefined in this case.

7.3. Spin quaternion

Considering that q_s is the quaternion representing only the rotation that creates the constant spin around e:

$$\begin{aligned} q_s \odot \begin{bmatrix} 0 \\ e \end{bmatrix} \odot q_s^* &= \begin{bmatrix} 0 \\ e \end{bmatrix} \\ q_s \odot \begin{bmatrix} 0 \\ e \end{bmatrix} &= \begin{bmatrix} 0 \\ e \end{bmatrix} \odot q_s \\ \begin{bmatrix} e \cdot s \\ e \times s \end{bmatrix} &= \begin{bmatrix} e \cdot s \\ -e \times s \end{bmatrix} \end{aligned} \tag{7.21}$$

Which gives $e \times s = -e \times s$, meaning that:

$$e \times s = 0 \tag{7.22}$$

So we know that, for a real scalar s:

$$s = se \tag{7.23}$$

And noting $s_\mathrm{r} = c$, we can rewrite q_s as:

$$q_s = \begin{bmatrix} c \\ se \end{bmatrix} \tag{7.24}$$

With the constraint that $|q_s|^2 = c^2 + s^2 = 1$. Defining a spin angle θ_s, this leads to the natural definition as:

$$\begin{aligned} c &= \cos(\theta_s/2) \\ s &= \sin(\theta_s/2) \end{aligned} \tag{7.25}$$

But to find these parameters for a given q, we can use 7.5 and 7.18 to find that:

$$\begin{aligned} q_s &= q_a^* \odot q \\ |(R+\mathbb{I})\,\boldsymbol{e}| \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \odot \begin{bmatrix} c \\ s\boldsymbol{e} \end{bmatrix} &= \begin{bmatrix} 0 \\ (R+\mathbb{I})\,\boldsymbol{e} \end{bmatrix} \odot \begin{bmatrix} q_\mathrm{r} \\ \boldsymbol{q} \end{bmatrix} \\ |(R+\mathbb{I})\,\boldsymbol{e}| \begin{bmatrix} -s \\ c\boldsymbol{e} \end{bmatrix} &= \begin{bmatrix} -\boldsymbol{q}\cdot(R+\mathbb{I})\,\boldsymbol{e} \\ (q_\mathrm{r}\mathbb{I} - \widehat{\boldsymbol{q}})\,(R+\mathbb{I})\,\boldsymbol{e} \end{bmatrix} \end{aligned} \tag{7.26}$$

Remembering Eq. 7.2:

$$\begin{aligned} (R+\mathbb{I})\,\boldsymbol{e} &= 2\,(\mathbb{I} + (q_\mathrm{r}\mathbb{I} + \widehat{\boldsymbol{q}})\,\widehat{\boldsymbol{q}})\,\boldsymbol{e} \\ &= 2\left(\mathbb{I} + q_\mathrm{r}\widehat{\boldsymbol{q}} + \widehat{\boldsymbol{q}}^2\right)\boldsymbol{e} \end{aligned} \tag{7.27}$$

To find s:

$$\begin{aligned} |\boldsymbol{e}_b^E + \boldsymbol{e}_b^B|s &= \boldsymbol{q}\cdot(\boldsymbol{e}_b^E + \boldsymbol{e}_b^B) \\ &= 2\boldsymbol{q}\cdot(\mathbb{I} + \widehat{\boldsymbol{q}}\,(q_\mathrm{r}\mathbb{I} + \widehat{\boldsymbol{q}}))\,\boldsymbol{e} \\ &= 2\left(\boldsymbol{q}\cdot\boldsymbol{e} + \left(\boldsymbol{q}^T\widehat{\boldsymbol{q}}\right)(q_\mathrm{r}\mathbb{I} + \widehat{\boldsymbol{q}})\,\boldsymbol{e}\right) \\ &= 2\boldsymbol{q}\cdot\boldsymbol{e} \end{aligned} \tag{7.28}$$

And finally:

$$s = \frac{2}{|(R+\mathbb{I})\boldsymbol{e}|}\,\boldsymbol{q}\cdot\boldsymbol{e} \tag{7.29}$$

For c:

$$|(R+\mathbb{I})\,\boldsymbol{e}|\,c\boldsymbol{e} = (q_\mathrm{r}\mathbb{I} - \widehat{\boldsymbol{q}})\,(R+\mathbb{I})\,\boldsymbol{e} \tag{7.30}$$

After some tedious algebra, we find that:

$$\begin{aligned} |(R+\mathbb{I})\,\boldsymbol{e}|\,c &= 2\boldsymbol{e}^T\,(q_\mathrm{r}\mathbb{I} + 2\widehat{\boldsymbol{q}})\,\boldsymbol{e} \\ &= 2q_\mathrm{r} \end{aligned} \tag{7.31}$$

Which gives:

$$c = \frac{2}{|(R+\mathbb{I})\,\boldsymbol{e}|}\,q_\mathrm{r} \tag{7.32}$$

And noting that $c^2 + s^2 = 1$ leads to $|(R + \mathbb{I})\,\boldsymbol{e}| = 2\sqrt{q_\mathrm{r}^2 + (\boldsymbol{q}\cdot\boldsymbol{e})^2}$:

$$\begin{aligned} q_s &= \frac{2}{|(R+\mathbb{I})\,\boldsymbol{e}|}\begin{bmatrix} q_\mathrm{r} \\ (\boldsymbol{q}\cdot\boldsymbol{e})\boldsymbol{e}\end{bmatrix} \\ &= \frac{1}{\sqrt{q_\mathrm{r}^2 + (\boldsymbol{q}\cdot\boldsymbol{e})^2}}\begin{bmatrix} q_\mathrm{r} \\ (\boldsymbol{q}\cdot\boldsymbol{e})\boldsymbol{e}\end{bmatrix} \end{aligned} \tag{7.33}$$

Which gives:

$$\theta_s = 2\ \mathrm{atan2}(\boldsymbol{q}\cdot\boldsymbol{e},\ q_\mathrm{r}) \tag{7.34}$$

In Appendix K, we demonstrate an equivalence between θ_s and Proper Euler angles. We can also note that:

$$\begin{bmatrix} 0 \\ \boldsymbol{m}\end{bmatrix} = q \odot \begin{bmatrix} c \\ -s\boldsymbol{e}\end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e}\end{bmatrix} \tag{7.35}$$

Which gives:

$$\begin{aligned} \boldsymbol{m} &= s\boldsymbol{q} + c(q_\mathrm{r}\mathbb{I} + \widehat{\boldsymbol{q}})\boldsymbol{e} \\ &= \frac{(\boldsymbol{q}\cdot\boldsymbol{e})\boldsymbol{q} + q_\mathrm{r}(q_\mathrm{r}\mathbb{I} + \widehat{\boldsymbol{q}})\boldsymbol{e}}{\sqrt{q_\mathrm{r}^2 + (\boldsymbol{q}\cdot\boldsymbol{e})^2}} \end{aligned} \tag{7.36}$$

Taking the dot product of Eq. 7.36 with $\boldsymbol{e}$ gives:

$$\begin{aligned} \boldsymbol{m}\cdot\boldsymbol{e} &= \frac{(\boldsymbol{q}\cdot\boldsymbol{e})\boldsymbol{q}\cdot\boldsymbol{e} + q_\mathrm{r}(q_\mathrm{r}\boldsymbol{e}\cdot\boldsymbol{e} + (\boldsymbol{q}\times\boldsymbol{e})\cdot\boldsymbol{e})}{\sqrt{q_\mathrm{r}^2 + (\boldsymbol{q}\cdot\boldsymbol{e})^2}} \\ &= \frac{(\boldsymbol{q}\cdot\boldsymbol{e})^2 + q_\mathrm{r}^2}{\sqrt{q_\mathrm{r}^2 + (\boldsymbol{q}\cdot\boldsymbol{e})^2}} = \sqrt{q_\mathrm{r}^2 + (\boldsymbol{q}\cdot\boldsymbol{e})^2} \end{aligned} \tag{7.37}$$

Which leads to the interesting relationship:

$$|\boldsymbol{e}_b^E + \boldsymbol{e}_b^B| = 2\boldsymbol{m}\cdot\boldsymbol{e} \tag{7.38}$$

7.4. Derivatives

Just as the orientation quaternion q can be decomposed into two components, the angular velocity $\boldsymbol{\omega}$ can also be decomposed as the angular velocity of these two components. We will study this in this section.

7.4.1. Derivative of spin quaternion

Analyzing the derivative of q_s:

$$\begin{aligned}\dot{q}_s &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\begin{bmatrix}\cos(\theta_s/2)\\ \sin(\theta_s/2)\boldsymbol{e}\end{bmatrix}\right)\\ &= \begin{bmatrix}-\sin(\theta_s/2)\\ \cos(\theta_s/2)\boldsymbol{e}\end{bmatrix}\frac{\dot{\theta}_s}{2}\\ &= \frac{1}{2}\begin{bmatrix}\cos(\theta_s/2)\\ \sin(\theta_s/2)\boldsymbol{e}\end{bmatrix}\odot\begin{bmatrix}0\\ \dot{\theta}_s\boldsymbol{e}\end{bmatrix}\end{aligned} \tag{7.39}$$

And defining $\boldsymbol{\omega}_s = \dot{\theta}_s\boldsymbol{e}$, we can finally write:

$$\dot{q}_s = \frac{1}{2}q_s\odot\begin{bmatrix}0\\ \boldsymbol{\omega}_s\end{bmatrix} \tag{7.40}$$

Using Eq. 7.34:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}(\tan\theta_s/2) &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\boldsymbol{q}\cdot\boldsymbol{e}}{q_\mathrm{r}}\right)\\ \frac{1}{2c^2}\dot{\theta}_s &= \frac{q_\mathrm{r}\dot{\mathbf{q}}_v - \dot{q}_r\boldsymbol{q}}{q_r^2}\cdot\boldsymbol{e}\end{aligned} \tag{7.41}$$

Moreover, we know that:

$$\begin{aligned}\dot{q} = \begin{bmatrix}\dot{q}_r\\ \dot{\mathbf{q}}_v\end{bmatrix} &= \frac{1}{2}\begin{bmatrix}q_\mathrm{r}\\ \boldsymbol{q}\end{bmatrix}\odot\begin{bmatrix}0\\ \boldsymbol{\omega}\end{bmatrix}\\ &= \frac{1}{2}\begin{bmatrix}-\boldsymbol{\omega}\cdot\boldsymbol{q}\\ (q_\mathrm{r}\mathbb{I}+\widehat{\boldsymbol{q}})\boldsymbol{\omega}\end{bmatrix}\end{aligned}$$

Which gives:

$$\begin{aligned}\frac{q_r^2}{c^2}\dot{\theta}_s &= (q_\mathrm{r}(q_\mathrm{r}\mathbb{I}+\widehat{\boldsymbol{q}})\boldsymbol{\omega} + (\boldsymbol{\omega}\cdot\boldsymbol{q})\boldsymbol{q})\cdot\boldsymbol{e}\\ &= q_\mathrm{r}(q_\mathrm{r}\mathbb{I}+\widehat{\boldsymbol{q}}\boldsymbol{\omega})\cdot\boldsymbol{e} + (\boldsymbol{\omega}\cdot\boldsymbol{q})(\boldsymbol{q}\cdot\boldsymbol{e})\\ &= \boldsymbol{\omega}^T q_\mathrm{r}(q_\mathrm{r}\mathbb{I}+\widehat{\boldsymbol{q}})^T\boldsymbol{e} + \boldsymbol{\omega}^T\boldsymbol{q}\boldsymbol{q}^T\boldsymbol{e}\\ &= \boldsymbol{\omega}^T\left(q_\mathrm{r}(q_\mathrm{r}\mathbb{I}-\widehat{\boldsymbol{q}}) + \boldsymbol{q}\boldsymbol{q}^T\right)\boldsymbol{e}\\ &= \boldsymbol{\omega}^T\left((q_\mathrm{r}^2+|\boldsymbol{q}|^2)\mathbb{I} - q_\mathrm{r}\widehat{\boldsymbol{q}} + \widehat{\boldsymbol{q}}^2\right)\boldsymbol{e}\\ &= \boldsymbol{\omega}^T\left(\mathbb{I} - q_\mathrm{r}\widehat{\boldsymbol{q}} + \widehat{\boldsymbol{q}}^2\right)\boldsymbol{e}\end{aligned} \tag{7.42}$$

And noting that:

$$\begin{aligned}\mathbb{I} - q_{\mathrm{r}}\widehat{\boldsymbol{q}} + \widehat{\boldsymbol{q}}^2 &= \frac{2\mathbb{I} - 2q_{\mathrm{r}}\widehat{\boldsymbol{q}} + 2\widehat{\boldsymbol{q}}^2}{2} \\ &= \frac{\mathbb{I} + \left(\mathbb{I} + 2q_{\mathrm{r}}\widehat{\boldsymbol{q}} + 2\widehat{\boldsymbol{q}}^2\right)^T}{2} \\ &= \frac{\mathbb{I} + R^T}{2} \\ &= \frac{R^T(\mathbb{I} + R)}{2}\end{aligned} \tag{7.43}$$

And according to Eqs. 7.27 and 7.32:

$$\begin{aligned}\dot{\theta}_s &= \frac{c^2}{q_{\mathrm{r}}^2}\boldsymbol{\omega} \cdot \frac{R^T(\boldsymbol{e}_b^E + \boldsymbol{e}_b^B)}{2} \\ \dot{\theta}_s &= 2\,\frac{(\boldsymbol{e}_b^E + \boldsymbol{e}_b^B)}{|\boldsymbol{e}_b^E + \boldsymbol{e}_b^B|^2} \cdot R\boldsymbol{\omega} \\ &= \frac{\boldsymbol{m} \cdot R\boldsymbol{\omega}}{\boldsymbol{m} \cdot \boldsymbol{e}}\end{aligned} \tag{7.44}$$

And we can finally state that:

$$\begin{aligned}\boldsymbol{\omega}_s &= 2\,\left(\frac{(\boldsymbol{e}_b^E + \boldsymbol{e}_b^B) \cdot R\boldsymbol{\omega}}{|\boldsymbol{e}_b^E + \boldsymbol{e}_b^B|^2}\right)\,\boldsymbol{e} \\ &= \left(\frac{\boldsymbol{m} \cdot R\boldsymbol{\omega}}{\boldsymbol{m} \cdot \boldsymbol{e}}\right)\boldsymbol{e}\end{aligned} \tag{7.45}$$

7.4.2. Derivative of $\boldsymbol{m}$

By definition:

$$\begin{bmatrix}0 \\ \boldsymbol{m}\end{bmatrix} = -q \odot q_s^* \odot \begin{bmatrix}0 \\ \boldsymbol{e}\end{bmatrix}^* \tag{7.46}$$

Which leads to:

$$\begin{aligned}
\begin{bmatrix} 0 \\ \dot{\boldsymbol{m}} \end{bmatrix} &= -\dot{q} \odot q_s^* \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix}^* - q \odot \dot{q}_s^* \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix}^* \\
&= -\frac{1}{2} q \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \odot q_s^* \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix}^* + \frac{1}{2} q \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega}_s \end{bmatrix} \odot q_s^* \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix}^* \\
&= -\frac{1}{2} q \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega} - \boldsymbol{\omega}_s \end{bmatrix} q_s^* \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix}^* \\
&= \frac{1}{2} q \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega} - \boldsymbol{\omega}_s \end{bmatrix} q^* \odot \begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \\
&= \frac{1}{2} \begin{bmatrix} 0 \\ R(\boldsymbol{\omega} - \boldsymbol{\omega}_s) \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix}
\end{aligned} \tag{7.47}$$

Which gives:

$$\begin{aligned}
\begin{bmatrix} 0 \\ \dot{\boldsymbol{m}} \end{bmatrix} &= \frac{1}{2} \begin{bmatrix} 0 \\ R(\boldsymbol{\omega} - \boldsymbol{\omega}_s) \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \\
&= \frac{1}{2} \begin{bmatrix} -\boldsymbol{m} \cdot R(\boldsymbol{\omega} - \boldsymbol{\omega}_s) \\ -\boldsymbol{m} \times R(\boldsymbol{\omega} - \boldsymbol{\omega}_s) \end{bmatrix}
\end{aligned} \tag{7.48}$$

Which finally gives:

$$\dot{\boldsymbol{m}} = -\frac{1}{2} \boldsymbol{m} \times R\left(\boldsymbol{\omega} - \boldsymbol{\omega}_s\right) \tag{7.49}$$

Moreover, since $\boldsymbol{m} \cdot R(\boldsymbol{\omega} - \boldsymbol{\omega}_s) = 0$, we can also write Eq. 7.48 as:

$$\begin{bmatrix} 0 \\ \dot{\boldsymbol{m}} \end{bmatrix} = -\frac{1}{2} \begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \odot \begin{bmatrix} 0 \\ R(\boldsymbol{\omega} - \boldsymbol{\omega}_s) \end{bmatrix} \tag{7.50}$$

7.5. Middle normal vector in rotating body frame

If instead of Eq. 7.5, we can also invert the order of rotations, defining instead:

$$q = q_s' \odot q_a' \tag{7.51}$$

To quickly derive this, we will use the solution found using the definition of Eq. 7.5. For a rotation q, with a corresponding rotation matrix of R, we have the following

decomposition:

$$\begin{aligned} q &= -\begin{bmatrix} 0 \\ \boldsymbol{m}^E \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \odot q_s \\ &= -\begin{bmatrix} 0 \\ \frac{(R+\mathbb{I})\boldsymbol{e}}{|(R+\mathbb{I})\boldsymbol{e}|} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \odot \frac{1}{|(R+\mathbb{I})\boldsymbol{e}|} \begin{bmatrix} q_\mathrm{r} \\ (\boldsymbol{q}\cdot\boldsymbol{e})\boldsymbol{e} \end{bmatrix} \end{aligned} \tag{7.52}$$

Then, for $q^* = \begin{bmatrix} q_\mathrm{r} & -\boldsymbol{q}^T \end{bmatrix}^T$, with a corresponding rotation matrix of R^T:

$$q^* = -\begin{bmatrix} 0 \\ \frac{(R^T+\mathbb{I})\boldsymbol{e}}{|(R^T+\mathbb{I})\boldsymbol{e}|} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \odot \frac{1}{|(R^T+\mathbb{I})\boldsymbol{e}|} \begin{bmatrix} q_\mathrm{r} \\ -(\boldsymbol{q}\cdot\boldsymbol{e})\boldsymbol{e} \end{bmatrix} \tag{7.53}$$

We note that:

$$\begin{aligned} (R^T+\mathbb{I})\boldsymbol{e} &= (R^T + R^T R)\boldsymbol{e} \\ &= R^T(\mathbb{I}+R)\boldsymbol{e} \end{aligned} \tag{7.54}$$

And $|(R^T+\mathbb{I})\boldsymbol{e}| = |(R+\mathbb{I})\boldsymbol{e}|$, leading to:

$$\begin{aligned} \frac{(R^T+\mathbb{I})\boldsymbol{e}}{|(R^T+\mathbb{I})\boldsymbol{e}|} &= R^T \frac{(R+\mathbb{I})\boldsymbol{e}}{|(R+\mathbb{I})\boldsymbol{e}|} \\ &= R^T\boldsymbol{m}^E \\ &= \boldsymbol{m}^B \end{aligned} \tag{7.55}$$

We can then rewrite Eq. 7.53 as:

$$q^* = -\begin{bmatrix} 0 \\ \boldsymbol{m}^B \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \odot q_s^* \tag{7.56}$$

Which leads to:

$$\begin{aligned} q &= -\left(\begin{bmatrix} 0 \\ \boldsymbol{m}^B \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \odot q_s^* \right)^* \\ &= -q_s \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix}^* \odot \begin{bmatrix} 0 \\ \boldsymbol{m}^B \end{bmatrix}^* \\ &= -q_s \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{m}^B \end{bmatrix} \end{aligned} \tag{7.57}$$

And finally:

$$\begin{aligned} q'_s &= q_s \\ q'_a &= -\begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{m}^B \end{bmatrix} \end{aligned} \tag{7.58}$$

Moreover, we can redefine the derivatives as:

$$\begin{aligned} \boldsymbol{\omega}_s &= \left(\frac{\boldsymbol{m} \cdot R\boldsymbol{\omega}}{\boldsymbol{m} \cdot \boldsymbol{e}} \right) \boldsymbol{e} \\ &= \left(\frac{R^T\boldsymbol{m} \cdot \boldsymbol{\omega}}{R^T\boldsymbol{m} \cdot R^T\boldsymbol{e}} \right) \boldsymbol{e} \\ &= \left(\frac{\boldsymbol{m}^B \cdot \boldsymbol{\omega}}{\boldsymbol{m}^B \cdot R^T\boldsymbol{e}} \right) \boldsymbol{e} \end{aligned} \tag{7.59}$$

And using the fact that:

$$\begin{aligned} \boldsymbol{m}^E &= R\boldsymbol{m}^B \\ \dot{\boldsymbol{m}}^E &= R\hat{\boldsymbol{\omega}}\boldsymbol{m}^B + R\dot{\boldsymbol{m}}^B \\ \dot{\boldsymbol{m}}^E &= R\left(\dot{\boldsymbol{m}}^B - \boldsymbol{m}^B \times \boldsymbol{\omega}\right) \end{aligned} \tag{7.60}$$

We show that:

$$\begin{aligned} \dot{\boldsymbol{m}}^E &= -\frac{1}{2}(RR^T\boldsymbol{m}^E) \times R\left(\boldsymbol{\omega} - \boldsymbol{\omega}_s\right) \\ R\left(\dot{\boldsymbol{m}}^B - \boldsymbol{m}^B \times \boldsymbol{\omega}\right) &= -\frac{1}{2}R\left(\boldsymbol{m}^B \times \left(\boldsymbol{\omega} - \boldsymbol{\omega}_s\right)\right) \\ \dot{\boldsymbol{m}}^B &= \frac{1}{2}\boldsymbol{m}^B \times \left(\boldsymbol{\omega} + \boldsymbol{\omega}_s\right) \end{aligned} \tag{7.61}$$

Which is slightly simpler than Eq. 7.49, since it does not have an added multiplication with R. I suspect that this form of the decomposition might be useful for analyzing the equations of motion in the body frame, given the lack of the added R term. This is arguably less practical from a control point of view, though, since both the actual and desired middle normal vectors are constantly changing with a spin rotation when represented in the moving body frame. In the fixed frame, however, if only the spin rotation is changing, both the actual and desired middle vectors are constant.

7.6. Final thoughts

In this chapter, a novel decomposition of the quaternion representing a full 3D rotation is given: the rotation can now be given by the quaternion product between

the middle normal vector $\boldsymbol{m}$, and a term dealing the spin rotation. This decomposition allows us to completely decouple the 2 controllable degrees of freedom of the rotation, expressed by $\boldsymbol{m}$, and the uncontrollable degree of freedom given by the spin angle. Moreover, the middle normal vector $\boldsymbol{m}$ has a clear geometrical interpretation and has a simple expression. In Chapter 8, we will present a non-linear controller that acts directly on $\boldsymbol{m}$ and corrects it, pointing the drone's normal vector to the desired direction while completely ignoring the spin rotation. We will also show that this controller renders the system asymptotically stable.

8. Controller architecture

In this section, we developed the cascaded controller architecture used for the simulations. This architecture consists of an inner non-linear attitude control loop, followed by an outer angular rate PID controller. This uncoupled controller is easier to implement, and compatible with commonly used flight controller architectures. The PX4 Flight Controller [37], for example, implements a nonlinear attitude controller based on [10] followed by an angular rate PID controller as well. This chapter has been submitted as a part of [4].

8.1. Middle normal vector and rotation decomposition

The system is underactuated, and we cannot control its yaw rotation (the robot will spin at all times around the z-axis). Instead of controlling the full attitude, we decomposed the rotation quaternion q to extract the uncontrollable spin, and control the remaining reduced attitude. We define a normal vector $\boldsymbol{n}_b$ that points upwards in the body frame $\mathcal{F}_B$:

$$\boldsymbol{n}_b^B \equiv \boldsymbol{e}_z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{8.1}$$

In this case, this vector in the inertial frame $\mathcal{F}_E$ can be calculated as:

$$\begin{aligned} \begin{bmatrix} 0 \\ \boldsymbol{n}_b^E \end{bmatrix} &= q \odot \begin{bmatrix} 0 \\ \boldsymbol{n}_b^B \end{bmatrix} \odot q^* \\ &= q \odot \begin{bmatrix} 0 \\ \boldsymbol{e}_z \end{bmatrix} \odot q^* \end{aligned} \tag{8.2}$$

Giving:

$$\boldsymbol{n}_b^E = \begin{bmatrix} 2(q_\mathrm{x} q_\mathrm{z} + q_\mathrm{r} q_\mathrm{y}) \\ 2(q_\mathrm{y} q_\mathrm{z} - q_\mathrm{r} q_\mathrm{x}) \\ 2\left(q_\mathrm{r}^2 + q_\mathrm{z}^2\right) - 1 \end{bmatrix} \tag{8.3}$$

Defining the middle normal vector $\boldsymbol{m}$ as the unit vector half-way between $\boldsymbol{n}_b^B$ and $\boldsymbol{n}_b^E$:

$$\boldsymbol{m} = \frac{\boldsymbol{n}_b^E + \boldsymbol{n}_b^B}{|\boldsymbol{n}_b^E + \boldsymbol{n}_b^B|} = \frac{1}{\sqrt{q_\mathrm{r}^2 + q_\mathrm{z}^2}} \begin{bmatrix} q_\mathrm{x} q_\mathrm{z} + q_\mathrm{r} q_\mathrm{y} \\ q_\mathrm{y} q_\mathrm{z} - q_\mathrm{r} q_\mathrm{x} \\ q_\mathrm{r}^2 + q_\mathrm{x}^2 \end{bmatrix} \tag{8.4}$$

Note that this has a singularity when in $\boldsymbol{n}_b^E = -\boldsymbol{n}_b^B = -\boldsymbol{e}_z$, as seen in Chapter 7. This position is usually not be plausible for the flying monospinner, but to address this problem in real-time, we can check if $\boldsymbol{m} \cdot \boldsymbol{e} \approx 0$ (Eq. 7.19) and apply 8.5:

$$\boldsymbol{m}_{k+1} \leftarrow \frac{\boldsymbol{m}_k - (\boldsymbol{m}_k \cdot \boldsymbol{e}_z)\boldsymbol{e}_z}{|\boldsymbol{m}_k - (\boldsymbol{m}_k \cdot \boldsymbol{e}_z)\boldsymbol{e}_z|} \tag{8.5}$$

We also define:

$$\begin{aligned} q_s &= \begin{bmatrix} c_s \\ s_s \boldsymbol{e}_z \end{bmatrix} = \begin{bmatrix} \cos(\theta_s/2) \\ \sin(\theta_s/2)\boldsymbol{e}_z \end{bmatrix} \\ \theta_s &= 2\ \mathrm{atan2}(q_\mathrm{z}, q_\mathrm{r}) \end{aligned} \tag{8.6}$$

Or, more directly:

$$q_s = \frac{1}{\sqrt{q_\mathrm{r}^2 + q_\mathrm{z}^2}} \begin{bmatrix} q_\mathrm{r} \\ 0 \\ 0 \\ q_\mathrm{z} \end{bmatrix} \tag{8.7}$$

And, as seen in Chapter 7, we can decompose q as:

$$q = -\begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e}_z \end{bmatrix} \odot q_s \tag{8.8}$$

Moreover:

$$\dot{q}_s = \frac{1}{2} q_s \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega}_s \end{bmatrix} \tag{8.9}$$

Where:

$$\boldsymbol{\omega}_s = \left(\frac{\boldsymbol{m} \cdot R\boldsymbol{\omega}}{\boldsymbol{m} \cdot \boldsymbol{e}_z} \right) \boldsymbol{e}_z \tag{8.10}$$

Also, we can define $\boldsymbol{m}$ with respect to q_s as follows:

$$\begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} = q \odot q_s^* \odot \begin{bmatrix} 0 \\ \boldsymbol{e}_z \end{bmatrix} \tag{8.11}$$

Moreover, its derivative can be given by:

$$\dot{\boldsymbol{m}} = -\frac{1}{2}\boldsymbol{m} \times R\left(\boldsymbol{\omega} - \boldsymbol{\omega}_s\right) \tag{8.12}$$

8.2. Rotation error definition

Define $\bar{q}$ as the desired orientation, and $\bar{\boldsymbol{n}} = \bar{\boldsymbol{n}}_E$ as the corresponding normal vector, and $\bar{\boldsymbol{m}}$ as the corresponding middle normal vector. We know that:

$$\bar{q} = -\begin{bmatrix} 0 \\ \bar{\boldsymbol{m}} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e}_z \end{bmatrix} \odot \bar{q}_s \tag{8.13}$$

But since we cannot control the rotation around the z-axis, we just set the goal $\bar{q}_s = q_s$:

$$\bar{q} = -\begin{bmatrix} 0 \\ \bar{\boldsymbol{m}} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e}_z \end{bmatrix} \odot q_s \tag{8.14}$$

To calculate the rotation quaternion between the desired quaternion and the current orientation, we can define a quaternion difference as:

$$\begin{aligned} q_\Delta &= q \odot \bar{q}^* \\ &= -\begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e}_z \end{bmatrix} \odot \bar{q}_s \odot q_s^* \odot \begin{bmatrix} 0 \\ \boldsymbol{e}_z \end{bmatrix}^* \odot \begin{bmatrix} 0 \\ \bar{\boldsymbol{m}} \end{bmatrix} \\ &= -\begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \bar{\boldsymbol{m}} \end{bmatrix} \\ &= \begin{bmatrix} \bar{\boldsymbol{m}} \cdot \boldsymbol{m} \\ \bar{\boldsymbol{m}} \times \boldsymbol{m} \end{bmatrix} \end{aligned} \tag{8.15}$$

Or equivalently:

$$q_\Delta = \frac{1}{|\boldsymbol{n} + \boldsymbol{e}_z||\bar{\boldsymbol{n}} + \boldsymbol{e}_z|} \begin{bmatrix} (\bar{\boldsymbol{n}} + \boldsymbol{e}_z) \cdot (\boldsymbol{n} + \boldsymbol{e}_z) \\ (\bar{\boldsymbol{n}} + \boldsymbol{e}_z) \times (\boldsymbol{n} + \boldsymbol{e}_z) \end{bmatrix} \tag{8.16}$$

When the robot's orientation is at the goal orientation, we know that, by definition, $q_\Delta = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T$. At this state, we know that:

$$\begin{bmatrix} \bar{\boldsymbol{m}} \cdot \boldsymbol{m} \\ \bar{\boldsymbol{m}} \times \boldsymbol{m} \end{bmatrix} = \begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} \tag{8.17}$$

And:

$$\bar{\boldsymbol{m}} \cdot \boldsymbol{m} = 1 \tag{8.18}$$

Or, equivalently:

$$(\boldsymbol{n} + \boldsymbol{e}_z) \cdot (\bar{\boldsymbol{n}} + \boldsymbol{e}_z) = |\boldsymbol{n} + \boldsymbol{e}_z||\bar{\boldsymbol{n}} + \boldsymbol{e}_z| \tag{8.19}$$

Which implies that $\boldsymbol{n} = \bar{\boldsymbol{n}}$.

8.3. Non-linear attitude controller

Consider the following family of Lyapunov candidate functions:

$$V = V(\boldsymbol{m}, \kappa) \equiv \begin{cases} -\ln\left(\boldsymbol{m} \cdot \bar{\boldsymbol{m}}^2\right) & \text{, if } \kappa = 0 \\ \frac{2S^\kappa}{\kappa}\left(\frac{1}{\boldsymbol{m} \cdot \bar{\boldsymbol{m}}^\kappa} - 1\right) & \text{, if } \kappa > 0 \end{cases} \tag{8.20}$$

Where $S = \text{sign}(\boldsymbol{m} \cdot \bar{\boldsymbol{m}})$ and $\kappa \in \mathbb{N}$ is a given parameter. We can see that, for any value of κ:

- $V = 0 \iff \boldsymbol{m} = \bar{\boldsymbol{m}}$, (the case when $\boldsymbol{n} = \bar{\boldsymbol{n}}$);
- $V > 0$ for any other value of $\boldsymbol{m}$;
- $\lim_{\boldsymbol{m} \to -\bar{\boldsymbol{m}}} V(\boldsymbol{m}) = +\infty$.

Moreover, we can see that, for any value of κ:

$$\begin{aligned} \dot{V} &= -2\frac{S^\kappa}{\boldsymbol{m} \cdot \bar{\boldsymbol{m}}^{\kappa+1}}\, \bar{\boldsymbol{m}} \cdot \dot{\boldsymbol{m}} \\ &= -2\frac{1}{|\boldsymbol{m} \cdot \bar{\boldsymbol{m}}|^\kappa}\, \frac{\bar{\boldsymbol{m}} \cdot \dot{\boldsymbol{m}}}{\boldsymbol{m} \cdot \bar{\boldsymbol{m}}} \end{aligned} \tag{8.21}$$

Introducing Eq. 8.12 into Eq. 8.21 gives:

$$\dot{V} = \frac{1}{|\boldsymbol{m} \cdot \bar{\boldsymbol{m}}|^\kappa}\, \frac{\bar{\boldsymbol{m}} \cdot (\boldsymbol{m} \times R\,(\boldsymbol{\omega} - \boldsymbol{\omega}_s))}{\boldsymbol{m} \cdot \bar{\boldsymbol{m}}} \tag{8.22}$$

Using the fact that $\boldsymbol{a} \cdot (\boldsymbol{b} \times \boldsymbol{c}) = \boldsymbol{c} \cdot (\boldsymbol{a} \times \boldsymbol{b})$, and that $\boldsymbol{a} \cdot R\boldsymbol{b} = R^T\boldsymbol{a} \cdot \boldsymbol{b}$, we can further simplify the expression as:

$$\dot{V} = (\boldsymbol{\omega} - \boldsymbol{\omega}_s) \cdot \left(\frac{1}{|\boldsymbol{m} \cdot \bar{\boldsymbol{m}}|^\kappa} R^T \frac{\bar{\boldsymbol{m}} \times \boldsymbol{m}}{\boldsymbol{m} \cdot \bar{\boldsymbol{m}}}\right) \tag{8.23}$$

Which leads to the natural definition that, for any 3×3 positive-definite matrix $\boldsymbol{P}$, setting:

$$\boldsymbol{\omega} - \boldsymbol{\omega}_s = -\boldsymbol{P}\left(\frac{1}{|\boldsymbol{m}\cdot\bar{\boldsymbol{m}}|^{\kappa}} R^T \frac{\bar{\boldsymbol{m}}\times\boldsymbol{m}}{\boldsymbol{m}\cdot\bar{\boldsymbol{m}}}\right) \tag{8.24}$$

Defining:

$$\boldsymbol{m}' = \frac{1}{|\boldsymbol{m}\cdot\bar{\boldsymbol{m}}|^{\kappa}} R^T \left(\frac{\bar{\boldsymbol{m}}\times\boldsymbol{m}}{\boldsymbol{m}\cdot\bar{\boldsymbol{m}}}\right) \tag{8.25}$$

We have:

$$\dot{V} = -\boldsymbol{m}'\cdot(\boldsymbol{P}\boldsymbol{m}') \leq 0 \tag{8.26}$$

Which is always negative, proving that V is a Lyapunov function and showing this system is asymptotically stable. Isolating $\boldsymbol{\omega}$ gives:

$$\boldsymbol{\omega} = \boldsymbol{\omega}_s + \boldsymbol{P}\left(\frac{1}{|\boldsymbol{m}\cdot\bar{\boldsymbol{m}}|^{\kappa}} R^T \frac{\boldsymbol{m}\cdot\bar{\boldsymbol{m}}}{\boldsymbol{m}\cdot\bar{\boldsymbol{m}}}\right) \tag{8.27}$$

In particular, setting $\bar{\boldsymbol{m}} = \bar{\boldsymbol{n}} = \boldsymbol{e}_z$, we know that S is always equal to $+1$, leading to:

$$\boldsymbol{\omega} = \left(\frac{\boldsymbol{m}\cdot R\boldsymbol{\omega}}{\boldsymbol{m}\cdot\boldsymbol{e}_z}\right)\boldsymbol{e}_z + \frac{\boldsymbol{P}}{|\boldsymbol{m}\cdot\boldsymbol{e}_z|^{\kappa}} R^T \left(\frac{\boldsymbol{m}\times\boldsymbol{e}_z}{\boldsymbol{m}\cdot\boldsymbol{e}_z}\right) \tag{8.28}$$

Arbitrarily choosing the simplest case with $\kappa = 0$ leads to the controller:

$$\boldsymbol{\omega} = \frac{(\boldsymbol{m}\cdot R\boldsymbol{\omega})\boldsymbol{e}_z + \boldsymbol{P}R^T(\boldsymbol{m}\times\boldsymbol{e}_z)}{\boldsymbol{m}\cdot\boldsymbol{e}_z} \tag{8.29}$$

Note however that different values of κ would create more aggressive controllers.

8.3.1. Angular rate controller

We can completely control the drone by setting the output of the attitude controller given by Eq. 8.27 as the input of an angular rate controller, given that the

separation principle holds[1]. We then a desired angular velocity $\boldsymbol{\omega}_r$:

$$\boldsymbol{\omega}_r = \boldsymbol{\omega}_s + \frac{\boldsymbol{P}}{|\boldsymbol{m} \cdot \bar{\boldsymbol{m}}|^{\kappa}} R^T \left(\frac{\boldsymbol{m} \times \bar{\boldsymbol{m}}}{\boldsymbol{m} \cdot \bar{\boldsymbol{m}}} \right) \tag{8.30}$$

And:

$$\Delta\boldsymbol{\omega} = \boldsymbol{\omega}_r - \boldsymbol{\omega} \tag{8.31}$$

And the correction torque $\boldsymbol{\tau}_c$ can be calculated as:

$$\frac{1}{h_P}\boldsymbol{\tau}_c = \boldsymbol{K}_p \, \Delta\boldsymbol{\omega} + \boldsymbol{K}_d \frac{\mathrm{d}\Delta\boldsymbol{\omega}}{\mathrm{d}t} + \boldsymbol{K}_i \int \Delta\boldsymbol{\omega} \, \mathrm{d}t \tag{8.32}$$

Where $\boldsymbol{K}_p$, $\boldsymbol{K}_d$ and $\boldsymbol{K}_i$ are the 3 PID terms. According to the motor's model, we know that the relationship between the thrust force and the torque is:

$$\begin{aligned} \boldsymbol{\tau}_p &= \boldsymbol{B}_\tau \boldsymbol{f}_p \\ &= (k\mathbb{I} + h_P \widehat{\boldsymbol{e}}_z) \boldsymbol{f}_p \end{aligned} \tag{8.33}$$

We assumed that the control force $\boldsymbol{f}_c$ and control torque $\boldsymbol{\tau}_c$ are given by the cross product part (see Fig. 5.3 and Eq. 5.48). Moreover, we also assumed that $\boldsymbol{e}_z \cdot \boldsymbol{f}_c = 0$. This leads to:

$$\begin{aligned} \boldsymbol{\tau}_c &= h_P \widehat{\boldsymbol{e}}_z \boldsymbol{f}_c \\ \widehat{\boldsymbol{e}}_z \boldsymbol{\tau}_c &= h_P \widehat{\boldsymbol{e}}_z^2 \boldsymbol{f}_c \\ \widehat{\boldsymbol{e}}_z \boldsymbol{\tau}_c &= h_P \left(\boldsymbol{e}_z \boldsymbol{e}_z^T - \mathbb{I} \right) \boldsymbol{f}_c \\ \widehat{\boldsymbol{e}}_z \boldsymbol{\tau}_c &= h_P \left(\boldsymbol{e}_z (\boldsymbol{e}_z \cdot \boldsymbol{f}_c) - \boldsymbol{f}_c \right) \end{aligned} \tag{8.34}$$

And finally:

$$\begin{aligned} \boldsymbol{f}_c &= \frac{1}{h_P} \boldsymbol{\tau}_c \times \boldsymbol{e}_z \\ &= \left(\boldsymbol{K}_p \, \Delta\boldsymbol{\omega} + \boldsymbol{K}_d \frac{\mathrm{d}\Delta\boldsymbol{\omega}}{\mathrm{d}t} + \boldsymbol{K}_i \int \Delta\boldsymbol{\omega} \, \mathrm{d}t \right) \times \boldsymbol{e}_z \end{aligned} \tag{8.35}$$

Adding a constant thrust term in the z direction, for some constant Ω:

$$\begin{aligned} \boldsymbol{f}_p &= \boldsymbol{f}_c + k_f \Omega^2 \boldsymbol{e}_z \\ &= \begin{bmatrix} +\tau_{c,y}/h_P \\ -\tau_{c,x}/h_P \\ k_f \Omega^2 \end{bmatrix} \end{aligned} \tag{8.36}$$

[1] The frequency of the inner rate controller is much higher than the frequency of the outer attitude controller.

Noting $\boldsymbol{f}_p = \begin{bmatrix} f_x & f_y & f_z \end{bmatrix}^T$, finally the motor velocity $\dot{\gamma}$, phase β and inclination angle α are calculated as:

$$\begin{aligned} \beta &= \text{atan2}(f_y, f_x) \\ \alpha &= \text{acos}\left(\frac{f_z}{|\boldsymbol{f}_p|}\right) = \text{atan}\left(\frac{|\boldsymbol{f}_c|}{f_z}\right) \\ \dot{\gamma} &= \sqrt{\frac{|\boldsymbol{f}_p|}{k_f}} = \frac{\Omega}{\sqrt{\cos(\alpha)}} \end{aligned} \tag{8.37}$$

Moreover, since $\boldsymbol{e}_z \times \boldsymbol{\omega}_s = \boldsymbol{0}$, we can simplify the desired angular velocity to:

$$\boldsymbol{\omega}_r = \frac{\boldsymbol{P}}{|\boldsymbol{m} \cdot \bar{\boldsymbol{m}}|^\kappa} R^T \left(\frac{\boldsymbol{m} \times \bar{\boldsymbol{m}}}{\boldsymbol{m} \cdot \bar{\boldsymbol{m}}}\right) \tag{8.38}$$

8.3.2. Influence of the controller order κ

Considering the error angle ψ between $\boldsymbol{n}$ and $\bar{\boldsymbol{n}}$, we know that:

$$\begin{aligned} \sin(\psi/2) &= |\boldsymbol{m} \times \bar{\boldsymbol{m}}| \\ \cos(\psi/2) &= \boldsymbol{m} \cdot \bar{\boldsymbol{m}} \end{aligned} \tag{8.39}$$

For a given controller order κ, we know that:

$$|\boldsymbol{m}'| = \left| \frac{\boldsymbol{m} \times \bar{\boldsymbol{m}}}{\boldsymbol{m} \cdot \bar{\boldsymbol{m}}^{\kappa+1}} \right| = \tan\left(\frac{\psi}{2}\right) \sec^\kappa\left(\frac{\psi}{2}\right) \tag{8.40}$$

Figure 8.1 compares the relation between the magnitude $|\boldsymbol{m}'|$ to the error angle ψ. For higher orders of κ, the controller will be increasingly aggressive for bigger error angles, while staying relatively the same for smaller error angles.

8.4. Final thoughts

In this chapter, a full cascaded controller architecture based on the middle normal vector decomposition is presented. This controller, together with the swashplateless system, makes it possible for us to control a reduced form of the drone's attitude. In the next chapter, we will show this controller and the equations of motion derived in Chapter 6 were all implemented in a single closed-loop simulation, demonstrating how this controller can stabilize the proposed UAV.

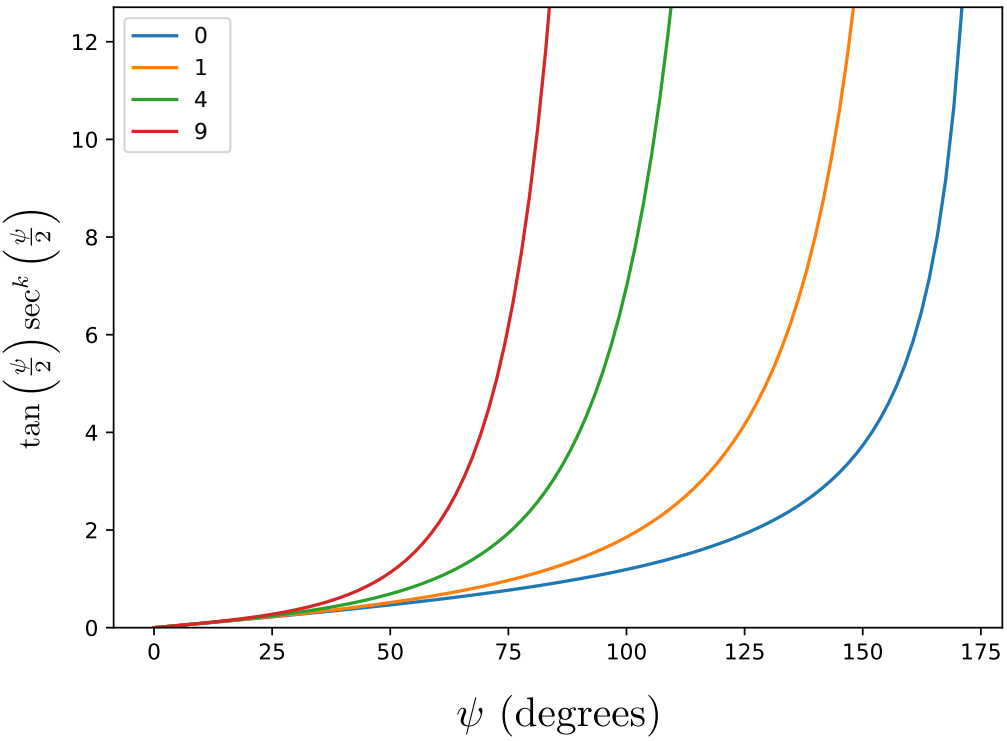


Figure 8.1.: Relation between the magnitude $|\boldsymbol{m}'|$ to the error angle ψ for 4 different values of the controller order κ: 0, 1, 4 and 9.

9. Simulation and spin drift study

In order to test the controller shown in Chapter 8, a closed-loop simulation of the Equations of Motion shown in Chapter 6 was implemented using Python[1]. In this section, we will analyze some implementation details and use the simulation to analyze the effect of a bad yaw estimation. A part of this chapter has been submitted in [4].

9.1. Controller

As shown in Chapter 8, the desired controller force is given by (according to Eq. 8.35):

$$\boldsymbol{f}_c = \left(\boldsymbol{K}_p \, \Delta\boldsymbol{\omega} + \boldsymbol{K}_d \frac{\mathrm{d}\Delta\boldsymbol{\omega}}{\mathrm{d}t} + \boldsymbol{K}_i \int \Delta\boldsymbol{\omega} dt \right) \times \boldsymbol{e}_z \tag{9.1}$$

Where:

$$\begin{aligned} \Delta\boldsymbol{\omega} &= \boldsymbol{\omega}_r - \boldsymbol{\omega} \\ \boldsymbol{\omega}_r &= \boldsymbol{P} \left(R^T \frac{1}{|\boldsymbol{m} \cdot \bar{\boldsymbol{m}}|^\kappa} \frac{\boldsymbol{m} \times \bar{\boldsymbol{m}}}{\boldsymbol{m} \cdot \bar{\boldsymbol{m}}} \right) \end{aligned} \tag{9.2}$$

And the positive definite matrices $\boldsymbol{P}$, $\boldsymbol{K}_p$, $\boldsymbol{K}_d$ and $\boldsymbol{K}_i$ and the controller order $\kappa \in \mathbb{N}^+$ are all adjusted parameters. The full propeller force is:

$$\boldsymbol{f}_p = \boldsymbol{f}_c + k_f \Omega^2 \boldsymbol{e}_z \tag{9.3}$$

[1] See github.com/evbernardes/monospinner_simulator

And according to Eq. 8.37, the rotor commands are calculated as (noting $\boldsymbol{f}_p = \begin{bmatrix} f_x & f_y & f_z \end{bmatrix}^T$):

$$\begin{aligned} \beta &= \text{atan2}(f_y, f_x) \\ \alpha &= \text{atan}\left(\frac{|\boldsymbol{f}_c|}{k_f \Omega^2}\right) \\ \dot{\gamma} &= \frac{\Omega}{\sqrt{\cos(\alpha)}} \end{aligned} \tag{9.4}$$

In a real robot, Ω would be manually controlled by the pilot. In our simulation, it was set as the necessary force for hover:

$$\Omega = \sqrt{\frac{mg}{k_f}} \tag{9.5}$$

To simulate the physical constraints of the system, two new parameters are introduced: $\alpha_{\max}$, the max inclination angle supposed to be 30 degrees, and Λ, the security angular velocity deviation ratio from Ω, fixed at 1.2. The motor control signals are then calculated as:

$$\begin{aligned} \beta &= \text{atan2}(f_y, f_x) \\ \alpha &= \min\left\{\text{atan}\left(\frac{|\boldsymbol{f}_c|}{k_f \Omega^2}\right),\ \alpha_{\max}\right\} \\ \dot{\gamma} &= \Omega\ \left[\min\left\{\frac{1}{\sqrt{\cos(\alpha)}},\ \Lambda\right\}\right] \end{aligned} \tag{9.6}$$

And the propeller thrust force is then calculated as follows:

$$\begin{aligned} \boldsymbol{r} &= \begin{bmatrix} s_\alpha c_\beta \\ s_\alpha s_\beta \\ c_\alpha \end{bmatrix} \\ \boldsymbol{f}_p &= k_f \dot{\gamma}^2 \boldsymbol{r} \end{aligned} \tag{9.7}$$

9.2. Equations of Motion and integration

We use the equations from Eq. 6.37, supposing that the torques generated by $\boldsymbol{\omega}_{B/P}^{B}$, the angular velocity of $\mathcal{F}_P$ w.r.t. $\mathcal{F}_B$ is negligible. As a reminder:

$$\mathbf{D}\begin{bmatrix}\dot{\boldsymbol{\omega}}\\ \dot{\boldsymbol{v}}\end{bmatrix} - \mathrm{ad}_{\eta}^{T}\mathbf{D}\begin{bmatrix}\boldsymbol{\omega}\\ \boldsymbol{v}\end{bmatrix} + \begin{bmatrix}\dot{\gamma}J_{pz}\,\boldsymbol{\omega}\times\boldsymbol{r}\\ 0\end{bmatrix} = F_{\text{ext}}^{B} \tag{9.8}$$

With :

$$\begin{aligned}\mathbf{D} &= \begin{bmatrix}\boldsymbol{J} & 0\\ 0 & m\,\mathbb{I}\end{bmatrix} + \Delta m\begin{bmatrix}0 & -\widehat{\boldsymbol{e}}_z\\ \widehat{\boldsymbol{e}}_z & 0\end{bmatrix}\\ F_{\text{ext}}^{B} &= \begin{bmatrix}f_p\boldsymbol{B}_\tau\boldsymbol{r} + \Delta m\,g\,\boldsymbol{e}_z\times\boldsymbol{n}_b^E - |\boldsymbol{\omega}|\boldsymbol{K}\boldsymbol{\omega}\\ f_p\boldsymbol{r} - mg\,\boldsymbol{n}_b^E\end{bmatrix}\end{aligned} \tag{9.9}$$

And:

$$\begin{aligned}m &= m_b + m_s + m_p\\ \Delta m &= m_s h_S - m_p h_P\\ \mathrm{ad}_{\eta} &= \begin{bmatrix}\widehat{\boldsymbol{\omega}} & 0\\ \widehat{\boldsymbol{v}} & \widehat{\boldsymbol{\omega}}\end{bmatrix}\end{aligned} \tag{9.10}$$

These equations are used to calculate the angular and linear accelerations:

$$\begin{bmatrix}\dot{\boldsymbol{\omega}}\\ \dot{\boldsymbol{v}}\end{bmatrix} = \mathbf{D}^{-1}\left(F_{\text{ext}}^{B} + \mathrm{ad}_{\eta}^{T}\mathbf{D}\begin{bmatrix}\boldsymbol{\omega}\\ \boldsymbol{v}\end{bmatrix} - \begin{bmatrix}\dot{\gamma}J_{pz}\,\boldsymbol{\omega}\times\boldsymbol{r}\\ 0\end{bmatrix}\right) \tag{9.11}$$

In order to test the robustness of the theoretical modelling of the equations of motion, all while simplifying the implementation, we decided to use the simplest and most basic method for numerical integration: the Euler method.

$$\begin{aligned}\boldsymbol{\omega}_{k+1} &= \boldsymbol{\omega}_k + \Delta t\,\dot{\boldsymbol{\omega}}_k\\ \boldsymbol{v}_{k+1} &= \boldsymbol{v}_k + \Delta t\,\dot{\boldsymbol{v}}_k\end{aligned} \tag{9.12}$$

With Δt the sampling time of the simulation. The orientation and position can then be reconstructed by applying the following equations:

$$\begin{aligned}\dot{q}_k &= \frac{1}{2}q_k \odot \begin{bmatrix}0\\ \boldsymbol{\omega}_k\end{bmatrix}\\ q_{k+1} &= q_k + \Delta t\,\dot{q}_k\\ \boldsymbol{s}_{k+1} &= \boldsymbol{s}_k + \Delta t\,R(q_k)\,\boldsymbol{v}_k\end{aligned} \tag{9.13}$$

Note that an extra normalization step is needed for q.

9.3. Full closed-loop simulation diagram

Figure 9.1 shows a diagram of the whole control scheme.

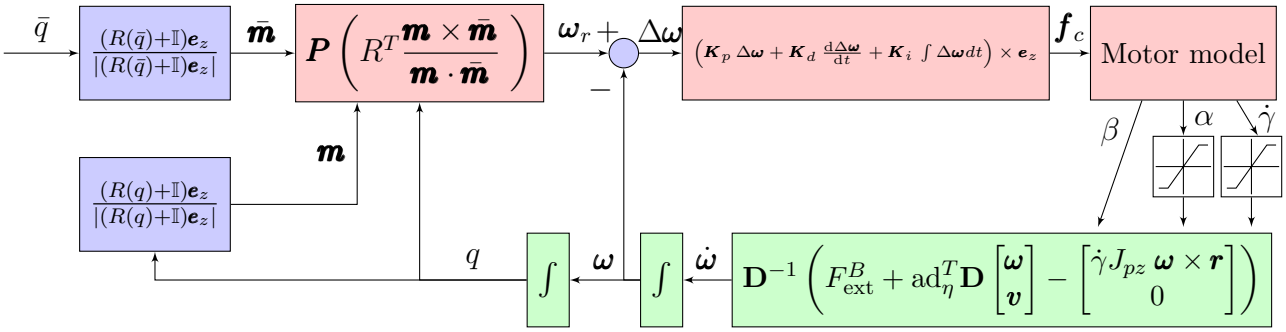


Figure 9.1.: Simulation diagram with a cascaded controller architecture. Controller (in red) composed of a non-linear reduced attitude controller cascaded with a simple PID for angular rate control, and finally the swashplateless motor model is used to extract the necessary commands to the motor. Equations of motion (in green) used to simulate the full system dynamics. Integration steps are used to extract the angular velocity and orientation. Purple blocks correspond to the extraction of $\boldsymbol{m}$ and $\bar{\boldsymbol{m}}$ from q and $\bar{q}$, respectively.

9.4. Parameters

The parameters are all based on a prototype designed at ISM, using 8 anti-torque fins as shown in Chapter 5. The physical and controller parameters are given respectively in Tables 9.1 and 9.2.

m_b	$0.2100kg$
m_s	$0.2007kg$
m_p	$0.0067kg$
h_P	$0.083m$
h_S	$0.145m$
J_{bx}	$0.0010026\,kg\,m^2$
J_{by}	$0.0010026\,kg\,m^2$
J_{bz}	$0.0001831\,kg\,m^2$
J_{sx}	$0.0001794\,kg\,m^2$
J_{sy}	$0.0000387\,kg\,m^2$
J_{sz}	$0.0001928\,kg\,m^2$
J_{pd}	$0.0000052\,kg\,m^2$
J_{pz}	$0.0001061\,kg\,m^2$
k_f	$0.0000052400\,kg\,m$
k_τ	$0.0000000108\,kg\,m^2$
K_z	$0.0000280908\,kg\,m^2$

Table 9.1.: Physical characteristics.

κ	1
$\boldsymbol{P}$	$\mathrm{diag}(1,1,1)$
$\boldsymbol{K}_p$	0.7
$\boldsymbol{K}_d$	0.0
$\boldsymbol{K}_i$	0.0
Ω	$945\mathrm{rad}/s$

Table 9.2.: Controller parameters.

9.5. Inputs and outputs

For simplicity, the inputs in the simulation are given as Euler angles in the ZYZ sequence. As seen in Chapter 7, the rotation can be decomposed as:

$$R_B^E = R(\theta_3 \boldsymbol{e}_z)\; R(\theta_2 \boldsymbol{e}_z)\; R(\theta_1 \boldsymbol{e}_z) \tag{9.14}$$

Where the θ_1, θ_2 and θ_3 (the precession, nutation and revolution respectively) are given by:

$$\begin{aligned}
\theta_1 &= \mathrm{atan2}(q_\mathrm{z}, q_\mathrm{r}) - \mathrm{atan2}(-q_\mathrm{x}, q_\mathrm{y}) \\
\theta_2 &= \mathrm{acos}\left(2\;\left(q_\mathrm{r}^2 + q_\mathrm{z}^2\right) - 1\right) \\
\theta_3 &= \mathrm{atan2}(q_\mathrm{z}, q_\mathrm{r}) + \mathrm{atan2}(-q_\mathrm{x}, q_\mathrm{y})
\end{aligned} \tag{9.15}$$

Equation 9.14 can be rewritten as:

$$R_B^E = R\left((\theta_1 + \theta_3)\,\boldsymbol{e}_z\right)\ R(\theta_1 \boldsymbol{e}_z)^T\ R(\theta_2 \boldsymbol{e}_z)\ R(\theta_1 \boldsymbol{e}_z) \tag{9.16}$$

And analyzing Eqs. 8.6, 9.15 and 9.16, we note that:

$$\begin{aligned} \theta_1 + \theta_3 &= 2\ \mathrm{atan2}(q_z, q_r) \\ &= \theta_s \end{aligned} \tag{9.17}$$

We will then use the angles θ_s instead of θ_3 to represent the robot's spin, since θ_s is consistent with the quaternion decomposition and Eq. 8.6. Moreover, θ_s is uniquely defined when θ_2 is near 0 and arguably more representative of the actual geometrical spin. A discussion of this can be seen in Appendix K. The inputs for the the simulation's goal orientation will be given as precessions and nutations:

$$\boldsymbol{\theta} = (\theta_1, \theta_2) \tag{9.18}$$

So that:

$$\bar{\boldsymbol{n}}_E = \begin{bmatrix} \cos(\theta_1)\sin(\theta_2) \\ \sin(\theta_1)\sin(\theta_2) \\ \cos(\theta_2) \end{bmatrix} \tag{9.19}$$

And[2]:

$$\bar{\boldsymbol{m}} = \frac{\bar{\boldsymbol{n}}_E + \boldsymbol{e}_z}{|\bar{\boldsymbol{n}}_E + \boldsymbol{e}_z|} = \begin{bmatrix} \cos(\theta_1)\sin(\theta_2/2) \\ \sin(\theta_1)\sin(\theta_2/2) \\ \cos(\theta_2/2) \end{bmatrix} \tag{9.20}$$

[2]Note that $\sin(x)/(\cos(x)+1) = \tan(x/2)$

9.6. Simulations

In this section, we will analyze some simulations with different inputs and conditions.

9.6.1. Simulation 1: orientation tracking without noise

This first simulation starts with the robot at rest. Then 4 orientation points in Table 9.3 are tracked in sequence, before going back to the initial orientation. In Fig.

θ_1	θ_2
45°	0°
45°	90°
45°	180°
45°	-90°
45°	0°

Table 9.3.: Orientation inputs for simulation 1.

9.2 **a)** and **b)** we can see how the controller manages to track all the points easily, acting directly on $(\boldsymbol{m}_x, \boldsymbol{m}_y)$, the components perpendicular to $\boldsymbol{e}_z$, and taking the shortest path. Figure 9.2 **c)** shows the angular velocities. The component around $\boldsymbol{e}_z$ approaches the theoretical steady-state velocity, while the other components approach zero. Figure 9.2 **d)** shows the calculated inputs to the motors. Both the inclination angle α and the rotor velocity $\dot{\gamma}$ have peaks whenever the goal is changed, limited by the physical constraints. Figure 9.3 shows the trajectory of the projection of $\boldsymbol{m}$ in the XY plane. Figure 9.4 shows the motor phase. We can note how the phase keeps increasing continuously to counter the monorotor's spin, with discontinuities when the goal changes.

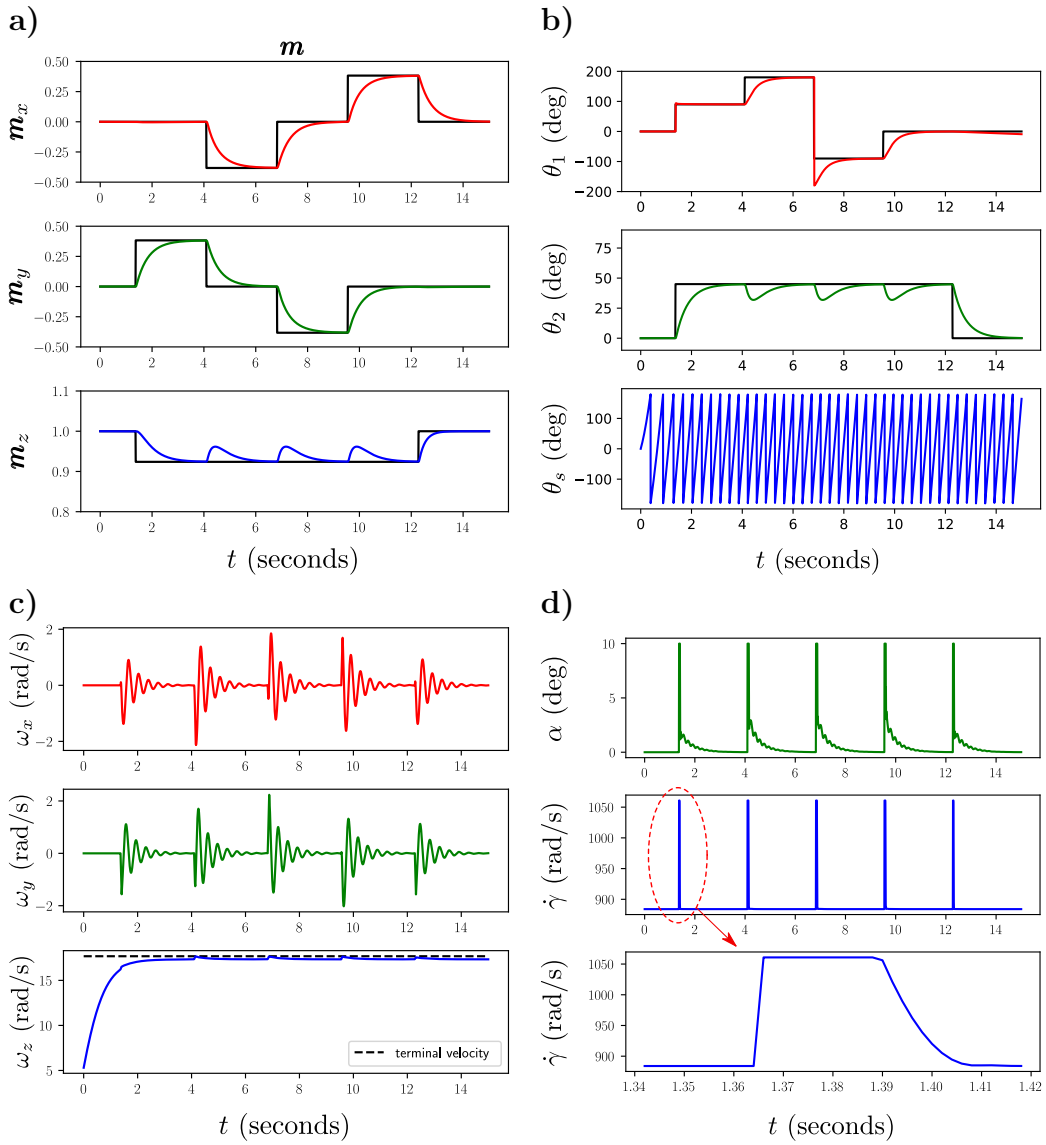


Figure 9.2.: Responses of the simulated mono spinner to changes in attitude angles. **a)** Time course of each component of $\boldsymbol{m}$. **b)** shows the Euler angles representation of the orientation on the ZYZ sequence: θ_1 is the precession (first rotation around the z-axis, representing the direction in which the robot is inclined), θ_2 represents the nutation (rotation around the y-axis, representing the inclination angle), and $\theta_s = \theta_1 + \theta_3$ is the uncontrollable spin. **c)** Angular velocity. **d)** Motor inclination and rotation velocity.

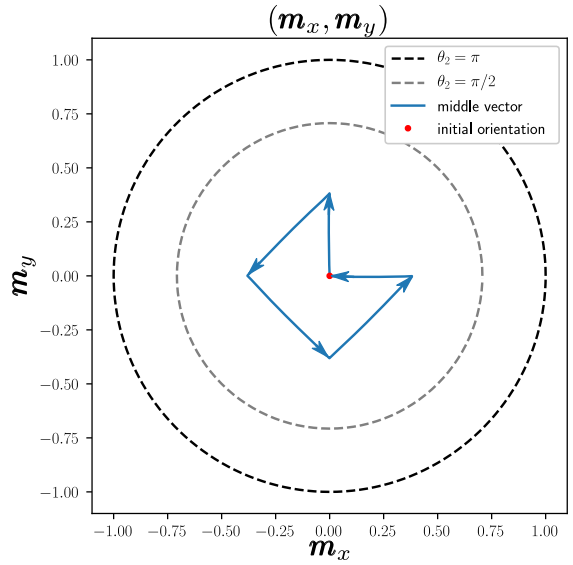


Figure 9.3.: Simulation of monorotor without noise, top view projection of $(\boldsymbol{m}_x, \boldsymbol{m}_y)$. The dashed circles indicate the values of $(\boldsymbol{m}_x, \boldsymbol{m}_y)$ when $\theta_2 = \pi$ (in black) and $\theta_2 = \pi/2$ (in gray).

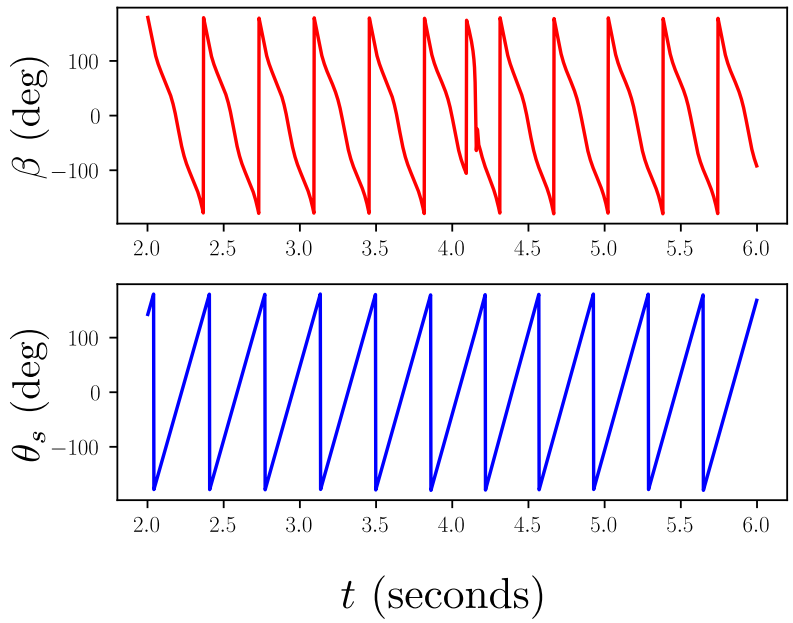


Figure 9.4.: Motor phase.

9.6.2. Simulation 2: vertical stability with random/impulsive noise

The second simulation starts with the robot at rest, then multiple random impulse torques with magnitude up to $26Nm$. It also contains random torques of up to $0.09Nm$ at all times. In Fig. 9.5 **a)** and **b)** we can see how the controller manages to compensate for these impulses. Figure 9.5 **d)** shows the motor inputs, and we note the effect of a noisy angular velocity. This may not be a problem in real systems where a low pass filter is usually applied to the orientation estimators, but an additional low pass filter might also be necessary in the controller output.

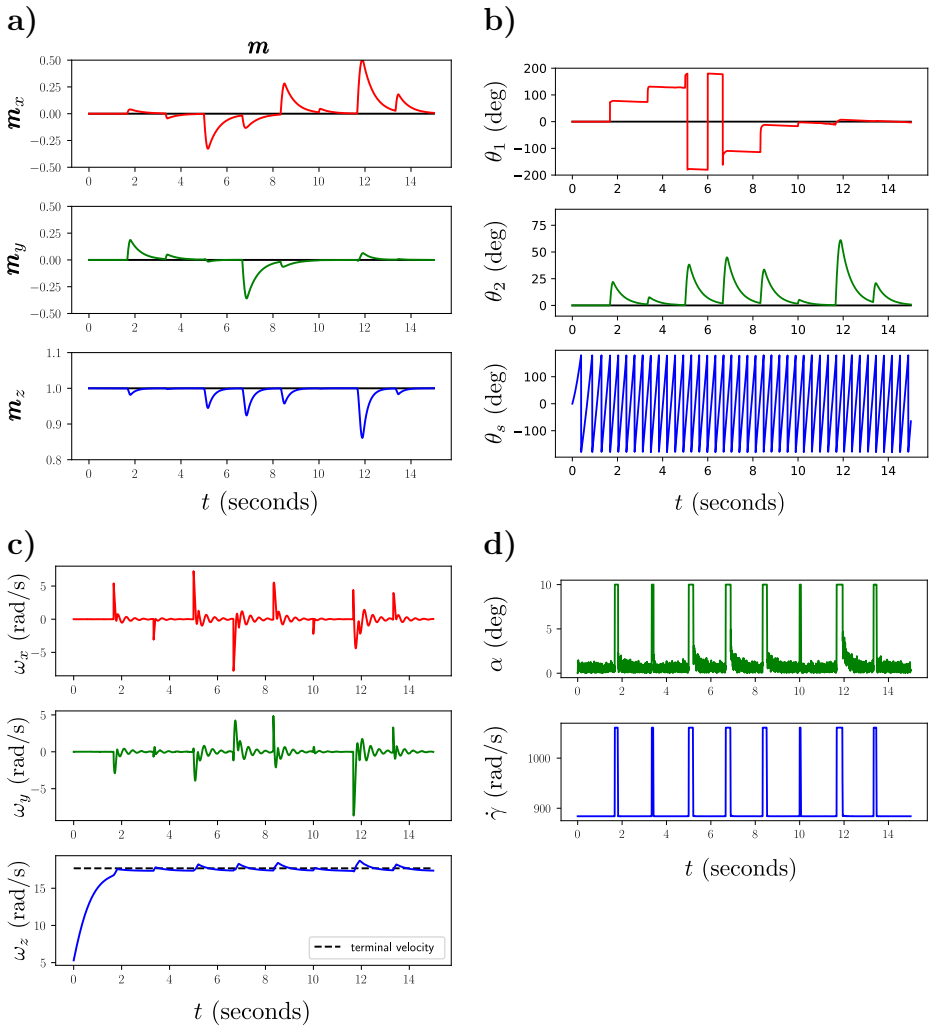


Figure 9.5.: Responses of the simulated mono spinner to perturbations. **a)** Time course of each component of $\boldsymbol{m}$. **b)** Euler angles in ZYZ sequence. **c)** Angular velocity. **d)** Motor inclination, phase, and rotation velocity.

9.7. Spin drift

Even though $\boldsymbol{m}$ is uncoupled from the spin angle in $\mathcal{F}_E$, a good measurement of the spin angle is still necessary for the control. Supposing there is an error of angle ψ in the measurements, generating a rotation q_ψ, whose equivalent rotation matrix is R_ψ, the measured rotation q' and R' are:

$$\begin{aligned} q' &= q \odot q_\psi \\ R' &= R\, R_\psi \end{aligned} \tag{9.21}$$

Where:

$$\begin{aligned} q_\psi &= \begin{bmatrix} \cos(\psi/2) \\ \sin(\psi/2)\boldsymbol{e}_z \end{bmatrix} \\ R_\psi &= \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{9.22}$$

These wrong measurements consequently generate a flawed desired angular velocity:

$$\boldsymbol{\omega}'_r = \boldsymbol{P}\left(R_\psi^T R^T \frac{1}{|\boldsymbol{m}\cdot\bar{\boldsymbol{m}}|^\kappa} \frac{\boldsymbol{m}\times\bar{\boldsymbol{m}}}{\boldsymbol{m}\cdot\bar{\boldsymbol{m}}} \right) \tag{9.23}$$

Which generates a flawed control output $\boldsymbol{f}'_c$.

9.7.1. Simulation 3: different spin angle errors

In this section, multiple simulations were performed starting the robot with an extreme inclination angle of 90°, and multiple simulated spin angle errors. The goal input was $\boldsymbol{\theta} = (0, 0)$. Figure 9.6 shows the top view of $\boldsymbol{m}$ for spin errors ranging from 0° to 75°. The arrows showing the evolution movement are plotted at $t = 0.4s$, giving also an idea of how long each simulation took to converge to the goal destination. We can see how the trajectory of $\boldsymbol{m}$ approaches different spirals. In Fig. 9.7 we clearly see the spiral generated by a spin error of 85°. In Fig. 9.8 we note that, as expected, errors greater than 90° cause the system to be completely unstable. These simulations indicate two things: first, that low values of spin measurement error might slow down convergence without rendering the system completely unstable, and second, that an analysis of the trajectory of $\boldsymbol{m}$ might give an indication of the current spin error.

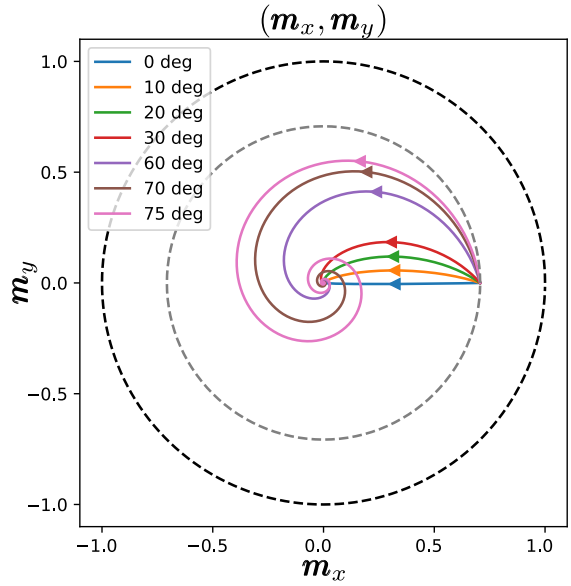


Figure 9.6.: Top view of $\boldsymbol{m}$ for multiple simulated spin error angles. The arrows show the direction of the movement and are plotted at $t = 0.4s$ for all the tests.

9.7.2. Spiral analysis

In this section, we will analyze the spiral generated by the projection of $\boldsymbol{m}$ in the xy-plane. We define:

$$\boldsymbol{\rho} = -\hat{\boldsymbol{e}}_z^2 \boldsymbol{m} = (\mathbb{I} - \boldsymbol{e}_z \boldsymbol{e}_z^T)\boldsymbol{m} = \begin{bmatrix} \boldsymbol{m}_x \\ \boldsymbol{m}_y \\ 0 \end{bmatrix} \tag{9.24}$$

And for a goal $\bar{\boldsymbol{m}}$, we also define $\bar{\boldsymbol{\rho}} = -\hat{\boldsymbol{e}}_z^2 \bar{\boldsymbol{m}}$. We also define the errors:

$$\begin{aligned} \Delta \boldsymbol{m} &= \boldsymbol{m} - \bar{\boldsymbol{m}} \\ \Delta \boldsymbol{\rho} &= \boldsymbol{\rho} - \bar{\boldsymbol{\rho}} = -\hat{\boldsymbol{e}}_z^2 \, \Delta \boldsymbol{m} \end{aligned} \tag{9.25}$$

And finally, the radius and phase of the spiral can be defined as:

$$\begin{aligned} r &= |\Delta \boldsymbol{\rho}| = \sqrt{\Delta \boldsymbol{m}_x^2 + \Delta \boldsymbol{m}_y^2} \\ \phi &= \operatorname{atan2}(\Delta \boldsymbol{m}_y, \Delta \boldsymbol{m}_x) \end{aligned} \tag{9.26}$$

We are interested in finding a function f such that:

$$r = f(\phi) \tag{9.27}$$

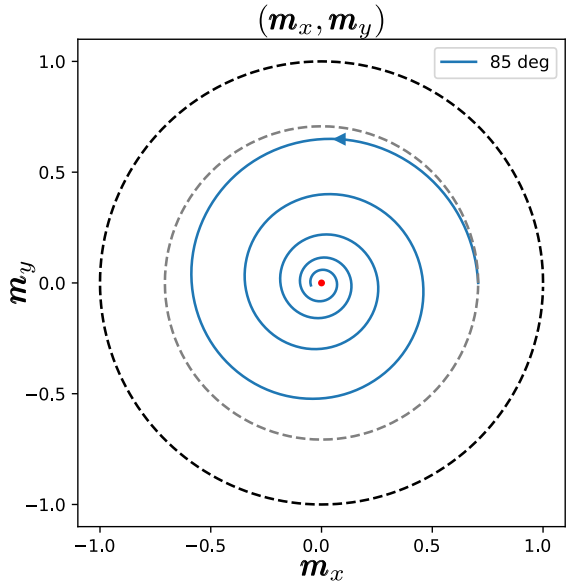


Figure 9.7.: Top view of $\boldsymbol{m}$ for $\psi = 85°$.

And with $\phi = \phi(t)$, the spiral can be found by:

$$\begin{aligned} \Delta \boldsymbol{m}_x &= f(\phi)\ \cos(\phi) \\ \Delta \boldsymbol{m}_y &= f(\phi)\ \sin(\phi) \end{aligned} \tag{9.28}$$

Using the same Figure 9.9 shows that $\ln(r)$, after the acceleration at the beginning of the simulation, approaches a straight line with negative slope in case of convergence and positive slope in case of divergence. Moreover, both ψ and $-\psi$ were observed to produce the same curve for r, hinting at an even relationship between r and ψ. This indicates that r, as expected, decays (or grows) exponentially. Figure 9.10 shows the same for ϕ, showing it also approaches a straight line. We can also note that the slope for $-\psi$ is the negative of the slope for ψ, hinting at an odd relationship between ϕ and ψ. Supposing that both $a = -\frac{\mathrm{d}\ln r}{\mathrm{d}t}$ and $b = \frac{\mathrm{d}\phi}{\mathrm{d}t}$ are constant, we can find the following formula for f:

$$\begin{aligned} r &= f(\phi) \\ &= r_0 \exp\left(-\frac{(\phi - \phi_0)}{b/a}\right) \end{aligned} \tag{9.29}$$

Figure 9.11**a)** and **b)** show the plot of a and b calculated for simulations with drift angle ψ varying from $-180°$ to $180°$. Figure 9.11**c)** shows the plot of b/a, and we can note how the curve strongly resembles a tangent function. Figure 9.11**d)** shows the plot of $\mathrm{atan2}(b, a)$, which shows that at least for $\psi \in [-90, 90]°$, we can state:

$$\psi = \mathrm{atan2}(b, a) \tag{9.30}$$

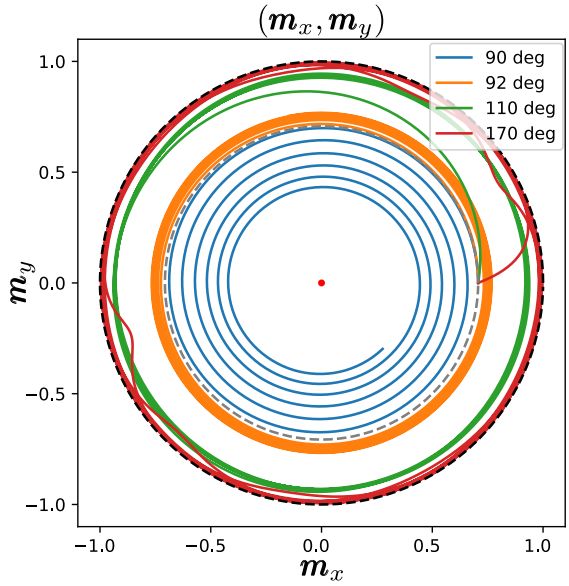


Figure 9.8.: Top view of $\boldsymbol{m}$ for multiple simulated extreme spin errors.
Top view of $\boldsymbol{m}$ for multiple simulated extreme spin errors.

And finally:

$$r = r_0 \exp\left(-\frac{(\phi - \phi_0)}{\tan(\psi)}\right) \tag{9.31}$$

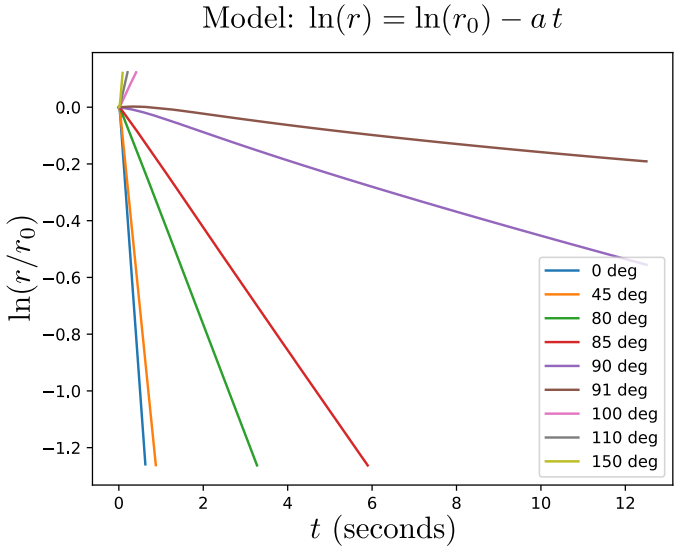


Figure 9.9.: Plot of the logarithm of r, illustrating that it approaches a straight line.

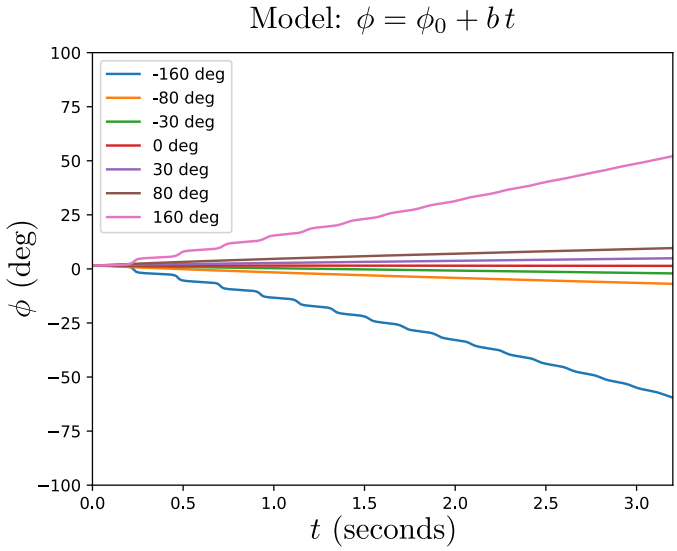


Figure 9.10.: Plot of the phase ϕ, illustrating that it also approaches a straight line.

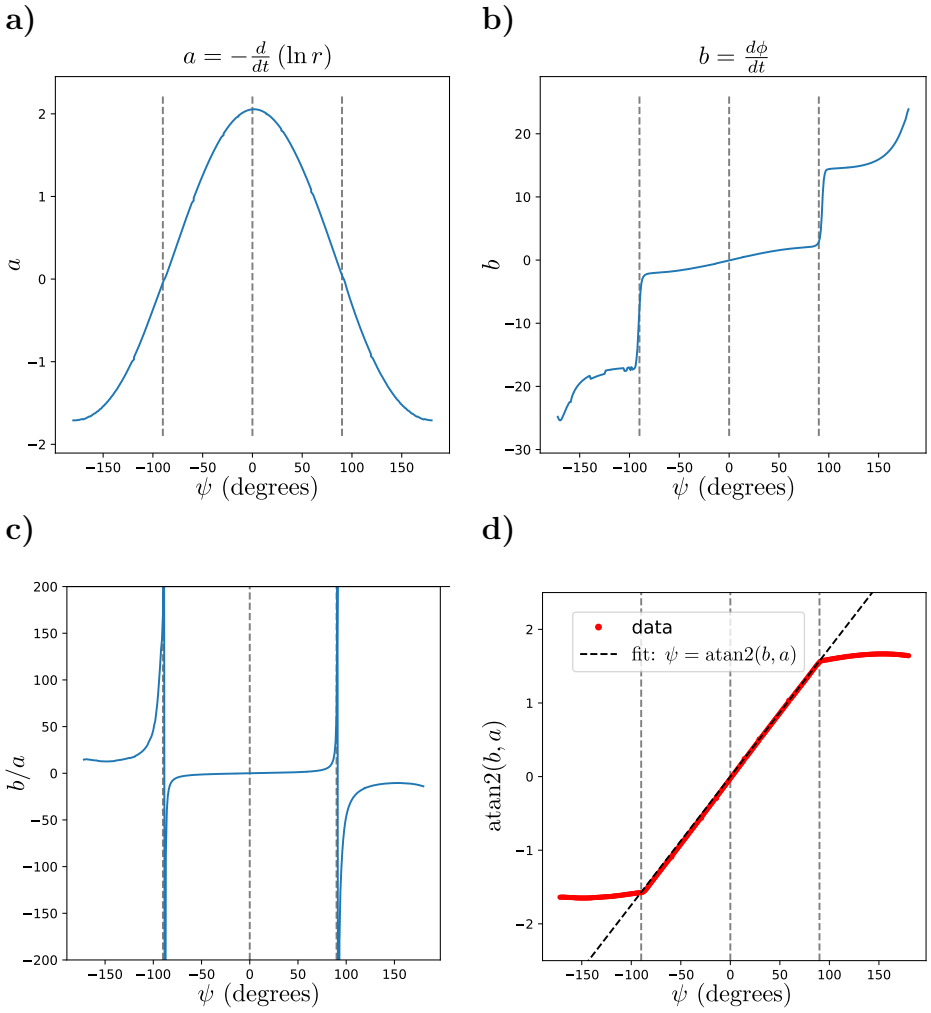


Figure 9.11.: **a)** plot of a, the slope of $-\ln(r)$. **b)** plot of b, ϕ. **c)** plot of the ratio b/a, suggesting a tangent-like relationship between ψ and this ratio. **d)** plot of $\mathrm{atan2}(b, a)$ further demonstrating this relationship.

9.7.3. Simplification of spin error formula

In this section, we will analyze the terms a and b of the spin error estimate and derive a more direct formula. By definition:

$$\begin{aligned} a &= -\frac{\mathrm{d}}{\mathrm{d}t}(\ln r) \\ &= -\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\left(\ln r^2\right) \\ &= -\frac{1}{2r^2}\frac{\mathrm{d}}{\mathrm{d}t}(\Delta\boldsymbol{\rho}\cdot\Delta\boldsymbol{\rho}) \\ &= -\frac{\Delta\boldsymbol{\rho}\cdot\Delta\dot{\boldsymbol{\rho}}}{r^2} \end{aligned} \tag{9.32}$$

Which is equivalent to:

$$\begin{aligned} a &= \frac{(\Delta\boldsymbol{m}\cdot\boldsymbol{e}_z)(\Delta\dot{\boldsymbol{m}}\cdot\boldsymbol{e}_z) - \Delta\boldsymbol{m}\cdot\Delta\dot{\boldsymbol{m}}}{r^2} \\ &= -\frac{(\boldsymbol{e}_z\times\Delta\boldsymbol{m})\cdot(\boldsymbol{e}_z\times\Delta\dot{\boldsymbol{m}})}{r^2} \end{aligned} \tag{9.33}$$

Noting that, since $\Delta\dot{\boldsymbol{m}} = \dot{\boldsymbol{m}}$, if $\bar{\boldsymbol{m}} = \mathbf{0}$, then the factor $\Delta\boldsymbol{m}\cdot\Delta\dot{\boldsymbol{m}}$ is equal to zero.

By definition:

$$\begin{aligned} b &= \frac{\mathrm{d}}{\mathrm{d}t}(\mathrm{atan2}(\Delta\boldsymbol{\rho}_y, \Delta\boldsymbol{\rho}_x)) \\ &= \frac{1}{\Delta\boldsymbol{\rho}_x^2 + \Delta\boldsymbol{\rho}_y^2}\left(\Delta\boldsymbol{\rho}_x\Delta\dot{\boldsymbol{\rho}}_y - \Delta\boldsymbol{\rho}_y\Delta\dot{\boldsymbol{\rho}}_x\right) \\ &= \frac{1}{r^2}\left(\Delta\boldsymbol{m}_x\Delta\dot{\boldsymbol{m}}_y - \Delta\boldsymbol{m}_y\Delta\dot{\boldsymbol{m}}_x\right) \\ &= \frac{1}{r^2}\left(\Delta\dot{\boldsymbol{m}}\cdot(\boldsymbol{e}_z\times\Delta\boldsymbol{m})\right) \end{aligned} \tag{9.34}$$

And finally:

$$b = \frac{\boldsymbol{e}_z\cdot(\Delta\boldsymbol{m}\times\Delta\dot{\boldsymbol{m}})}{r^2} \tag{9.35}$$

Using Eqs. 9.33 and 9.35, and noting that r^2 is positive and can be ignored inside the arc tangent function:

$$\begin{aligned} \psi &= \mathrm{atan2}\left(\boldsymbol{e}_z\cdot(\Delta\boldsymbol{m}\times\Delta\dot{\boldsymbol{m}}),\ -(\boldsymbol{e}_z\times\Delta\boldsymbol{m})\cdot(\boldsymbol{e}_z\times\Delta\dot{\boldsymbol{m}})\right) \\ &= \mathrm{atan2}\left(\boldsymbol{e}_z\cdot(\Delta\boldsymbol{m}\times\Delta\dot{\boldsymbol{m}}),\ (\Delta\boldsymbol{m}\cdot\boldsymbol{e}_z)(\Delta\dot{\boldsymbol{m}}\cdot\boldsymbol{e}_z) - \Delta\boldsymbol{m}\cdot\Delta\dot{\boldsymbol{m}}\right) \end{aligned} \tag{9.36}$$

9.7.4. Test of spiral fit

Figure 9.12 uses Eq. 9.31 to predict the trajectory of $\boldsymbol{m}$ in a simulation with $\psi = 80°$ for a starting at two different points. We can see how the prediction is better when we do not pick the first point as the initial point, since the both a and b must first converge to their values. We can note, however, from Figure 9.12**a)** that even when the fit is not perfect, it can still predict the trajectory very well.

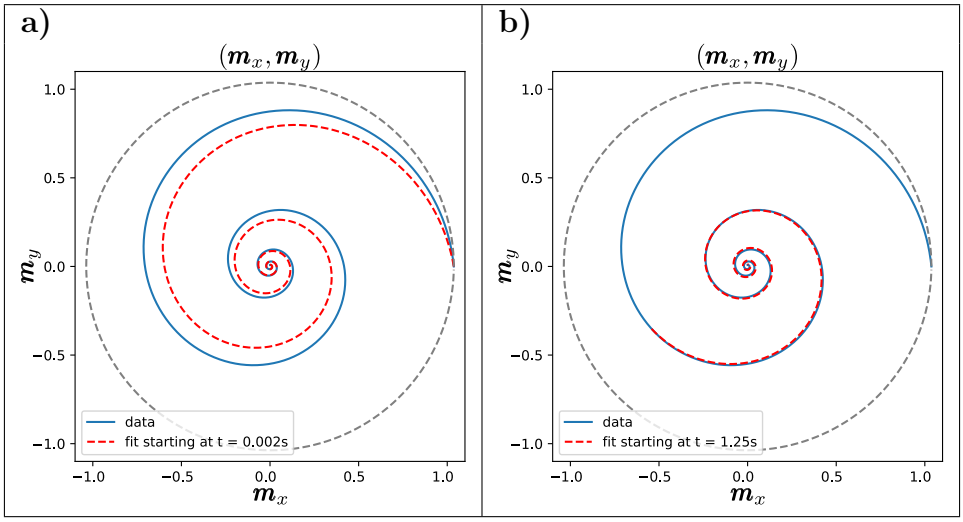


Figure 9.12.: Prediction of $\boldsymbol{m}$ trajectory for a simulation with $\psi = 80°$, starting with a 90° inclination. **a)** fit starting at $t = 0.002$s, and **b)** fit starting at $t = 1.25$s.

9.7.5. Simulation 4: drifting spin error

In this simulation, we aim to analyze the most common case of spin error: the case of a drifting error. We will start the simulation at the extreme case of $\boldsymbol{\theta} = (0°, 150°)$ and the goal is $\boldsymbol{\theta} = (0°, 0°)$. The spin error start at $\psi = 0°$, but it will keep increasing with $\dot{\psi} = 25°/s$ (an extreme drift, but useful to analyze the results of our spin error estimation). Figure 9.13 shows $\boldsymbol{m}$ for this test. We can see the controller starts well, but then something happens, and the system becomes unstable. This is expected, since the error ψ will arrive at some point at a value that is too high for the system to be able to converge. Figure 9.14 shows the estimated ψ for simulation 4. In green, the safe region where $|\psi| < 90°$. In blue, the unsafe region where $|\psi| > 90°$. Note that even though our model did not predict the formula would work in these cases, it still sometimes works, but it is not reliable. In red, the region where $r < 10^{-5}$. In this region, the robot has converged to the desired goal and no more movement is expected (apart from the spin). This means that $a = 0, b = 0$ and $\text{atan2}(b, a)$ is undefined, meaning that we will only get numerical floating point noise.

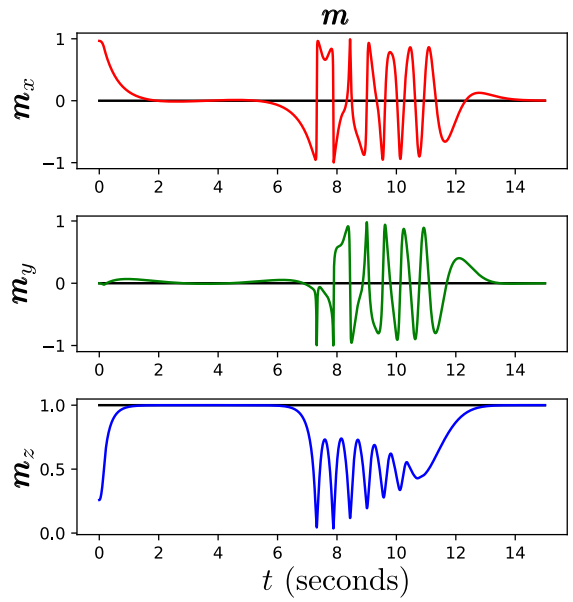


Figure 9.13.: Simulation 4. We see that the controller manages to stabilize the system at first, but then it gets out of control when the drift error gets too big.

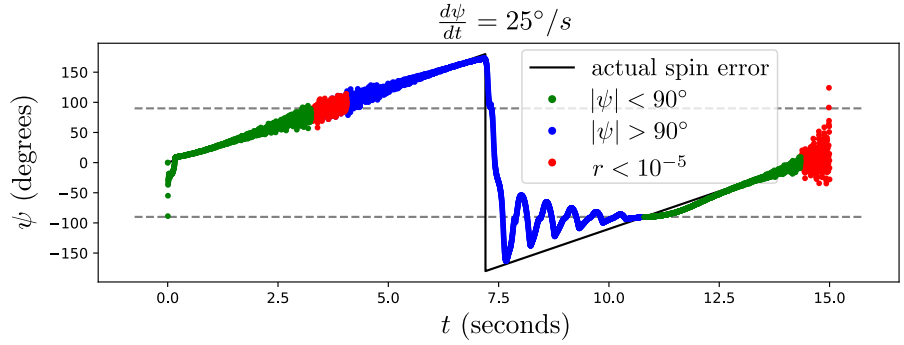


Figure 9.14.: Drift estimation for simulation 4. In green, the safe region where $|\psi| < 90°$. In blue, the unsafe region where $|\psi| > 90°$. Note that even though our model did not predict the formula would work in these cases, it still sometimes works, but it is not reliable. In red, the region where $r < 10^{-5}$.

9.7.6. Theoretical analysis of spin error formula

In this section, we will demonstrate that the formula from Equation 9.36 can be easily demonstrated with a few simple assumptions. The controller designed in Chapter 8 works, in the ideal case, by directly approaching $\boldsymbol{\rho}$ to $\bar{\boldsymbol{\rho}}$, minimizing $\Delta\boldsymbol{\rho}$ by a straight line. shown in Fig 9.15 **a)**. Denoting the unit vector that goes from $\boldsymbol{\rho}$ to $\bar{\boldsymbol{\rho}}$ as $\boldsymbol{n}$, we know that:

$$\boldsymbol{n} = -\frac{\Delta\boldsymbol{\rho}}{|\Delta\boldsymbol{\rho}|} \tag{9.37}$$

And in the ideal case, shown in Fig 9.15 **b)**, the unit vector in the direction of $\Delta\dot{\boldsymbol{\rho}}$ is aligned with it:

$$\frac{\Delta\dot{\boldsymbol{\rho}}}{|\Delta\dot{\boldsymbol{\rho}}|} = \boldsymbol{n} \tag{9.38}$$

For some spin error ψ, we can expect that the controller force will be rotated by R_ψ^T, equivalent to a rotation by $-\psi$ around the axis $\boldsymbol{e}_z$. We show this in Fig 9.15 **c)**.

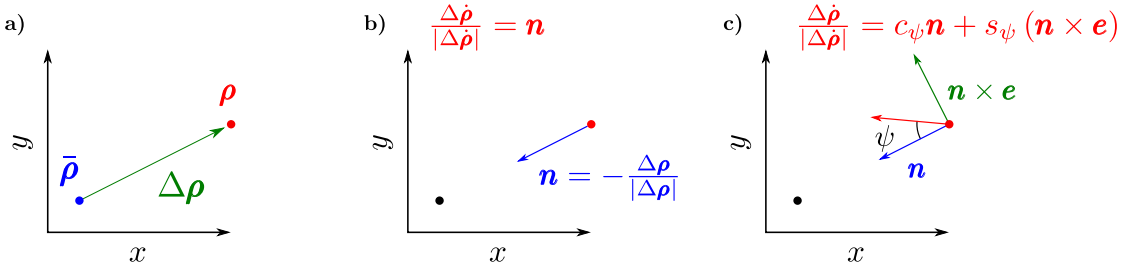


Figure 9.15.: Diagrams for theoretical analysis of spin drift effect. **a)** Plot of $\boldsymbol{\rho}$, the projection of $\boldsymbol{m}$ in the xy-plane, $\bar{\boldsymbol{\rho}}$ the goal, and $\Delta\boldsymbol{\rho}$, the difference between them. **b)** Optimal scenario: the controller works as expected, and $\boldsymbol{\rho}$ approaches $\bar{\boldsymbol{\rho}}$, and $\Delta\dot{\boldsymbol{\rho}}$ point in the direction of $-\Delta\boldsymbol{\rho}$. **c)** Spin error scenario: the controller has an angle error of ψ.

Defining $\lambda = |\Delta\boldsymbol{\rho}|/|\Delta\dot{\boldsymbol{\rho}}|$, we can state:

$$-\lambda\Delta\dot{\boldsymbol{\rho}} = c_\psi\Delta\boldsymbol{\rho} + s_\psi(\Delta\boldsymbol{\rho}\times\boldsymbol{e}_z) \tag{9.39}$$

Calculating the cross product with $\Delta\boldsymbol{\rho}$, and noting that $\Delta\boldsymbol{\rho}\cdot\boldsymbol{e}_z = 0$:

$$\begin{aligned} -\lambda\Delta\boldsymbol{\rho}\times\Delta\dot{\boldsymbol{\rho}} &= s_\psi\widehat{\Delta\boldsymbol{\rho}}^2\boldsymbol{e}_z \\ &= s_\psi\left((\Delta\boldsymbol{\rho}\cdot\boldsymbol{e}_z)\boldsymbol{e}_z - |\Delta\boldsymbol{\rho}|^2\boldsymbol{e}_z\right) \\ &= -s_\psi|\Delta\boldsymbol{\rho}|^2\boldsymbol{e}_z \end{aligned} \tag{9.40}$$

Applying now the dot product with $\boldsymbol{e}_z$ yields:

$$s_\psi = \frac{\lambda}{r^2}\boldsymbol{e}_z\cdot(\Delta\boldsymbol{\rho}\times\Delta\dot{\boldsymbol{\rho}}) \tag{9.41}$$

Moreover:

$$\begin{aligned} \Delta\boldsymbol{\rho}\times\Delta\dot{\boldsymbol{\rho}} &= (\Delta\boldsymbol{m} - (\Delta\boldsymbol{m}\cdot\boldsymbol{e}_z)\,\boldsymbol{e}_z)\times(\Delta\dot{\boldsymbol{m}} - (\Delta\dot{\boldsymbol{m}}\cdot\boldsymbol{e}_z)\,\boldsymbol{e}_z) \\ &= \Delta\boldsymbol{m}\times\Delta\dot{\boldsymbol{m}} - (\Delta\boldsymbol{m}\cdot\boldsymbol{e}_z)(\boldsymbol{e}_z\times\Delta\dot{\boldsymbol{m}}) - (\Delta\dot{\boldsymbol{m}}\cdot\boldsymbol{e}_z)(\Delta\boldsymbol{m}\times\boldsymbol{e}_z) \end{aligned} \tag{9.42}$$

And finally:

$$s_\psi = \frac{\lambda}{r^2} \boldsymbol{e}_z \cdot (\Delta \boldsymbol{m} \times \Delta \dot{\boldsymbol{m}}) \tag{9.43}$$

Calculating this time the cross product of 9.39 with $\boldsymbol{e}_z$:

$$-\lambda \Delta \dot{\boldsymbol{\rho}} \times \boldsymbol{e}_z = c_\psi (\Delta \boldsymbol{\rho} \times \boldsymbol{e}_z) + s_\psi (\Delta \boldsymbol{\rho} \times \boldsymbol{e}_z) \times \boldsymbol{e}_z \tag{9.44}$$

And calculating now the dot product with $(\Delta \boldsymbol{\rho} \times \boldsymbol{e}_z)$:

$$-\lambda (\Delta \dot{\boldsymbol{\rho}} \times \boldsymbol{e}_z) \cdot (\Delta \boldsymbol{\rho} \times \boldsymbol{e}_z) = c_\psi |\Delta \boldsymbol{\rho} \times \boldsymbol{e}_z|^2 \tag{9.45}$$

Noting that $|\Delta \boldsymbol{\rho} \times \boldsymbol{e}_z|^2 = |\Delta \boldsymbol{\rho}|^2$ and that $(\Delta \dot{\boldsymbol{\rho}} \times \boldsymbol{e}_z) = (\Delta \dot{\boldsymbol{m}} \times \boldsymbol{e}_z)$:

$$c_\psi = -\frac{\lambda}{r^2} (\boldsymbol{e}_z \times \Delta \boldsymbol{m}) \cdot (\boldsymbol{e}_z \times \Delta \dot{\boldsymbol{m}}) \tag{9.46}$$

Using Eqs. 9.43 and 9.46, and noting that λ / r is a common positive factor, we can finally state that:

$$\psi = \operatorname{atan2} \left(\boldsymbol{e}_z \cdot (\Delta \boldsymbol{m} \times \Delta \dot{\boldsymbol{m}}),\ -(\boldsymbol{e}_z \times \Delta \boldsymbol{m}) \cdot (\boldsymbol{e}_z \times \Delta \dot{\boldsymbol{m}}) \right) \tag{9.47}$$

Note that the theoretically demonstrated Eq. 9.47 and the simplified experimental formula from Eq. 9.36 are the same.

9.8. Final thoughts

In this chapter, the full simulation of the proposed monorotor UAV was described. Here we present all the theoretical and practical details used for the creation of a closed-loop simulation assembling the work detailed in the previous chapters: the model of the motor system equipped with the swashplateless mechanism, the equations of motion derived with the Euler-Poincaré equations, the rotation decomposition and subsequent non-linear controller architecture. We were able to show that this controller has the exact desired effect on the robot's rotation: it acts directly on the middle vector $\boldsymbol{m}$, while ignoring the spin rotation.

Moreover, we also used this simulation to study the effect of a poor estimation of the spin rotation on the controller's performance. This led to the expression of a new observer that uses the dynamics of the controllable part of the rotation to improve the estimation of the uncontrollable spin.

10. A concise and general formula for the direct conversion between a quaternion and the Euler angles in any sequence

Current methods of the conversion between a rotation quaternion and Euler angles are either a complicated set of multiple sequence-specific implementations, or a complicated method relying on multiple matrix multiplications. In this paper, a general formula is presented for extracting the Euler angles in any desired sequence from a unit quaternion. This is a direct method, in that no intermediate conversion step is required (no quaternion-to-rotation matrix conversion, for example) and it is general because it works with all 12 possible sequences of rotations. A closed formula was first developed for extracting angles in any of the 12 possible sequences, both "Proper Euler angles" and "Tait-Bryan angles". The resulting algorithm was compared with a popular implementation of the matrix-to-Euler angle algorithm, which involves a quaternion-to-matrix conversion in the first computational step. Lastly, a single-page pseudocode implementation of this algorithm is presented, illustrating its conciseness and straightforward implementation. With an execution speed 30 times faster than the classical method, our algorithm can be of great interest in every aspect. Then, we will see how this can be extended to the other six sequences with a simple transformation.

This work has been published in [6][1], and the official SciPy development branch has since replaced the algorithm on [58] with this algorithm.

10.1. Introduction

When dealing with 3D orientation problems, many formalisms can be used to describe a given rotation [60], each of which has its own set of advantages and shortcomings. Arguably the most direct representation of a 3D rotation is a matrix $R \in \mathrm{SO}(3)$, where SO(3) is the group of invertible 3×3 matrices such that for

[1] Sadly, some minor (but annoying) definition errors passed the reviews and were published. These errors were corrected for this chapter.

any R, $\det(R) = 1$ and $RR^T = R^T R = \mathbb{I}$, where $\mathbb{I}$ is the identity matrix. These rotation matrices represent direct linear transformations such that, with $\boldsymbol{v} \in \mathbb{R}^3$:

$$\boldsymbol{v}_{\text{rotated}} = R\,\boldsymbol{v} \tag{10.1}$$

Apart from being simple to use, a rotation matrix also has the advantage of being continuous, and a simple matrix multiplication can be used to compose rotations: $R = R_2\, R_1$ is the rotation matrix corresponding to a rotation by R_1 followed by a rotation by R_2. 3D rotation matrices have some numerical shortcomings, however. For example, as many as 9 numbers (and 6 constraints) are required to represent a 3 degree of freedom rotation, and it can be difficult and computationally costly to orthogonalize a rotation matrix numerically [54] (i.e., to check that the matrix has its determinant equal to 1 and its inverse equal to its transpose, which is necessary to compensate for the accumulated floating point errors).

However, it is possible to parametrize this rotation matrix with a smaller set of numbers [64]. One of the most usual set of parameters are the Euler angles. The approach consists in decomposing the 3D rotation matrix into the product of three rotations:

$$R = R_{\theta_3\,\boldsymbol{e}_3}\; R_{\theta_2\,\boldsymbol{e}_2}\; R_{\theta_1\,\boldsymbol{e}_1} \tag{10.2}$$

Where $R_{\theta\,\boldsymbol{e}}$ is a rotation by the angle θ around the axis $\boldsymbol{e}$, and the consecutive axes are orthogonal ($\boldsymbol{e}_1 \cdot \boldsymbol{e}_2 = \boldsymbol{e}_2 \cdot \boldsymbol{e}_3 = 0$). The advantages of using Euler angles include the fact that only three numbers have to be stored, and due to their familiarity, they can be more easily understood, which explains why they are still being so widely used, even in cases where other forms of representation may be more appropriate. The use of Euler angles also has several disadvantages. For example, they are discontinuous, and it is difficult to directly compose two 3D rotations expressed in Euler angles. Euler angles are also affected by the phenomenon commonly called "gymbal lock": when two axes become aligned, making the system underdetermined, special care has to be taken. In addition, since there are 12 possible axis sequences (24, when considering the difference between "intrinsic" and "extrinsic" rotations), the correct sequence has to be checked in the case of each application. An arguably preferable parametrization are quaternions. A quaternion is a hypercomplex number defined by one real part and three distinct imaginary parts (which can also be regarded as the vector part). When the norm of a quaternion is equal to 1, quaternions are a useful and efficient representation of 3D orientation: they can be composed as easily as rotation matrices, they are continuous, and they are easily constructed from the axis-angle representation. In addition, quaternions can be normalized trivially, which is much more efficient than having to cope with the corresponding matrix orthogonalization problem. For these reasons, most 3D graphical applications and rotation engines carry quaternions under the hood. Besides these advantages, Euler angles are still being preferred by many authors: Euler angles are the most familiar concept to most engineers and researchers. In addition, in the case of many

problems in which there exists only one degree of freedom, angles can suffice.

To be able to perform fast calculations with quaternions and at the same time analyze rotations using angles, it might be necessary to have an efficient method of converting the one set of parameters to the other. Calculating the corresponding quaternion (or rotation matrix) for a given set of Euler angles is trivial. Extracting the Euler angles is much harder, however. One of the following two methods has generally been used up to now. The first method consists in adopting a different set of formulas for each possible angle sequence [26], which is difficult to implement and debug. The second method is that described in [58]. SciPy [71], for example, a widely used scientific library for the Python programming language, implements this method. It converts rotation matrices into Euler angles and involves many matrix multiplications, including the inverse trigonometric functions required, which are naturally computationally costly. In addition, if rotations are stored in the form of quaternions (as is usually the case in many of the 3D rendering software tools dealing with rotations), an additional conversion step from quaternions to rotation matrices is necessary.

Since many robotic, graphic and other high-level applications involve the use of quaternions (even if they are hidden from the user), it can be necessary to have a concise, efficient method for the conversion between quaternions and Euler angles. The direct conversion formula from quaternions to Euler angles presented here requires fewer computational steps and less expensive computational resources. Moreover, this conversion formula is much simpler to implement and debug, making it a great option for any new applications needing to implement this kind of conversions.

10.2. Formula development

In the section, the formula for the conversion between a quaternion and any of the 6 proper Euler angle sequences is derived, and then an adaptation for the 6 remaining Tait-Bryan sequences is demonstrated.

10.2.1. Case 1: Proper Euler angles

Assuming $q = [q_\mathrm{r},\, \boldsymbol{q}^T]^T$ is unit and known, it can be decomposed as follows:

$$q = \begin{bmatrix} c_3 \\ s_3\boldsymbol{e} \end{bmatrix} \odot \begin{bmatrix} c_2 \\ s_2\boldsymbol{e}' \end{bmatrix} \odot \begin{bmatrix} c_1 \\ s_1\boldsymbol{e} \end{bmatrix} \tag{10.3}$$

In which (for $0 \leq \theta_2 \leq \pi$):

$$\begin{aligned} s_1 &\equiv \sin(\theta_1/2),\ c_1 \equiv \cos(\theta_1/2) \\ s_2 &\equiv \sin(\theta_2/2),\ c_2 \equiv \cos(\theta_2/2) \\ s_3 &\equiv \sin(\theta_3/2),\ c_3 \equiv \cos(\theta_3/2) \end{aligned} \tag{10.4}$$

Where $s_2 \geq 0,\ c_2 \geq 0$. Note that we **choose** a solution in which $c_2 > 0$ and $s_2 > 0$. Appendix J shows how we can come up with a different set of solutions. Taking $\boldsymbol{e}$ and $\boldsymbol{e}'$ to be orthogonal unit vectors ($\boldsymbol{e} \cdot \boldsymbol{e}' = 0$), there is a third unit vector which is orthogonal to the other two, such that:

$$\boldsymbol{e}'' \equiv \varepsilon\, \boldsymbol{e} \times \boldsymbol{e}' \tag{10.5}$$

Where $\varepsilon = (\boldsymbol{e} \times \boldsymbol{e}') \cdot \boldsymbol{e}'' = \pm 1$. Together, $\boldsymbol{e}, \boldsymbol{e}'$ and $\boldsymbol{e}''$ form an orthonormal basis. We also define:

$$\begin{aligned} \theta_+ &= \frac{\theta_3 + \theta_1}{2} \\ \theta_- &= \frac{\theta_3 - \theta_1}{2} \end{aligned} \tag{10.6}$$

And:

$$\begin{aligned} s_+ &\equiv \sin(\theta_+) = c_1 s_3 + s_1 c_3 \\ s_- &\equiv \sin(\theta_-) = c_1 s_3 - s_1 c_3 \\ c_+ &\equiv \cos(\theta_+) = c_1 c_3 - s_1 s_3 \\ c_- &\equiv \cos(\theta_-) = c_1 c_3 + s_1 s_3 \end{aligned} \tag{10.7}$$

Analyzing Eq. 10.3:

$$\begin{aligned} q &= \begin{bmatrix} c_3 \\ s_3 \boldsymbol{e} \end{bmatrix} \odot \begin{bmatrix} c_2 \\ s_2 \boldsymbol{e}' \end{bmatrix} \odot \begin{bmatrix} c_1 \\ s_1 \boldsymbol{e} \end{bmatrix} \\ &= c_2 \begin{bmatrix} c_3 \\ s_3 \boldsymbol{e} \end{bmatrix} \odot \begin{bmatrix} c_1 \\ s_1 \boldsymbol{e} \end{bmatrix} + s_2 \begin{bmatrix} c_3 \\ s_3 \boldsymbol{e} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e}' \end{bmatrix} \odot \begin{bmatrix} c_1 \\ s_1 \boldsymbol{e} \end{bmatrix} \\ &= c_2 \begin{bmatrix} c_+ \\ s_+ \boldsymbol{e} \end{bmatrix} + s_2 \begin{bmatrix} 0 \\ c_- \boldsymbol{e}' + s_- \boldsymbol{e} \times \boldsymbol{e}' \end{bmatrix} \end{aligned} \tag{10.8}$$

And noting that $\boldsymbol{e} \times \boldsymbol{e}' = \varepsilon\, \boldsymbol{e}''$:

$$q = \begin{bmatrix} c_2 c_+ \\ c_2 s_+ \boldsymbol{e} + s_2 c_- \boldsymbol{e}' + s_2 s_- \varepsilon \boldsymbol{e}'' \end{bmatrix} \tag{10.9}$$

Defining the following four components:

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \equiv \begin{bmatrix} q_{\mathrm{r}} \\ \boldsymbol{q} \cdot \boldsymbol{e} \\ \boldsymbol{q} \cdot \boldsymbol{e}' \\ \varepsilon\, \boldsymbol{q} \cdot \boldsymbol{e}'' \end{bmatrix} \tag{10.10}$$

We obtain:

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} c_2 c_+ \\ c_2 s_+ \\ s_2 c_- \\ s_2 s_- \end{bmatrix} \tag{10.11}$$

Alternatively, we can see that $\begin{bmatrix} b & c & d \end{bmatrix}^T$ is simply a permutation of the components of $\boldsymbol{q}$:

$$\begin{bmatrix} b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \boldsymbol{e} & \boldsymbol{e}' & \boldsymbol{e} \times \boldsymbol{e}' \end{bmatrix}^T \boldsymbol{q} \tag{10.12}$$

10.2.1.1. Extraction of angles

Using complex numbers, we can define:

$$\begin{aligned} z_+ &\equiv a + ib = c_2(c_+ + is_+) \\ z_- &\equiv c + id = s_2(c_- + is_-) \end{aligned} \tag{10.13}$$

Since $c_2, s_2 \geq 0$, we know that $|z_+| = c_2$, $\arg(z_+) = \theta_+$, $|z_-| = s_2$ and $\arg(z_-) = \theta_-$. We can then rewrite:

$$\begin{aligned} z_+ &= a + ib = c_2 \exp(i\theta_+) \\ z_- &= c + id = s_2 \exp(i\theta_-) \end{aligned} \tag{10.14}$$

And we know that:

$$\begin{aligned} \theta_+ &= \frac{\theta_3 + \theta_1}{2} = \arg\{a + ib\} = \operatorname{atan2}(b, a) \\ \theta_- &= \frac{\theta_3 - \theta_1}{2} = \arg\{c + id\} = \operatorname{atan2}(d, c) \end{aligned} \tag{10.15}$$

10.2.1.2. Singularities

There are two different singularities in these expressions. When $\theta_2 = 0$, we have $s_2 = 0$ and θ_- is undefined. When $\theta_2 = \pi$, we have $c_2 = 0$ and θ_+ is undefined. In

both cases, one degree of freedom is lost, and we can argue that θ_1 (or alternatively, θ_3) loses its geometrical meaning. We can then either set θ_1 to zero, or keep it fixed in its latest value (for example, when updating an estimator, for the sake of continuity). Defining:

$$\theta_1 \equiv \hat{\theta}_1 \text{ , if } \theta_2 = 0 \text{ or } \theta_2 = \pi \tag{10.16}$$

Taking $\hat{\theta}_1$ to be some constant (zero or otherwise), we can calculate:

$$\begin{cases} \theta_3 = 2\,\text{atan2}(b, a) - \hat{\theta}_1 \text{ , when } \theta_2 = 0 \\ \theta_3 = 2\,\text{atan2}(d, c) + \hat{\theta}_1 \text{ , when } \theta_2 = \pi \end{cases} \tag{10.17}$$

10.2.1.3. General formula for θ_1 and θ_3 in the absence of singularities

If $\theta_2 \neq 0$ and $\theta_2 \neq \pi$, multiplying z_+ and z_- yields:

$$\begin{aligned} z_+\, z_- &= (a + ib)(c + id) \\ &= c_2 s_2 \exp\left(i\frac{\theta_3 + \theta_1 + \theta_3 - \theta_1}{2}\right) \\ &= c_2 s_2 \exp(i\theta_3) \end{aligned} \tag{10.18}$$

On similar lines, multiplying z_+ and the conjugate of z_- yields:

$$\begin{aligned} z_+\, z_-^* &= (a + ib)(c - id) \\ &= c_2 s_2 \exp(i\theta_1) \end{aligned} \tag{10.19}$$

The angles can then be obtained using:

$$\begin{aligned} \theta_1 &= \arg(z_+\, z_-^*) = \arg((a + ib)(c - id)) \\ \theta_3 &= \arg(z_+\, z_-) = \arg((a + ib)(c + id)) \end{aligned} \tag{10.20}$$

Or, more simply, from Eq. 10.15:

$$\begin{aligned} \theta_1 &= \arg(a + ib) - \arg(c + id) \\ \theta_3 &= \arg(a + ib) + \arg(c + id) \end{aligned} \tag{10.21}$$

Or:

$$\begin{aligned} \theta_1 &= \theta_+ - \theta_- \\ \theta_3 &= \theta_+ + \theta_- \end{aligned} \tag{10.22}$$

It is worth noting that Eq. 10.21 requires fewer operations than Eq. 10.20: only 2 calls to atan2, one addition and one subtraction, but a final wrapping step may be required in order to either keep the angles either in $(-\pi, \pi]$ or $[0, 2\pi)$.

10.2.1.4. General formulas for calculating θ_2

From Eq. 10.14, we know that:

$$\begin{aligned} c_2 &= \cos\left(\theta_2/2\right) = |z_+| = \sqrt{a^2 + b^2} \\ s_2 &= \sin\left(\theta_2/2\right) = |z_-| = \sqrt{c^2 + d^2} \end{aligned} \tag{10.23}$$

And we can use any of the following equivalent formulas obtained directly from the definition:

$$\theta_2 = 2\,\mathrm{asin}\left(\sqrt{\frac{c^2 + d^2}{n^2}}\right) = 2\,\mathrm{acos}\left(\sqrt{\frac{a^2 + b^2}{n^2}}\right) = 2\,\mathrm{atan}\left(\sqrt{\frac{c^2 + d^2}{a^2 + b^2}}\right) \tag{10.24}$$

Where the factor $n^2 = a^2 + b^2 + c^2 + d^2 = |q|^2$ can be ignored if the quaternion is already normalized. Using the properties of inverse trigonometric functions, we can also find the following formula, which avoids the need for a square root:

$$\theta_2 = \mathrm{acos}\left(2\,\frac{a^2 + b^2}{n^2} - 1\right) \tag{10.25}$$

Discussions during the implementation of this algorithm in the SciPy library led me to realize that usual implementations of acos and asin might lead to problems near zero. The consequence is that using an atan2 formula for θ_2 is also preferred, not for set problems but rather for numerical reasons[2]:

$$\theta_2 = 2\,\mathrm{atan2}\left(\sqrt{c^2 + d^2}, \sqrt{a^2 + b^2}\right) \tag{10.26}$$

10.2.2. Case 2: Tait-Bryan angles

We now define:

$$q = \begin{bmatrix} c_3 \\ s_3\boldsymbol{e}'' \end{bmatrix} \odot \begin{bmatrix} c_2 \\ s_2\boldsymbol{e}' \end{bmatrix} \odot \begin{bmatrix} c_1 \\ s_1\boldsymbol{e} \end{bmatrix} \tag{10.27}$$

[2] I would like to personally thank maintainer Nikolay Mayorov for pointing that out: https://github.com/scipy/scipy/pull/17392

Where $-\pi/2 < \phi_2 < \pi/2$. Again assuming that $\boldsymbol{e}$, $\boldsymbol{e}'$ and $\boldsymbol{e}''$ are orthogonal unit vectors and $\boldsymbol{e}'' = \varepsilon\,\boldsymbol{e} \times \boldsymbol{e}'$, where $\varepsilon = (\boldsymbol{e} \times \boldsymbol{e}') \cdot \boldsymbol{e}'' = \pm 1$, we define:

$$\lambda \equiv \begin{bmatrix} \cos(\pi/4) \\ \sin(\pi/4)\boldsymbol{e}' \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ \boldsymbol{e}' \end{bmatrix} \tag{10.28}$$

We note that:

$$\begin{aligned} \lambda^* \odot \begin{bmatrix} c_3 \\ s_3\varepsilon\boldsymbol{e} \end{bmatrix} \odot \lambda &= \begin{bmatrix} c_3 \\ s_3\varepsilon\boldsymbol{e} \times \boldsymbol{e}' \end{bmatrix} \\ &= \begin{bmatrix} c_3 \\ s_3\boldsymbol{e}'' \end{bmatrix} \end{aligned} \tag{10.29}$$

Which gives:

$$\begin{aligned} q &= \begin{bmatrix} c_3 \\ s_3\boldsymbol{e}'' \end{bmatrix} \odot \begin{bmatrix} c_2 \\ s_2\boldsymbol{e}' \end{bmatrix} \odot \begin{bmatrix} c_1 \\ s_1\boldsymbol{e} \end{bmatrix} \\ q &= \lambda^* \odot \begin{bmatrix} c_3 \\ s_3\varepsilon\boldsymbol{e} \end{bmatrix} \odot \lambda \odot \begin{bmatrix} c_2 \\ s_2\boldsymbol{e}' \end{bmatrix} \odot \begin{bmatrix} c_1 \\ s_1\boldsymbol{e} \end{bmatrix} \\ q' &= \begin{bmatrix} c_3' \\ s_3'\boldsymbol{e} \end{bmatrix} \odot \begin{bmatrix} c_2' \\ s_2'\boldsymbol{e}' \end{bmatrix} \odot \begin{bmatrix} c_1 \\ s_1\boldsymbol{e} \end{bmatrix} \end{aligned} \tag{10.30}$$

Where:

$$\begin{aligned} s_2' &\equiv \sin\theta_2'/2 \ (\geq 0) \\ c_2' &\equiv \cos\theta_2'/2 \ (\geq 0) \\ s_3' &\equiv \sin\theta_3'/2 \\ c_3' &\equiv \cos\theta_3'/2 \end{aligned} \tag{10.31}$$

Where $\theta_2' = \theta_2 + \pi/2$ and $\theta_3' = \varepsilon\theta_3$, and:

$$\begin{aligned} q' &\equiv \lambda \odot q \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ \boldsymbol{e}' \end{bmatrix} \odot \begin{bmatrix} a \\ b\boldsymbol{e} + c\boldsymbol{e}' + d\boldsymbol{e}'' \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} a - c \\ (b+d)\boldsymbol{e} + (c+a)\boldsymbol{e}' + (d-b)\boldsymbol{e}'' \end{bmatrix} \end{aligned} \tag{10.32}$$

Using Eq. 10.32, we can define:

$$\begin{bmatrix} a' \\ b' \\ c' \\ d' \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} a - c \\ b + d \\ c + a \\ d - b \end{bmatrix} \tag{10.33}$$

And then calculate θ_1, θ'_2 and θ'_3 using the formulas obtained in the proper case. Using the acos formula for θ_2, we have:

$$\begin{aligned}\theta_2 = \theta'_2 - \pi/2 &= \operatorname{acos}\left(2\,\frac{a'^2 + b'^2}{n'^2} - 1\right) - \pi/2 \\ &= \operatorname{acos}\left(\frac{a'^2 + b'^2 - c'^2 - d'^2}{n'^2}\right) - \pi/2\end{aligned} \tag{10.34}$$

Moreover, using the fact that $\operatorname{acos}(x) - \pi/2 = \operatorname{asin}(-x)$:

$$\begin{aligned}\theta_2 &= \operatorname{asin}\left(-2\,\frac{a'^2 + b'^2}{n'^2} + 1\right) \\ &= \operatorname{asin}\left(\frac{c'^2 + d'^2 - a'^2 - b'^2}{n'^2}\right)\end{aligned} \tag{10.35}$$

Which results in singularities when $\theta'_2 = 0$ or $\theta'_2 = \pi$, which is equivalent to $\theta_2 = -\pi/2$ or $\theta_2 = \pi/2$, as was to be expected. In addition, we know that when no singularities are present:

$$\begin{aligned}\theta_1 &= \operatorname{atan2}(b', a') - \operatorname{atan2}(d', c') \\ \theta_3 &= \varepsilon\left(\operatorname{atan2}\left(b', a'\right) + \operatorname{atan2}\left(d', c'\right)\right)\end{aligned} \tag{10.36}$$

10.2.3. Example of a proper sequence: ZYZ

If we decide to use the sequence ZYZ, then $\boldsymbol{e} = \boldsymbol{e}_z$, $\boldsymbol{e}' = \boldsymbol{e}_y$ and $\boldsymbol{e}'' = \boldsymbol{e}_z \times \boldsymbol{e}_y = -\boldsymbol{e}_x$. This leads to:

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} q_\mathrm{r} \\ q_\mathrm{z} \\ q_\mathrm{y} \\ -q_\mathrm{x} \end{bmatrix} \tag{10.37}$$

And the general formulas for θ_1, θ_2 and θ_3 (when no singularities are present, and assuming q to have been normalized) are:

$$\begin{aligned}\theta_1 &= \operatorname{atan2}(q_\mathrm{z}, q_\mathrm{r}) - \operatorname{atan2}(-q_\mathrm{x}, q_\mathrm{y}) \\ \theta_2 &= \operatorname{acos}\left(2\,\left(q_\mathrm{r}^2 + q_\mathrm{z}^2\right) - 1\right) \\ \theta_3 &= \operatorname{atan2}(q_\mathrm{z}, q_\mathrm{r}) + \operatorname{atan2}(-q_\mathrm{x}, q_\mathrm{y})\end{aligned} \tag{10.38}$$

10.2.4. Example of a Tait-Bryan sequence: ZYZ

Using the sequence XYZ, equivalent to the common aeronautical angles, then $\boldsymbol{e} = \boldsymbol{e}_x$, $\boldsymbol{e}' = \boldsymbol{e}_y$ and $\boldsymbol{e}'' = \boldsymbol{e}_z$. This leads to:

$$\begin{bmatrix} a' \\ b' \\ c' \\ d' \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} q_\mathrm{r} - q_\mathrm{y} \\ q_\mathrm{x} + q_\mathrm{z} \\ q_\mathrm{y} + q_\mathrm{r} \\ q_\mathrm{z} - q_\mathrm{x} \end{bmatrix} \tag{10.39}$$

And the general formulas for θ_1, θ_2 and θ_3 are:

$$\begin{aligned} \theta_1 &= \operatorname{atan2}(q_\mathrm{x} + q_\mathrm{z}, q_\mathrm{r} - q_\mathrm{y}) - \operatorname{atan2}(q_\mathrm{z} - q_\mathrm{x}, q_\mathrm{y} + q_\mathrm{r}) \\ \theta_2 &= \operatorname{acos}\left((q_\mathrm{r} - q_\mathrm{y})^2 + (q_\mathrm{x} + q_\mathrm{z})^2 - 1\right) - \pi/2 \\ \theta_3 &= \operatorname{atan2}(q_\mathrm{x} + q_\mathrm{z}, q_\mathrm{r} - q_\mathrm{y}) + \operatorname{atan2}(q_\mathrm{z} - q_\mathrm{x}, q_\mathrm{y} + q_\mathrm{r}) \end{aligned} \tag{10.40}$$

10.3. Complete algorithm

Algorithm 1, presented in this section, implements the conversion method from this work. Assuming that our inputs are $q \in \mathbb{R}^4$, the rotation quaternion and i, j and $k \in \mathbb{N}$, an array of integers defining the sequence of angles (for example, $[i, j, k] = [323]$ is equivalent to the sequence ZYZ). A Python implementation can be found on [3].[3]

Many operations are required to convert a quaternion into a rotation matrix. Using the homogeneous formula from Eq. A.51, for example, if special care is taken in order not to repeat any operations, we have to perform at least $4^2 = 16$ floating point multiplications (all the possible products between two different components of the quaternion, plus all the squares of each component), $4 \times 3 = 12$ multiplications by 2 and $3 \times 3 + 6 = 15$ additions/subtractions. This conversion step alone is more than enough to make an algorithm based on [58] much slower than the proposed method. In addition, multiple matrix multiplications also have to be computed. By comparison, our algorithm replaces all the conversions and matrix multiplications by a simple permutation of the quaternion elements and in the case of Tait-Bryan angles, only 5 additional additions/subtractions and possibly a sign change are required.

[3]Note that this algorithm handles gymbal lock by setting $\theta_1 = 0$.

Input: $q \in \mathbb{R}^4$, and $i, j, k \in \{1, 2, 3\}$, where $i \neq j, j \neq k$
Output: $\theta_1, \theta_2, \theta_3$
if $i == k$ **then**
  $not\_proper \leftarrow$ False
  $k \leftarrow 6 - i - j$ // because $i + j + k = 1 + 2 + 3 = 6$
else
  $not\_proper \leftarrow$ True
end
$\varepsilon \leftarrow (i - j) \times (j - k) \times (k - i)/2$ // equivalent to the Levi-Civita symbol
if not_proper **then**
  $a \leftarrow q[0] - q[j]$
  $b \leftarrow q[i] + q[k] \times \varepsilon$
  $c \leftarrow q[j] + q[0]$
  $d \leftarrow q[k] \times \varepsilon - q[i]$
else
  $a \leftarrow q[0]$
  $b \leftarrow q[i]$
  $c \leftarrow q[j]$
  $d \leftarrow q[k] \times \varepsilon$
end
$\theta_2 \leftarrow 2\,\mathrm{atan2}\left(\sqrt{c^2 + d^2},\ \sqrt{a^2 + b^2}\right)$
$\theta^+ \leftarrow \mathrm{atan2}(b, a)$
$\theta^- \leftarrow \mathrm{atan2}(d, c)$
switch *value of* θ_2 **do**
  case 0 **do**
    $\theta_1 \leftarrow 0$ // For simplicity, we are setting $\hat{\theta}_1 = 0$
    $\theta_3 \leftarrow 2 \times \theta^+ - \theta_1$
  case π **do**
    $\theta_1 \leftarrow 0$
    $\theta_3 \leftarrow 2 \times \theta^- + \theta_1$
  otherwise do
    $\theta_1 \leftarrow \theta^+ - \theta^-$
    $\theta_3 \leftarrow \theta^+ + \theta^-$
  end
end
if not_proper **then**
  $\theta_3 \leftarrow \varepsilon \times \theta_3$
  $\theta_2 \leftarrow \theta_2 - \pi/2$
end
$\theta_1, \theta_3 \leftarrow \mathtt{wrap}(\theta_1, \theta_3)$ // ''wrap'' assures $\theta_1, \theta_3 \in (-\pi, \pi]$ or $\theta_1, \theta_3 \in [0, 2\pi)$

Algorithm 1: Complete implementation of conversion between a rotation quaternion and Euler angles in any sequence, setting $\theta_1 = 0$ in case of singularity.

10.4. Results

In this section, a performance comparison between our method and the Shuster method is presented. We adapted the SciPy library in order to compile the algorithm as described in Section 10.3. A real data set comprising the orientation of a spinning object with 3284 data points was used to compare the efficiency of the two algorithms. The full implementation and data set can be downloaded from [3]. First we noted that both methods give the same results: adding the absolute value of the differences between the two methods in a whole data set gives an error of the order of 10^{-12}. The execution times required in our tests for each sequence (and their ratios) are presented in the Table 10.1. From this test, it can be clearly seen that the method presented here is about 30 times faster.

seq	new method	[58] implemented in [71]	ratio
ZYZ	0.487 s	13.770 s	28.261
ZXZ	0.384 s	13.361 s	34.805
XYX	0.382 s	13.381 s	34.998
XZX	0.414 s	13.187 s	31.832
YXY	0.359 s	13.029 s	36.262
YZY	0.375 s	13.078 s	34.884
ZYX	0.371 s	13.152 s	35.408
ZXY	0.364 s	13.124 s	36.048
XYZ	0.373 s	13.170 s	35.291
XZY	0.385 s	13.157 s	34.213
YXZ	0.365 s	13.087 s	35.838
YZX	0.425 s	13.122 s	30.844

Table 10.1.: Comparison of execution times between the two methods. The Python module *timeit* was used to check the execution time required to convert the whole data set 500 times on an Intel® Core™ i3-4030U CPU with a 1.90GHz clock speed.

10.5. Conclusion

The Euler angles are still a useful intuitive 3D orientation parametrization. A fast method of conversion to/from any other set of parameters can therefore be of great interest for displaying or analyzing data, for instance. In this study, we therefore developed a general formula for this conversion which is concise, easy to implement and easy to debug. In addition, the fact that our method is about 30 times faster than the method proposed by [58], which required an intermediate conversion into rotation matrices, we believe that our proposed method can be of great interest. This faster execution time also makes this method suitable for use in embedded real time applications such as inertial measurement units (IMUs). We propose

that this method could be adopted as the new standard method for converting quaternions into Euler angles, and we are now planning to contributing to several scientific libraries accordingly.

10.6. Extra: generalization for Davenport angles

This section, left out from the article, shows how to generalize the formula for the generalized Euler angles [16], also known as Davenport angles. The decomposition studied will be of the form:

$$q = \begin{bmatrix} c_3 \\ s_3 \boldsymbol{e}_k \end{bmatrix} \odot \begin{bmatrix} c_2 \\ s_2 \boldsymbol{e}_j \end{bmatrix} \odot \begin{bmatrix} c_1 \\ s_1 \boldsymbol{e}_i \end{bmatrix} \tag{10.41}$$

Where:

$$\begin{aligned} s_1 &\equiv \sin(\theta_1/2),\ c_1 \equiv \cos(\theta_1/2) \\ s_2 &\equiv \sin(\theta_2/2),\ c_2 \equiv \cos(\theta_2/2) \\ s_3 &\equiv \sin(\theta_3/2),\ c_3 \equiv \cos(\theta_3/2) \end{aligned} \tag{10.42}$$

As seen in [59], the following properties of Euler angles are also required when generalizing the Euler angles to Davenport angles:

$$\boldsymbol{e}_i \cdot \boldsymbol{e}_j = \boldsymbol{e}_j \cdot \boldsymbol{e}_k = 0 \tag{10.43}$$

This extra relation is required for Euler angles:

$$\boldsymbol{e}_i \cdot \boldsymbol{e}_k = \begin{cases} 0 & \text{, in case of the Proper Euler angles} \\ \pm 1 & \text{, in case of the Tait-Bryan angles} \end{cases} \tag{10.44}$$

For the Davenport angles, the value $\boldsymbol{e}_i \cdot \boldsymbol{e}_k$ can be any value. As a consequence of Eq. 10.43, and $\boldsymbol{e}_k$ can be transformed in $\boldsymbol{e}_i$ by a simple rotation of an angle the angle θ_λ around the axis $\boldsymbol{e}_j$, as seen in Fig. 10.1:

$$\boldsymbol{e}_k = \cos(\theta_\lambda)\,\boldsymbol{e}_i + \sin(\theta_\lambda)\,(\boldsymbol{e}_i \times \boldsymbol{e}_j) \tag{10.45}$$

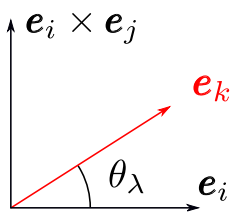


Figure 10.1.: Illustration of angle between Davenport axes.

From Eq. 10.45, we note that:

$$\begin{aligned}\cos\left(\theta_\lambda\right) &= \boldsymbol{e}_k \cdot \boldsymbol{e}_i \\ \sin\left(\theta_\lambda\right) &= \boldsymbol{e}_k \cdot \left(\boldsymbol{e}_i \times \boldsymbol{e}_j\right)\end{aligned} \tag{10.46}$$

And:

$$\theta_\lambda = \operatorname{atan2}\left(\boldsymbol{e}_k \cdot \left(\boldsymbol{e}_i \times \boldsymbol{e}_j\right), \boldsymbol{e}_k \cdot \boldsymbol{e}_i\right) \tag{10.47}$$

We note from Eq. 10.46 that the formula for $\sin\left(\theta_\lambda\right)$ is exactly the formula for the parity ε! Except that, in this case, it can be any value between -1 and 1. Defining $c_\lambda = \cos(\theta_\lambda/2)$, $s_\lambda = \sin(\theta_\lambda/2)$ and λ the quaternion that rotates $\boldsymbol{e}_k$ into $\boldsymbol{e}_i$, we have, from Fig 10.1:

$$\begin{aligned}\begin{bmatrix}0\\ \boldsymbol{e}_i\end{bmatrix} &= \lambda \odot \begin{bmatrix}0\\ \boldsymbol{e}_k\end{bmatrix} \odot \lambda^* \\ \begin{bmatrix}0\\ \boldsymbol{e}_k\end{bmatrix} &= \lambda^* \odot \begin{bmatrix}0\\ \boldsymbol{e}_i\end{bmatrix} \odot \lambda\end{aligned} \tag{10.48}$$

$$\lambda = \begin{bmatrix}\cos(\theta_\lambda/2)\\ \sin(\theta_\lambda/2)(-\boldsymbol{e}_i) \times \left(\boldsymbol{e}_i \times \boldsymbol{e}_j\right)\end{bmatrix} \tag{10.49}$$

From Eq. 10.43 we can note that:

$$\begin{aligned}(-\boldsymbol{e}_i) \times \left(\boldsymbol{e}_i \times \boldsymbol{e}_j\right) &= -\widehat{\boldsymbol{e}_i}^{\,2}\boldsymbol{e}_j \\ &= -\left(\boldsymbol{e}_i\boldsymbol{e}_i^T - \mathbb{I}\right)\boldsymbol{e}_j \\ &= -\left(\boldsymbol{e}_i\boldsymbol{e}_i^T - \mathbb{I}\right)\boldsymbol{e}_j \\ &= \boldsymbol{e}_j\end{aligned} \tag{10.50}$$

And we can then state that:

$$\lambda = \begin{bmatrix}\cos(\theta_\lambda/2)\\ \sin(\theta_\lambda/2)\boldsymbol{e}_j\end{bmatrix} \tag{10.51}$$

And we note that Eq. 10.28 and Eq. 10.51 are equivalent in the case where $\theta_\lambda = \pi/2$, as expected. Analyzing Eqs. 10.41, 10.48 and 10.51 we can state that:

$$\begin{aligned}q &= \begin{bmatrix}c_3\\ s_3\boldsymbol{e}_k\end{bmatrix} \odot \begin{bmatrix}c_2\\ s_2\boldsymbol{e}_j\end{bmatrix} \odot \begin{bmatrix}c_1\\ s_1\boldsymbol{e}_i\end{bmatrix} \\ &= \lambda^* \odot \begin{bmatrix}c_3\\ s_3\boldsymbol{e}_i\end{bmatrix} \odot \lambda \odot \begin{bmatrix}c_2\\ s_2\boldsymbol{e}_j\end{bmatrix} \odot \begin{bmatrix}c_1\\ s_1\boldsymbol{e}_i\end{bmatrix}\end{aligned} \tag{10.52}$$

And finally:

$$q' = \begin{bmatrix} c_3 \\ s_3 \boldsymbol{e}_i \end{bmatrix} \odot \begin{bmatrix} c_2' \\ s_2' \boldsymbol{e}_j \end{bmatrix} \odot \begin{bmatrix} c_1 \\ s_1 \boldsymbol{e}_i \end{bmatrix} \tag{10.53}$$

Where:

$$\begin{aligned} q' &\equiv \lambda \odot q \equiv \begin{bmatrix} q_\mathrm{r}' \\ \boldsymbol{q}' \end{bmatrix} \\ s_2' &\equiv \sin\left(\frac{\theta_2 + \theta_\lambda}{2}\right) \\ c_2' &\equiv \cos\left(\frac{\theta_2 + \theta_\lambda}{2}\right) \end{aligned} \tag{10.54}$$

And we suppose $\theta_2 + \lambda > 0$. And as seen in this chapter, we can solve Eq. 10.53 (in the general case without singularities) with:

$$\begin{aligned} \theta_2 + \lambda &= 2\,\mathrm{asin}\left(\sqrt{\frac{c^2+d^2}{n^2}}\right) = 2\,\mathrm{acos}\left(\sqrt{\frac{a^2+b^2}{n^2}}\right) = 2\,\mathrm{atan}\left(\sqrt{\frac{c^2+d^2}{a^2+b^2}}\right) \\ &= \mathrm{acos}\left(\sqrt{2\frac{a^2+b^2}{n^2} - 1}\right) \end{aligned} \tag{10.55}$$

And:

$$\begin{aligned} \frac{\theta_3 + \theta_1}{2} &= \mathrm{atan2}(b, a) \\ \frac{\theta_3 - \theta_1}{2} &= \mathrm{atan2}(d, c) \end{aligned} \tag{10.56}$$

Where:

$$\begin{aligned} a &= q_\mathrm{r}' \\ \begin{bmatrix} b \\ c \\ d \end{bmatrix} &= \begin{bmatrix} \boldsymbol{e}_i & \boldsymbol{e}_j & \boldsymbol{e}_i \times \boldsymbol{e}_j \end{bmatrix}^T \boldsymbol{q}' \\ n^2 &= a^2 + b^2 + c^2 + d^2 \end{aligned} \tag{10.57}$$

Note that using this general Davenport solution for the computation of Euler angles (specially Tait-Bryan angles) might lead to a set problem, since the angles are not unique. Appendix J explains this problem and shows a solution.

Conclusion

Origami and robot shape We made an extensive review of the existing origami structures of interest in the literature. In order to rapidly design and create the structures, a Pattern-drawing extension for the graphical software Inkscape was implemented, and different ways of assembling origami structures were tested. With the help of the ICube team, different possible shape changing structures were analyzed according to their possible use cases, in specific:

- *Magic ball*: rigidly-foldable origami, very flexible, useful if a big shape change is needed. We decided not to continue studying this particular structure since it would not be particularly useful for the project.
- *Flasher*: mostly studied by ICube, this origami structure can be easily deployed, changing its diameter. We also decided not to use this particular origami structure for the OrigaBot project.
- *Kresling tower*: this structure transforms a rotational movement into a linear displacement and vice-versa, all while being a bistable structure that, when well built, does not need a constant source of power in order to stay in one of its two possible states. This structure was chosen for the prototypes.

We decided to concentrate our efforts into developing a monorotor spinning drone that uses a Kresling tower as its main body. In order to create this monorotor drone, we decided to re-implement a swashplateless solution as described in [46, 44] in order to have a vector control of the rotor's torque and thrust force.

Origami Bendy Straw We noticed that the literature lacked a simple origami-based structure that bends. This lead us to the development of the *Origami Bendy Straw*, a multistable origami structure that, when modified with a Pop-Through Defect, has a stable bending position. This structure was implemented in the Pattern-drawing Inkscape extension, as a FOLD file generator, and a low-cost force-measuring bench was created in order to test the forces necessary to deploy a set of paper origami bendy straws.

Decomposition of orientation quaternion Rotary wing aircraft vehicles, be they manned or unmanned, usually have an even number of rotary wings, half of them spinning clockwise and half of them spinning counterclockwise. This is

a critical design decision that is important in order to neutralize the total torque produced to the body. A monorotor drone will normally have a non-zero torque in one constant direction, making it spin. In order to control the 2 controllable degrees-of-freedom of the rotation, while accepting that our robot will spin freely and uncontrollably around the z-axis, we analyzed alternative ways of representing the orientation. We have showed that, for some constant axis $\boldsymbol{e}$, any orientation quaternion can be decomposed into two independent components: A component that purely represents the rotation around the axis $\boldsymbol{e}$ (which, for us, is uncontrollable) and a vector $\boldsymbol{m}$ (the middle normal vector) that represents the other two controllable degrees of freedom. The vector $\boldsymbol{m}$ is simply the middle vector between $\boldsymbol{e}$ before and after the full rotation, as represented in the inertial frame. This is closely related to the fact that a quaternion's elements are proportional to the sine and cosine of half of the rotation angles. Moreover, we analyzed the derivatives of both decomposed elements and showed some of their properties.

Nonlinear control Using the normal middle vector $\boldsymbol{m}$ as the rotation variable, we derived a fully nonlinear control law that stabilizes the system while ignoring its spin around the z-axis, proving it via a Lyapunov function. This control law gives the goal angular velocity as output. We then derived a complete orientation controller composed of the formerly discussed controller as an inner loop, and a PID controller as an outer loop. We have decided on this architecture because it is well-known and easy to implement on the PX4 flying controller (as a matter of fact, the external PID loop is already implemented). The output of this controller was then adapted to be used with the single rotor attached to a swashplateless mechanism.

Equations of Motion with Euler-Poincaré Equation We decided to make a complete analysis of the robot's movement without any linearization and with a minimal set of simplifying assumptions. We did this modeling using the Euler-Poincaré method. This method is an alternative form of the energy-based Euler-Lagrange method that can be used when the generalized coordinates of the system can be described as an element of a Lie group. In our case, we are studying the linear and rotational degrees of freedom of a flying drone, which can both be represented as elements of SE(3). This method is arguably more straightforward than the usual Euler-Lagrange, but is also less known by the scientific community.

Simulation and findings We implemented a Python class for the simulation of the whole system, integrating our Poincaré method-derived equations of motion, quaternion decomposition-based nonlinear control architecture, swashplateless mechanism model and parameters derived from a real prototype. We used the simulation to further indicate that the controller architecture can be used to stabilize

the robot's orientation. Moreover, the simulation was used to analyze the effect of a bad orientation estimation on the robot's stabilization. The rapid spinning of the robot, together with the limits of the IMU integrated circuits, can lead to a rapid degrading of the robot's spin (or yaw) estimation. We did an extensive simulation study of the effect of spin estimation error on the trajectory of the reduced attitude middle vector $\boldsymbol{m}$. This study was used to derive an observer that could theoretically use the trajectory of $\boldsymbol{m}$ (equivalent to roll and pitch, which are usually more precise) to estimate an error on the estimation of the spin (or yaw, which is almost always less precise).

Quaternion to Euler angles conversion As an extra to the monorotor project, I was able to derive a new formula for the direct conversion of an orientation quaternion to three Euler angles in any of the possible 12 sequences. This formula has the convenience of being much simpler and more concise to implement than the other available methods, and has the slight advantage of being more than 30 times faster than the very well-known conversion method described by [58]. Moreover, it is a direct mathematical formula, making it possible to use it in theoretical development. Since the first publication of this formula and algorithm in November 2022, it has already been merged into both SciPy and SymPy, among other smaller open source projects.

Future The next steps for the OrigaBot project are, firstly, experimental tests in order to confirm all the theoretical development made during my thesis project. Not only the dynamical model and the control architecture need to be proved, but also the observer that estimates spin error must be implemented and tested. Moreover, the OrigaBot project's initial ambitions were that of creating a fully multi-modal vehicle that can either fly, for full mobility, or roll on the ground, for battery-saving purposes. It would be interesting to dedicate the next steps, after the ending of the monorotor project, to study the problem of designing and constructing such a multi-modal robot.

Bibliography

[1] V. Baiocchi, D. Dominici, and M. Mormile. "UAV application in post seismic environment". In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XL-1/W2 (2013), pp. 21–25. DOI: 10.5194/isprsarchives-XL-1-W2-21-2013 (cit. on p. 68).

[2] Nakul P. Bende, Tian Yu, Nicholas A. Corbin, et al. "Overcurvature induced multistability of linked conical frusta: How a 'bendy straw' holds its shape". In: *Soft Matter* 14.42 (2018), pp. 8636–8642. ISSN: 17446848. DOI: 10.1039/c8sm01355a. arXiv: 1808.02450 (cit. on pp. 51, 52, 54, 60).

[3] Evandro Bernardes. *Quaternion to Euler Scipy implementation*. URL: https://github.com/evbernardes/quaternion_to_euler (visited on 11/09/2022) (cit. on pp. 170, 172).

[4] Evandro Bernardes, Fréféric Boyer, and Stéphane Viollet. "Modelling, control and simulation of a single rotor UAV with swashplateless torque modulation". In: *Aerospace Science and Technology* (2023, submitted) (cit. on pp. 69, 96, 110, 117, 119, 132, 140).

[5] Evandro Bernardes and Stéphane Viollet. "Design of an Origami Bendy Straw for Robotic Multistable Structures". In: *Journal of Mechanical Design* 144.3 (Sept. 2021). 033301. ISSN: 1050-0472. DOI: 10.1115/1.4052222. eprint: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/144/3/033301/6763497/md_144_3_033301.pdf. URL: https://doi.org/10.1115/1.4052222 (cit. on p. 50).

[6] Evandro Bernardes and Stéphane Viollet. "Quaternion to Euler angles conversion: a direct, general and computationally efficient method". In: *PLOS ONE* (Nov. 2022) (cit. on p. 161).

[7] John Berre, François Geiskopf, Lennart Rubbert, et al. "Toward the Design of Kresling Tower Origami As a Compliant Building Block". In: *Journal of Mechanisms and Robotics* 14 (Dec. 2021), pp. 1–12. DOI: 10.1115/1.4053378 (cit. on p. 44).

[8] John Berre, François Geiskopf, Lennart Rubbert, et al. "Towards a Synthesis Method of Kresling Tower Used as a Compliant Building Block". In: Aug. 2021. DOI: 10.1115/DETC2021-68904 (cit. on p. 44).

[9] Frédéric Boyer and Mathieu Porez. "Multibody system dynamics for bio-inspired locomotion: from geometric structures to computational aspects". en. In: *Bioinspiration & Biomimetics* 10.2 (Mar. 2015). Publisher: IOP Publishing, p. 025007. ISSN: 1748-3190. DOI: 10.1088/1748-3190/10/2/025007. (Visited on 03/27/2023) (cit. on p. 111).

[10] Dario Brescianini, Markus Hehn, and Raffaello D'Andrea. "Nonlinear Quadrocopter Attitude Control". In: *ETH Zürich Research Collection* (2013). DOI: 10.3929/ethz-a-009970340 (cit. on p. 132).

[11] David Carabis. "The Design of a Maneuverable Rolling Robot". In: (2013) (cit. on p. 44).

[12] Jaimie Carlson, Jason Friedman, Christopher Kim, et al. "REBOund: Untethered Origami Jumping Robot with Controllable Jump Height". In: *Proceedings - IEEE International Conference on Robotics and Automation* (2020), pp. 10089–10095. ISSN: 10504729. DOI: 10.1109/ICRA40945.2020.9196534 (cit. on p. 51).

[13] Nan Chen, Fanze Kong, Wei Xu, et al. "A self-rotating, single-actuated UAV with extended sensor field of view for autonomous navigation". In: *Science Robotics* 8.76 (Mar. 2023). DOI: 10.1126/scirobotics.ade4538 (cit. on p. 110).

[14] Venkata Siva C. Chillara, Leon M. Headings, and Marcelo J. Dapino. "Self-folding laminated composites for smart origami structures". In: *ASME 2015 Conference on Smart Materials, Adaptive Structures and Intelligent Systems, SMASIS 2015* 2.May 2018 (2015). DOI: 10.1115/SMASIS2015-8968 (cit. on p. 51).

[15] Matteo Cianchetti, Tommaso Ranzani, Giada Gerboni, et al. "STIFF-FLOP Surgical Manipulator mechanical design and". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2013), pp. 3576–3581 (cit. on p. 51).

[16] Paul B. Davenport. "Rotations about nonorthogonal axes." In: *AIAA Journal* 11.6 (1973), pp. 853–857. DOI: 10.2514/3.6842. eprint: https://doi.org/10.2514/3.6842. URL: https://doi.org/10.2514/3.6842 (cit. on p. 173).

[17] Erik D. Demaine and Joseph O'Rourke. *Geometric folding algorithms: Linkages, origami, polyhedra*. Vol. 9780521857. 2007, pp. 1–472. ISBN: 9780511735172. DOI: 10.1017/CBO9780511735172 (cit. on p. 37).

[18] Evelin Villegas Diaz, Robert J. Wood, John J. Stampfli, et al. "A Vacuum-driven Origami 'Magic-ball' Soft Gripper". In: *Prof. Rus* (2019). ISBN: 9781538660263. URL: https://dspace.mit.edu/handle/1721.1/120930 (cit. on pp. 25, 38, 39, 51).

[19] Mario Doria and Lionel Birglen. "Design of an underactuated compliant gripper for surgery using nitinol". In: *Journal of Medical Devices, Transactions of the ASME* 3.1 (2009), pp. 1–7. ISSN: 19326181. DOI: 10.1115/1.3089249 (cit. on p. 25).

[20] Bryce J. Edmondson, Landen A. Bowen, Clayton L. Grames, et al. "Oriceps: Origami-Inspired Forceps". en. In: *Volume 1: Development and Characterization of Multifunctional Materials; Modeling, Simulation and Control of Adaptive Systems; Integrated System Design and Implementation*. American Society of Mechanical Engineers, Sept. 2013, V001T01A027. ISBN: 978-0-7918-5603-1. DOI: 10.1115/SMASIS2013-3299 (cit. on p. 26).

[21] E. T. Filipov, G. H. Paulino, and T. Tachi. "Origami tubes with reconfigurable polygonal cross-sections". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 472.2185 (2016). ISSN: 14712946. DOI: 10.1098/rspa.2015.0607 (cit. on p. 51).

[22] Evgueni T. Filipov, Tomohiro Tachi, Glaucio H. Paulino, et al. "Origami tubes assembled into stiff, yet reconfigurable structures and metamaterials". In: *Proceedings of the National Academy of Sciences of the United States of America* 112.40 (2015), pp. 12321–12326. ISSN: 10916490. DOI: 10.1073/pnas.1509465112 (cit. on p. 51).

[23] Amanda Ghassaei. *Origami Simulator*. URL: https://origamisimulator.org/ (visited on 09/14/2022) (cit. on p. 33).

[24] Amanda Ghassaei, Erik D Demaine, and Neil Gershenfeld. "Fast, Interactive Origami Simulation Using GPU Computation". In: *Origami$7̂$: Proceedings of the 7th International Meeting on Origami in Science, Mathematics and Education (OSME 2018)* (2018) (cit. on pp. 33, 54).

[25] Mojtaba Hedayatpour, Mehran Mehrandezh, and Farrokh Janabi-Sharifi. "Optimal-power Configurations for Hover Solutions in Mono-spinners". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 1344–1349. DOI: 10.1109/IROS45743.2020.9341648 (cit. on p. 68).

[26] Dm Henderson. "Euler Angles, Quaternions, and Transformation Matrices". In: *NASA JSC Report* (1977), p. 42 (cit. on p. 163).

[27] Larry L. Howell, Spencer P. Magleby, and Brian M. Olsen. *Handbook of Compliant Mecahnisms*. 2013. 2019, p. 326. ISBN: 9781119953456 (cit. on p. 25).

[28] Meredith Johnson, Yue Chen, Sierra Hovet, et al. "Fabricating biomedical origami: a state-of-the-art review". In: *International Journal of Computer Assisted Radiology and Surgery* 12.11 (2017), pp. 2023–2032. ISSN: 18616429. DOI: 10.1007/s11548-017-1545-1 (cit. on p. 26).

[29] N. Kidambi and K. W. Wang. "Dynamics of Kresling origami deployment". In: *Physical Review E* 101.6 (2020), p. 63003. ISSN: 2470-0045. DOI: 10.1103/physreve.101.063003. URL: https://doi.org/10.1103/PhysRevE.101.063003 (cit. on p. 40).

[30] Sukjun Kim, Dae-Young Lee, Gwang-Pil Jung, et al. "An origami-inspired, self-locking robotic arm that can be folded flat". In: *Science Robotics* 3 (Mar. 2018), eaar2915. DOI: 10.1126/scirobotics.aar2915 (cit. on p. 66).

[31] Je-sung Koh, Sa-reum Kim, and Kyu-jin Cho. "Self-Folding Origami Using Torsion Shape Memory Alloy Wire Actuators". In: c (2014), V05BT08A043. DOI: 10.1115/DETC2014-34822 (cit. on p. 66).

[32] Jack B. Kuipers. *Quaternions and rotation sequences : a primer with applications to orbits, aerospace, and virtual reality*. Princeton, NJ: Princeton Univ. Press, 1999. ISBN: 0691058725 9780691058726 (cit. on p. 116).

[33] Robert J. Lang. *Twists, Tilings and Tesselations - Mathematicals Methods for Geometrical Origami*. Publication Title: CRC Press. 2018. ISBN: 978-1-138-56306-3 (cit. on p. 30).

[34] Robert J. Lang, Kyler A. Tolman, Erica B. Crampton, et al. "A Review of Thickness-Accommodation Techniques in Origami-Inspired Engineering". In: *Applied Mechanics Reviews* 70.1 (2018). ISSN: 00036900. DOI: 10.1115/1.4039314 (cit. on p. 37).

[35] Dae-Young Lee, Jae-Kyeong Kim, Chang-Young Sohn, et al. "High–load capacity origami transformable wheel". In: *Science Robotics* 6 (2021). DOI: 10.1126/scirobotics.abe0201 (cit. on p. 50).

[36] Dae-Young Lee, Sa-Reum Kim, Ji-Suk Kim, et al. "Origami Wheel Transformer: A Variable-Diameter Wheel Drive Robot Using an Origami Structure". In: *Soft Robotics* 4.2 (2017), pp. 163–180. ISSN: 2169-5172. DOI: 10.1089/soro.2016.0038 (cit. on pp. 26, 38, 50).

[37] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 6235–6240. DOI: 10.1109/ICRA.2015.7140074 (cit. on p. 132).

[38] Mark W. Mueller and Raffaello D'Andrea. "Relaxed hover solutions for multicopters: Application to algorithmic redundancy and novel vehicles". In: *International Journal of Robotics Research* 35.8 (2016), pp. 873–889. ISSN: 17413176. DOI: 10.1177/0278364915596233 (cit. on p. 68).

[39] Mark W. Mueller and Raffaello D'Andrea. "Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 45–52. DOI: 10.1109/ICRA.2014.6906588 (cit. on p. 68).

[40] Richard M. Murray, S. Shankar Sastry, and Li Zexiang. *A Mathematical Introduction to Robotic Manipulation*. 1st. USA: CRC Press, Inc., 1994. ISBN: 0849379814 (cit. on pp. 111, 190).

[41] Ake Norberg. "Autorotation, self-stability, and structure of single-winged fruits and seeds (samaras) with comparative remarks on animal flight". In: *Biological Reviews* 48.4 (1973), pp. 561–596. DOI: https://doi.org/10.1111/j.1469-185X.1973.tb01569.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1469-185X.1973.tb01569.x (cit. on p. 68).

[42] Alexander Pagano, Brandon Leung, Brian Chien, et al. "Multi-Stable Origami Structure for Crawling Locomotion". In: *Proceedings of the ASME 2016 Conference on Smart Materials, Adaptive Structures and Intelligent Systems (SMASIS2016)* September 2016 (2016), V002T06A005. DOI: 10.1115/smasis2016-9071 (cit. on p. 40).

[43] Alexander Pagano, Tongxi Yan, Brian Chien, et al. "A crawling robot driven by multi-stable origami". In: *Smart Materials and Structures* 26.9 (2017). ISSN: 1361665X. DOI: 10.1088/1361-665X/aa721e (cit. on pp. 39, 40, 209).

[44] James Paulos, Bennet Caraher, and Mark Yim. "Emulating a Fully Actuated Aerial Vehicle Using Two Actuators". In: *Proceedings - IEEE International Conference on Robotics and Automation* (2018), pp. 7011–7016. ISSN: 10504729. DOI: 10.1109/ICRA.2018.8462975 (cit. on pp. 69, 77, 78, 81, 176).

[45] James Paulos and Mark Yim. "An underactuated propeller for attitude control in micro air vehicles". In: *IEEE International Conference on Intelligent Robots and Systems* (2013), pp. 1374–1379. ISSN: 21530858. DOI: 10.1109/IROS.2013.6696528 (cit. on pp. 68, 77–79, 98).

[46] James Paulos and Mark Yim. "Cyclic blade pitch control for small UAV without a swashplate". In: *AIAA Atmospheric Flight Mechanics Conference, 2017* (2017). DOI: 10.2514/6.2017-1186 (cit. on pp. 68, 77, 79, 98, 176).

[47] Matthew Piccoli and Mark Yim. "Passive stability of a single actuator micro aerial vehicle". In: *Proceedings - IEEE International Conference on Robotics and Automation* (2014), pp. 5510–5515. ISSN: 10504729. DOI: 10.1109/ICRA.2014.6907669 (cit. on p. 68).

[48] Matthew Piccoli and Mark Yim. "Piccolissimo: The smallest micro aerial vehicle". In: *Proceedings - IEEE International Conference on Robotics and Automation* (2017), pp. 3328–3333. ISSN: 10504729. DOI: 10.1109/ICRA.2017.7989378 (cit. on p. 68).

[49] Henri Poincaré. "Sur une forme nouvelle des équations de la mécanique". In: *Comptes rendus de l'Académie des Sciences de Paris* 132 (1901), pp. 369–371 (cit. on p. 69).

[50] Youming Qin, Nan Chen, Yixi Cai, et al. "Gemini II: Design, Modeling, and Control of a Compact Yet Efficient Servoless Bi-copter". In: *IEEE/ASME Transactions on Mechatronics* (2022), pp. 1–12. DOI: 10.1109/TMECH.2022.3153587 (cit. on pp. 69, 77).

[51] Youming Qin, Wei Xu, Adrian Lee, et al. "Gemini: A Compact Yet Efficient Bi-Copter UAV for Indoor Applications". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3213–3220. ISSN: 23773766. DOI: 10.1109/LRA.2020.2974718 (cit. on pp. 68, 69, 77).

[52] Valentin Riviere, Augustin Manecy, and Stéphane Viollet. "Agile Robotic Fliers: A Morphing-Based Approach". In: *Soft Robotics* 5.5 (2018), pp. 541–553. ISSN: 21695180. DOI: 10.1089/soro.2017.0120 (cit. on p. 68).

[53] Daniela Rus and Cynthia Sung. "Spotlight on origami robots". In: *Science Robotics* 3.15 (2018), pp. 2–4. ISSN: 24709476. DOI: 10.1126/scirobotics.aat0938 (cit. on p. 25).

[54] Soheil Sarabandi, Arya Shabani, Josep M. Porta, et al. "On Closed-Form Formulas for the 3-D Nearest Rotation Matrix Problem". In: *IEEE Transactions on Robotics* 36.4 (2020), pp. 1333–1339. DOI: 10.1109/TRO.2020.2973072 (cit. on p. 162).

[55] Pooya Sareh. "The least symmetric crystallographic derivative of the developable double corrugation surface: Computational design using underlying conic and cubic curves". In: *Materials & Design* 183 (2019), p. 108128. ISSN: 0264-1275. DOI: https://doi.org/10.1016/j.matdes.2019.108128. URL: https://www.sciencedirect.com/science/article/pii/S0264127519305660 (cit. on p. 50).

[56] Pooya Sareh and Yao Chen. "Intrinsic non-flat-foldability of two-tile DDC surfaces composed of glide-reflected irregular quadrilaterals". In: *International Journal of Mechanical Sciences* 185 (2020), p. 105881. ISSN: 0020-7403. DOI: https://doi.org/10.1016/j.ijmecsci.2020.105881. URL: https://www.sciencedirect.com/science/article/pii/S0020740319348283 (cit. on p. 50).

[57] Pooya Sareh, Pisak Chermprayong, Marc Emmanuelli, et al. "Rotorigami: A rotary origami protective system for robotic rotorcraft". In: *Science Robotics* 3.22 (2018), eaah5228. DOI: 10.1126/scirobotics.aah5228 (cit. on p. 51).

[58] Malcolm Shuster and Landis Markley. "General Formula for Extracting the Euler Angles". In: *Journal of Guidance Control and Dynamics* 29 (Jan. 2006), pp. 215–221. DOI: 10.2514/1.16622 (cit. on pp. 161, 163, 170, 172, 178).

[59] Malcolm Shuster and Landis Markley. "Generalization of the Euler Angles". In: *Journal of the Astronautical Sciences* 51 (Apr. 2003), pp. 123–132. DOI: 10.1007/BF03546304 (cit. on p. 173).

[60] Malcolm D. Shuster. "Survey of attitude representations". In: *Journal of the Astronautical Sciences* 41.4 (1993), pp. 439–517 (cit. on p. 161).

[61] Jesse L. Silverberg, Arthur A. Evans, Lauren McLeod, et al. "Using origami design principles to fold reprogrammable mechanical metamaterials". In: *Science* 345.6197 (2014), pp. 647–650. ISSN: 10959203. DOI: 10.1126/science.1252876 (cit. on p. 60).

[62] Origami Simulator. *https://origamisimulator.org/. Accessed: 2021-01-12.* URL: https://origamisimulator.org/ (visited on 01/12/2021) (cit. on p. 54).

[63] Hannes Sommer, Igor Gilitschenski, Michael Bloesch, et al. "Why and how to avoid the flipped quaternion multiplication". In: *Aerospace* 5.3 (2018), pp. 1–15. ISSN: 22264310. DOI: 10.3390/aerospace5030072. arXiv: 1801.07478 (cit. on p. 116).

[64] John Stuelpnagel. "On the Parametrization of the Three-Dimensional Rotation Group". In: *Siam Review* 6.4 (1964), pp. 422–430 (cit. on p. 162).

[65] Tomohiro Tachi. "Rigid-Foldable Thick Origami". In: *Origami 5* (2011), pp. 253–263. DOI: 10.1201/b10971-24 (cit. on p. 37).

[66] Chris E. Thorne and Mark Yim. "Design and Analysis of a Gyroscopically Controlled Micro Air Vehicle". In: *Journal of Intelligent & Robotic Systems* 65.1 (Jan. 2012), pp. 417–435. ISSN: 1573-0409. DOI: 10.1007/s10846-011-9644-7 (cit. on p. 68).

[67] Physlets Tracker. *https://physlets.org/tracker/. Accessed: 2021-02-08.* URL: https://physlets.org/tracker/ (visited on 02/08/2021) (cit. on p. 62).

[68] Firdaus E. Udwadia and Aaron D. Schutte. "An alternative derivation of the quaternion equations of motion for rigid-body rotational dynamics". In: *Journal of Applied Mechanics, Transactions ASME* 77.4 (2010), pp. 1–4. ISSN: 00218936. DOI: 10.1115/1.4000917 (cit. on p. 111).

[69] Firdaus E. Udwadia and Aaron D. Schutte. "Equations of motion for general constrained systems in lagrangian mechanics". In: *Acta Mechanica* 213.1-2 (2010), pp. 111–129. ISSN: 00015970. DOI: 10.1007/s00707-009-0272-2 (cit. on p. 111).

[70] Roberto G. Valenti, Ivan Dryanovski, and Jizhong Xiao. "Keeping a good attitude: A quaternion-based orientation filter for IMUs and MARGs". In: *Sensors (Switzerland)* 15.8 (2015), pp. 19302–19330. ISSN: 14248220. DOI: 10.3390/s150819302 (cit. on p. 119).

[71] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2 (cit. on pp. 163, 172).

[72] Wikipedia contributors. *Conjugacy class — Wikipedia, The Free Encyclopedia*. [Online; accessed 11-November-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=Conjugacy_class&oldid=1119487271 (cit. on p. 246).

[73] Shane Kyi Hla Win, Luke Soe Thura Win, Danial Sufiyan, et al. "An Agile Samara-Inspired Single-Actuator Aerial Robot Capable of Autorotation and Diving". In: *IEEE Transactions on Robotics* 38.2 (2022), pp. 1033–1046. DOI: 10.1109/TRO.2021.3091275 (cit. on p. 68).

[74] Shane Kyi Hla Win, Luke Soe Thura Win, Danial Sufiyan, et al. "Design and control of the first foldable single-actuator rotary wing micro aerial vehicle". In: *Bioinspiration & Biomimetics* 16.6 (Nov. 2021), p. 066019. DOI: 10.1088/1748-3190/ac253a (cit. on p. 68).

[75] Tomi Ylikorpi, Pekka Forsman, Aarne Halme, et al. "Unified representation of decoupled dynamic models for pendulum-driven ball-shaped robots". In: *Proceedings - 28th European Conference on Modelling and Simulation, ECMS 2014* June (2014), pp. 411–420. DOI: 10.7148/2014-0411 (cit. on p. 46).

[76] Wenjie Yu and Zhenkuan Pan. "Dynamical equations of multibody systems on Lie groups". In: *Advances in Mechanical Engineering* 7.3 (2015). tex.eprint: https://doi.org/10.1177/1687814015575959, p. 1687814015575959. DOI: 10.1177/1687814015575959. URL: https://doi.org/10.1177/1687814015575959 (cit. on p. 69).

[77] Qian Zhang. "Bistable behaviour of a deployable cylinder with Kresling pattern". In: *7th International Meeting on Origami in Science, Mathematics and Education (7OSME)* (2018) (cit. on pp. 40, 105).

[78] Weixuan Zhang, Mark W. Mueller, and Raffaello D'Andrea. "A controllable flying vehicle with a single moving part". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 3275–3281. DOI: 10.1109/ICRA.2016.7487499 (cit. on p. 68).

[79] Weixuan Zhang, Mark W. Mueller, and Raffaello D'Andrea. "Design, modeling and control of a flying vehicle with a single moving part that can be positioned anywhere in space". In: *Mechatronics* 61.October 2018 (2019), pp. 117–130. ISSN: 09574158. DOI: 10.1016/j.mechatronics.2019.06.004 (cit. on pp. 68, 225).

[80] Shannon A. Zirbel, Spencer P. Magleby, Larry L. Howell, et al. "Accommodating Thickness in Origami-Based Deployable Arrays". In: *Volume 6B: 37th Mechanisms and Robotics Conference* 135.November 2013 (2013). ISBN: 978-0-7918-5594-2, V06BT07A027. DOI: 10.1115/DETC2013-12348. URL: http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?doi=10.1115/DETC2013-12348 (cit. on p. 37).

Part III.

Appendices

A. Mathematical Definitions

A.1. Useful matrices, vectors, and operations

1. $\boldsymbol{e}_x$, $\boldsymbol{e}_y$ and $\boldsymbol{e}_z$ are the unit base vectors in the x, y and z axes respectively:

$$\boldsymbol{e}_x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \ \boldsymbol{e}_y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \ \boldsymbol{e}_z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{A.1}$$

2. For $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$, $\widehat{\boldsymbol{a}} = (\boldsymbol{a})\widehat{\ }$ is a 3×3 skew-symmetric matrix such that $\boldsymbol{a} \times \boldsymbol{b} = \widehat{\boldsymbol{a}}\,\boldsymbol{b}$, given by:

$$\widehat{\boldsymbol{a}} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{A.2}$$

3. For any rotation matrix R, the hat operator $(\cdot)\widehat{\ }$ has the following property[1]:

$$(R\boldsymbol{a})\widehat{\ } = R\widehat{\boldsymbol{a}}R^T \tag{A.3}$$

4. $\widehat{\boldsymbol{e}}_x$, $\widehat{\boldsymbol{e}}_y$ and $\widehat{\boldsymbol{e}}_z$ are, according to Eq. A.2:

$$\widehat{\boldsymbol{e}}_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \ \widehat{\boldsymbol{e}}_y = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \ \widehat{\boldsymbol{e}}_z = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{A.4}$$

5. $(\cdot)\check{\ }$ is the inverse of $(\cdot)\widehat{\ }$ such that:

$$(\widehat{\boldsymbol{\omega}})\check{\ } = \boldsymbol{\omega} \tag{A.5}$$

6. The rotation matrix equivalent to a rotation of an angle θ around an axis $\boldsymbol{e}$ is denoted by $R_{\theta\boldsymbol{e}}$. Moreover, the rotation matrices around the canonical axes

[1] https://math.stackexchange.com/questions/4252273/if-a-times-is-a-matrix-such-that-a-times-b-a-4252360#4252360

are also denoted by:

$$R_{\theta \boldsymbol{e}_x} = R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$
$$R_{\theta \boldsymbol{e}_y} = R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$
$$R_{\theta \boldsymbol{e}_z} = R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.6}$$

7. $\mathbb{I}_n$ is the $n \times n$ identity matrix, and $\mathbb{I}$ is a shorthand notation for $\mathbb{I}_3$ (the 3×3 identity matrix).
8. $\mathbf{0}_n$ (note: the boldface type) is the $n \times 1$ zero vector. For example, $\mathbf{0}_3$ is the 3×1 zero vector: $\mathbf{0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$.
9. $\mathbf{0}_{n,m}$ is the $n \times m$ zero matrix (n lines and m columns).
10. $\mathbf{0}$ (without any subscripts) is used as a shorthand notation to either $\mathbf{0}_n$ or $\mathbf{0}_{n,m}$ when the dimensions are obvious from context (mostly used in place of $\mathbf{0}_3$ and $\mathbf{0}_{3,3}$).
11. $\tilde{(\cdot)}$ means the "quaternion version" of its value. When applied to a real number a:
$$\tilde{a} = \begin{bmatrix} a \\ \mathbf{0} \end{bmatrix}$$
And when applied to a vector $\boldsymbol{v} \in \mathbb{R}^3$:
$$\tilde{\boldsymbol{v}} = \begin{bmatrix} 0 \\ \boldsymbol{v} \end{bmatrix}$$
12. Defining R_P^B the rotation matrix from $\mathcal{F}_P$ to $\mathcal{F}_B$ and s_P^B the position of $\mathcal{F}_P$ in $\mathcal{F}_B$, the total transformation $g_P^B \in SE(3)$ from frame $\mathcal{F}_P$ to $\mathcal{F}_B$ is given by:
$$g_P^B = \begin{bmatrix} R_P^B & \boldsymbol{s}_P^B \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{A.7}$$
13. Superscripts on a vector (linear or angular) indicate the frame of reference,

for example:

$$\boldsymbol{v}^B \equiv \text{ vector } \boldsymbol{v} \text{ in frame } \mathcal{F}_B \tag{A.8}$$

14. Subscripts on a velocity vector (linear or angular) explicitly indicates the frames (or, sometimes, bodies):

$$\boldsymbol{\omega}_{P/B} \equiv \text{ angular velocity vector } \boldsymbol{\omega} \text{ of frame } \mathcal{F}_P \text{ w.r.t. } \mathcal{F}_B \tag{A.9}$$

15. A single subscript indicates the velocity is w.r.t. the inertial frame $\mathcal{F}_E$:

$$\boldsymbol{\omega}_P \equiv \boldsymbol{\omega}_{P/E} \equiv \text{ angular velocity of frame } \mathcal{F}_P \text{ w.r.t. } \mathcal{F}_E \tag{A.10}$$

16. Both superscripts from 13 and subscripts from 14 and 15 can be used together:

$$\boldsymbol{\omega}_P^P \equiv \boldsymbol{\omega}_{P/E}^P \equiv \text{angular velocity of frame } \mathcal{F}_P \text{ w.r.t. } \mathcal{F}_E, \text{ represented in } \mathcal{F}_P \tag{A.11}$$

17. For the inertial twist defined in 18, $\widehat{\eta}$ is a 4×4 matrix given by:

$$\widehat{\eta} = \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix}^{\widehat{}} = \begin{bmatrix} \widehat{\boldsymbol{\omega}} & \boldsymbol{v} \\ \mathbf{0}^T & 0 \end{bmatrix} \tag{A.12}$$

18. For $\boldsymbol{\omega}, \boldsymbol{v} \in \mathbb{R}^3$ representing, respectively, the angular and linear velocity vectors, we define the inertial twist:

$$\eta = \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} \tag{A.13}$$

19. Defining g_P^B the transformation from some frame $\mathcal{F}_P$ to some frame $\mathcal{F}_B$ (as seen in 12), to transform a twist η from one frame to another, by composition of velocities, we can state:

$$\eta_P^B = \mathrm{Ad}_P^B \, \eta_P^P + \eta_{P/B}^P \tag{A.14}$$

Where, as seen in [40], the Ad operator is given by:

$$\begin{aligned} \mathrm{Ad}_P^B &\equiv \begin{bmatrix} R_P^B & 0 \\ \widehat{\boldsymbol{s}_P^B} R_P^B & R_P^B \end{bmatrix} \\ \left(\mathrm{Ad}_P^B\right)^{-1} = \mathrm{Ad}_B^P &\equiv \begin{bmatrix} \left(R_P^B\right)^T & 0 \\ -\left(R_P^B\right)^T \widehat{\boldsymbol{s}_P^B} & \left(R_P^B\right)^T \end{bmatrix} \end{aligned} \tag{A.15}$$

A.2. Matrix calculus definitions

Since there are many conventions on how to define the derivatives of matrices and vectors in relation to matrices and vectors[2], we will give in this chapter the definitions (and some properties) used throughout this work. For a scalar $a \in \mathbb{R}$, column vectors $\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^n$ and matrix $A \in \mathbb{R}^{m \times n}$:

$$\begin{aligned}
\boldsymbol{x} &= \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T \\
\boldsymbol{v} &= \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix}^T \\
\boldsymbol{w} &= \begin{bmatrix} w_1 & w_2 & \dots & w_n \end{bmatrix}^T \\
A &= \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{bmatrix} = \begin{bmatrix} A_{(1)}^T \\ A_{(2)}^T \\ \vdots \\ A_{(m)}^T \end{bmatrix}
\end{aligned} \tag{A.16}$$

Where:

$$A_{(i)}^T = \begin{bmatrix} A_{i1} & A_{i2} & \dots & A_{in} \end{bmatrix} \tag{A.17}$$

A.2.1. Definitions

In this work, our definitions will be mostly compatible with the denominator layout convention[3]

1. $\forall A = A(a)$:

$$\frac{\partial \mathbf{A}}{\partial a} = \begin{bmatrix} \frac{\partial A_{11}}{\partial a} & \dots & \frac{\partial A_{1n}}{\partial a} \\ \vdots & \ddots & \vdots \\ \frac{\partial A_{m1}}{\partial a} & \dots & \frac{\partial A_{mn}}{\partial a} \end{bmatrix} \tag{A.18}$$

2. From Eq. A.18, we know that $\forall a = a(\boldsymbol{x})$:

$$\frac{\partial a}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial a}{\partial x_1} \\ \frac{\partial a}{\partial x_2} \\ \vdots \\ \frac{\partial a}{\partial x_n} \end{bmatrix} \tag{A.19}$$

[2] As can be seen, for example, in: `https://en.wikipedia.org/wiki/Matrix_calculus#Notation`

[3] Since, for the Partial Differential Equations presented in this work, I find it more practical to define the derivatives as column vectors.

3. $\forall \boldsymbol{v} = \boldsymbol{v}(a)$:

$$\frac{\partial \boldsymbol{v}}{\partial a} = \begin{bmatrix} \frac{\partial v_1}{\partial a} \\ \frac{\partial v_2}{\partial a} \\ \vdots \\ \frac{\partial v_n}{\partial a} \end{bmatrix} \tag{A.20}$$

And:

$$\frac{\partial \boldsymbol{v}}{\partial a}^T = \begin{bmatrix} \frac{\partial v_1}{\partial a} & \frac{\partial v_2}{\partial a} & \dots & \frac{\partial v_n}{\partial a} \end{bmatrix} \tag{A.21}$$

4. from Eq. A.19, we know that $\forall \boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x})$, $\frac{\partial v_i}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial v_i}{\partial x_1} & \frac{\partial v_i}{\partial x_2} & \dots & \frac{\partial v_i}{\partial x_n} \end{bmatrix}^T$, so:

$$\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial v_1}{\partial x_1} & \dots & \frac{\partial v_n}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial v_1}{\partial x_n} & \dots & \frac{\partial v_n}{\partial x_n} \end{bmatrix}$$

And finally:

$$\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial \boldsymbol{v}^T}{\partial x_1} \\ \vdots \\ \frac{\partial \boldsymbol{v}^T}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial v_1}{\partial \boldsymbol{x}} & \dots & \frac{\partial v_n}{\partial \boldsymbol{x}} \end{bmatrix} \tag{A.22}$$

Note that:

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}} = \mathbb{I}_n \tag{A.23}$$

Where $\mathbb{I}_n$ is the $n \times n$ identity matrix.

A.2.2. Useful properties

1. $\forall \boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x})$, $\forall \boldsymbol{w} = \boldsymbol{w}(\boldsymbol{x})$ and $d = \boldsymbol{w} \cdot \boldsymbol{v} = \boldsymbol{w}^T \boldsymbol{v}$:

$$\begin{aligned} d &= \begin{bmatrix} w_1 & w_2 & \dots & w_n \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \\ &= w_1 v_1 + \dots + w_n v_n \\ \frac{\partial d}{\partial x_1} &= \frac{\partial w_1}{\partial x_1} v_1 + \frac{\partial v_1}{\partial x_1} w_1 + \dots + \frac{\partial w_n}{\partial x_1} v_n + \frac{\partial v_n}{\partial x_1} w_n \end{aligned}$$

Rearranging:

$$\begin{aligned}
\frac{\partial d}{\partial x_1} &= \begin{bmatrix} \frac{\partial w_1}{\partial x_1} & \cdots & \frac{\partial w_n}{\partial x_1} \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} + \begin{bmatrix} \frac{\partial v_1}{\partial x_1} & \cdots & \frac{\partial v_n}{\partial x_1} \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} \\
&= \left(\frac{\partial \boldsymbol{w}}{\partial x_1} \right)^T \boldsymbol{v} + \left(\frac{\partial \boldsymbol{v}}{\partial x_1} \right)^T \boldsymbol{w} \\
\frac{\partial d}{\partial \boldsymbol{x}} &= \begin{bmatrix} \frac{\partial d}{\partial x_1} \\ \frac{\partial d}{\partial x_n} \end{bmatrix} \\
&= \begin{bmatrix} \left(\frac{\partial \boldsymbol{w}}{\partial x_1} \right)^T \\ \vdots \\ \left(\frac{\partial \boldsymbol{w}}{\partial x_n} \right)^T \end{bmatrix} \boldsymbol{v} + \begin{bmatrix} \left(\frac{\partial \boldsymbol{v}}{\partial x_1} \right)^T \\ \vdots \\ \left(\frac{\partial \boldsymbol{v}}{\partial x_n} \right)^T \end{bmatrix} \boldsymbol{w}
\end{aligned}$$

And finally:

$$\frac{\partial \boldsymbol{w} \cdot \boldsymbol{v}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{w}}{\partial \boldsymbol{x}} \boldsymbol{v} + \frac{\partial \mathbf{v}}{\partial \boldsymbol{x}} \boldsymbol{w} \tag{A.24}$$

2. If A is a constant matrix and $\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x})$, and defining $\boldsymbol{w} = A\boldsymbol{v}$:

$$\begin{aligned}
\boldsymbol{w} &= \begin{bmatrix} A_{(1)}^T \\ A_{(2)}^T \\ \vdots \\ A_{(m)}^T \end{bmatrix} \boldsymbol{v} \\
&= \begin{bmatrix} A_{(1)}^T \boldsymbol{v} \\ A_{(2)}^T \boldsymbol{v} \\ \vdots \\ A_{(m)}^T \boldsymbol{v} \end{bmatrix}
\end{aligned}$$

For every i, using Eq. A.24, and remembering that A is constant:

$$\begin{aligned}
\frac{\partial w_i}{\partial x_j} &= \frac{\partial \boldsymbol{v}}{\partial x_j}^T A_{(i)} = A_{(i)}^T \frac{\partial \boldsymbol{v}}{\partial x_j} \\
\frac{\partial \boldsymbol{w}}{\partial x_j} &= \begin{bmatrix} A_{(1)}^T \frac{\partial \boldsymbol{v}}{\partial x_j} \\ \vdots \\ A_{(m)}^T \frac{\partial \boldsymbol{v}}{\partial x_j} \end{bmatrix} = A \frac{\partial \boldsymbol{v}}{\partial x_j}
\end{aligned}$$

And from Eq. A.22:

$$\begin{aligned}\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{x}} &= \begin{bmatrix}\frac{\partial \boldsymbol{w}}{\partial x_1} & \dots & \frac{\partial \boldsymbol{w}}{\partial x_n}\end{bmatrix}^T \\ &= \begin{bmatrix}A\frac{\partial \boldsymbol{v}}{\partial x_1} & \dots & A\frac{\partial \boldsymbol{v}}{\partial x_n}\end{bmatrix}^T \\ &= \begin{bmatrix}\frac{\partial \boldsymbol{v}}{\partial x_1} & \dots & \frac{\partial \boldsymbol{v}}{\partial x_n}\end{bmatrix}^T A^T\end{aligned}$$

And finally:

$$\frac{\partial A\boldsymbol{v}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}} A^T \tag{A.25}$$

3. If A is constant and square ($m = n$), $\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x})$ and defining $e = \boldsymbol{v}^T A\boldsymbol{v}$, and using Eq. A.24:

$$\begin{aligned}\frac{\partial e}{\partial \boldsymbol{x}} &= \frac{\partial \boldsymbol{v} \cdot (A\boldsymbol{v})}{\partial \boldsymbol{x}} \\ &= \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}} A\boldsymbol{v} + \frac{\partial A\boldsymbol{v}}{\partial \boldsymbol{x}} \boldsymbol{v}\end{aligned}$$

Using Eq. A.25:

$$\frac{\partial e}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}} A\boldsymbol{v} + \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}} A^T \boldsymbol{v}$$

And finally:

$$\frac{\partial \boldsymbol{v}^T A\boldsymbol{v}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}}^T (A + A^T)\boldsymbol{v} \tag{A.26}$$

And from Eqs. A.23 and A.26:

$$\frac{\partial \boldsymbol{x}^T A\boldsymbol{x}}{\partial \boldsymbol{x}} = (A + A^T)\boldsymbol{v} \tag{A.27}$$

A.3. Quaternion algebra summary

The orientation of the robot can (and **will**, in this work) be expressed as a unit quaternion. Since the definitions concerning quaternion algebra are not perfectly consistent in the literature, we will show in this section some important notation and definitions used in this work.

A quaternion q is a hypercomplex number composed of four components:

$$\mathrm{q} = q_\mathrm{r} + q_\mathrm{x}\boldsymbol{i} + q_\mathrm{y}\boldsymbol{j} + q_\mathrm{z}\boldsymbol{k} \tag{A.28}$$

Where $q_{\mathrm{r}}, q_{\mathrm{x}}, q_{\mathrm{y}}, q_{\mathrm{z}} \in \mathbb{R}$. And all the properties of quaternions can be derived using the properties:

- $\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 \equiv -1$.
- $\boldsymbol{ij} = -\boldsymbol{ji} = \boldsymbol{k}$.
- $\boldsymbol{jk} = -\boldsymbol{kj} = \boldsymbol{i}$.
- $\boldsymbol{ki} = -\boldsymbol{ik} = \boldsymbol{j}$.

Using the properties above, the product of two quaternions q and p can be given by the Hamilton product:

$$\begin{aligned} \mathrm{q\ p} = {} & (p_r q_{\mathrm{r}} - p_x q_{\mathrm{x}} - p_y q_{\mathrm{y}} - p_z q_{\mathrm{z}}) + (p_r q_{\mathrm{x}} + p_x q_{\mathrm{r}} - p_y q_{\mathrm{z}} + p_z q_{\mathrm{y}})\,\boldsymbol{i} \\ & + (p_r q_{\mathrm{y}} + p_x q_{\mathrm{z}} + p_y q_{\mathrm{r}} - p_z q_{\mathrm{x}})\,\boldsymbol{j} + (p_r q_{\mathrm{z}} - p_x q_{\mathrm{y}} + p_y q_{\mathrm{x}} + p_z q_{\mathrm{r}})\,\boldsymbol{k} \end{aligned} \tag{A.29}$$

For simplicity, quaternions will be written in this work as 4×1 vectors:

$$q = \begin{bmatrix} q_{\mathrm{r}} \\ q_{\mathrm{x}} \\ q_{\mathrm{y}} \\ q_{\mathrm{z}} \end{bmatrix} \tag{A.30}$$

And the Hamilton product between two quaternions in 4-vector form will be written as[4]:

$$q \odot p = \begin{bmatrix} q_{\mathrm{r}} \\ q_{\mathrm{x}} \\ q_{\mathrm{y}} \\ q_{\mathrm{z}} \end{bmatrix} \odot \begin{bmatrix} p_r \\ p_x \\ p_y \\ p_z \end{bmatrix} \tag{A.31}$$

Defining q_{r} as the real part and $\boldsymbol{q} = \begin{bmatrix} q_{\mathrm{x}} & q_{\mathrm{y}} & q_{\mathrm{z}} \end{bmatrix}^T$ the imaginary/vector part of q, we can rewrite q as:

$$q = \begin{bmatrix} q_{\mathrm{r}} \\ \boldsymbol{q} \end{bmatrix} \tag{A.32}$$

And the Hamilton product between two quaternions in 4-vector form can be denoted by:

$$q \odot p = \begin{bmatrix} q_{\mathrm{r}} \\ \boldsymbol{q} \end{bmatrix} \odot \begin{bmatrix} p_{\mathrm{r}} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} q_{\mathrm{r}} p_{\mathrm{r}} - \boldsymbol{q} \cdot \boldsymbol{p} \\ q_{\mathrm{r}} \boldsymbol{p} + p_{\mathrm{r}} \boldsymbol{q} + \boldsymbol{q} \times \boldsymbol{p} \end{bmatrix} \tag{A.33}$$

Defining the conjugate $q^* = \begin{bmatrix} q_{\mathrm{r}} \\ -\boldsymbol{q} \end{bmatrix}$ and the absolute value as $|q| = \sqrt{q_{\mathrm{r}}^2 + q_{\mathrm{x}}^2 + q_{\mathrm{y}}^2 + q_{\mathrm{z}}^2}$, the inverse q^{-1} of q is given by:

$$q^{-1} = \frac{q^*}{|q|^2} \tag{A.34}$$

[4]When $\odot$ notation is not used, consider common matrix/vector products.

And for any quaternion q:

$$q \odot q^{-1} = q^{-1} \odot q = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{A.35}$$

A.3.1. Matrix representation of the Hamilton product

The product $q \odot p$ can be also rewritten as a simple product between a 4×4 matrix and quaternion 4-vector:

$$q \odot p = \mathcal{L}(q)p = \mathcal{R}(p)q \tag{A.36}$$

Where $\mathcal{L}(\cdot)$ represents a Hamilton product from the **left** and $\mathcal{R}(\cdot)$ represents a Hamilton product from the **right**. Both of these matrices have multiple possible definitions[5], but we will use:

$$\begin{aligned} \mathcal{L}(q) &= \begin{bmatrix} q_\mathrm{r} & -q_\mathrm{x} & -q_\mathrm{y} & -q_\mathrm{z} \\ q_\mathrm{x} & q_\mathrm{r} & -q_\mathrm{z} & q_\mathrm{y} \\ q_\mathrm{y} & q_\mathrm{z} & q_\mathrm{r} & -q_\mathrm{x} \\ q_\mathrm{z} & -q_\mathrm{y} & q_\mathrm{x} & q_\mathrm{r} \end{bmatrix} = q_\mathrm{r}\, \mathbb{I}_4 + \begin{bmatrix} 0 & -\boldsymbol{q}^T \\ \boldsymbol{q} & \widehat{\boldsymbol{q}} \end{bmatrix} \\ \mathcal{R}(q) &= \begin{bmatrix} q_\mathrm{r} & -q_\mathrm{x} & -q_\mathrm{y} & -q_\mathrm{z} \\ q_\mathrm{x} & q_\mathrm{r} & q_\mathrm{z} & -q_\mathrm{y} \\ q_\mathrm{y} & -q_\mathrm{z} & q_\mathrm{r} & q_\mathrm{x} \\ q_\mathrm{z} & q_\mathrm{y} & -q_\mathrm{x} & q_\mathrm{r} \end{bmatrix} = q_\mathrm{r}\, \mathbb{I}_4 + \begin{bmatrix} 0 & -\boldsymbol{q}^T \\ \boldsymbol{q} & -\widehat{\boldsymbol{q}} \end{bmatrix} \end{aligned} \tag{A.37}$$

Note the very useful basic properties:

$$\begin{aligned} \mathcal{L}(q^*) &= \mathcal{L}(q)^T \\ \mathcal{R}(q^*) &= \mathcal{R}(q)^T \\ \mathcal{L}(\lambda\, q) &= \lambda\, \mathcal{L}(q) \\ \mathcal{R}(\lambda\, q) &= \lambda\, \mathcal{R}(q) \end{aligned} \tag{A.38}$$

For $\lambda \in \mathbb{R}$. The following reduced definitions are also useful to keep in mind:

$$\begin{aligned} \mathcal{L}_1 = \mathcal{L}_1(q) &= \begin{bmatrix} q_\mathrm{x} & q_\mathrm{r} & -q_\mathrm{z} & q_\mathrm{y} \\ q_\mathrm{y} & q_\mathrm{z} & q_\mathrm{r} & -q_\mathrm{x} \\ q_\mathrm{z} & -q_\mathrm{y} & q_\mathrm{x} & q_\mathrm{r} \end{bmatrix} = \begin{bmatrix} \boldsymbol{q} & (q_\mathrm{r}\, \mathbb{I} + \widehat{\boldsymbol{q}}) \end{bmatrix} \\ \mathcal{R}_1 = \mathcal{R}_1(q) &= \begin{bmatrix} q_\mathrm{x} & q_\mathrm{r} & q_\mathrm{z} & -q_\mathrm{y} \\ q_\mathrm{y} & -q_\mathrm{z} & q_\mathrm{r} & q_\mathrm{x} \\ q_\mathrm{z} & q_\mathrm{y} & -q_\mathrm{x} & q_\mathrm{r} \end{bmatrix} = \begin{bmatrix} \boldsymbol{q} & (q_\mathrm{r}\, \mathbb{I} - \widehat{\boldsymbol{q}}) \end{bmatrix} \end{aligned} \tag{A.39}$$

[5] https://en.wikipedia.org/wiki/Quaternion#Matrix_representations

Such that:

$$\begin{aligned}\mathcal{L}(q) &= \begin{bmatrix}(q*)^T \\ \mathcal{L}_1\end{bmatrix} \\ \mathcal{R}(q) &= \begin{bmatrix}(q*)^T \\ \mathcal{R}_1\end{bmatrix}\end{aligned} \tag{A.40}$$

These multiplicative properties are important:

$$\begin{aligned}\mathcal{L}(q \odot p) &= \mathcal{L}(q)\mathcal{L}(p) \\ \mathcal{R}(q \odot p) &= \mathcal{R}(p)\mathcal{R}(q) \\ \mathcal{L}(q)\mathcal{R}(p) &= \mathcal{R}(p)\mathcal{L}(q)\end{aligned} \tag{A.41}$$

Defining the conjugation matrix $\mathcal{C} = \begin{bmatrix}1 & \mathbf{0} \\ \mathbf{0} & -\mathbb{I}\end{bmatrix}$, we can state:

$$q^* = \mathcal{C}\, q \tag{A.42}$$

And we can use it to find the following relationships:

$$\begin{aligned}\mathcal{L}(q) &= \mathcal{C}\ \mathcal{R}(q)^T\ \mathcal{C} \\ \mathcal{R}(q) &= \mathcal{C}\ \mathcal{L}(q)^T\ \mathcal{C}\end{aligned} \tag{A.43}$$

A final practical definition used in this work[6]:

$$\begin{aligned}E(q) &= \mathcal{L}(q)^T = \mathcal{L}(q^*) \\ &= \begin{bmatrix} q_\mathrm{r} & q_\mathrm{x} & q_\mathrm{y} & q_\mathrm{z} \\ -q_\mathrm{x} & q_\mathrm{r} & q_\mathrm{z} & -q_\mathrm{y} \\ -q_\mathrm{y} & -q_\mathrm{z} & q_\mathrm{r} & q_\mathrm{x} \\ -q_\mathrm{z} & q_\mathrm{y} & -q_\mathrm{x} & q_\mathrm{r} \end{bmatrix} \\ &= \begin{bmatrix} q^T \\ E_1 \end{bmatrix}\end{aligned} \tag{A.44}$$

Where:

$$E_1 = E_1(q) = \begin{bmatrix} -q_\mathrm{x} & q_\mathrm{r} & q_\mathrm{z} & -q_\mathrm{y} \\ -q_\mathrm{y} & -q_\mathrm{z} & q_\mathrm{r} & q_\mathrm{x} \\ -q_\mathrm{z} & q_\mathrm{y} & -q_\mathrm{x} & q_\mathrm{r} \end{bmatrix} \tag{A.45}$$

[6] Useful definition for consistency with [**Udwadia2010**].

And we can note that:

$$\begin{aligned}\mathcal{L}(q) = E(q)^T &= \begin{bmatrix} q^T \\ E_1 \end{bmatrix}^T \\ &= \begin{bmatrix} q & E_1^T \end{bmatrix}\end{aligned} \tag{A.46}$$

A useful property:

$$\mathcal{L}_1(q)q^* = \mathcal{R}_1(q)q^* = E_1(q)q = \mathbf{0} \tag{A.47}$$

And if $|q| = 1$:

$$\begin{aligned}\mathcal{L}(q)\mathcal{L}(q)^T = \mathcal{R}(q)\mathcal{R}(q)^T = E(q)E(q)^T = \mathbb{I}_4 \\ \mathcal{L}_1(q)\mathcal{L}_1(q)^T = \mathcal{R}_1(q)\mathcal{R}_1(q)^T = E_1(q)E_1(q)^T = \mathbb{I}\end{aligned} \tag{A.48}$$

A.3.2. Unit quaternions as rotations

If q is a unit quaternion[7], meaning that $|q| = 1$, it can be used to represent the rotation between two frames of reference. Also, in this case:

$$q^{-1} = q^* \tag{A.49}$$

Denoting $\boldsymbol{v}^A$ and $\boldsymbol{v}^B$ a vector $\boldsymbol{v}$ in frames A and B respectively and $q = q_A^B$ the **unit** quaternion that represent the rotation from A to B:

$$\begin{bmatrix} 0 \\ \boldsymbol{v}^B \end{bmatrix} = q_A^B \odot \begin{bmatrix} 0 \\ \boldsymbol{v}^A \end{bmatrix} \odot (q_A^B)^* \tag{A.50}$$

The equivalent rotation matrix is given by[8]:

$$R_A^B = \begin{bmatrix} q_\mathrm{r}^2 + q_\mathrm{x}^2 - q_\mathrm{y}^2 - q_\mathrm{z}^2 & -2q_\mathrm{r}q_\mathrm{z} + 2q_\mathrm{x}q_\mathrm{y} & 2q_\mathrm{r}q_\mathrm{y} + 2q_\mathrm{x}q_\mathrm{z} \\ 2q_\mathrm{r}q_\mathrm{z} + 2q_\mathrm{x}q_\mathrm{y} & q_\mathrm{r}^2 - q_\mathrm{x}^2 + q_\mathrm{y}^2 - q_\mathrm{z}^2 & -2q_\mathrm{r}q_\mathrm{x} + 2q_\mathrm{y}q_\mathrm{z} \\ -2q_\mathrm{r}q_\mathrm{y} + 2q_\mathrm{x}q_\mathrm{z} & 2q_\mathrm{r}q_\mathrm{x} + 2q_\mathrm{y}q_\mathrm{z} & q_\mathrm{r}^2 - q_\mathrm{x}^2 - q_\mathrm{y}^2 + q_\mathrm{z}^2 \end{bmatrix} \tag{A.51}$$

[7] Also known as the S^3 group

[8] This is just one of the possible forms for this matrix. Many others can be found given the constraint that $|q|^2 = 1$.

Which can be found using:

$$\begin{aligned}
\begin{bmatrix} 0 \\ \boldsymbol{v}^B \end{bmatrix} &= q \odot \begin{bmatrix} 0 \\ \boldsymbol{v}^A \end{bmatrix} \odot q^* \\
&= \mathcal{L}(q)\mathcal{R}(q^*) \begin{bmatrix} 0 \\ \boldsymbol{v}^A \end{bmatrix} \\
&= \mathcal{L}(q)\mathcal{R}(q)^T \begin{bmatrix} 0 \\ \boldsymbol{v}^A \end{bmatrix} \\
&= \begin{bmatrix} (q^*)^T \\ \mathcal{L}_1 \end{bmatrix} \begin{bmatrix} q^* & \mathcal{R}_1^T \end{bmatrix} \begin{bmatrix} 0 \\ \boldsymbol{v}^A \end{bmatrix} \\
&= \begin{bmatrix} (q^*)^T q^* & (q^*)^T \mathcal{R}_1^T \\ \mathcal{L}_1 q^* & \mathcal{L}_1 \mathcal{R}_1^T \end{bmatrix} \begin{bmatrix} 0 \\ \boldsymbol{v}^A \end{bmatrix} \\
&= \begin{bmatrix} q \cdot q & (\mathcal{R}_1 q^*)^T \\ \mathcal{L}_1 q^* & \mathcal{L}_1 \mathcal{R}_1^T \end{bmatrix} \begin{bmatrix} 0 \\ \boldsymbol{v}^A \end{bmatrix}
\end{aligned} \tag{A.52}$$

And from Eq. A.47:

$$\begin{bmatrix} q \cdot q & (\mathcal{R}_1 q^*)^T \\ \mathcal{L}_1 q^* & \mathcal{L}_1 \mathcal{R}_1^T \end{bmatrix} = \begin{bmatrix} |q|^2 & 0 \\ 0 & \mathcal{L}_1 \mathcal{R}_1^T \end{bmatrix} \tag{A.53}$$

So:

$$\begin{aligned}
\begin{bmatrix} 0 \\ \boldsymbol{v}^B \end{bmatrix} &= \begin{bmatrix} 0 \\ \mathcal{L}_1 \mathcal{R}_1^T \boldsymbol{v}^A \end{bmatrix}, \\
R(q) &= \mathcal{L}_1(q)\mathcal{R}_1(q)^T
\end{aligned} \tag{A.54}$$

Which gives (noting that for any $\boldsymbol{v} \in \mathbb{R}^3$, we have $\boldsymbol{v}\boldsymbol{v}^T = |\boldsymbol{v}|^2\mathbb{I} + \widehat{\boldsymbol{v}}^2$):

$$\begin{aligned}
R(q) &= (q_\mathrm{r}^2 - |\boldsymbol{q}|^2)\mathbb{I} + 2\boldsymbol{q}\boldsymbol{q}^T + 2q_\mathrm{r}\widehat{\boldsymbol{q}} \\
&= \mathbb{I} + 2q_\mathrm{r}\widehat{\boldsymbol{q}} + 2\widehat{\boldsymbol{q}}^{\,2}
\end{aligned} \tag{A.55}$$

A.3.3. Derivative of rotation quaternion and angular velocity

Defining R_A^B and $q = q_A^B$ the rotation matrix and quaternion respectively from Frame A to frame B, and $\boldsymbol{\omega} = \boldsymbol{\omega}_{A/B}^A$ as the angular velocity of Frame A in respect to B (as seen from frame A), we have the relation[9]:

$$\dot{q} = \frac{1}{2} q \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \tag{A.56}$$

[9] If $\boldsymbol{\omega}$ is the velocity seen from frame B, the relationship is different, but this will not be used in this work.

Or:

$$\begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} = 2q^* \odot \dot{q} \tag{A.57}$$

Equation A.57 can be rewritten as:

$$\begin{aligned} \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} &= 2\mathcal{L}(q)^T \dot{q} \\ &= 2E(q)\dot{q} = 2 \begin{bmatrix} q \\ E_1 \end{bmatrix} \dot{q} \end{aligned} \tag{A.58}$$

Or simply:.

$$\boldsymbol{\omega} = 2E_1\dot{q} \tag{A.59}$$

Finally, from A.59, we can also observe:

$$q \cdot \dot{q} = 0 \tag{A.60}$$

Alternatively, Eq. A.59 can also be rewritten as:

$$\begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} = 2 \begin{bmatrix} \dot{q}_r & \dot{q}_x & \dot{q}_y & \dot{q}_z \\ \dot{q}_x & -\dot{q}_r & -\dot{q}_z & \dot{q}_y \\ \dot{q}_y & +\dot{q}_z & -\dot{q}_r & -\dot{q}_x \\ \dot{q}_z & -\dot{q}_y & \dot{q}_x & -\dot{q}_r \end{bmatrix} q \tag{A.61}$$

Which can be rewritten, thanks to Eq. A.60, as:

$$\begin{aligned} \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} &= 2 \begin{bmatrix} -\dot{q}_r & -\dot{q}_x & -\dot{q}_y & -\dot{q}_z \\ \dot{q}_x & -\dot{q}_r & -\dot{q}_z & \dot{q}_y \\ \dot{q}_y & +\dot{q}_z & -\dot{q}_r & -\dot{q}_x \\ \dot{q}_z & -\dot{q}_y & \dot{q}_x & -\dot{q}_r \end{bmatrix} q \\ &= 2E(-\dot{q})q \end{aligned} \tag{A.62}$$

$$\tag{A.63}$$

And finally:

$$\boldsymbol{\omega} = -2\dot{E}_1 q \tag{A.64}$$

Both Eqs. A.59 and A.64 will be useful.

B. Designing thick origami with OrigamiPatterns

In this section, a simple method to use Inkscape and the OrigamiPatterns extension for the design of thick origami with the membrane method will be explained, step-by-step. First, we will see every step needed to draw each of the necessary laser-cut cuts. Then, we will see how to cut this with a laser-cutting machine.

The material used was monolayer acrylic[1] with $0.8mm$ of thickness. The original pattern we want to design can be seen in Fig. B.1.

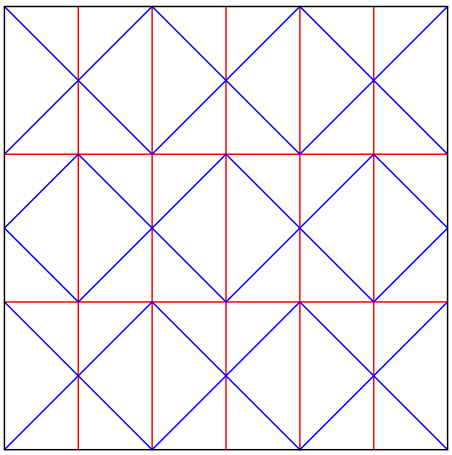


Figure B.1.: Original pattern used as a guideline for the construction of a thicker version.

B.1. Drawing each cut

In this section, we will describe the procedure used to generate all the necessary cut drawings with the aid of Inkscape's built-in functions.

[1] Trotec TroLase ADA Signage

B.1.1. Cut 1

1. Draw valleys with width equal to twice the thickness of the material (in this case, $1.6mm$), mountains to some thin width ($0.1mm$, for example), the vertices with radius sufficiently big ($2.0mm$ in this example), getting Fig. B.2**a**.
2. Ungroup once to separate valleys, mountains, and vertices in three groups `Object -> Ungroup`.
3. Select vertices, ungroup until each vertex is separated, then combine them on a same path object with `Path -> Combine`.
4. Repeat for valley strokes and then for mountain strokes.
5. Selecting the valleys (that are now a single path), select `Path -> Stroke to Path`. Then select `Object -> Fill and Stroke`, remove the fill and add a thin contour. The result will be the one on Fig. B.2**b**.
6. Combine valleys and mountains.
7. First, assuring that the vertices are **above** the valleys, select `Object -> Clip -> Set Inverse (LPE)`. You will have, finally, the result from B.2**c**.

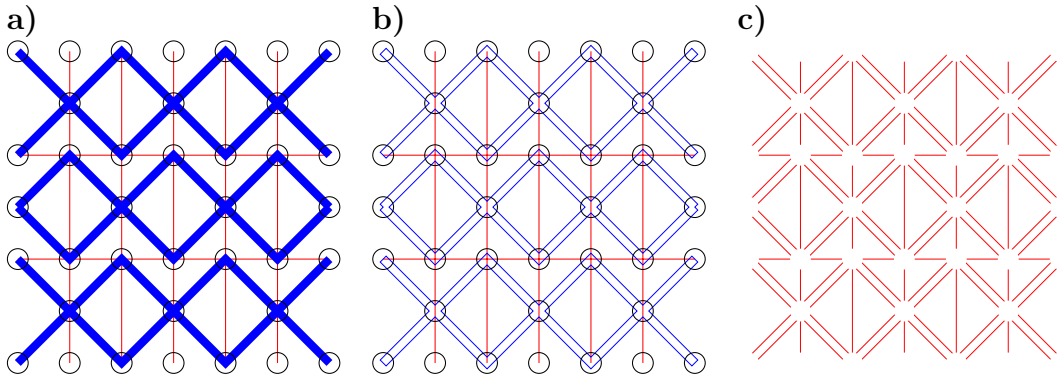


Figure B.2.: Steps for Cut 1.

B.1.2. Cut 2

1. Draw valleys (same thick as before width), edges (same width as valleys) and vertices (same radius as before). See Fig. B.3**a**.
2. Repeat steps 2 and 3 as seen in the instructions for Cut 1.
3. Select valleys and edges and combine them in a single image, then repeat step 5 from Cut 1, getting Fig. B.3**b**.
4. Selecting everything, apply `Path -> Union`, getting Fig. B.3**c**.

5. Select `Path -> Break Apart`, delete the outer edge, then combine again with `Path -> Combine`, finally getting Fig. B.3**d**.

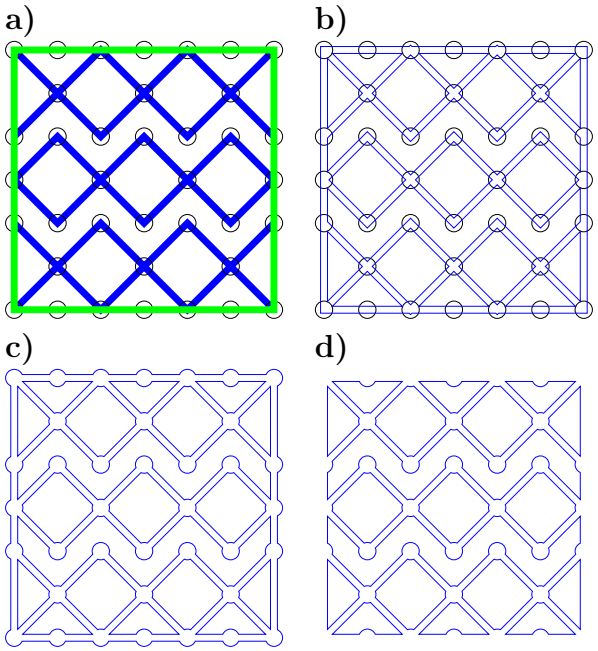


Figure B.3.: Steps for Cut 2.

B.1.3. Cut 3 (optional)

1. Copy Cut 1.
2. Go to `Object -> Fill and Stroke`, go to `Stroke style` options and change stroke type to dashed. See Fig. B.4.

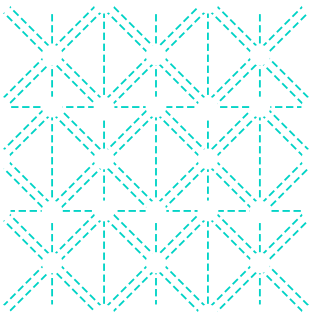


Figure B.4.: Steps for Cut 3. This cut is optional.

B.1.4. Cut 4

1. Draw thin edges and vertices with the same radius as before.
2. Option 1: Simply combine everything, getting Fig. B.5**a**, and the Cut is done.
3. Option 2: Combine every vertex, then select edges and apply `Path -> Intersection`. Then draw new edges, combine everything together and get Fig. B.5**b**.

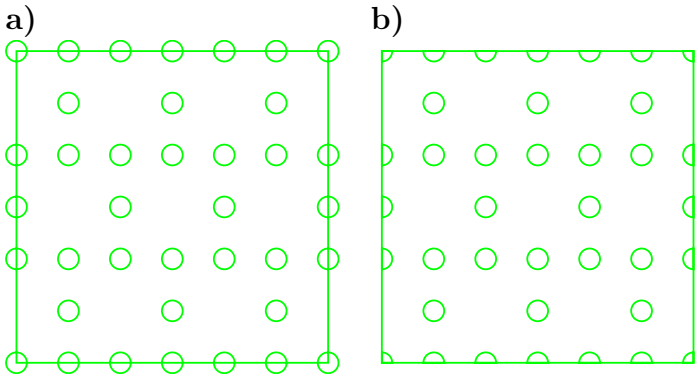


Figure B.5.: Cut 4. Either do Step 1 and Step 2 to get **a)**, or do Step 1 and Step 3 to get **b)**.

l

B.2. Assembling and cutting

1. Organize cut images in this order:
   a) **Cut 1**: Thick material settings (acrylic);
   b) **Cut 2**: Paper settings;
   c) **Cut 3**: Membrane settings (also paper, in this case), optional;
   d) **Cut 4**: Thick material settings (acrylic), this should be able to pierce every layer. Cut multiple times if necessary.

   In our particular laser-cutting machine, the material is chosen according to the color of the strokes. We assembled everything in a same image, every cut with a different color, as seen in Fig. B.6.
2. Put thick material (acrylic) in laser-cutting machine and apply **Cut 1** with thick material settings. See Fig. B.7.
3. Cover with membrane material (paper) and glue borders with scotch tape, see Fig. B.8**a**.

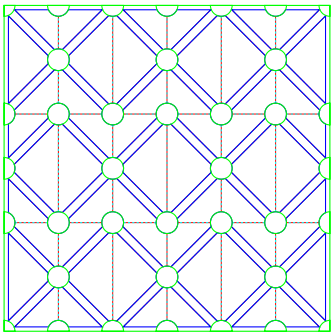


Figure B.6.: Every cut images assembled.

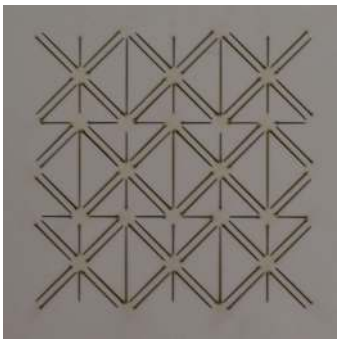


Figure B.7.: Applying Cut 1.

4. Apply **Cut 2** with paper settings, see Fig. B.8**b**.
5. Remove paper free pieces and leave everything that is connected to the glued borders. The remaining paper layer (Fig. B.8**c**) will stop the scotch layer from attaching to small pieces that should be removed, and from gluing parts when folded.

a) **b)** **c)**

Figure B.8.: **a)** Covered with paper. **b)** Applying **Cut 2** and **c)** Unattached paper parts removed.

6. Cover with double-layer scotch tape. See Fig. B.9**a**.

7. Cover with membrane. See Fig. B.9**b**.
8. Optional. Apply **Cut 3**. See Fig. B.9**c**.

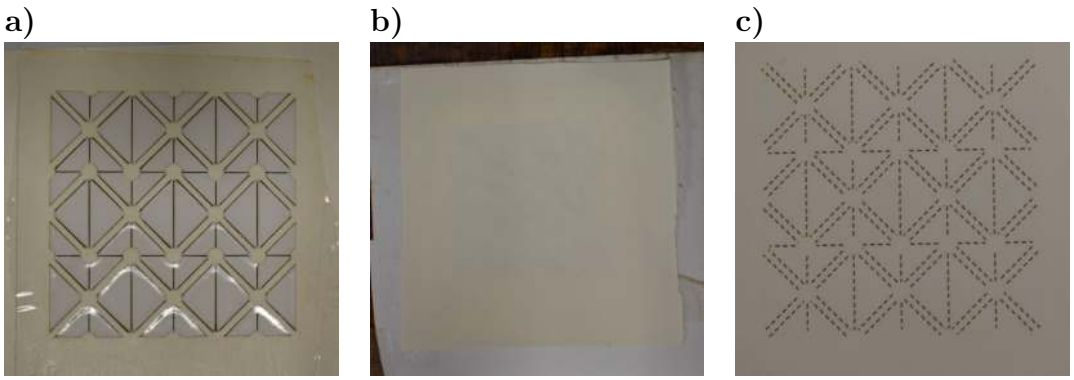


Figure B.9.: **a)** Covered with double-layer scotch tape. **b)** Cover with membrane material and **c)** apply **Cut 3**.

9. Apply **Cut 4** with thick material settings, liberating the piece. This cut has to pierce every layer (thick material, paper, scotch tape and membrane). Apply multiple times if needed. See Fig. B.10**a**.
10. Remove liberated piece. See Fig. B.10**b**.
11. Fold membrane according to pattern. Mountains can only be folded in one direction. See Fig. See Fig. B.10**c**.

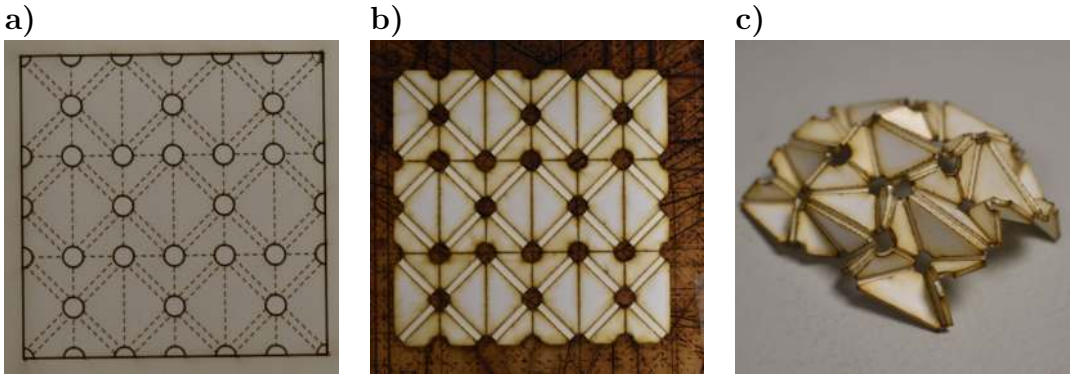


Figure B.10.: **a)** Apply **Cut 4**. **b)** Finished flipped piece before folding and **c)** after folding.

C. Radial ratio for Kresling tower origami

The external radius is usually the parameter that defines the whole scale of the Kresling tower. The internal radius, however, also has some practical importance: in order to fit all the necessary elements of the drone inside the structure (electronics, battery, etc.), we must be sure the internal radius of the closed structure is big enough. In this subsection, I will briefly document an analysis of this internal radius.

Defining the new variable:

$$\gamma = \frac{r_i}{R} \tag{C.1}$$

We can interpret it as a ratio between the internal and external radii of the Kresling tower. Using the relationships in Eq. 1.1, we can show that:

$$\gamma = \sin\left(\frac{\pi(n-2)}{2n}\left(1-\lambda\right)\right)$$

Or, equivalently:

$$\gamma = \sin\left(\pi\left(\frac{1}{2}-\frac{1}{n}\right)\left(1-\lambda\right)\right) \tag{C.2}$$

C.1. Maximum and minimum values

Since $\frac{1}{2} \leq \lambda \leq 1$, we can derive that:

$$0 \leq 1-\lambda \leq \frac{1}{2} \tag{C.3}$$

And since $n \geq 3$ (otherwise it wouldn't define a polygon), we can also show that:

$$\frac{1}{6} \leq \left(\frac{1}{2}-\frac{1}{n}\right) \leq \frac{1}{2} \tag{C.4}$$

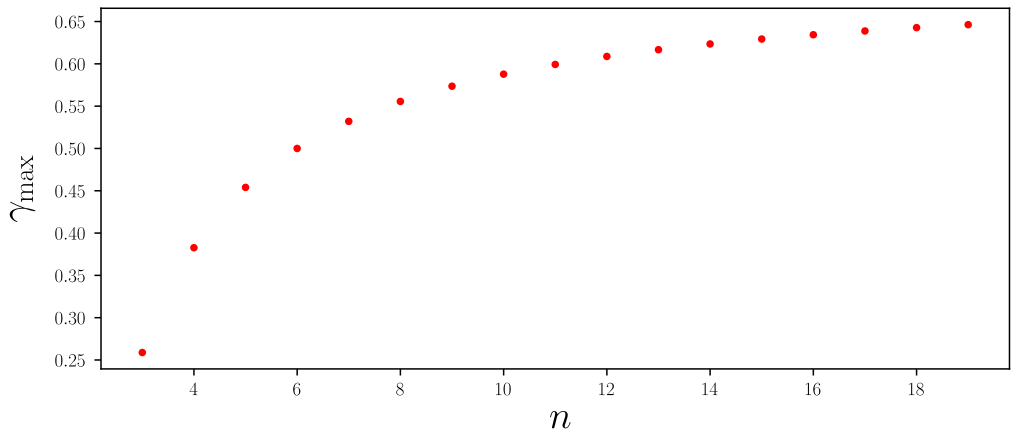


Figure C.1.: Max radial ratio γ_{max} for Kresling towers of different number of sides n.

With this, we can conclude that:

$$0 \leq \pi \left(\frac{1}{2} - \frac{1}{n}\right)(1-\lambda) \leq \frac{\pi}{4} \tag{C.5}$$

We know that:

$$\frac{d}{dx}\sin(x) \geq 0, \text{ for } x \subset [0, \pi/4] \tag{C.6}$$

We can conclude that the greater the biggest possible value for γ happens at the biggest possible value of $\pi\left(\frac{1}{2} - \frac{1}{n}\right)(1-\lambda)$. Moreover, we know from Eq. C.5 that:

$$0 \leq \sin\left(\pi \left(\frac{1}{2} - \frac{1}{n}\right)(1-\lambda)\right) \leq \frac{1}{\sqrt{2}} \tag{C.7}$$

So we know that the minimum possible value for γ is $\gamma_{min} = 0$ for $\lambda = 0.5$, and that the maximum theoretical value is $\gamma_{max} = \frac{1}{\sqrt{2}} \approx 0.707$ for $\lambda = 1$ and a very large n.

C.2. Required n for given γ

For a given constant n, the highest possible value of γ still happens when $\lambda = 0.5$ (Eq. C.6) and has a value of:

$$\gamma^n_{max} = \sin\left(\frac{\pi}{2}\left(\frac{1}{2} - \frac{1}{n}\right)\right) \tag{C.8}$$

This is plotted on Figure C.1. Inverting Eq. C.8 we can find the minimum n needed for a given γ:

$$n_{min} = \left\lceil \frac{2\pi}{\pi - 4\sin^{-1}(\gamma)} \right\rceil \tag{C.9}$$

n	Maximum γ
3	0.25881
4	0.38268
5	0.45399
6	0.50000
7	0.53203
8	0.55557
9	0.57357

Table C.1.: Minimum n needed for a desired γ.

Where $\lceil x \rceil$ is the ceiling function. We can see on Table C.1 the minimum n needed for a desired γ.

C.3. Computing all the other parameters

Using the definition of γ in Eq. C.10, we can invert the equation to find that:

$$\lambda = 1 - \frac{2n}{\pi(n-2)} \sin^{-1}(\gamma) \tag{C.10}$$

So, knowing n and γ, we also know λ. If we also know R, then we have all we need ([43]).

D. Kresling tower height/rotation relationship

In this chapter, I will study the relationship between the height of a Kresling tower cell and its rotation angle.

D.1. Node position

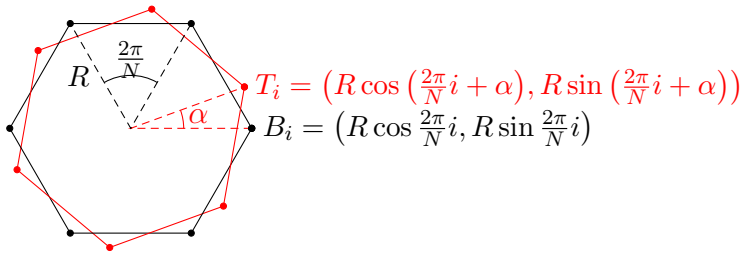


Figure D.1.: Projection of top (in red) and bottom (in black) polygons on the XY plane.

Analyzing the projections of the top and bottom polygons of a single Kresling tower cell on Figure D.1, and taking into account the height z of the Kresling tower cell, defined as the difference in the z direction between both planes, we can define the full 3D position of each point with the following equations:

$$B_i = R\begin{bmatrix}\cos\frac{2\pi}{N}i\\ \sin\frac{2\pi}{N}i\\ 0\end{bmatrix} \tag{D.1}$$

$$T_i = R\begin{bmatrix}\cos\left(\frac{2\pi}{N}i + \alpha\right)\\ \sin\left(\frac{2\pi}{N}i + \alpha\right)\\ \frac{z}{R}\end{bmatrix}$$

D.2. Relationship between R and z

In order to find the theoretical relationship, we must find some kind of assumption. We will explore the two easiest assumptions (which are but approximations of the real tower).

D.2.1. Constant l

Assuming the length of each l is constant during the whole movement of the tower, and analyzing the tower, we can state that:

$$l^2 = |T_{i+2} - B_i|^2 = R^2 \left\| \begin{bmatrix} \cos\left(\frac{2\pi}{N}(i+2)+\alpha\right) - \cos\left(\frac{2\pi}{N}i\right) \\ \sin\left(\frac{2\pi}{N}(i+2)+\alpha\right) - \sin\left(\frac{2\pi}{N}i\right) \\ \frac{z}{R} \end{bmatrix} \right\|^2 \tag{D.2}$$

Fixing $i = 0$ (any value of i will simplify to the same result), we can rewrite Eq. D.2 as the following:

$$\begin{aligned} l^2 &= R^2 \left\| \begin{bmatrix} \cos\left(\frac{4\pi}{N}+\alpha\right) - 1 \\ \sin\left(\frac{4\pi}{N}+\alpha\right) \\ \frac{z}{R} \end{bmatrix} \right\|^2 \\ \frac{l^2}{R^2} &= \left(\cos\left(\frac{4\pi}{N}+\alpha\right) - 1\right)^2 + \left(\sin\left(\frac{4\pi}{N}+\alpha\right)\right)^2 + \frac{z^2}{R^2} \\ \frac{l^2 - z^2}{R^2} &= \left(\cos\left(\frac{4\pi}{N}+\alpha\right)\right)^2 - 2\cos\left(\frac{4\pi}{N}+\alpha\right) + 1 + \left(\sin\left(\frac{4\pi}{N}+\alpha\right)\right)^2 \\ \frac{l^2 - z^2}{R^2} &= 2\left(1 - \cos\left(\frac{4\pi}{N}+\alpha\right)\right) \end{aligned}$$

And finally, we can write it in the following final form:

$$2\,\frac{l^2 - z^2}{(2R)^2} = 1 - \cos\left(\frac{4\pi}{N}+\alpha\right) \tag{D.3}$$

or:

$$\alpha = \operatorname{acos}\left(1 - 2\,\frac{l^2 - z^2}{(2R)^2}\right) - \frac{4\pi}{N}$$

D.2.2. Constant b

If we decide instead to assume that b is constant:

$$b^2 = |T_{i+1} - B_i|^2 \tag{D.4}$$

we can find by the same kind of development the following formula:

$$2\,\frac{b^2 - z^2}{(2R)^2} = 1 - \cos\left(\frac{2\pi}{N} + \alpha\right) \tag{D.5}$$

or:

$$\alpha = \operatorname{acos}\left(1 - 2\,\frac{b^2 - z^2}{(2R)^2}\right) - \frac{2\pi}{N} \tag{D.6}$$

D.3. l and b constant

If we assume both l and b are constant, then we can cross both functions and find the low-energy configurations of the tower cell (since the cell will have no elastic potential energy stored when all the sides are not curved). Defining $B = \frac{\sqrt{b^2-z^2}}{2R}$ and $L = \frac{\sqrt{l^2-z^2}}{2R}$ for simpler of notation, and substituting α on Eq. D.3 by the expression found on Eq. D.6, we can state that:

$$2L^2 = 1 - \cos\left(\frac{2\pi}{N} + \operatorname{acos}\big(1 - 2B^2\big)\right) \tag{D.7}$$

Using the property $\operatorname{acos}(1 - 2x^2) = 2\operatorname{asin}(|x|)$, we can simplify it as:

$$2L^2 = 1 - \cos\left(2\left(\frac{\pi}{N} + \operatorname{asin}(B)\right)\right) \tag{D.8}$$

Using the property $1 - \cos x = 2\sin^2 x$, it can be once again rewritten as:

$$2L^2 = 2\sin^2\left(\frac{\pi}{N} + \operatorname{asin}(B)\right) \tag{D.9}$$

And finally, using property $\sin(a+b) = \sin a \cos b + \cos a \sin b$ and $\cos(\operatorname{asin} x) = \sqrt{1-x^2}$:

$$L = \sqrt{1 - B^2}\sin\left(\frac{\pi}{N}\right) + B\cos\left(\frac{\pi}{N}\right) \tag{D.10}$$

Defining $S = \sin\left(\frac{\pi}{N}\right)$ and $C = \cos\left(\frac{\pi}{N}\right)$ for simplicity, we can rewrite the equation as:

$$\begin{aligned} L &= \sqrt{1-B^2}S + BC \\ L - BC &= \sqrt{1-B^2}S \\ (L-BC)^2 &= (1-B^2)S^2 \\ L^2 - 2LBC + B^2C^2 &= S^2 - B^2S^2 \\ L^2 + B^2 - S^2 &= 2LBC \end{aligned}$$

Since both B and L are defined as square roots, we can square the equation once again to simplify it:

$$\begin{aligned} (L^2 + B^2 - S^2)^2 &= 4L^2B^2C^2 \\ (L^2+B^2)^2 - 2(L^2+B^2)S^2 + S^4 &= 4L^2B^2C^2 \end{aligned}$$

Defining $K = \frac{\sqrt{l^2+b^2}}{2R}$, $P = \frac{\sqrt{lb}}{2R}$ and $h = \frac{z}{2R}$, we can show the following relationships:

$$\begin{aligned} L^2 + B^2 &= K^2 - 2h^2 \\ L^2B^2 &= P^4 - K^2h^2 + h^4 \end{aligned}$$

And the equation can be rewritten as:

$$\begin{aligned} (K^2-2h^2)^2 - 2(K^2-2h^2)S^2 + S^4 &= 4(P^4 - K^2h^2 + h^4)C^2 \\ K^4 - 4K^2h^2 + 4h^4 - 4K^2S^2 + 4h^2S^2 + S^4 &= 4P^4C^2 - 4K^2h^2C^2 + 4h^4C^2 \\ 4(1-C^2)h^4 + (K^4 - 2K^2S^2 + S^4) + (4K^2C^2 - 4K^2 + 4S^2)h^2 &= 4P^4C^2 \end{aligned}$$

And finally:

$$\begin{aligned} 4(1-C^2)h^4 + \left[4K^2(C^2-1) + 4S^2\right]h^2 + (K^2-S^2)^2 - 4P^4C^2 &= 0 \\ S^2h^4 + \left[S^2(1-K^2)\right]h^2 + \left[\frac{(K^2-S^2)^2}{4} - P^4C^2\right] &= 0 \end{aligned}$$

Which is a biquadratic equation for h and can be solved using the quadratic formula:

$$h^2 = \frac{S(K^2-1) \pm C\sqrt{4P^4 - K^4 + S^2}}{2S}$$

Using the fact that the height is non-negative, we can eliminate the 2 possible negatives solutions:

$$h = \sqrt{\frac{S(K^2-1) \pm C\sqrt{4P^4 - K^4 + S^2}}{2S}} \tag{D.11}$$

D.4. Numerical simulations

To test the solution, here is the plot of both functions (constant l and constant b) with two vertical bars representing the analytical solutions using Eq. D.11: To

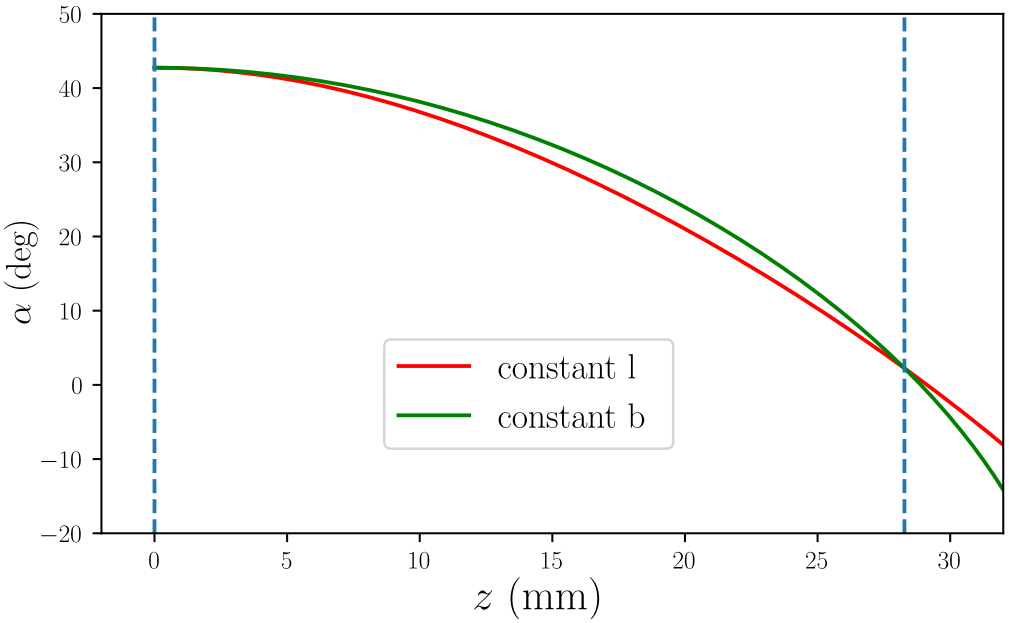


Figure D.2.: Numerical solution of the Kresling tower heights.

test the effect of the parameter λ, we plotted on Figure 1.15 the results of h for λ varying between 0.5 and 1: Here we can observe that:

- One of the solutions appears to be always equal to 0;
- One of the solutions tends towards 0 when λ tends towards 0.5.

Analyzing Eq. D.11, if one of the solutions is always equal to zero, we can state that:

$$S(K^2-1) = C\sqrt{4P^4 - K^4 + S^2} \tag{D.12}$$

Which leads to the following simplification:

$$\begin{aligned} h^2 &= \frac{S(K^2-1) \pm C\sqrt{4P^4 - K^4 + S^2}}{2S} \\ h^2 &= \frac{S(K^2-1) \pm S(K^2-1)}{2S} \\ h^2 &= \frac{(K^2-1)}{2}(1 \pm 1) \end{aligned}$$

And finally:

$$\begin{aligned} h^+ &= \sqrt{K^2-1} \\ h^- &= 0 \end{aligned} \tag{D.13}$$

Or:

$$\begin{aligned} z^+ &= 2R\sqrt{\left(\frac{l^2+b^2}{2R}\right)^2 - 1} = \sqrt{\left(l^2+b^2\right)^2 - 4R^2} \\ z^- &= 0 \end{aligned} \tag{D.14}$$

E. Autopilot technical details

To better match the end goal of a lightweight drone, the chosen hardware was an important decision. Our first choice was to use the Elle0 card produced by 1BitSquared [1], seen in Fig. E.1**a**. This card has a 32bit ARM Cortex M4 microprocessor and its size is only $30.5mm \times 30.5mm$. In the end, we decided to switch to a Pixracer card produced by mRobotics[2]. This card has an equivalent 32-bit STM32F427 Cortex M4 processor and has size $36mm \times 36mm$ (Fig. E.1**c**.).

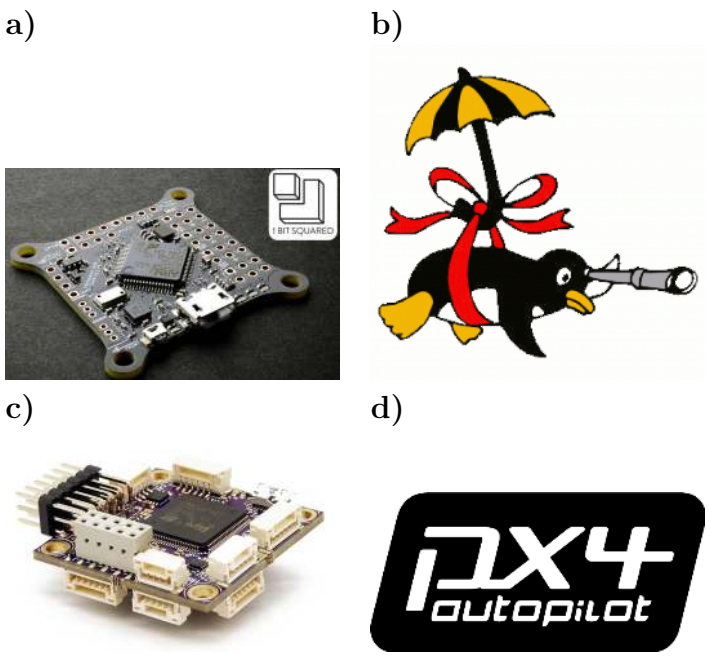


Figure E.1.: Autopilots for the project. **a)** Elle0 autopilot by 1BitSquared (discontinued), that uses the Parapazzi UAV **b)**. **c)** Pixracer autopilot by mRobotics, that uses the PX4 UAV **d)**.

Even though it is (slightly) bigger, this choice was motivated by their software: while the Elle0 is powered by Paparazzi[3], the Pixracer card is powered by PX4 [4].

[1] https://1bitsquared.com/products/elle0-autopilot
[2] https://store.mrobotics.io/mRo-PixRacer-R14-Official-p/auav-pxrcr-r14-mr.htm
[3] https://wiki.paparazziuav.org/wiki/Main_Page
[4] https://px4.io/

PX4 has two advantages for us in the project: it is more widely used and known, making it easier to develop, and, more importantly, PX4 is the Autopilot firmware used by the original developers of Vertiq. This made it possible for us to use their original code as an inspiration for our own PX4 fork, implementing the integration of the Vertiq module firmware into the Autopilot firmware[5].

For wireless telemetry and communication with the drone, the Pixracer card comes bundled with an ESP8266 Wi-Fi chip[6], flashed with the MavESP8266 firmware[7]. The messaging protocol used for communications between PX4 and the Ground Control is called MavLink[8]. The MavESP8266 is a specialized firmware that transforms the Wi-Fi Access Point into a MavLink bridge.

The connection with the ESP8266 chip worked well *in the beginning of the project*, before I tried integrating everything in the Flying Arena (see Fig. E.2), a space equipped with Vicon[9] infrared motion-capture cameras used for flight tests at the ISM Biorobotics.

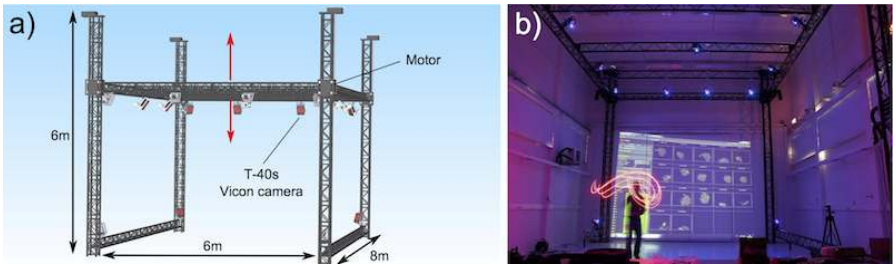


Figure E.2.: **a)** CAD of the Flying Arena composed of a motorized metallic tubular structure equipped with 17 T-40s Vicon cameras. **b)** The Flying Arena during the calibration process. Source: http://www.ism.univ-amu.fr/viollet/arena_english.html.

Integrating the usual communications exchange between the Pixracer card with Ground Control with the Flying Arena position/orientation data and the extra commands needed for testing the Swashplateless ended up being more challenging than expected, and the ESP8266 chip was not powerful enough to handle all communication with it reliably. We decided instead to add an extra *RaspberryPi Zero W 2*[10] to the drone (see Fig. E.3). This low-cost[11] miniature Linux computer has a size of only $65mm \times 30mm$, weights only $11g$ and has integrated Wi-Fi. This board running Raspberry Pi OS[12], connects directly by an UART connection to the

[5]My fork can be found here: https://github.com/evbernardes/PX4-Autopilot
[6]https://en.wikipedia.org/wiki/ESP8266
[7]https://github.com/dogmaphobic/mavesp8266
[8]https://docs.px4.io/main/en/middleware/mavlink.html
[9]https://www.vicon.com/
[10]https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/
[11]15 euros in pre-COVID times.
[12]A specialized Linux distribution made for Raspberry Pi, formerly called Raspbian: https://www.raspberrypi.com/software/

Figure E.3.: Raspberry Pi Zero W2 board.

Pixracer board using MAVROS[13], a compatibility layer that translates MavLink messages into ROS messages and vice-versa. This makes it possible to make a direct connection from the Drone to the Flying Arena system without an intermediate connection with a ground computer, making the connection much more robust.

[13]http://wiki.ros.org/mavros

F. Relationship between angular velocity and derivative of Euler angles

In this appendix, we will analyze the derivatives of full rotation matrices when decomposed into any sequence of Euler angles, and then develop a general solution to the problem of computing the derivative of Euler angles from the actual orientation and the robot's angular velocity.

F.1. Derivative of full rotation matrix

We suppose a rotation matrix $R = R_B^E$ represents the orientation of some body. We know that for a matrix $R_{\phi \boldsymbol{u}}$ representing the rotation of an angle ϕ around some axis $\boldsymbol{u}$:

$$\frac{\mathrm{d}R_{\phi \boldsymbol{u}}}{\mathrm{d}t} = \frac{\mathrm{d}\phi}{\mathrm{d}t} R_{\phi \boldsymbol{u}} \widehat{\boldsymbol{u}} \tag{F.1}$$

The rotation matrix R can then be decomposed as:

$$R = R_3 \, R_2 \, R_1 \tag{F.2}$$

Where:

$$\begin{aligned} R_1 &= R_{\theta_1 \boldsymbol{e}_1} \\ R_2 &= R_{\theta_2 \boldsymbol{e}_2} \\ R_3 &= R_{\theta_3 \boldsymbol{e}_3} \end{aligned} \tag{F.3}$$

And defining $\boldsymbol{\omega}$ the body's angular velocity in the body frame:

$$\frac{\mathrm{d}R}{\mathrm{d}t} = R \widehat{\boldsymbol{\omega}} \tag{F.4}$$

This leads to:

$$\begin{aligned} R\widehat{\boldsymbol{\omega}} &= \frac{\mathrm{d}R}{\mathrm{d}t} \\ &= \frac{\mathrm{d}R_3}{\mathrm{d}t}\, R_2\, R_1 + R_3\, \frac{\mathrm{d}R_2}{\mathrm{d}t}\, R_1 + R_3\, R_2\, \frac{\mathrm{d}R_1}{\mathrm{d}t} \\ &= \dot{\theta}_3 R_3\, \widehat{\boldsymbol{e}}_3\, R_2\, R_1 + \dot{\theta}_2 R_3\, R_2\, \widehat{\boldsymbol{e}}_2\, R_1 + \dot{\theta}_1 R_3\, R_2\, R_1\, \widehat{\boldsymbol{e}}_1 \end{aligned} \tag{F.5}$$

Multiplying both sides by $R^T = R_1^T\, R_2^T\, R_3^T$ on the left:

$$\widehat{\boldsymbol{\omega}} = \dot{\theta}_3 R_1^T\, R_2^T\, \widehat{\boldsymbol{e}}_3\, R_2\, R_1 + \dot{\theta}_2 R_1^T\, \widehat{\boldsymbol{e}}_2\, R_1 + \dot{\theta}_1 \widehat{\boldsymbol{e}}_1 \tag{F.6}$$

And applying the $(\cdot)^{\check{}}$ operator define in Eq. A.5:

$$\boldsymbol{\omega} = (\widehat{\boldsymbol{\omega}})^{\check{}} = \dot{\theta}_3 \left(R_1^T\, R_2^T\, \widehat{\boldsymbol{e}}_3\, R_2\, R_1\right)^{\check{}} + \dot{\theta}_2 \left(R_1^T\, \widehat{\boldsymbol{e}}_2\, R_1\right)^{\check{}} + \dot{\theta}_1 \left(\widehat{\boldsymbol{e}}_1\right)^{\check{}} \tag{F.7}$$

And noting $(R\boldsymbol{a})\widehat{\ } = R\,\widehat{\boldsymbol{a}}\,R^T$ for any rotation R and $\boldsymbol{a} \in \mathbb{R}^3$:

$$\boldsymbol{\omega} = \dot{\theta}_3 \left(R_1^T\, R_2^T\right) \boldsymbol{e}_3 + \dot{\theta}_2 \left(R_1^T\right) \boldsymbol{e}_2 + \dot{\theta}_1 \boldsymbol{e}_1 \tag{F.8}$$

Noting that $R_3\, \boldsymbol{e}_3 = R_3^T\, \boldsymbol{e}_3 = \boldsymbol{e}_3$, we can also rewrite Eq. F.8 as:

$$\boldsymbol{\omega} = \dot{\theta}_3\, R^T \boldsymbol{e}_3 + \dot{\theta}_2\, R_1^T \boldsymbol{e}_2 + \dot{\theta}_1 \boldsymbol{e}_1 \tag{F.9}$$

Or, in matrix form:

$$\boldsymbol{\omega} = R_1^T\, R_2^T \begin{bmatrix} (R_2\, \boldsymbol{e}_1)^T \\ (\boldsymbol{e}_2)^T \\ (\boldsymbol{e}_3)^T \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} \tag{F.10}$$

F.2. Derivative of full rotation matrix in ZYZ sequence

In this section, we will apply the result in Eq. F.8 to a rotation matrix defined in the ZYZ sequence. For the ZYZ sequence, we can note:

$$\boldsymbol{e}_1 = \boldsymbol{e}_3 = \boldsymbol{e}_z,\ \boldsymbol{e}_2 = \boldsymbol{e}_y, \text{ and } (\theta_1, \theta_2, \theta_3) = (\gamma, \alpha, \beta) \tag{F.11}$$

Where $\boldsymbol{e}_y$ and $\boldsymbol{e}_z$ are unit vectors along axes y and z. As defined in Eq. A.6, the rotation matrix of an angle θ around axis y and z are, respectively:

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix},\ R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{F.12}$$

Notating $R_\beta = R_z(\beta)$, $R_\alpha = R_y(\alpha)$ and $R_\gamma = R_z(\gamma)$, and defining $\boldsymbol{\Omega}_{zyz}$ as the angular velocity, we have:

$$\begin{aligned} R &= R_\beta\, R_\alpha\, R_\gamma \\ \dot{R} &= R\, \widehat{\boldsymbol{\Omega}}_{zyz} \end{aligned} \tag{F.13}$$

And inserting the definitions in Eqs. F.11 and F.13 into Eq. F.8:

$$\begin{aligned} \boldsymbol{\Omega}_{zyz} &= \dot{\theta}_3 \left(R_1^T\ R_2^T\right) \boldsymbol{e}_3 + \dot{\theta}_2 \left(R_1^T\right) \boldsymbol{e}_2 + \dot{\theta}_1 \boldsymbol{e}_1 \\ &= \dot{\beta} \left(R_\gamma^T\ R_\alpha^T\right) \boldsymbol{e}_z + \dot{\alpha} \left(R_\gamma^T\right) \boldsymbol{e}_y + \dot{\gamma} \boldsymbol{e}_z \\ &= R_\gamma^T \left(R_\alpha^T \dot{\beta} \boldsymbol{e}_z + \dot{\alpha} \boldsymbol{e}_y\right) + \dot{\gamma} \boldsymbol{e}_z \end{aligned} \tag{F.14}$$

Or:

$$\boldsymbol{\Omega}_{zyz} = \begin{bmatrix} -\sin\alpha\cos\gamma & \sin\gamma & 0 \\ \sin\alpha\sin\gamma & \cos\gamma & 0 \\ \cos\alpha & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\beta} \\ \dot{\alpha} \\ \dot{\gamma} \end{bmatrix} \tag{F.15}$$

Also note that we can define:

$$\boldsymbol{\omega}_{zy} = R_\alpha^T \dot{\beta} \boldsymbol{e}_z + \dot{\alpha} \boldsymbol{e}_y = \begin{bmatrix} -\sin\alpha & 0 & 0 \\ 0 & 1 & 0 \\ \cos\alpha & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\beta} \\ \dot{\alpha} \\ 0 \end{bmatrix} \tag{F.16}$$

And noting that $R_\gamma^T \boldsymbol{e}_z = \boldsymbol{e}_z$, we can write, for the general case:

$$\boldsymbol{\Omega}_{zyz} = R_\gamma^T \left(\boldsymbol{\omega}_{zy} + \dot{\gamma} \boldsymbol{e}_z\right) \tag{F.17}$$

F.3. General relationship between angular velocity and angles in matrix form

As a reminder, Eq. F.9 states that:

$$\boldsymbol{\omega} = \dot{\theta}_3 \, R^T \boldsymbol{e}_3 + \dot{\theta}_2 \, R_1^T \boldsymbol{e}_2 + \dot{\theta}_1 \boldsymbol{e}_1 \tag{F.18}$$

Or, equivalently:

$$R_2 R_1 \boldsymbol{\omega} = \dot{\theta}_3 \boldsymbol{e}_3 + \dot{\theta}_2 \boldsymbol{e}_2 + \dot{\theta}_1 \, R_2 \, \boldsymbol{e}_1 \tag{F.19}$$

Moreover, we remember that for any Euler angles sequence chosen (or, more generally, any Davenport decomposition), $\boldsymbol{e}_2$ must be orthogonal to both $\boldsymbol{e}_1$ and $\boldsymbol{e}_3$:

$$\boldsymbol{e}_2 \cdot \boldsymbol{e}_1 = \boldsymbol{e}_2 \cdot \boldsymbol{e}_3 = 0 \tag{F.20}$$

And it is trivial to show that:

$$\boldsymbol{e}_2 \cdot R_2 \, \boldsymbol{e}_1 = 0 \tag{F.21}$$

Calculating the dot product of both sides of Eq. F.19 with $R_2 \, \boldsymbol{e}_1$ gives:

$$(R_2 \, \boldsymbol{e}_1) \cdot R_2 R_1 \boldsymbol{\omega} = \dot{\theta}_3 \boldsymbol{e}_3 \cdot (R_2 \, \boldsymbol{e}_1) + \dot{\theta}_1 \tag{F.22}$$

Calculating the dot product with $\boldsymbol{e}_2$:

$$\boldsymbol{e}_2 \cdot R_2 R_1 \boldsymbol{\omega} = \dot{\theta}_2 \tag{F.23}$$

And finally, calculating the dot product with $\boldsymbol{e}_3$:

$$\boldsymbol{e}_3 \cdot R_2 R_1 \boldsymbol{\omega} = \boldsymbol{e}_3 + \dot{\theta}_1 \, \boldsymbol{e}_2 \cdot R_2 \, \boldsymbol{e}_1 \tag{F.24}$$

To summarize, Eqs. F.22, F.23 and F.24 can be rewritten in matrix form as:

$$\boldsymbol{A} \, \boldsymbol{\omega} = \boldsymbol{B} \, \dot{\boldsymbol{\theta}} \tag{F.25}$$

Where $\dot{\boldsymbol{\theta}} = \begin{bmatrix} \dot{\theta}_1 & \dot{\theta}_2 & \dot{\theta}_3 \end{bmatrix}^T$, and:

$$\boldsymbol{A} \equiv \begin{bmatrix} (R_2 \, \boldsymbol{e}_1)^T \\ (\boldsymbol{e}_2)^T \\ (\boldsymbol{e}_3)^T \end{bmatrix} R_2 R_1, \; \boldsymbol{B} \equiv \begin{bmatrix} 1 & 0 & \boldsymbol{e}_3 \cdot (R_2 \, \boldsymbol{e}_1) \\ 0 & 1 & 0 \\ \boldsymbol{e}_3 \cdot (R_2 \, \boldsymbol{e}_1) & 0 & 1 \end{bmatrix} \tag{F.26}$$

F.4. Derivative of angles by matrix inversion

In this section, we will develop a formula for $\dot{\boldsymbol{\theta}}$. The system degenerates and is under-determined when the rotation around $\boldsymbol{e}_2$ aligns $\boldsymbol{e}_1$ with $\boldsymbol{e}_3$. As seen in Chapter 10, it can be useful to define:

$$\begin{aligned} \theta_+ &= (\theta_3 + \theta_1)/2 \\ \theta_- &= (\theta_3 - \theta_1)/2 \end{aligned} \tag{F.27}$$

Such that:

$$\begin{aligned} \theta_3 &= \theta_+ + \theta_- \\ \theta_1 &= \theta_+ - \theta_- \end{aligned} \tag{F.28}$$

And we can finally rewrite $\dot{\boldsymbol{\theta}}$ as:

$$\dot{\boldsymbol{\theta}} = \boldsymbol{C}\,\dot{\boldsymbol{\theta}}' \tag{F.29}$$

Where:

$$\begin{aligned} \boldsymbol{C} &= \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \\ \dot{\boldsymbol{\theta}}' &= \begin{bmatrix} \dot{\theta}_+ \\ \dot{\theta}_2 \\ \dot{\theta}_- \end{bmatrix} \end{aligned} \tag{F.30}$$

We can then rewrite Eq. F.25 as:

$$\begin{aligned} \boldsymbol{B}\,\boldsymbol{C}\,\dot{\boldsymbol{\theta}}' &= \boldsymbol{A}\,\boldsymbol{\omega} \\ \dot{\boldsymbol{\theta}}' &= (\boldsymbol{B}\,\boldsymbol{C})^{-1}\,\boldsymbol{A}\,\boldsymbol{\omega} \end{aligned} \tag{F.31}$$

Where, defining $\Lambda = \boldsymbol{e}_3 \cdot (R_2\,\boldsymbol{e}_1)$:

$$\boldsymbol{B}\,\boldsymbol{C} = \begin{bmatrix} 1+\Lambda & 0 & -(1-\Lambda) \\ 0 & 1 & 0 \\ 1+\Lambda & 0 & 1-\Lambda \end{bmatrix} \tag{F.32}$$

And:

$$(\boldsymbol{B}\,\boldsymbol{C})^{-1} = \frac{1}{2}\begin{bmatrix} \frac{1}{1+\Lambda} & 0 & \frac{1}{1+\Lambda} \\ 0 & 2 & 0 \\ \frac{-1}{1-\Lambda} & 0 & \frac{1}{1-\Lambda} \end{bmatrix} \tag{F.33}$$

Where, as expected, either θ_+ or θ_- is undefined (when $\Lambda = -1$ or $\Lambda = 1$, respectively). As seen in Eq. 10.47, defining:

$$\theta_\lambda = \text{atan2}\left(\boldsymbol{e}_3 \cdot (\boldsymbol{e}_1 \times \boldsymbol{e}_2), \boldsymbol{e}_3 \cdot \boldsymbol{e}_1\right) \tag{F.34}$$

We can state that $\boldsymbol{e}_1 = R_{\theta_\lambda \boldsymbol{e}_2}\, \boldsymbol{e}_3$, and:

$$\begin{aligned} \boldsymbol{e}_3 \cdot (R_2\, \boldsymbol{e}_1) &= \boldsymbol{e}_3 \cdot (R_2\, R_{\theta_\lambda \boldsymbol{e}_2}\, \boldsymbol{e}_1) \\ &= \boldsymbol{e}_3 \cdot (R_{(\theta_2+\theta_\lambda)\boldsymbol{e}_2}\, \boldsymbol{e}_3) \end{aligned} \tag{F.35}$$

Using, for example, Rodrigues' formula for $R_{(\theta_2+\theta_\lambda)\boldsymbol{e}_2}$ gives:

$$R_{(\theta_2+\theta_\lambda)\boldsymbol{e}_2} = \left(\cos(\theta_2+\theta_\lambda)\mathbb{I} + \widehat{\boldsymbol{e}}_2 \sin(\theta_2+\theta_\lambda) + \boldsymbol{e}_2\boldsymbol{e}_2^T(1-\cos(\theta_2+\theta_\lambda))\right) \tag{F.36}$$

And finally:

$$\Lambda = \boldsymbol{e}_3 \cdot (R_2\, \boldsymbol{e}_1) = \cos(\theta_2+\theta_\lambda) \tag{F.37}$$

Note that for symmetric / proper Euler sequences, $\theta_\lambda = 0$ or $\theta_\lambda = \pi$, while for symmetric / Tait-Bryan sequences, $\theta_\lambda \pm \pi/2$.

As seen in Chapter 10, in any of these cases, we can set any constant value for θ_1, define $\dot{\theta}_1 = 0$ and:

- $\dot{\theta}_3 = 2\dot{\theta}_+$ when $\Lambda = 1$, or $\theta_2 + \theta_\lambda = 2k\pi$ for $k \in \mathbb{Z}$.
- $\dot{\theta}_3 = 2\dot{\theta}_-$ when $\Lambda = -1$, or $\theta_2 + \theta_\lambda = (2k-1)\pi$ for $k \in \mathbb{Z}$.

For all the other cases, then:

$$\begin{aligned} \dot{\boldsymbol{\theta}} = \boldsymbol{C}\,\dot{\boldsymbol{\theta}}' &= \boldsymbol{C}\,(\boldsymbol{B}\boldsymbol{C})^{-1}\,\boldsymbol{A}\boldsymbol{\omega} \\ &= \boldsymbol{B}^{-1}\,\boldsymbol{A}\boldsymbol{\omega} \end{aligned} \tag{F.38}$$

Where:

$$\boldsymbol{B}^{-1} = \begin{bmatrix} \frac{1}{1-\Lambda^2} & 0 & \frac{-\Lambda}{1-\Lambda^2} \\ 0 & 1 & 0 \\ \frac{-\Lambda}{1-\Lambda^2} & 0 & \frac{1}{1-\Lambda^2} \end{bmatrix} \tag{F.39}$$

And the full matrix can be given as:

$$\boldsymbol{M} = \boldsymbol{B}^{-1}\boldsymbol{A} = \begin{bmatrix} \frac{1}{1-\Lambda^2} & 0 & \frac{-\Lambda}{1-\Lambda^2} \\ 0 & 1 & 0 \\ \frac{-\Lambda}{1-\Lambda^2} & 0 & \frac{1}{1-\Lambda^2} \end{bmatrix} \begin{bmatrix} (R_2\,\boldsymbol{e}_1)^T \\ (\boldsymbol{e}_2)^T \\ (\boldsymbol{e}_3)^T \end{bmatrix} R_2 R_1 \tag{F.40}$$

Which can be used to generate the desired formulas. [1]

The SymPy code from Listing F.1 calculates the derivative matrix for the usual roll, pitch, and yaw angles, and returns the formula used, for example, in [79]:

$$\boldsymbol{M} = \begin{bmatrix} 1 & \sin(\theta_{\text{roll}})\tan(\theta_{\text{pitch}}) & \cos(\theta_{\text{roll}})\tan(\theta_{\text{pitch}}) \\ 0 & \cos(\theta_{\text{roll}}) & -\sin(\theta_{\text{roll}}) \\ 0 & \sin(\theta_{\text{roll}})/\cos(\theta_{\text{pitch}}) & \cos(\theta_{\text{roll}})/\cos(\theta_{\text{pitch}}) \end{bmatrix} \tag{F.41}$$

```
from sympy import Matrix, symbols, trigsimp
from sympy import rot_axis1, rot_axis2, rot_axis3

# defining angles as symbolic variables
ang1, ang2, ang3 = symbols(
        '\\theta_{\\text{roll}},'
        '\\theta_{\\text{pitch}},'
        '\\theta_{\\text{yaw}}')

e1 = Matrix([1, 0, 0]) # extrinsic ZYX sequence
e2 = Matrix([0, 1, 0]) # is equivalent to intrinsic xyz
e3 = Matrix([0, 0, 1])

R1 = rot_axis1(-ang1) # rot_axis[1,2,3] are implemented
R2 = rot_axis2(-ang2) # in SymPy as clockwise rotations,
R3 = rot_axis3(-ang3) # hence the negative signs

lamb = e3.dot(R3 * R2 * e1)
if abs(lamb) == 1:
    raise ValueError('Degenerate case')

A = Matrix.hstack(R2*e1, e2, e3).T * R2 * R1

B = Matrix([
        [   1, 0, lamb],
        [   0, 1,    0],
        [lamb, 0,    1]])

M = B.inv() * A
M = trigsimp(M) # simplify with trigonometric identities
```

Listing F.1: SymPy code generating the derivative matrix for roll, pitch, and yaw angles, for non-degenerate cases.

[1] Note that $1 - \Lambda^2 = \sin^2(\theta_2 + \theta_\lambda)$.

G. Validity of disc approximation

In Chapter 5, specifically in Eq. 5.3, it is stated that we use and approximated form for the expression of the rotor's inertia matrix by averaging its inertia matrix around one full rotation around the z-axis. In this section, we will demonstrate this.

G.1. Exact kinetic energy

In order to analyze the real system, we must do the following modifications: First, instead of the virtual frame $\mathcal{F}_P$, we will define everything in the real frame $\mathcal{F}_{P*}$, and the rotation from $\mathcal{F}_{P*}$ to $\mathcal{F}_P$ is: $R^P_{P*} = R_z(\gamma)$, giving the full rotation matrix seen in Appendix F.2. We include a third term in the rotation between $\mathcal{F}_P$ and $\mathcal{F}_B$, replacing R^B_P by $R^B_P R_z(\gamma)$:

$$\begin{aligned} R^B_{P*} &\equiv R^B_P R_\gamma \\ &= R_z(\beta)\, R_y(\alpha)\, R_z(\gamma) \end{aligned} \tag{G.1}$$

And we define $R_\gamma \equiv R_z(\gamma)$ for simplicity. Defining the angular velocity of the virtual frame $\mathcal{F}_P$ w.r.t. $\mathcal{F}_B$ as:

$$\boldsymbol{\omega}^P_{P/B} \equiv \left((R^B_P)^T \frac{\mathrm{d}R^B_P}{\mathrm{d}t} \right)^\vee \tag{G.2}$$

The full expression for the angular velocity of $\mathcal{B}_p$ w.r.t. $\mathcal{B}_b$ in $\mathcal{F}_{P*}$ is[1]:

$$\begin{aligned} \boldsymbol{\Omega}^{P*}_{P/B} &\equiv \left((R^B_{P*})^T \frac{\mathrm{d}R^B_{P*}}{\mathrm{d}t} \right)^\vee \\ &= R^T_\gamma (\boldsymbol{\omega}^P_{P/B} + \dot{\gamma} \boldsymbol{e}_z) \\ &= R^T_\gamma \boldsymbol{\Omega}^P_{P/B} \end{aligned} \tag{G.3}$$

[1] This comes directly from the final solution of Appendix F.2 with $\boldsymbol{\Omega}^{P*}_{P/B} = \boldsymbol{\Omega}_{zyz}$ and $\boldsymbol{\omega}^P_{P/B} = \boldsymbol{\omega}_{zy}$.

And finally, we suppose the propeller's diagonal inertia matrix has three distinct values:

$$J_{p*}^{P*} = \begin{bmatrix} J_{px} & 0 & 0 \\ 0 & J_{py} & 0 \\ 0 & 0 & J_{pz} \end{bmatrix} \tag{G.4}$$

In which, in general, $J_{px} \neq J_{py}$. The new full velocities of the propeller w.r.t. to the Inertial frame on the frame $\mathcal{F}_{P*}$ is:

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\Omega}_P^{P*} \\ \boldsymbol{v}_P^{P*} \end{bmatrix} &= \left(\mathrm{Ad}_B^{P*}\right) \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Omega}_{P/B}^{P*} \\ \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \left(R_P^B R_\gamma\right)^T & 0 \\ -h_P \left(R_P^B R_\gamma\right)^T \widehat{\boldsymbol{e}}_z & \left(R_P^B R_\gamma\right)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \end{bmatrix} + \begin{bmatrix} R_\gamma^T \boldsymbol{\Omega}_{P/B}^P \\ \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} (R_\gamma)^T \left(R_P^B\right)^T & 0 \\ -h_P (R_\gamma)^T \left(R_P^B\right)^T \widehat{\boldsymbol{e}}_z & (R_\gamma)^T \left(R_P^B\right)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \end{bmatrix} + \begin{bmatrix} R_\gamma^T \boldsymbol{\Omega}_{P/B}^P \\ \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} R_\gamma^T & \mathbf{0} \\ \mathbf{0} & R_\gamma^T \end{bmatrix} \left(\begin{bmatrix} \left(R_P^B\right)^T & 0 \\ -h_P \left(R_P^B\right)^T \widehat{\boldsymbol{e}}_z & \left(R_P^B\right)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Omega}_{P/B}^P \\ \mathbf{0} \end{bmatrix} \right) \end{aligned} \tag{G.5}$$

And finally:

$$\begin{bmatrix} \boldsymbol{\Omega}_P^{P*} \\ \boldsymbol{v}_P^{P*} \end{bmatrix} = \left(\mathrm{Ad}_P^{P*}\right) \begin{bmatrix} \boldsymbol{\Omega}_P^P \\ \boldsymbol{v}_P^P \end{bmatrix} \tag{G.6}$$

Where:

$$\left(\mathrm{Ad}_P^{P*}\right) = \begin{bmatrix} R_\gamma^T & \mathbf{0} \\ \mathbf{0} & R_\gamma^T \end{bmatrix} \tag{G.7}$$

Which leads to: And the rotor's exact kinetic energy then is:

$$\begin{aligned} T_{P*} &= \frac{m_p}{2} \left(\boldsymbol{v}_P^{P*}\right)^T \boldsymbol{v}_P^{P*} + \frac{1}{2} \left(\boldsymbol{\Omega}_P^{P*}\right)^T J_{p*}^{P*} \boldsymbol{\Omega}_P^{P*} \\ &= \frac{1}{2} \begin{bmatrix} \boldsymbol{\Omega}_P^{P*} \\ \boldsymbol{v}_P^{P*} \end{bmatrix}^T \begin{bmatrix} J_{p*}^{P*} & \mathbf{0} \\ \mathbf{0} & m_p \mathbb{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Omega}_P^{P*} \\ \boldsymbol{v}_P^{P*} \end{bmatrix} \\ &= \frac{1}{2} \begin{bmatrix} \boldsymbol{\Omega}_P^{P} \\ \boldsymbol{v}_P^{P} \end{bmatrix}^T \left(\mathrm{Ad}_P^{P*}\right)^T \begin{bmatrix} J_{p*}^{P*} & \mathbf{0} \\ \mathbf{0} & m_p \mathbb{I} \end{bmatrix} \left(\mathrm{Ad}_P^{P*}\right) \begin{bmatrix} \boldsymbol{\Omega}_P^{P} \\ \boldsymbol{v}_P^{P} \end{bmatrix} \\ &= \frac{1}{2} \begin{bmatrix} \boldsymbol{\Omega}_P^{P} \\ \boldsymbol{v}_P^{P} \end{bmatrix}^T \begin{bmatrix} R_\gamma & \mathbf{0} \\ \mathbf{0} & R_\gamma \end{bmatrix} \begin{bmatrix} J_{p*}^{P*} & \mathbf{0} \\ \mathbf{0} & m_p \mathbb{I} \end{bmatrix} \begin{bmatrix} R_\gamma^T & \mathbf{0} \\ \mathbf{0} & R_\gamma^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\Omega}_P^{P} \\ \boldsymbol{v}_P^{P} \end{bmatrix} \end{aligned} \tag{G.8}$$

Finally giving:

$$T_{P*} = \frac{1}{2} \begin{bmatrix} \boldsymbol{\Omega}_P^{P} \\ \boldsymbol{v}_P^{P} \end{bmatrix}^T \begin{bmatrix} R_\gamma J_{p*}^{P*} (R_\gamma)^T & \mathbf{0} \\ \mathbf{0} & m_p \mathbb{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Omega}_P^{P} \\ \boldsymbol{v}_P^{P} \end{bmatrix} \tag{G.9}$$

Following the same steps as described in Section 5.2.2, we get that:

$$T_{P*} = \frac{1}{2} \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \\ \boldsymbol{\Omega}_{P/B}^P \end{bmatrix}^T \boldsymbol{M}_{P*} \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \\ \boldsymbol{\Omega}_{P/B}^P \end{bmatrix}$$

$$\boldsymbol{M}_{P*} = \begin{bmatrix} J_{p*}^B - m_p h_P^2 \left(\widehat{\boldsymbol{e}}_z\right)^2 & m_p h_P \widehat{\boldsymbol{e}}_z & R_P^B J_{p*}^P \\ -m_p h_P \widehat{\boldsymbol{e}}_z & m_b \mathbb{I} & \mathbf{0} \\ \left(R_P^B J_{p*}^P\right)^T & \mathbf{0} & J_{p*}^P \end{bmatrix} \tag{G.10}$$

Where $J_{p*}^P = R_\gamma J_{p*}^{P*} (R_\gamma)^T$ and $J_{p*}^B = R_P^B J_{p*}^P (R_P^B)^T = R_P^B R_\gamma J_{p*}^{P*} (R_\gamma)^T (R_P^B)^T$. Finally, the total kinetic energy of the system is:

$$T^* = T_B + T_S + T_{P*} \tag{G.11}$$

G.2. Time approximations

We make two key assumptions in order to find the approximation. First, the mean speed $\dot{\gamma}$ of the rotor is much greater than $\boldsymbol{v}_B^B$, $\boldsymbol{\omega}_B^B$, $\dot{\beta}$ and $\dot{\alpha}$, such that in the time interval $\mathcal{T}$ that γ takes for a complete revolution (from 0 to 2π), all the other variables can be considered constant. Moreover, $\dot{\gamma}$ is considered approximately constant. Inserting Eq. G.11 into 6.2:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T^*}{\partial \eta}\right) - \mathrm{ad}_\eta^T \left(\frac{\partial T^*}{\partial \eta}\right) = F_{\text{ext}}^B \tag{G.12}$$

Our objective is to replace each term of Eq. G.12 by its average on a full revolution of γ. We define the average as:

$$\text{mean}\,(\cdot) = \frac{1}{T}\int_0^T (\cdot) dt \tag{G.13}$$

But since all the terms are approximated as constants in the small interval $T = \frac{2\pi}{\dot{\gamma}}$, an easier way is to define:

$$\text{mean}_\gamma\,(\cdot) = \frac{1}{2\pi}\int_0^{2\pi} (\cdot) d\gamma \tag{G.14}$$

Defining $T = \text{mean}_\gamma\left(T^*\right)$ and applying Eq. G.14 to G.12 leads to:

$$\begin{aligned}
\text{mean}_\gamma\left(\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T^*}{\partial \eta}\right) - \text{ad}_\eta^T\left(\frac{\partial T^*}{\partial \eta}\right)\right) &= \text{mean}_\gamma\left(F_{\text{ext}}^B\right) \\
\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial}{\partial \eta}(\text{mean}_\gamma\left(T^*\right))\right) - \text{ad}_\eta^T\left(\frac{\partial}{\partial \eta}(\text{mean}_\gamma\left(T^*\right))\right) &= F_{\text{ext}}^B \frac{1}{2\pi}\int_0^{2\pi} 1 d\gamma \\
\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T}{\partial \eta}\right) - \text{ad}_\eta^T\left(\frac{\partial T}{\partial \eta}\right) &= F_{\text{ext}}^B
\end{aligned} \tag{G.15}$$

Which is equivalent to Eq. 6.2. Analyzing $\text{mean}_\gamma\left(T^*\right)$, we see that the only terms of T^* that depend on γ is the term T_{P*}:

$$\begin{aligned}
T &= \text{mean}_\gamma\left(T^*\right) \\
&= \text{mean}_\gamma\left(T_B + T_S + T_{P*}\right) \\
&= T_B + T_S + \text{mean}_\gamma\left(T_{P*}\right)
\end{aligned} \tag{G.16}$$

Calculating the mean of the rotor kinetic energy:

$$\begin{aligned}
T_P &= \text{mean}_\gamma\left(T_{P*}\right) \\
&= \frac{1}{2}\begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \\ \boldsymbol{\Omega}_{P/B}^P \end{bmatrix}^T \text{mean}_\gamma\left(\boldsymbol{M}_{P*}\right) \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \\ \boldsymbol{\Omega}_{P/B}^P \end{bmatrix} \\
&= \frac{1}{2}\begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \\ \boldsymbol{\Omega}_{P/B}^P \end{bmatrix}^T \boldsymbol{M}_P \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \\ \boldsymbol{\Omega}_{P/B}^P \end{bmatrix}
\end{aligned} \tag{G.17}$$

Where:

$$\begin{aligned}
\boldsymbol{M}_P &= \text{mean}_\gamma\left(\boldsymbol{M}_{P*}\right) \\
&= \begin{bmatrix} \text{mean}_\gamma\left(J_{p*}^B\right) - m_p h_P^2 \left(\widehat{\boldsymbol{e}}_z\right)^2 & m_p h_P \widehat{\boldsymbol{e}}_z & R_P^B\left[\text{mean}_\gamma\left(J_{p*}^P\right)\right] \\ -m_p h_P \widehat{\boldsymbol{e}}_z & m_b \mathbb{I} & \boldsymbol{0} \\ \left(R_P^B\left[\text{mean}_\gamma\left(J_{p*}^P\right)\right]\right)^T & \boldsymbol{0} & J_{p*}^P \end{bmatrix}
\end{aligned} \tag{G.18}$$

And the only integral we must compute is $\text{mean}_\gamma\left(J_{p*}^P\right) = \frac{1}{2\pi}\int_0^{2\pi} J_{p*}^P d\gamma$. Defining:

$$\begin{aligned}
J_p^P &\equiv \text{mean}_\gamma\left(J_{p*}^P\right) \\
&= \frac{1}{2\pi}\int_0^{2\pi} J_{p*}^P d\gamma \\
&= \frac{1}{2\pi}\int_0^{2\pi} R_\gamma J_{p*}^{P*} R_\gamma^T d\gamma \\
&= \frac{1}{2\pi}\int_0^{2\pi} R_\gamma J_{p*}^{P*} R_\gamma^T d\gamma \\
&= \frac{1}{2\pi}\int_0^{2\pi} \begin{bmatrix} c_\gamma & -s_\gamma & 0 \\ s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} J_{px} & 0 & 0 \\ 0 & J_{py} & 0 \\ 0 & 0 & J_{pz} \end{bmatrix} \begin{bmatrix} c_\gamma & s_\gamma & 0 \\ -s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} d\gamma \\
&= \frac{1}{2\pi}\int_0^{2\pi} \begin{bmatrix} J_{px}c_\gamma^2 + J_{py}s_\gamma^2 & (J_{px} - J_{py})s_\gamma c_\gamma & 0 \\ (J_{px} - J_{py})s_\gamma c_\gamma & J_{px}s_\gamma^2 + J_{py}c_\gamma^2 & 0 \\ 0 & 0 & J_{pz} \end{bmatrix} d\gamma
\end{aligned} \tag{G.19}$$

We can use the following relationships:

$$\begin{aligned}
\int_0^{2\pi} (\sin t)^2 dt = \int_0^{2\pi} (\cos t)^2 dt = \pi \\
\int_0^{2\pi} \sin t \cos t dt = 0
\end{aligned} \tag{G.20}$$

To find that:

$$\begin{aligned}
J_p^P &= \frac{1}{2\pi}\begin{bmatrix} J_{px}\pi + J_{py}\pi & (J_{px} - J_{py})0 & 0 \\ (J_{px} - J_{py})0 & J_{px}\pi + J_{py}\pi & 0 \\ 0 & 0 & 2\pi J_{pz} \end{bmatrix} \\
&= \begin{bmatrix} \frac{J_{px}+J_{py}}{2} & 0 & 0 \\ 0 & \frac{J_{px}+J_{py}}{2} & 0 \\ 0 & 0 & J_{pz} \end{bmatrix}
\end{aligned} \tag{G.21}$$

Moreover, we get:

$$\begin{aligned}
J_p^B &\equiv \text{mean}_\gamma\left(J_{p*}^B\right) \\
&= \text{mean}_\gamma\left(R_P^B J_{p*}^P (R_P^B)^T\right) \\
&= R_P^B\left[\text{mean}_\gamma\left(J_{p*}^P\right)\right](R_P^B)^T \\
&= R_P^B J_p^P (R_P^B)^T
\end{aligned} \tag{G.22}$$

Which proves that, as expected, the disc approximate inertia matrix J_p^P's terms for x and y are identical. We can finally state that:

$$T_P = \frac{1}{2} \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \\ \boldsymbol{\Omega}_{P/B}^P \end{bmatrix}^T \begin{bmatrix} J_p^B - m_p h_P^2 \left(\widehat{\boldsymbol{e}}_z\right)^2 & m_p h_P \widehat{\boldsymbol{e}}_z & R_P^B J_p^P \\ -m_p h_P \widehat{\boldsymbol{e}}_z & m_b \mathbb{I} & \mathbf{0} \\ \left(R_P^B J_p^P\right)^T & \mathbf{0} & J_{p*}^P \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_B^B \\ \boldsymbol{v}_B^B \\ \boldsymbol{\Omega}_{P/B}^P \end{bmatrix} \tag{G.23}$$

H. Derivative of a rotated vector with respect to the rotation quaternion

In this section, we will derive the derivative of a rotated vector in respect to the rotation quaternion. [1]

H.1. Derivative expression

Defining $q = [q_\mathrm{r}, \boldsymbol{q}^T]^T$ a unit rotation quaternion, $\boldsymbol{v} \in \mathbb{R}^3$ and $\boldsymbol{v}'$ the vector $\boldsymbol{v}$ rotated by q, we know that:

$$\tilde{\boldsymbol{v}}' = q \odot \tilde{\boldsymbol{v}} \odot q^* \tag{H.1}$$

Where $\tilde{\boldsymbol{v}} = \begin{bmatrix} 0 \\ \boldsymbol{v} \end{bmatrix}$ and $\tilde{\boldsymbol{v}}' = \begin{bmatrix} 0 \\ \boldsymbol{v}' \end{bmatrix}$. For a parameter t, we know that:

$$\begin{aligned} \frac{\partial \tilde{\boldsymbol{v}}'}{\partial t} &= \frac{\partial}{\partial t}(q \odot \tilde{\boldsymbol{v}} \odot q^*) \\ &= \frac{\partial q}{\partial t} \odot \tilde{\boldsymbol{v}} \odot q^* + q \odot \tilde{\boldsymbol{v}} \odot \frac{\partial q^*}{\partial t} \end{aligned} \tag{H.2}$$

Using the definitions in Eq. A.37, we can rewrite Eq. H.2 in Matrix form:

$$\frac{\partial \tilde{\boldsymbol{v}}'}{\partial t} = \mathcal{R}(\tilde{\boldsymbol{v}} \odot q^*)\frac{\partial q}{\partial t} + \mathcal{L}(q \odot \tilde{\boldsymbol{v}})\frac{\partial q^*}{\partial t} \tag{H.3}$$

And:

$$\frac{\partial \tilde{\boldsymbol{v}}'}{\partial q} = \mathcal{R}(\tilde{\boldsymbol{v}} \odot q^*)\frac{\partial q}{\partial q} + \mathcal{L}(q \odot \tilde{\boldsymbol{v}})\frac{\partial q^*}{\partial q} \tag{H.4}$$

[1] Originally, a version of the EOM was derived using the Euler-Lagrange method and had a quaternion variable as the generalized coordinates vector, and a Passivity-Based controller that depended on the results of this Appendix was derived for it. Since I could not find this derivation elsewhere in the literature, I decided to leave it here, even though it is not used anymore in this work.

We know that:

$$\begin{aligned}\frac{\partial q}{\partial q} &= \mathbb{I}_4 \\ \frac{\partial q^*}{\partial q} &= \mathcal{C}\end{aligned} \tag{H.5}$$

Where, from Eq. A.42:

$$\mathcal{C} = \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & -\mathbb{I} \end{bmatrix} \tag{H.6}$$

Which gives:

$$\frac{\partial \tilde{\boldsymbol{v}}'}{\partial q} = \mathcal{R}(\tilde{\boldsymbol{v}} \odot q^*) + \mathcal{L}(q \odot \tilde{\boldsymbol{v}})\,\mathcal{C} \tag{H.7}$$

Using Eq. A.43, we know that:

$$\begin{aligned}\mathcal{L}(q \odot \tilde{\boldsymbol{v}}) &= \mathcal{C}\,\mathcal{R}\left((q \odot \tilde{\boldsymbol{v}})^*\right)\,\mathcal{C} \\ &= -\mathcal{C}\,\mathcal{R}\left(\tilde{\boldsymbol{v}} \odot q^*\right)\,\mathcal{C}\end{aligned} \tag{H.8}$$

Adding that to Eq. H.7, and noting that $\mathcal{C}^2 = \mathbb{I}_4$:

$$\begin{aligned}\frac{\partial \tilde{\boldsymbol{v}}'}{\partial q} &= \mathcal{R}(\tilde{\boldsymbol{v}} \odot q^*) - \mathcal{C}\,\mathcal{R}\left(\tilde{\boldsymbol{v}} \odot q^*\right) \\ &= \left(\mathbb{I}_4 - \mathcal{C}\right)\mathcal{R}(\tilde{\boldsymbol{v}} \odot q^*) \\ &= \left(\begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \mathbb{I} \end{bmatrix} - \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & -\mathbb{I} \end{bmatrix}\right)\mathcal{R}(\tilde{\boldsymbol{v}} \odot q^*)\end{aligned} \tag{H.9}$$

And finally:

$$\frac{\partial \tilde{\boldsymbol{v}}'}{\partial q} = 2\begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \mathbb{I} \end{bmatrix}\mathcal{R}(\tilde{\boldsymbol{v}} \odot q^*) \tag{H.10}$$

And its transpose is:

$$\begin{aligned}\left(\frac{\partial \tilde{\boldsymbol{v}}'}{\partial q}\right)^T &= 2\mathcal{R}\left((\tilde{\boldsymbol{v}} \odot q^*)^*\right)\begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \mathbb{I} \end{bmatrix} \\ &= -2\mathcal{R}\left(q \odot \tilde{\boldsymbol{v}}\right)\begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \mathbb{I} \end{bmatrix}\end{aligned} \tag{H.11}$$

H.2. Dot product

Supposing a vector $\boldsymbol{w} \in \mathbb{R}^3$ and $\tilde{\boldsymbol{w}} = (0, \boldsymbol{w}^T)^T$, the dot product $\boldsymbol{v}' \cdot \boldsymbol{w}$ can be rewritten as:

$$\begin{aligned} \boldsymbol{v}' \cdot \boldsymbol{w} &= \begin{bmatrix} 0 \\ \boldsymbol{v}' \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \boldsymbol{w} \end{bmatrix} \\ &= \tilde{\boldsymbol{v}}' \cdot \tilde{\boldsymbol{w}} \end{aligned} \tag{H.12}$$

Using Eq. A.24:

$$\begin{aligned} \frac{\partial}{\partial q}(\boldsymbol{v}' \cdot \boldsymbol{w}) &= \frac{\partial}{\partial q}(\tilde{\boldsymbol{v}}' \cdot \tilde{\boldsymbol{w}}) \\ &= \left(\frac{\partial \tilde{\boldsymbol{v}}'}{\partial q}\right)^T \tilde{\boldsymbol{w}} \end{aligned} \tag{H.13}$$

According to Eq. H.11:

$$\begin{aligned} \frac{\partial}{\partial q}(\boldsymbol{v}' \cdot \boldsymbol{w}) &= -2\mathcal{R}(q \odot \tilde{\boldsymbol{v}})\, \tilde{\boldsymbol{w}} \\ &= -2\, \tilde{\boldsymbol{w}} \odot q \odot \tilde{\boldsymbol{v}} \end{aligned} \tag{H.14}$$

I. Inverse of B_τ

The matrix $\boldsymbol{B}_\tau = (k\mathbb{I} + h_P\,\widehat{\boldsymbol{e}}_z)$ is of the form:

$$\boldsymbol{M} = a\mathbb{I} + \widehat{\boldsymbol{b}} = \begin{bmatrix} a & -b_z & b_y \\ b_z & a & -b_x \\ -b_y & b_x & a \end{bmatrix} \tag{I.1}$$

With $a = k$ and $\boldsymbol{b} = h_P\,\boldsymbol{e}_z$. Any matrix $\boldsymbol{M}$ of the form in Eq. I.1, with $a \in \mathbb{R}$, $a \neq 0$ and $\boldsymbol{b} \in \mathbb{R}^3$ has an inverse of the form:

$$\boldsymbol{M}^{-1} = \frac{a\boldsymbol{M}^T + \boldsymbol{b}\boldsymbol{b}^T}{a(a^2 + |\boldsymbol{b}|^2)} \tag{I.2}$$

To prove this, we can multiply Eq. I.1 and Eq. I.2:

$$\begin{aligned} \boldsymbol{M}\boldsymbol{M}^{-1} &= \left(a\mathbb{I} + \widehat{\boldsymbol{b}}\right) \frac{\left(a\boldsymbol{M}^T + \boldsymbol{b}\boldsymbol{b}^T\right)}{a(a^2 + |\boldsymbol{b}|^2)} \\ &= \left(a\mathbb{I} + \widehat{\boldsymbol{b}}\right) \frac{\left(a^2\mathbb{I} - a\widehat{\boldsymbol{b}} + \boldsymbol{b}\boldsymbol{b}^T\right)}{a(a^2 + |\boldsymbol{b}|^2)} \\ &= \frac{\left(a^2\mathbb{I} + \boldsymbol{b}\boldsymbol{b}^T - \widehat{\boldsymbol{b}}^2\right)}{a^2 + |\boldsymbol{b}|^2} \end{aligned} \tag{I.3}$$

And since $\boldsymbol{b}\boldsymbol{b}^T - \widehat{\boldsymbol{b}}^2 = |\boldsymbol{b}|^2\mathbb{I}$:

$$\begin{aligned} \boldsymbol{M}\boldsymbol{M}^{-1} &= \frac{(a^2\mathbb{I} + |\boldsymbol{b}|^2\mathbb{I})}{a^2 + |\boldsymbol{b}|^2} \\ &= \frac{a^2 + |\boldsymbol{b}|^2}{a^2 + |\boldsymbol{b}|^2}\mathbb{I} \\ &= \mathbb{I} \end{aligned} \tag{I.4}$$

J. Non-uniqueness of Euler and Davenport angles

For a quaternion q, and three unit axes $\boldsymbol{e}_i$, $\boldsymbol{e}_j$ and $\boldsymbol{e}_k$, we have seen that, if $\boldsymbol{e}_i \cdot \boldsymbol{e}_j = \boldsymbol{e}_j \cdot \boldsymbol{e}_k = 0$, Eq. 10.41 shows that we can decompose q as:

$$q = \begin{bmatrix} c_3 \\ s_3 \boldsymbol{e}_k \end{bmatrix} \odot \begin{bmatrix} c_2 \\ s_2 \boldsymbol{e}_j \end{bmatrix} \odot \begin{bmatrix} c_1 \\ s_1 \boldsymbol{e}_i \end{bmatrix} \tag{J.1}$$

Where:

$$\begin{aligned} s_1 &\equiv \sin(\theta_1/2), \; c_1 \equiv \cos(\theta_1/2) \\ s_2 &\equiv \sin(\theta_2/2), \; c_2 \equiv \cos(\theta_2/2) \\ s_3 &\equiv \sin(\theta_3/2), \; c_3 \equiv \cos(\theta_3/2) \end{aligned} \tag{J.2}$$

Moreover, for $\theta_\lambda = \operatorname{atan2}\left(\boldsymbol{e}_k \cdot (\boldsymbol{e}_i \times \boldsymbol{e}_j), \boldsymbol{e}_k \cdot \boldsymbol{e}_i\right)$, Eq. 10.51 shows that by defining:

$$\lambda = \begin{bmatrix} \cos(\theta_\lambda/2) \\ \sin(\theta_\lambda/2) \boldsymbol{e}_j \end{bmatrix} \tag{J.3}$$

We can rewrite Eq. J.1 as Eq. 10.53:

$$\begin{aligned} q' = \lambda \odot q &= \begin{bmatrix} c_3 \\ s_3 \boldsymbol{e}_i \end{bmatrix} \odot \begin{bmatrix} c_2' \\ s_2' \boldsymbol{e}_j \end{bmatrix} \odot \begin{bmatrix} c_1 \\ s_1 \boldsymbol{e}_i \end{bmatrix} \\ s_2' \equiv \sin((\theta_2 + \theta_\lambda)/2), \; & c_2' \equiv \cos((\theta_2 + \theta_\lambda)/2) \end{aligned} \tag{J.4}$$

Defining $\boldsymbol{q}'$ as the vector part of q, we obtain, as seen in Eqs. 10.10 and 10.11:

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \equiv \begin{bmatrix} q_\mathrm{r}' \\ \boldsymbol{q}' \cdot \boldsymbol{e}_i \\ \boldsymbol{q}' \cdot \boldsymbol{e}_j \\ \boldsymbol{q}' \cdot (\boldsymbol{e}_i \times \boldsymbol{e}_j) \end{bmatrix} = \begin{bmatrix} c_2' c_+ \\ c_2' s_+ \\ s_2' c_- \\ s_2' s_- \end{bmatrix} \tag{J.5}$$

Then, as seen in Eq. 10.13:

$$\begin{aligned} z_+ \equiv a + ib &= c_2'(c_+ + i s_+) = c_2' \exp(i\theta_+) \\ z_- \equiv c + id &= s_2'(c_- + i s_-) = s_2' \exp(i\theta_-) \end{aligned} \tag{J.6}$$

Where:

$$\begin{aligned}\theta_+ &= (\theta_1 + \theta_3)/2 \\ \theta_- &= (\theta_1 - \theta_3)/2\end{aligned} \tag{J.7}$$

The only added assumption done in the last sections was to suppose $c_2' > 0$ and $s_2' > 0$. In the following subsections, we will consider every possible case.

J.1. Case 1: Positive-Positive

Assuming $c_2' > 0$ and $s_2' > 0$ leads to:

$$\begin{aligned}|z_+| &= c_2' \\ |z_-| &= s_2' \\ \arg(z_+) &= \theta_+ \\ \arg(z_-) &= \theta_-\end{aligned} \tag{J.8}$$

Which can be rewritten as:

$$\begin{aligned}c_2' &= \sqrt{a^2 + b^2} \\ s_2' &= \sqrt{c^2 + d^2} \\ \theta_+ &= \operatorname{atan2}(b, a) \\ \theta_- &= \operatorname{atan2}(d, c)\end{aligned} \tag{J.9}$$

Which leads to the original solutions:

$$\begin{aligned}\theta_1 &= \theta_+ - \theta_- \\ \theta_2 &= 2\operatorname{atan2}(|z_-|, |z_+|) - \theta_\lambda \\ \theta_3 &= \theta_+ + \theta_-\end{aligned} \tag{J.10}$$

Or:

$$\begin{aligned}\theta_1 &= \operatorname{atan2}(b, a) - \operatorname{atan2}(d, c) \\ \theta_2 &= 2\operatorname{atan2}(\sqrt{c^2 + d^2}, \sqrt{a^2 + b^2}) - \theta_\lambda \\ \theta_3 &= \operatorname{atan2}(b, a) + \operatorname{atan2}(d, c)\end{aligned} \tag{J.11}$$

J.2. Case 2: Positive-Negative

Assuming $c_2' > 0$ and $s_2' < 0$:

$$z_- \equiv c + id = s_2'(c_- + is_-) = s_2' \exp(i\theta_-) \tag{J.12}$$

We know that $-s_2'$ is positive, so:

$$z_- = (-s_2')(-(c_- + is_-)) \tag{J.13}$$

Its absolute value is:

$$\begin{aligned} s_2' &= -|z_-| \\ &= -\sqrt{c^2 + d^2} \end{aligned} \tag{J.14}$$

Using Eq. J.14 to calculate θ_2 now leads to:

$$\begin{aligned} \theta_2 + \theta_\lambda &= 2\,\mathrm{atan2}(-|z_-|, |z_+|) \\ &= -2\,\mathrm{atan2}(|z_-|, |z_+|) \\ &= -2\,\mathrm{atan2}(\sqrt{c^2 + d^2}, \sqrt{a^2 + b^2}) \end{aligned} \tag{J.15}$$

Moreover, the phase of z_- is:

$$\begin{aligned} \arg z_- &= \arg(-(c_- + is_-)) \\ &= \arg((c_- + is_-)(-1 + i0)) \\ &= \arg(c_- + is_-) + \arg(-1 + i0) \\ &= \theta_- + \pi \end{aligned} \tag{J.16}$$

Leading to:

$$\begin{aligned} \theta_- &= \arg z_- - \pi \\ &= \mathrm{atan2}(d, c) - \pi \end{aligned} \tag{J.17}$$

Putting it all together:

$$\begin{aligned} \theta_1 &= \mathrm{atan2}(b, a) - \mathrm{atan2}(d, c) + \pi \\ \theta_2 &= -2\,\mathrm{atan2}(\sqrt{c^2 + d^2}, \sqrt{a^2 + b^2}) - \theta_\lambda \\ \theta_3 &= \mathrm{atan2}(b, a) + \mathrm{atan2}(d, c) - \pi \end{aligned} \tag{J.18}$$

Note that both θ_1 and θ_3 have a difference of π compared to case **PP**, and θ_2 has a change in sign.

J.3. Case 3: Negative-Positive

In this case, assuming $c_2' < 0$ and $s_2' > 0$:

$$z_+ \equiv a + ib = c_2'(c_+ + is_+) = c_2' \exp(i\theta_+) \tag{J.19}$$

We know that $-c_2'$ is positive, so:

$$z_+ = (-c_2')(-(c_+ + is_+)) \tag{J.20}$$

Its absolute value is:

$$\begin{aligned} c_2' &= -|z_+| \\ &= -\sqrt{a^2 + b^2} \end{aligned} \tag{J.21}$$

Using Eq. J.21 to calculate θ_2 now leads to:

$$\begin{aligned} \theta_2 + \theta_\lambda &= 2\,\mathrm{atan2}(|z_-|, -|z_+|) \\ &= 2\,(\pi - \mathrm{atan2}(|z_-|, |z_+|)) \\ &= 2\pi - 2\,\mathrm{atan2}(\sqrt{c^2 + d^2}, \sqrt{a^2 + b^2}) \end{aligned} \tag{J.22}$$

For the phase, similarly:

$$\begin{aligned} \arg z_+ &= \arg(-(c_+ + is_+)) \\ &= \theta_+ + \pi \end{aligned} \tag{J.23}$$

And:

$$\begin{aligned} \theta_+ &= \arg z_+ - \pi \\ &= \mathrm{atan2}(b, a) - \pi \end{aligned} \tag{J.24}$$

Putting it all together:

$$\begin{aligned} \theta_1 &= \mathrm{atan2}(b, a) - \mathrm{atan2}(d, c) - \pi \\ \theta_2 &= -2\,\mathrm{atan2}(\sqrt{c^2 + d^2}, \sqrt{a^2 + b^2}) - \theta_\lambda + 2\pi \\ \theta_3 &= \mathrm{atan2}(b, a) + \mathrm{atan2}(d, c) - \pi \end{aligned} \tag{J.25}$$

Note, however, that if we consider our angles between $[0, 2\pi]$ or $[-\pi, \pi]$ (which is the case, in order to use the atan2 function), the additional 2π term can be ignored. Moreover, both adding and subtracting a term of π have the exact same effect. This means that the solutions for cases **PN** and **NP** are equivalent.

J.4. Case 4: Negative-Negative

In this case, we assume $c_2' < 0$ and $s_2' < 0$ and combine results from both cases **PN** and **NP**. The absolute values are:

$$\begin{aligned} c_2' &= -|z_+| = -\sqrt{a^2 + b^2} \\ s_2' &= -|z_-| = -\sqrt{c^2 + d^2} \end{aligned} \tag{J.26}$$

Using Eq. J.26 to calculate θ_2 now leads to:

$$\begin{aligned} \theta_2 + \theta_\lambda &= 2\,\mathrm{atan2}(-|z_-|, -|z_+|) \\ &= 2\,(\pi + \mathrm{atan2}(|z_-|, |z_+|)) \\ &= 2\pi + 2\,\mathrm{atan2}(\sqrt{c^2 + d^2}, \sqrt{a^2 + b^2}) \end{aligned} \tag{J.27}$$

For the phases, similarly:

$$\begin{aligned} \arg z_+ &= \theta_+ + \pi \\ \arg z_- &= \theta_- + \pi \end{aligned} \tag{J.28}$$

Or:

$$\begin{aligned} \theta_+ &= \mathrm{atan2}(b, a) - \pi \\ \theta_- &= \mathrm{atan2}(d, c) - \pi \end{aligned} \tag{J.29}$$

Putting it all together:

$$\begin{aligned} \theta_1 &= \mathrm{atan2}(b, a) - \mathrm{atan2}(d, c) \\ \theta_2 &= 2\,\mathrm{atan2}(\sqrt{c^2 + d^2}, \sqrt{a^2 + b^2}) - \theta_\lambda + 2\pi \\ \theta_3 &= \mathrm{atan2}(b, a) + \mathrm{atan2}(d, c) - 2\pi \end{aligned} \tag{J.30}$$

Note that, again, the additional 2π terms can be ignored. This means that the solutions for cases **PP** and **NN** are equivalent.

J.5. Comparison between both sets

As seen in the last sections, from the 4 possibilities, only two of them are unique:

1. The original solution, equivalent to both cases **PP** and **NN**, given by:

$$\begin{aligned}\theta_1 &= \operatorname{atan2}(b,a) - \operatorname{atan2}(d,c)\\ \theta_2 &= 2\operatorname{atan2}(\sqrt{c^2+d^2},\sqrt{a^2+b^2}) - \theta_\lambda\\ \theta_3 &= \operatorname{atan2}(b,a) + \operatorname{atan2}(d,c)\end{aligned} \tag{J.31}$$

2. An alternative solution, equivalent to both cases **PN** and **NP**, given by:

$$\begin{aligned}\phi_1 &= \operatorname{atan2}(b,a) - \operatorname{atan2}(d,c) - \pi\\ \phi_2 &= -2\operatorname{atan2}(\sqrt{c^2+d^2},\sqrt{a^2+b^2}) - \theta_\lambda\\ \phi_3 &= \operatorname{atan2}(b,a) + \operatorname{atan2}(d,c) - \pi\end{aligned} \tag{J.32}$$

We can convert from one set to the other with:

$$\begin{aligned}\phi_1 &= \theta_1 \pm \pi\\ \phi_2 + \theta_\lambda &= -(\theta_2 + \theta_\lambda)\\ \phi_3 &= \theta_3 \pm \pi\end{aligned} \tag{J.33}$$

We define $\varepsilon_1 = \pm 1$ and $\varepsilon_3 = \pm 1$. By noticing that:

$$\begin{bmatrix}\cos((\theta+\varepsilon\pi)/2)\\ \sin((\theta+\varepsilon\pi)/2)\boldsymbol{e}_i\end{bmatrix} = \begin{bmatrix}\cos(\theta/2)\\ \sin(\theta/2)\boldsymbol{e}_i\end{bmatrix} \odot \begin{bmatrix}\cos(\varepsilon\pi/2)\\ \sin(\varepsilon\pi/2)\boldsymbol{e}_i\end{bmatrix} = \begin{bmatrix}\cos(\theta/2)\\ \sin(\theta/2)\boldsymbol{e}_i\end{bmatrix} \odot \begin{bmatrix}0\\ \varepsilon\boldsymbol{e}_i\end{bmatrix} \tag{J.34}$$

Where $\begin{bmatrix}0\\ \varepsilon\boldsymbol{e}_i\end{bmatrix}$, independent of the sign of ε, is a rotation of π radians around the axis $\boldsymbol{e}_i$. We can insert Eq. J.33 into Eq. J.4 to show how they are equivalent:

$$\begin{aligned}q' &= \begin{bmatrix}\cos((\theta_3+\varepsilon_3\pi)/2)\\ \sin((\theta_3+\varepsilon_3\pi)/2)\boldsymbol{e}_i\end{bmatrix} \odot \begin{bmatrix}\cos(-(\theta_2+\theta_\lambda)/2)\\ \sin(-(\theta_2+\theta_\lambda)/2)\boldsymbol{e}_j\end{bmatrix} \odot \begin{bmatrix}\cos((\theta_1+\varepsilon_1\pi)/2)\\ \sin((\theta_1+\varepsilon_1\pi)/2)\boldsymbol{e}_i\end{bmatrix}\\ &= -\varepsilon_1\varepsilon_3 \begin{bmatrix}c_3\\ s_3\boldsymbol{e}_i\end{bmatrix} \odot \begin{bmatrix}0\\ \boldsymbol{e}_i\end{bmatrix} \odot \begin{bmatrix}c_2'\\ s_2'(-\boldsymbol{e}_j)\end{bmatrix} \odot \begin{bmatrix}0\\ \boldsymbol{e}_i\end{bmatrix}^* \odot \begin{bmatrix}c_1\\ s_1\boldsymbol{e}_i\end{bmatrix}\end{aligned} \tag{J.35}$$

Since $\boldsymbol{e}_i \cdot \boldsymbol{e}_j = 0$, we know that:

$$\begin{bmatrix}0\\ \boldsymbol{e}_i\end{bmatrix} \odot \begin{bmatrix}c_2'\\ s_2'(-\boldsymbol{e}_j)\end{bmatrix} \odot \begin{bmatrix}0\\ \boldsymbol{e}_i\end{bmatrix}^* = \begin{bmatrix}c_2'\\ s_2'\boldsymbol{e}_j\end{bmatrix} \tag{J.36}$$

And since $-\varepsilon_1\varepsilon_3 = \pm 1$, we find:

$$q' = \pm \begin{bmatrix} c_3 \\ s_3 \boldsymbol{e}_i \end{bmatrix} \odot \begin{bmatrix} c'_2 \\ s'_2 \boldsymbol{e}_j \end{bmatrix} \odot \begin{bmatrix} c_1 \\ s_1 \boldsymbol{e}_i \end{bmatrix} \tag{J.37}$$

Which show the equivalency, since both $+q$ and $-q$ represent the same rotation.

J.6. Using the general Davenport formula to compute Tait-Bryan angles

If the sequence has odd parity, we know that $\varepsilon = \boldsymbol{e}_k \cdot (\boldsymbol{e}_i \times \boldsymbol{e}_j) = -1$. To calculate the Euler angles in the YXZ sequence, for example, we suppose $\boldsymbol{e}_i = \boldsymbol{e}_y$, $\boldsymbol{e}_j = \boldsymbol{e}_x$ and $\boldsymbol{e}_k = \boldsymbol{e}_z$.

We can calculate θ_λ as:

$$\begin{aligned} \theta_\lambda &= \operatorname{atan2}\left(\boldsymbol{e}_k \cdot (\boldsymbol{e}_i \times \boldsymbol{e}_j), \boldsymbol{e}_k \cdot \boldsymbol{e}_i\right) \\ &= \operatorname{atan2}\left(-1, 0\right) = -\pi/2 \end{aligned} \tag{J.38}$$

If we use the formula from the cases **PP** and **NN**, we find:

$$\begin{aligned} \theta_2 &= 2\operatorname{atan2}(\sqrt{c^2 + d^2}, \sqrt{a^2 + b^2}) - \theta_\lambda \\ &= 2\operatorname{atan2}(\sqrt{c^2 + d^2}, \sqrt{a^2 + b^2}) + \pi/2 \end{aligned} \tag{J.39}$$

In this case, θ_2 will belong to the interval $[\pi/2, 3\pi/2]$. This is not what is usually expected: Normally, Tait-Bryan angles are in the interval $[-\pi/2, +\pi/2]$. The alternative set of angles, though, give the correct range: $[-3\pi/2, \pi/2]$ which is equivalent to $[-\pi/2, +\pi/2]$.

A practical way of implementing the Euler angles with the general Davenport formula is the following algorithm:

1. Calculate θ_λ.
2. If θ_λ is negative, replace $\boldsymbol{e}_j$ by $-\boldsymbol{e}_j$ (and consequently, θ_λ by $-\theta_\lambda$) and set a flag: `invert = True`.
3. Calculate the angles with the original formula.
4. If `invert == True`, replace the solution θ_2 by $-\theta_2$.

This algorithm gives a fast and concise way of always having the simplest solution without needing to implement both formulas.

K. Analyzing the singularities of proper Euler angles: an alternative set of angles

From the 12 possible sequences for Euler angles decomposition, 6 of them are called the Proper Euler angles: those with a symmetric sequence, such as ZYZ, YXY, etc. These angles can be very useful, but having the first and third axes align leads to a very unfortunate singularity: one that is located at the origin of the second angle. Here, I will show how this singularity arises from an unnatural definition on these angles. I will propose a more natural and arguably geometrically more meaningful definition.

Moreover, this definition leads to a connection with the spin decomposition presented in Chapter 7. Particularly, we show a direct relationship between Proper Euler angles and the spin angle given by Eq. 7.34:

$$\theta_s = 2\ \mathrm{atan2}(\boldsymbol{q} \cdot \boldsymbol{e},\ q_\mathrm{r}) \tag{K.1}$$

K.1. Singularity problem

Define R the rotation matrix representing the total rotation of some body, and q the equivalent rotation quaternion. We define[1]:

- The function $\boldsymbol{\phi}_{xyz} : \mathbb{R}^{3\times3} \to \mathbb{R}^3$ as the decomposition of a rotation in the sequence XYZ, such that $\boldsymbol{\phi}_{xyz}\{R\} = (\phi_1, \phi_2, \phi_3)$ and:

$$\begin{aligned} R &= R_z(\phi_3)\, R_y(\phi_2)\, R_x(\phi_1) \\ q &= \begin{bmatrix} c_{(\phi_3/2)} \\ s_{(\phi_3/2)}\boldsymbol{e}_z \end{bmatrix} \odot \begin{bmatrix} c_{(\phi_2/2)} \\ s_{(\phi_2/2)}\boldsymbol{e}_y \end{bmatrix} \odot \begin{bmatrix} c_{(\phi_1/2)} \\ s_{(\phi_1/2)}\boldsymbol{e}_x \end{bmatrix} \end{aligned} \tag{K.2}$$

- The function $\boldsymbol{\theta}_{zyz} : \mathbb{R}^{3\times3} \to \mathbb{R}^3$ as the decomposition of a rotation in the

[1] Note that in this section, we use **extrinsic** rotations, meaning that the first rotation is the rightmost rotation matrix / quaternion.

sequence ZYZ, such that $\boldsymbol{\theta}_{zyz}\{R\} = (\theta_1, \theta_2, \theta_3)$ and:

$$\begin{aligned} R &= R_z(\theta_3)\, R_y(\theta_2)\, R_z(\theta_1) \\ q &= \begin{bmatrix} c_{(\theta_3/2)} \\ s_{(\theta_3/2)}\boldsymbol{e}_z \end{bmatrix} \odot \begin{bmatrix} c_{(\theta_2/2)} \\ s_{(\theta_2/2)}\boldsymbol{e}_y \end{bmatrix} \odot \begin{bmatrix} c_{(\theta_1/2)} \\ s_{(\theta_1/2)}\boldsymbol{e}_z \end{bmatrix} \end{aligned} \tag{K.3}$$

To represent a simple pitch rotation $R_y(\psi_{\text{pitch}})$, the angles are the same in both representations:

$$\begin{aligned} \boldsymbol{\phi}_{xyz}\{R_y(\psi_{\text{pitch}})\} &= (0, \psi_{\text{pitch}}, 0) \\ \boldsymbol{\theta}_{zyz}\{R_y(\psi_{\text{pitch}})\} &= (0, \psi_{\text{pitch}}, 0) \end{aligned} \tag{K.4}$$

Decomposing a simple roll rotation $R_x(\psi_{\text{roll}})$ is also straightforward in the XYZ sequence. However, in the ZYZ sequence this is slightly more complicated:

1. First, a rotation of $-\frac{\pi}{2}$ around the z-axis is needed in order to align the body's y-axis with the x-axis of the inertial frame.
2. Then, a rotation of ψ_{roll} around the (rotated) y-axis is applied.
3. Finally, the first rotation is reversed with a rotation of $\frac{\pi}{2}$ around the z-axis.

This gives:

$$\begin{aligned} \boldsymbol{\phi}_{xyz}\{R_x(\psi_{\text{roll}})\} &= (\psi_{\text{roll}}, 0, 0) \\ \boldsymbol{\theta}_{zyz}\{R_x(\psi_{\text{roll}})\} &= \left(\frac{\pi}{2}, \psi_{\text{roll}}, -\frac{\pi}{2}\right) \end{aligned} \tag{K.5}$$

Adding a final yaw rotation $R_z(\psi_{\text{yaw}})$ gives:

$$\begin{aligned} \boldsymbol{\phi}_{xyz}\{R_z(\psi_{\text{yaw}})R_x(\psi_{\text{roll}})\} &= (\psi_{\text{roll}}, 0, \psi_{\text{yaw}}) \\ \boldsymbol{\theta}_{zyz}\{R_z(\psi_{\text{yaw}})R_x(\psi_{\text{roll}})\} &= \left(\frac{\pi}{2}, \psi_{\text{roll}}, -\frac{\pi}{2} + \psi_{\text{yaw}}\right) \end{aligned} \tag{K.6}$$

What is arguably even worse, a pure yaw rotation $R_z(\psi_{\text{yaw}})$ is not completely defined in ZYZ without any extra assumption. As seen in Chapter 10, this happens because when $\theta_2 = 0$, there is a singularity in this decomposition, and the only **uniquely**[2] defined value is:

$$\theta_1 + \theta_3 = \psi_{\text{yaw}} \tag{K.7}$$

[2] In the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

Adding, arbitrarily, the extra assumption that $\theta_1 \equiv b$[3], we can state for any value of b that:

$$\begin{aligned} \boldsymbol{\phi}_{xyz}\left\{R_z(\psi_{\text{yaw}})\right\} &= (0, 0, \psi_{\text{yaw}}) \\ \boldsymbol{\theta}_{zyz}\left\{R_z(\psi_{\text{yaw}})\right\} &= (b, 0, -b + \psi_{\text{yaw}}) \end{aligned} \tag{K.8}$$

This hints at an unnecessary coupling between the first and third angles in proper Euler sequences.

K.2. Rotation around a rotated axis

We can generalize both the simple roll and the simple pitch rotation in a Proper sequence (like the ZYZ sequence) in the following way: Suppose an axis $\boldsymbol{e}_j$ in which we want to rotate the body with an angle of ψ. This rotation is represented by the quaternion $\begin{bmatrix} c_{(\psi/2)} \\ s_{(\psi/2)}\boldsymbol{e}_j \end{bmatrix}$.

Now, define $\boldsymbol{e}'_j$ as a rotated version $\boldsymbol{e}_j$ around axis $\boldsymbol{e}_i$ by an angle $-\theta_1$, where $\boldsymbol{e}_i \cdot \boldsymbol{e}_j = 0$. We can state that:

$$\begin{bmatrix} 0 \\ \boldsymbol{e}'_j \end{bmatrix} = \begin{bmatrix} c_{(\theta_1/2)} \\ s_{(\theta_1/2)}\boldsymbol{e}_i \end{bmatrix}^* \odot \begin{bmatrix} 0 \\ \boldsymbol{e}_j \end{bmatrix} \odot \begin{bmatrix} c_{(\theta_1/2)} \\ s_{(\theta_1/2)}\boldsymbol{e}_i \end{bmatrix} \tag{K.9}$$

The quaternion q' representing the full rotation of an angle θ_2 around the rotated axis $\boldsymbol{e}'_j$ is given by:

$$\begin{aligned} q' &= \begin{bmatrix} c_{(\theta_2/2)} \\ s_{(\theta_2/2)}\boldsymbol{e}'_j \end{bmatrix} \\ &= c_{(\theta_2/2)} \begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} + s_{(\theta_2/2)} \begin{bmatrix} 0 \\ \boldsymbol{e}'_j \end{bmatrix} \\ &= c_{(\theta_2/2)} \begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} + s_{(\theta_2/2)} \begin{bmatrix} c_{(\theta_1/2)} \\ s_{(\theta_1/2)}\boldsymbol{e}_i \end{bmatrix}^* \odot \begin{bmatrix} 0 \\ \boldsymbol{e}_j \end{bmatrix} \odot \begin{bmatrix} c_{(\theta_1/2)} \\ s_{(\theta_1/2)}\boldsymbol{e}_i \end{bmatrix} \\ &= \begin{bmatrix} c_{(\theta_1/2)} \\ s_{(\theta_1/2)}\boldsymbol{e}_i \end{bmatrix}^* \odot \left(c_{(\theta_2/2)} \begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} + s_{(\theta_2/2)} \begin{bmatrix} 0 \\ \boldsymbol{e}_j \end{bmatrix} \right) \odot \begin{bmatrix} c_{(\theta_1/2)} \\ s_{(\theta_1/2)}\boldsymbol{e}_i \end{bmatrix} \\ &= \begin{bmatrix} c_{(\theta_1/2)} \\ s_{(\theta_1/2)}\boldsymbol{e}_i \end{bmatrix}^* \odot \begin{bmatrix} c_{(\theta_2/2)} \\ s_{(\theta_2/2)}\boldsymbol{e}_j \end{bmatrix} \odot \begin{bmatrix} c_{(\theta_1/2)} \\ s_{(\theta_1/2)}\boldsymbol{e}_i \end{bmatrix} \end{aligned} \tag{K.10}$$

[3]Usually the chosen value is $b = 0$.

And the equivalent rotation matrix R' is[4]:

$$\begin{aligned} R' &= \left(R_{\theta_1 \boldsymbol{e}_i}\right)^T R_{\theta_2 \boldsymbol{e}_j} R_{\theta_1 \boldsymbol{e}_i} \\ &= R_{-\theta_1 \boldsymbol{e}_i} R_{\theta_2 \boldsymbol{e}_j} R_{\theta_1 \boldsymbol{e}_i} \end{aligned} \tag{K.11}$$

And defining $\boldsymbol{\theta}_{iji} : \mathbb{R}^{3\times3} \to \mathbb{R}^3$ the decomposition of a rotation R in the Euler sequence IJI, we know that:

$$\boldsymbol{\theta}_{iji}\left\{R'\right\} = (\theta_1, \theta_2, -\theta_1) \tag{K.12}$$

Supposing that the sequence IJI is equivalent to the sequence ZYZ, if $\theta_1 = 0$ we find Eq. K.4 and if $\theta_1 = -\frac{\pi}{2}$ we find Eq. K.5

K.3. Definition of a new set of angles

We redefine, more generally, $\boldsymbol{\theta} : \mathbb{R}^{3\times3} \to \mathbb{R}^3$, the decomposition of a rotation in any Proper sequence, such that $\boldsymbol{\theta}\left\{R\right\} = (\theta_1, \theta_2, \theta_3)$ and:

$$\begin{aligned} R &= R_{\theta_3 \boldsymbol{e}}\, R_{\theta_2 \boldsymbol{e}'}\, R_{\theta_1 \boldsymbol{e}} \\ q &= \begin{bmatrix} c_{(\theta_3/2)} \\ s_{(\theta_3/2)}\boldsymbol{e} \end{bmatrix} \odot \begin{bmatrix} c_{(\theta_2/2)} \\ s_{(\theta_2/2)}\boldsymbol{e}' \end{bmatrix} \odot \begin{bmatrix} c_{(\theta_1/2)} \\ s_{(\theta_1/2)}\boldsymbol{e} \end{bmatrix} \end{aligned} \tag{K.13}$$

Inspired by Eq. K.7, we define $\tilde{\theta}_3$ such that:

$$\begin{aligned} \tilde{\theta}_3 &= \theta_1 + \theta_3 \\ \theta_3 &= \tilde{\theta}_3 - \theta_1 \end{aligned} \tag{K.14}$$

Inserting Eq. K.14 into Eq. K.13, we have:

$$\begin{aligned} R &= R_{(\tilde{\theta}_3 - \theta_1)\boldsymbol{e}}\, R_{\theta_2 \boldsymbol{e}'}\, R_{\theta_1 \boldsymbol{e}} \\ &= R_{\tilde{\theta}_3 \boldsymbol{e}}\, \left(R_{-\theta_1 \boldsymbol{e}}\, R_{\theta_2 \boldsymbol{e}'}\, R_{\theta_1 \boldsymbol{e}}\right) \end{aligned} \tag{K.15}$$

And, in quaternion form:

$$\begin{aligned} q &= \begin{bmatrix} c_{\left((\tilde{\theta}_3 - \theta_1)/2\right)} \\ s_{\left((\tilde{\theta}_3 - \theta_1)/2\right)}\boldsymbol{e} \end{bmatrix} \odot \begin{bmatrix} c_{(\theta_2/2)} \\ s_{(\theta_2/2)}\boldsymbol{e}' \end{bmatrix} \odot \begin{bmatrix} c_{(\theta_1/2)} \\ s_{(\theta_1/2)}\boldsymbol{e} \end{bmatrix} \\ &= \begin{bmatrix} c_{\left(\tilde{\theta}_3/2\right)} \\ s_{\left(\tilde{\theta}_3/2\right)}\boldsymbol{e} \end{bmatrix} \odot \left(\begin{bmatrix} c_{(\theta_1/2)} \\ s_{(\theta_1/2)}\boldsymbol{e} \end{bmatrix}^* \odot \begin{bmatrix} c_{(\theta_2/2)} \\ s_{(\theta_2/2)}\boldsymbol{e}' \end{bmatrix} \odot \begin{bmatrix} c_{(\theta_1/2)} \\ s_{(\theta_1/2)}\boldsymbol{e} \end{bmatrix} \right) \end{aligned} \tag{K.16}$$

[4] Unsurprisingly, this is a conjugation operation, as studied in Group Theory [72].

In both forms, the term inside the parentheses are equivalent to the rotation by a rotated axis in Eq. K.11

Defining a new decomposition operation $\tilde{\boldsymbol{\theta}}_{iji} : \mathbb{R}^{3\times3} \to \mathbb{R}^3$, such that $\tilde{\boldsymbol{\theta}}_{iji}\{R\} = (\tilde{\theta}_3, \theta_2, \theta_1)$, and applying it in the simple roll, pitch and yaw rotation examples in the ZYZ sequence, we have:

$$\begin{aligned}
\tilde{\boldsymbol{\theta}}_{zyz}\left\{R_x(\psi_{\text{roll}})\right\} &= \left(\frac{\pi}{2}, \psi_{\text{roll}}, 0\right) \\
\tilde{\boldsymbol{\theta}}_{zyz}\left\{R_y(\psi_{\text{pitch}})\right\} &= (0, \psi_{\text{pitch}}, 0) \\
\tilde{\boldsymbol{\theta}}_{zyz}\left\{R_z(\psi_{\text{yaw}})\right\} &= (b, 0, \psi_{\text{yaw}}) \\
\tilde{\boldsymbol{\theta}}_{zyz}\left\{R_z(\psi_{\text{yaw}})R_x(\psi_{\text{roll}})\right\} &= \left(\frac{\pi}{2}, \psi_{\text{roll}}, \psi_{\text{yaw}}\right) \\
\tilde{\boldsymbol{\theta}}_{zyz}\left\{R_z(\psi_{\text{yaw}})R_y(\psi_{\text{pitch}})\right\} &= (0, \psi_{\text{pitch}}, \psi_{\text{yaw}})
\end{aligned} \tag{K.17}$$

Where, for the pure yaw rotation, $\theta_1 = b$, as before, can be any value. The difference is that $\tilde{\theta}_3$ is now uncoupled from θ_1, and its value is independent of b[5].

Moreover, the actual general value of $\tilde{\theta}_3$ is:

$$\tilde{\theta}_3 = 2\ \text{atan2}(\boldsymbol{q}\cdot\boldsymbol{e}, q_{\text{r}}) \tag{K.18}$$

This is, unsurprisingly, the same formula for the spin (or twist) angle found in Chapter 7! Which is yet another indication that the angle $\tilde{\theta}_3$ has, in fact, a more meaningful geometrical interpretation than θ_3.

[5]This is not valid for the singularity at $\theta_2 = \pi$, but this is arguably a less common and less important singularity.