Modelling, control and simulation of a single rotor UAV with swashplateless torque modulation

Evandro Bernardes, Frédéric Boyer, Stéphane Viollet

Modelling, control and simulation of a single rotor UAV with swashplateless torque modulation

Evandro Bernardes[a], Frédéric Boyer[b], Stéphane Viollet[a,*]

[a]*Aix Marseille Univ, CNRS, ISM, Marseille, France*
[b]*LS2N Laboratory, Institut Mines Telecom Atlantique, Nantes, France*

Abstract

Single rotor-base unmanned aerial vehicles (UAVs) represent the most minimalistic design to implement an aerial vehicle. We describe here a mono-rotor UAV, also named mono-spinner, that uses torque modulation (also called swashplateless mechanism) to provide roll and pitch control without the need of a complicated swashplate mechanism. A full non-linear model of the mono-spinner based on the Euler-Poincaré theoretical framework is given including a description of the structure of the drone and its rotor control system. Mechanical properties, kinetic energy and all of the external forces of interest to derive the equations of motion are described. A novel, Lyapunov-stable, quaternion-based nonlinear control law based on the decomposition of the orientation quaternion into a spin component and a reduced orientation component is provided. Finally, simulation results based on parameters measured from a real implementation of a mono-spinner are produced.

Keywords: UAV, mono-rotor, drone, spinner, autopilot

1. 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, quad-copters) 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

*Corresponding author.
Email address: stephane.viollet@univ-amu.fr (Stéphane Viollet)

within a broad range.

Nevertheless, different UAV configurations exist with different sets of advantages and disadvantages. The Gemini [2] 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. Highly underactuated UAVs using a single rotor are also of great interest: they can be easier to miniaturize, like the Piccolissimo [3], and even be designed in such a way that an active control is not needed for achieving stable hovering flight [4]. An interesting subclass of these mono-rotor drones are mono-wing drones, like the ones proposed in [5, 6] inspired by the flight of the samara winged fruit/seed [7] and featuring a high efficiency [8]. In [9] the problem of finding solutions for a relaxed definition of hover for different configurations of motors pointing in the same direction is studied. In [10, 11] 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, [12] studies the design of a gyroscopically controlled micro air vehicle. Another elegant effort to achieve the same controllability is cyclic flapping by torque modulation [13, 14]: By attaching the tip of each blade of a 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 [15], and more recently, the Gemini-II [16], a follow-up to [2] that replaces both servomotors by a swashplateless cyclical system was also demonstrated successfully. It is worth noting that a very recent study sheds a light on the advantages of implementing such swashplateless mechanism combined with a single rotor configuration [17]. Usually, the uncontrollable spin is an undesirable feature and many studies of spinning drones dealt with the problem of safety when a drone loses control of one or more of its rotors [18]. Multirotors like quadrotors have been extensively studied in the literature, and many models with varying levels of detail and precision have been presented (e.g. [19] for a review). Quadrotors can also generate very aggressive maneuvers [20] thanks to higher roll and pitch torques than a single rotor configuration of the same size. However, the mono-spinner named PULSAR consumes 26.7% less power than a quadrotor having the same payload and total propeller disk area [17]. As its has been shown for a quadrotor [21], PULSAR features a high controllability as it can also achieve

dynamic ball avoidance and even navigation in a cluttered environment.

Our work presents a full theoretical analysis of a mono-rotor drone, and aims at providing a full theoretical framework to model this type of UAVs that feature both internal (rotor pitch and roll) and external (angular velocities and 3D positions) degrees of freedom. It can be seen as a complementary work to [17] and present two main contributions. First, the Euler-Poincaré equation was used to model the Equations of Motion of the drone. This is a special form of the Euler-Lagrange equation that can be used when the space of configurations is a member of a Lie group [22, 23]. Using this equation allowed us to have a very complete theoretical model of the dynamics of the drone, using a minimal set of approximations. Second, a new treatment of the orientation is presented: we introduce a novel decomposition of the orientation, effectively separating the uncontrollable spin component from the controllable reduced orientation component, and we demonstrate the relationship between this expression of the spin rotation with the Euler angles in the ZYZ sequence. Based on this decomposition, a new non-linear control law acting on the controllable reduced orientation is presented. Moreover, a simulation of the drone comprising both the complete dynamics and the attitude controller is implemented.

This work is organized as follows: Section 2 describes the different parts of the vehicle and its general working principles. Section 3 analyzes the linear and angular velocities of each part of the drone, and uses them to model the expression of the total kinetic energy. Section 4 models the external forces acting on the drone. Section 5 presents the equations of motion used for the simulation and describes the methodology used to derive them by means of the Euler-Poincaré equation. Section 6 describes a quaternion-based nonlinear rate control for the attitude, using a special kind of quaternion decomposition (that is further explained in Appendix C). Section 7 shows the simulation results.

2. System overview

The robot is composed of 3 separated rigid bodies, as seen on Figure 2:

- 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 of 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 comprises of the battery, which is a big part of the overall mass;

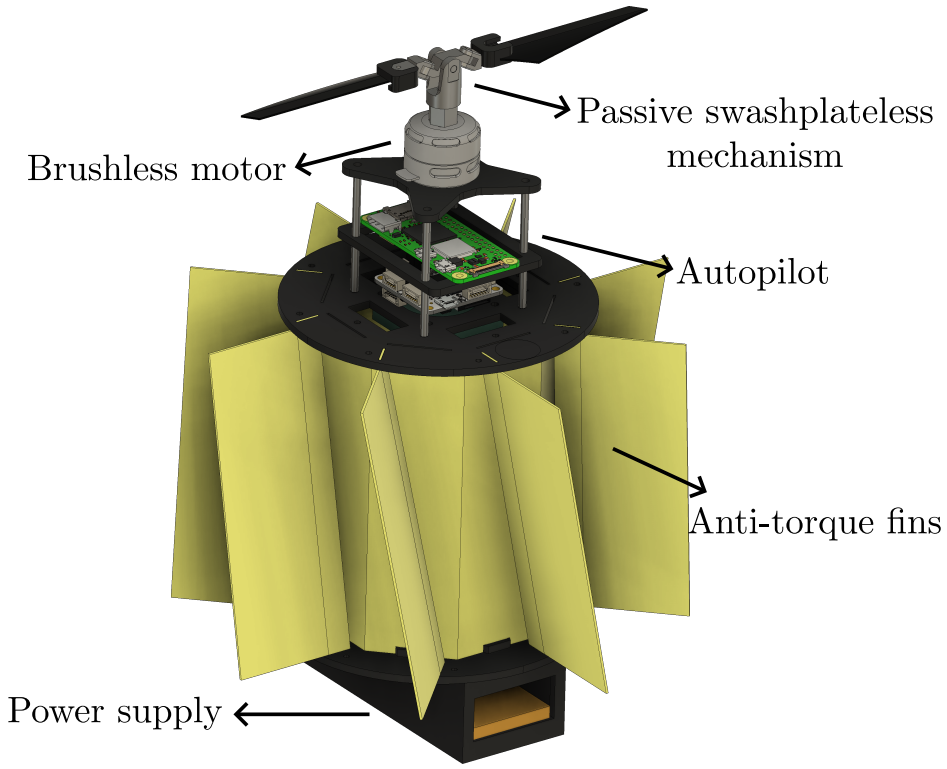

Figure 1: Overview of the proposed spinning UAV. The thrust is produced by a brushless motor, the driver of which is capable of controlling the rotor's speed with a sinusoidal velocity. A passive swashplateless mechanism based on [13, 14] is attached to the rotor. The swashplateless mechanism allows to control the direction of the thrust vector with 3 degrees of freedom. Multiple anti-torque of dimensions $136mm \times 50mm$ fins are mounted radially around the UAV's main body, in order to reduce the spin rotation drastically. The power supply, located at the bottom, presents a great point of concentrated 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.

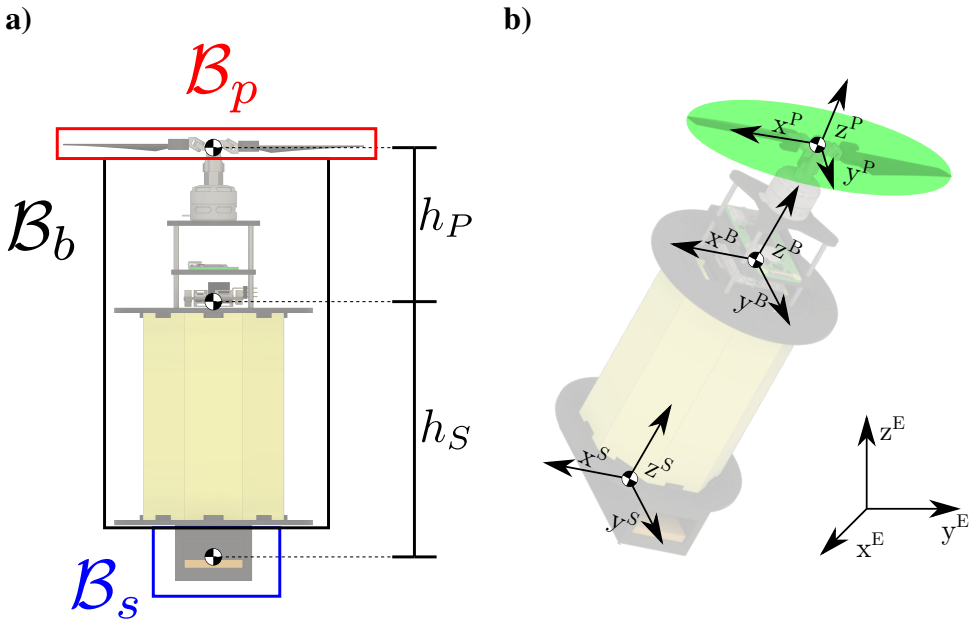

Figure 2: **a)** Illustration of the 3 separated bodies that compose the monorotor, and the distances between their centers of mass. $\mathcal{B}_p$ is the propeller's body, $\mathcal{B}_b$ is the main body and $\mathcal{B}_s$ is the power supply body. **b)** Diagram showing the inertial coordinate frame $\mathcal{F}_E$, 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.

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 $h_S = |\overrightarrow{\mathcal{F}_S\mathcal{F}_B}|$ between the centers of mass of the main body and the battery, specially if this distance is variable.

2.1. Masses and inertia matrices

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 $\mathcal{B}_s$:

$$\begin{aligned} J_b^B &= \text{diag}(J_{bx}, J_{by}, J_{bz}) \\ J_s^B = J_s^S &= \text{diag}(J_{sx}, J_{sy}, J_{sz}) \end{aligned} \tag{1}$$

For the propeller's inertia matrix, since the angular speed of the rotor is large compared to all of the other angular velocities, we approximated the propeller by averaging its inertia matrix around one rotation around the z-axis. The real inertia matrix of the propeller is:

$$J_{p*}^{P*} = \text{diag}(J_{px}, J_{py}, J_{pz}) \tag{2}$$

Where, in general, $J_{px} \neq J_{py}$. We use instead the following mean:

$$J_p^P = \frac{1}{2\pi} \int_0^{2\pi} R_z(\gamma) \left(J_{p*}^{P*} \right) R_z(\gamma)^T d\gamma \tag{3}$$

which gives:

$$J_p^P = \text{diag}(J_{pd}, J_{pd}, J_{pz}) \tag{4}$$

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:

$$J_p^B = R_P^B \, J_p^P \, \left(R_P^B \right)^T \tag{5}$$

We also define $m = m_b + m_s + m_p$, the mass of the whole drone, and the full inertia matrix is

$$\boldsymbol{J} = J_b^B + J_s^B + J_p^B + J_m^B \tag{6}$$

Where J_m^B is the inertia component created by the masses of $\mathcal{B}_p$ and $\mathcal{B}_s$, given by:

$$J_m^B = (m_s h_S^2 + m_p h_P^2) \left(-\widehat{\boldsymbol{e}}_z^{\,2} \right) \tag{7}$$

Note that the hat operator $\widehat{(\cdot)}$ gives the skew-symmetric matrix form of the cross product:

$$\widehat{\boldsymbol{a}} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{8}$$

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:

$$\widehat{\boldsymbol{e}}_z = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{9}$$

3. Pose and velocities

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.

3.1. Body pose and velocities

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

$$g_B^E = \begin{bmatrix} R_B^E & \boldsymbol{s}_B^E \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{10}$$

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:

$$\eta_B^B = \begin{bmatrix} \boldsymbol{\omega}_b^B \\ \boldsymbol{v}_b^B \end{bmatrix} \tag{11}$$

3.2. 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. This technology consists of attaching the blades on passive asymmetric lag-pitch hinges (as seen in Fig. 3), that are excited by using a motor speed controller that adds a sinusoidal speed component of frequency equals to 1/rev. A complete description of this mechanism can be read in [13, 14].

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 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 $h_P = |\overrightarrow{\mathcal{F}_E\mathcal{F}_B}|$ from the the center of $\mathcal{F}_B$. The rotation matrix from the propeller frame $\mathcal{F}_P$ to the body frame $\mathcal{F}_B$ is:

$$R_P^B \equiv R_z(\beta)\, R_y(\alpha) = \begin{bmatrix} c_\alpha c_\beta & -s_\beta & s_\alpha c_\beta \\ c_\alpha s_\beta & c_\beta & s_\alpha s_\beta \\ -s_\alpha & 0 & c_\alpha \end{bmatrix} \tag{12}$$

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

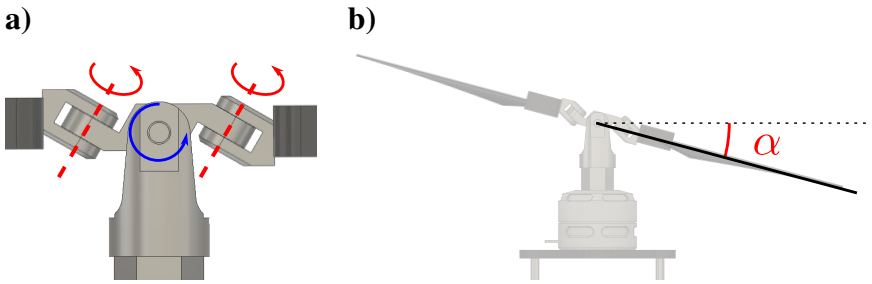

Figure 3: Swashplateless assembly based on [13, 14]. **a)** The lag-pitch hinge rotations (detailed in red) are responsible for creating the oscillatory flapping due to their inclination angle. The center teethering hinge helps reducing higher order components on the oscillation. **b)** Angle α between 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 teethering hinge as the center of the controlled inclination. This approximates well the behavior of the system.

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:

$$g_P^B = \begin{bmatrix} R_P^B & h_P \boldsymbol{e}_z \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{13}$$

Where $\boldsymbol{e}_z = [0,\ 0,\ 1]^T$. We define $\boldsymbol{\Omega}_{P/B}^P$ as the full angular velocity between the propeller and the body in the rotor frame $\mathcal{F}_P$ (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):

$$\boldsymbol{\Omega}_{P/B}^P = \boldsymbol{\omega}_{P/B}^P + \dot{\gamma}\boldsymbol{e}_z \tag{14}$$

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

$$\boldsymbol{\omega}_{P/B}^P = \begin{bmatrix} -s_\alpha \dot{\beta} \\ \dot{\alpha} \\ c_\alpha \dot{\beta} \end{bmatrix}$$

$$\dot{R}_P^B = R_P^B \widehat{\boldsymbol{\omega}}_{P/B}^P \tag{15}$$

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 $\boldsymbol{\omega}_{P/B}^P \approx 0$ and:

$$\boldsymbol{\Omega}_{P/B}^P \approx \dot{\gamma}\boldsymbol{e}_z \tag{16}$$

4. External forces

In this section, we will analyze the external forces and torques acting on the drone. There are three components:

$$F^B_{\text{ext}} = F^B_{\text{grav}} + F^B_p + F^B_{\text{aero}} \tag{17}$$

Where F^B_{grav} represents the external forces and torques generated by gravity, F^B_p represents the forces and torques generated by the rotor and F^B_{aero} represents the aerodynamic torque generated by friction with the air.

4.1. Gravity force

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

$$\begin{aligned} F^B_{\text{grav}} &= \begin{bmatrix} 0 \\ m_b \boldsymbol{g}^B \end{bmatrix} + \begin{bmatrix} R^B_P & h_P \widehat{\boldsymbol{e}_z} R^B_P \\ 0 & R^B_P \end{bmatrix} \begin{bmatrix} 0 \\ m_p \boldsymbol{g}^P \end{bmatrix} + \begin{bmatrix} R^B_S & -h_S \widehat{\boldsymbol{e}_z} R^B_S \\ 0 & R^B_S \end{bmatrix} \begin{bmatrix} 0 \\ m_s \boldsymbol{g}^S \end{bmatrix} \\ &= \begin{bmatrix} -\Delta_m \boldsymbol{e}_z \times \boldsymbol{g}^B \\ m \boldsymbol{g}^B \end{bmatrix} \end{aligned} \tag{18}$$

Assuming $\boldsymbol{g}^E = (0, 0, -g) = -g\boldsymbol{e}_z$ where g is the gravitational constant, $R^B_S = \mathbb{I}$ and $\boldsymbol{g}^B = \boldsymbol{g}^S = R^T \boldsymbol{g}^E$ and $\Delta_m = m_s h_S - m_p h_P$.

4.2. Motor thrust and torque

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

$$\boldsymbol{r} = R^B_P \boldsymbol{e}_z = \begin{bmatrix} s_\alpha c_\beta \\ s_\alpha s_\beta \\ c_\alpha \end{bmatrix} \tag{19}$$

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

$$\begin{aligned} \boldsymbol{f}^B_p &= f_p \boldsymbol{r} \\ \boldsymbol{\tau}^B_p &= \tau_p \boldsymbol{r} \end{aligned} \tag{20}$$

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:

$$\left|\boldsymbol{\Omega}^P_p\right|^2 = \left|\left(R^B_P\right)^T \boldsymbol{\omega}^B_b + \boldsymbol{\Omega}^P_{P/B}\right|^2 \tag{21}$$

And taking into account that:

$$\left|\boldsymbol{\Omega}_{P/B}^{P}\right| \gg \left|\boldsymbol{\omega}_{b}^{B}\right| \tag{22}$$

We can simplify the expression of the rotor velocity as:

$$\left|\boldsymbol{\Omega}_{p}^{P}\right| \approx \left|\boldsymbol{\Omega}_{P/B}^{P}\right| \tag{23}$$

We then model the magnitudes as:

$$\begin{aligned} \left|\boldsymbol{f}_{p}\right| &= f_{p} \approx k_{f} \left|\boldsymbol{\Omega}_{P/B}^{P}\right|^{2} \\ \left|\boldsymbol{\tau}_{p}\right| &= \tau_{p} \approx k_{\tau} \left|\boldsymbol{\Omega}_{P/B}^{P}\right|^{2} \end{aligned} \tag{24}$$

As seen in [11], we assume that the propeller torque is linear w.r.t. to the propeller thrust. Defining:

$$k = \frac{k_{\tau}}{k_{f}}, \tag{25}$$

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

$$\boldsymbol{\tau}_{p}^{B} = k\boldsymbol{f}_{p}^{B} \tag{26}$$

Moreover, given $h_{P}\,\boldsymbol{e}_{z}$ the position vector of propeller in the body frame $\mathcal{F}_{B}$, there is also a crossed thrust torque term that is added, as seen in Fig. 4:

$$\boldsymbol{\tau}_{\text{cross}}^{B} = h_{P}\,\boldsymbol{e}_{z} \times \boldsymbol{f}_{p}^{B} \tag{27}$$

Adding both terms of torque, we have:

$$\boldsymbol{\tau}_{\text{total}} = \boldsymbol{\tau}_{p}^{B} + \boldsymbol{\tau}_{\text{cross}}^{B} = \boldsymbol{B}_{\tau}\boldsymbol{f}_{p}^{B} \tag{28}$$

With:

$$\boldsymbol{B}_{\tau} = (k\mathbb{I} + h_{P}\widehat{\boldsymbol{e}_{z}}) = \begin{bmatrix} k & -h_{P} & 0 \\ h_{P} & k & 0 \\ 0 & 0 & k \end{bmatrix} \tag{29}$$

And finally, the 6×1 vector of all torques and forces produced by the propeller is:

$$F_{p}^{B} = \begin{bmatrix} \boldsymbol{B}_{\tau} f_{p} \boldsymbol{r} \\ f_{p}\boldsymbol{r} \end{bmatrix} \tag{30}$$

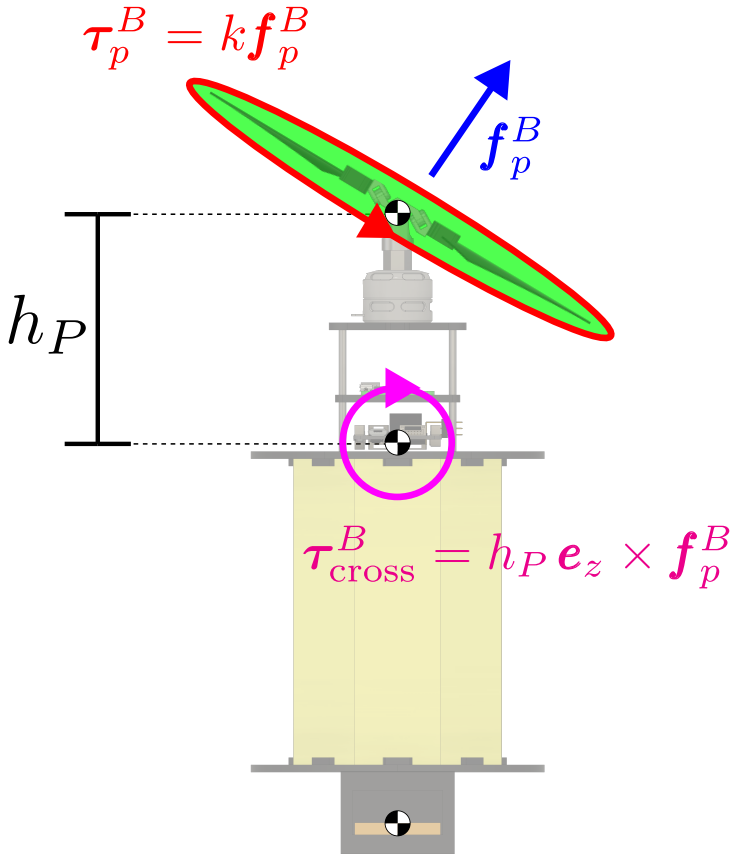

Figure 4: All of the forces and torques produced by the rotor. Thrust vector (in red) considered to be proportional to the square of the rotor mean angular speed. First torque component (in green) considered linear in the thrust force. Second torque component (in magenta) generated by the distance between the propeller thrust center and the center of mass of the body.

4.3. Aerodynamic drag

The aerodynamic drag can be modeled as:

$$F_{\text{aero}}^B = \begin{bmatrix} -\left|\boldsymbol{\omega}_b^B\right| \boldsymbol{K}\,\boldsymbol{\omega}_b^B \\ \mathbf{0} \end{bmatrix} \tag{31}$$

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:

$$\left|\boldsymbol{\omega}_b^B\right| \boldsymbol{K}\,\boldsymbol{\omega}_b^B = \text{sign}(\omega_z) K_z\, \omega_z^2\, \boldsymbol{e}_z \tag{32}$$

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

$$K_z = \text{sign}(\omega_z) k_\tau \left(1 + \frac{\dot{\gamma}}{\omega_z}\right)^2 \tag{33}$$

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 mono-rotor drone's body was higher than the limits of the Pixracer's IMU. In order to overcome this problem, several fins of dimensions $136mm \times 50mm$ 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. 1. We used our prototype containing a Pixracer card with PX4 firmware to test the steady-state velocity of the mono-spinner 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. 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.

4.4. Sum of external forces

Introducing Eqs. 18, 30 and 31 into Eq. 17, we have the total expression of the external forces:

$$\begin{aligned} F^B_{\text{ext}} &= F^B_{\text{grav}} + F^B_p + F^B_{\text{aero}} \\ &= \begin{bmatrix} f_p \boldsymbol{B}_\tau \boldsymbol{r} + \Delta_m \boldsymbol{e}_z \times \boldsymbol{g}^B - \left|\boldsymbol{\omega}^B_b\right| \boldsymbol{K}\, \boldsymbol{\omega}^B_b \\ f_p \boldsymbol{r} - m\boldsymbol{g}^B \end{bmatrix} \end{aligned} \tag{34}$$

5. Equations of Motion with Euler-Poincaré equations

In this section, we will apply the Euler-Poincaré equations to our system. Originally published in a short two-page note in [22], these equations are a generalization of the Euler-Lagrange equations to systems whose configuration space is a Lie group of transformations, instead of a vector space of generalized coordinates [24].

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

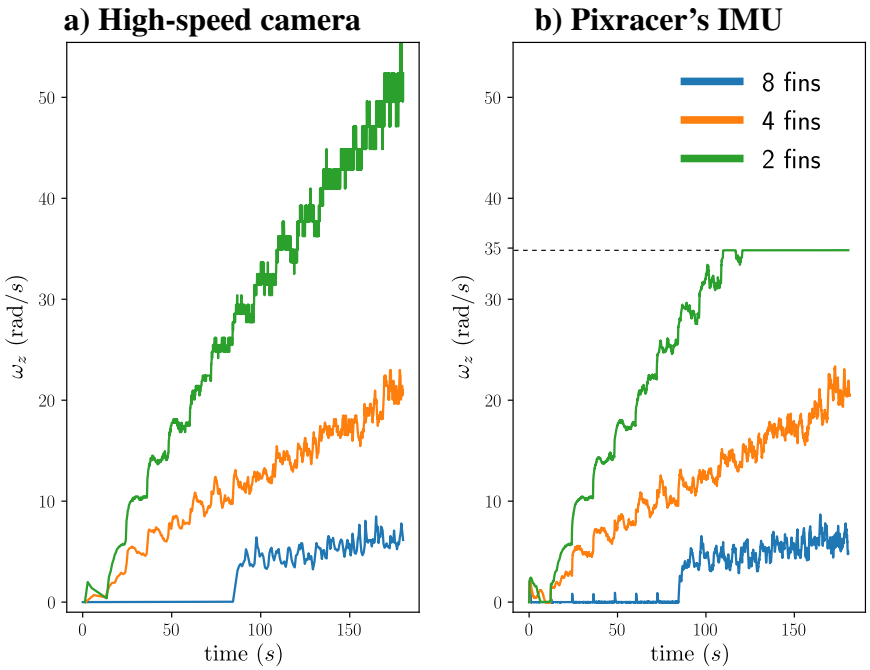

Figure 5: Steady-state angular angular velocity of mono-rotor with 2, 4 and 8 fins, while varying the motor's speed from 80 to 800 rad/s. **a)** Angular velocity measured from a Phantom high-speed camera. **b)** Angular velocity measured from the mono-rotor's IMU's rate gyro. The latter saturates at around 35.68 rad/s.

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 [17]. 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:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial L}{\partial \boldsymbol{q}} = \boldsymbol{Q} \tag{35}$$

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 [25]. 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 [26].

Any solution to Eq. 35 is also a solution to the Euler Poincaré equations [27]. 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.

The Euler-Poincaré equations on the configuration Lie group SE(3) of our system are:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T}{\partial \eta}\right) - \mathrm{ad}_{\eta}^{T}\left(\frac{\partial T}{\partial \eta}\right) = F_{\mathrm{ext}}^{B} \tag{36}$$

Where the ad_{η} operator is given by [28]:

$$\mathrm{ad}_{\eta} = \begin{bmatrix} \widehat{\boldsymbol{\omega}} & 0 \\ \widehat{\boldsymbol{v}} & \widehat{\boldsymbol{\omega}} \end{bmatrix} \tag{37}$$

And, for simplicity, $\boldsymbol{\omega} = \boldsymbol{\omega}_b^B$, $\boldsymbol{v} = \boldsymbol{v}_b^B$ and $\boldsymbol{\Omega} = \boldsymbol{\Omega}_{P/B}^P$. A quick comparison between Eq. 36 and Eq. 35 shows that the Euler-Poincaré equations avoid the use of the usual generalized coordinates of Lagrangian mechanics, and the nonlinearities/singularities related to their use. Moreover, the most tedious, albeit straightforward, step in using the Euler-Poincaré equations in our work was to model the kinetic energies of the system, which is also a required step for Lagrangian mechanics nonetheless. As shown in Appendix B:

$$T = \frac{1}{2}\begin{bmatrix} \eta \\ \boldsymbol{\Omega} \end{bmatrix}^{T} \mathbf{M} \begin{bmatrix} \eta \\ \boldsymbol{\Omega} \end{bmatrix} \tag{38}$$

Where, defining $\Delta_m = m_s h_S - m_p h_P$:

$$\mathbf{M} = \begin{bmatrix} \boldsymbol{J} & -\Delta_m \widehat{\boldsymbol{e}_z} & R_P^B J_p^P \\ \Delta_m \widehat{\boldsymbol{e}_z} & m\,\mathbb{I} & 0 \\ \left(R_P^B J_p^P\right)^T & 0 & J_p^P \end{bmatrix} \tag{39}$$

Calculating the partial derivative gives:

$$\frac{\partial T}{\partial \eta} = \left(\begin{bmatrix} \boldsymbol{J} & 0 \\ 0 & m\,\mathbb{I} \end{bmatrix} + \boldsymbol{\Delta_m}\right)\eta + \begin{bmatrix} R_P^B J_p^P \boldsymbol{\Omega} \\ 0 \end{bmatrix} \tag{40}$$

Where:

$$\boldsymbol{\Delta_m} \equiv \Delta_m \begin{bmatrix} 0 & -\widehat{\boldsymbol{e}_z} \\ \widehat{\boldsymbol{e}_z} & 0 \end{bmatrix} \tag{41}$$

And $\boldsymbol{\Omega} = \dot{\gamma}\boldsymbol{e}_z$ with constant $\dot{\gamma}$. Moreover, differentiating Eq. 40 w.r.t. time, recalling that $\frac{\mathrm{d}\boldsymbol{J}}{\mathrm{d}t} = \boldsymbol{0}$ since we assume $\boldsymbol{\omega}^P_{P/B} \approx 0$, we have:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T}{\partial \eta}\right) = \left(\begin{bmatrix} \boldsymbol{J} & 0 \\ 0 & m\,\mathbb{I} \end{bmatrix} + \boldsymbol{\Delta_m}\right)\dot{\eta} \tag{42}$$

For the second term, we have:

$$-\mathrm{ad}^T_\eta \frac{\partial T}{\partial \eta} = \left(\begin{bmatrix} \widehat{\boldsymbol{\omega}}\boldsymbol{J} & 0 \\ 0 & \widehat{\boldsymbol{\omega}}m \end{bmatrix} - \mathrm{ad}^T_\eta \boldsymbol{\Delta_m}\right)\eta + J_{pz}\dot{\gamma}\begin{bmatrix} \widehat{\boldsymbol{\omega}}\,\boldsymbol{r} \\ 0 \end{bmatrix} \tag{43}$$

Finally giving:

$$\mathbf{D}\begin{bmatrix} \dot{\boldsymbol{\omega}} \\ \dot{\boldsymbol{v}} \end{bmatrix} + \mathbf{C}\begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} + \begin{bmatrix} \dot{\gamma} J_{pz}\,\boldsymbol{\omega} \times \boldsymbol{r} \\ 0 \end{bmatrix} = F^B_{\text{ext}} \tag{44}$$

With:

$$\begin{aligned} \mathbf{D} &= \begin{bmatrix} \boldsymbol{J} & 0 \\ 0 & m\,\mathbb{I} \end{bmatrix} + \boldsymbol{\Delta_m} \\ \mathbf{C} &= -\mathrm{ad}^T_\eta \mathbf{D} \\ F^B_{\text{ext}} &= \begin{bmatrix} f_p \boldsymbol{B}_\tau \boldsymbol{r} + \Delta_m\,\boldsymbol{e}_z \times \boldsymbol{g}^B - \left|\boldsymbol{\omega}^B_b\right| \boldsymbol{K}\,\boldsymbol{\omega}^B_b \\ f_p \boldsymbol{r} - m\boldsymbol{g}^B \end{bmatrix} \end{aligned} \tag{45}$$

This is the equation used for all simulations. Isolating the angular accelerations, we have the following equation, which will be useful for the simulation implementation:

$$\begin{bmatrix} \dot{\boldsymbol{\omega}} \\ \dot{\boldsymbol{v}} \end{bmatrix} = \mathbf{D}^{-1}\left(F^B_{\text{ext}} + \mathrm{ad}^T_\eta \mathbf{D}\begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} - \begin{bmatrix} \dot{\gamma} J_{pz}\,\boldsymbol{\omega} \times \boldsymbol{r} \\ 0 \end{bmatrix}\right) \tag{46}$$

Which, as expected, expresses the evolution of the system as a function of its velocities. The position $\boldsymbol{s} = \boldsymbol{s}^E_B$ and rotation matrix R^E_B can then be integrated with the reconstruction equations:

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{s}}{\mathrm{d}t} &= R\,\boldsymbol{v} \\ \frac{\mathrm{d}R}{\mathrm{d}t} &= R\,\widehat{\boldsymbol{\omega}} \end{aligned} \tag{47}$$

Instead of directly using rotation matrices, however, we parameterize the orientation of the drone using quaternions (more precisely, classical Hamilton quaternions [29]). 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 [30]. A summary of quaternion algebra can be seen on Appendix A. In this work, we will denote quaternions as 4-vectors:

$$q = \begin{bmatrix} q_{\mathrm{r}} \\ \boldsymbol{q} \end{bmatrix} \tag{48}$$

Where q_{r} and $\boldsymbol{q}$ are, respectively, the scalar (or real) and vector (or imaginary) parts of q. Supposing $q = q_B^E$ represents the rotation from $\mathcal{F}_B$ to $\mathcal{F}_E$, then, for some vector $\boldsymbol{a}$:

$$\begin{bmatrix} 0 \\ \boldsymbol{a}^E \end{bmatrix} = q \odot \begin{bmatrix} 0 \\ \boldsymbol{a}^B \end{bmatrix} \odot q^* \tag{49}$$

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

$$\dot{q} = \frac{1}{2} q \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \tag{50}$$

6. Controller architecture

Given that the separation principle holds[1], we can control the UAV's orientation using a cascaded 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 [31], for example, implements a nonlinear attitude controller based on [32] followed by an angular rate PID controller as well.

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

6.1. 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 vector $\boldsymbol{n}_b^B$ that points upwards in the body frame $\mathcal{F}_B$:

$$\boldsymbol{n}_b^B \equiv \boldsymbol{e}_z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{51}$$

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

$$\begin{aligned} \begin{bmatrix} 0 \\ \boldsymbol{n}_b^E \end{bmatrix} &= q \odot \begin{bmatrix} 0 \\ \boldsymbol{n}_b^B \end{bmatrix} \odot q^* \\ &= q \odot \begin{bmatrix} 0 \\ \boldsymbol{e}_z \end{bmatrix} \odot q^* \end{aligned} \tag{52}$$

Giving:

$$\boldsymbol{n}_b^E = \begin{bmatrix} 2(q_\mathrm{x} q_\mathrm{z} + q_\mathrm{r} q_\mathrm{y}) \\ 2(q_\mathrm{y} q_\mathrm{z} - q_\mathrm{r} q_\mathrm{x}) \\ 2\left(q_\mathrm{r}^2 + q_\mathrm{z}^2\right) - 1 \end{bmatrix} \tag{53}$$

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

$$\boldsymbol{m} = \frac{\boldsymbol{n}_b^E + \boldsymbol{n}_b^B}{\left|\boldsymbol{n}_b^E + \boldsymbol{n}_b^B\right|} = \frac{1}{\sqrt{q_\mathrm{r}^2 + q_\mathrm{z}^2}} \begin{bmatrix} q_\mathrm{x} q_\mathrm{z} + q_\mathrm{r} q_\mathrm{y} \\ q_\mathrm{y} q_\mathrm{z} - q_\mathrm{r} q_\mathrm{x} \\ q_\mathrm{r}^2 + q_\mathrm{x}^2 \end{bmatrix} \tag{54}$$

Note that this has a singularity when in $\boldsymbol{n}_b^E = -\boldsymbol{n}_b^B = -\boldsymbol{e}_z$, which can not be plausible for the flying mono spinner here. We also defined:

$$\begin{aligned} q_s &= \begin{bmatrix} c_s \\ s_s \boldsymbol{e}_z \end{bmatrix} = \begin{bmatrix} \cos(\theta_s/2) \\ \sin(\theta_s/2) \boldsymbol{e}_z \end{bmatrix} \\ \theta_s &= 2\,\mathrm{atan2}(q_\mathrm{z}, q_\mathrm{r}) \end{aligned} \tag{55}$$

Or, more directly:

$$q_s = \frac{1}{\sqrt{q_\mathrm{r}^2 + q_\mathrm{z}^2}} \begin{bmatrix} q_\mathrm{r} \\ 0 \\ 0 \\ q_\mathrm{z} \end{bmatrix} \tag{56}$$

And we can decompose q as (see Appendix C for a complete demonstration):

$$q = -\begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e}_z \end{bmatrix} \odot q_s \tag{57}$$

Moreover:

$$\dot{q}_s = \frac{1}{2} q_s \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega}_s \end{bmatrix} \tag{58}$$

Where:

$$\boldsymbol{\omega}_s = \left(\frac{\boldsymbol{m} \cdot R\boldsymbol{\omega}}{\boldsymbol{m} \cdot \boldsymbol{e}_z} \right) \boldsymbol{e}_z \tag{59}$$

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

$$\begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} = q \odot q_s^* \odot \begin{bmatrix} 0 \\ \boldsymbol{e}_z \end{bmatrix} \tag{60}$$

Moreover, its derivative can be given by:

$$\dot{\boldsymbol{m}} = -\frac{1}{2} \boldsymbol{m} \times R\left(\boldsymbol{\omega} - \boldsymbol{\omega}_s\right) \tag{61}$$

6.2. Rotation error definition

Define $\bar{q}$ as the desired orientation, and $\bar{\boldsymbol{n}}_E$ as the corresponding normal vector, and $\bar{\boldsymbol{m}}$ as the corresponding middle normal vector. Moreover, since we cannot control the rotation around the z-axis, we just set the goal spin as $\bar{q}_s = q_s$:

$$\bar{q} = -\begin{bmatrix} 0 \\ \bar{\boldsymbol{m}} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e}_z \end{bmatrix} \odot q_s \tag{62}$$

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

$$q_\Delta = q \odot \bar{q}^* = \begin{bmatrix} \bar{\boldsymbol{m}} \cdot \boldsymbol{m} \\ \bar{\boldsymbol{m}} \times \boldsymbol{m} \end{bmatrix} \tag{63}$$

Or equivalently:

$$q_\Delta = \frac{1}{|\boldsymbol{n} + \boldsymbol{e}_z||\bar{\boldsymbol{n}} + \boldsymbol{e}_z|} \begin{bmatrix} (\bar{\boldsymbol{n}} + \boldsymbol{e}_z) \cdot (\boldsymbol{n} + \boldsymbol{e}_z) \\ (\bar{\boldsymbol{n}} + \boldsymbol{e}_z) \times (\boldsymbol{n} + \boldsymbol{e}_z) \end{bmatrix} \tag{64}$$

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:

$$\begin{bmatrix} \bar{\boldsymbol{m}} \cdot \boldsymbol{m} \\ \bar{\boldsymbol{m}} \times \boldsymbol{m} \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} \tag{65}$$

Leading to:

$$(\boldsymbol{n} + \boldsymbol{e}_z) \cdot (\bar{\boldsymbol{n}} + \boldsymbol{e}_z) = |\boldsymbol{n} + \boldsymbol{e}_z||\bar{\boldsymbol{n}} + \boldsymbol{e}_z| \tag{66}$$

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

6.3. Angular controller

The inner non-linear loop is defined by:

$$\boldsymbol{\omega}_r = \boldsymbol{\omega}_s + \boldsymbol{P}\left(R^T \frac{\boldsymbol{m} \times \bar{\boldsymbol{m}}}{\boldsymbol{m} \cdot \bar{\boldsymbol{m}}}\right) \tag{67}$$

Which we demonstrate to stabilize the system with the use of a Lyapunov function (see Appendix C.5). We then define:

$$\Delta\boldsymbol{\omega} = \boldsymbol{\omega}_r - \boldsymbol{\omega} \tag{68}$$

And using an outer PID loop, the correction torque can be calculated as:

$$\boldsymbol{\tau}_c = \boldsymbol{K}_p \, \Delta\boldsymbol{\omega} + \boldsymbol{K}_d \frac{\mathrm{d}\Delta\boldsymbol{\omega}}{\mathrm{d}t} + \boldsymbol{K}_i \int \Delta\boldsymbol{\omega} dt \tag{69}$$

According to the motor's model, we know that the equivalent control force is:

$$\boldsymbol{\tau} = \boldsymbol{B}_\tau \boldsymbol{f} = (k\mathbb{I} + h_P \widehat{\boldsymbol{e}_z})\boldsymbol{f} \tag{70}$$

We assumed that the control is given by the cross product part (see Fig. 4). Moreover, we also assumed that $\boldsymbol{e}_z \cdot \boldsymbol{f} = 0$, since no control of rotations around $\boldsymbol{e}_z$ is supposed possible (and assuming the torque generated by the helices inertia is negligible). This leads to:

$$\begin{aligned} \boldsymbol{\tau}_c &= h_P \, \boldsymbol{e}_z \times \boldsymbol{f}_c \\ \boldsymbol{f}_c &= \frac{1}{h_P} \boldsymbol{\tau}_c \times \boldsymbol{e}_z = \frac{1}{h_P} \begin{bmatrix} +\tau_{c,y} \\ -\tau_{c,x} \\ 0 \end{bmatrix} \end{aligned} \tag{71}$$

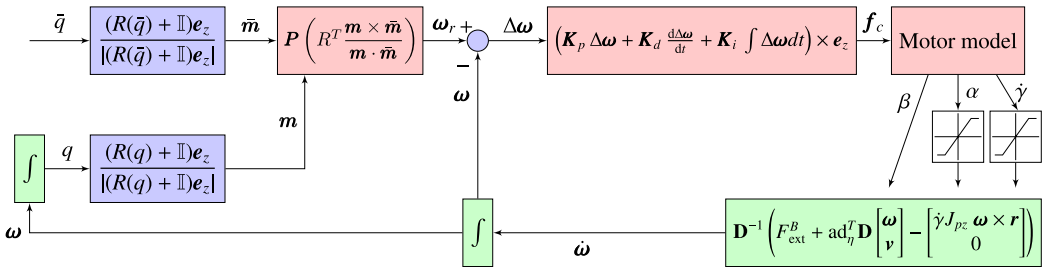

Figure 6: Simulation diagram with a cascaded controller architecture. Controller (in red) composed of a non-linear reduced attitude controller cascaded with a simple PID for angular rate control, and finally the swashplateless motor model is used to to extract the necessary commands to the motor. Equations 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.

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

$$\boldsymbol{f} = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = \boldsymbol{f}_c + k_f \Omega^2 \boldsymbol{e}_z = \begin{bmatrix} +\tau_{c,y}/h_P \\ -\tau_{c,x}/h_P \\ k_f \Omega^2 \end{bmatrix} \tag{72}$$

And finally the motor velocity $\dot{\gamma}$, phase β and inclination angle α are calculated as:

$$\begin{aligned} \beta &= \text{atan2}(f_y, f_x) \\ \alpha &= \text{acos}\left(\frac{f_z}{|\boldsymbol{f}|}\right) \\ \dot{\gamma} &= \sqrt{\frac{|\boldsymbol{f}|}{k_f}} = \frac{1}{\sqrt{\cos(\alpha)}}\,\Omega \end{aligned} \tag{73}$$

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

$$\boldsymbol{\omega}_r = \boldsymbol{P}\left(R^T \frac{\boldsymbol{m} \times \bar{\boldsymbol{m}}}{\boldsymbol{m} \cdot \bar{\boldsymbol{m}}}\right) \tag{74}$$

7. Simulation

In order to test the controller shown in Section 6, a simulation of the Equations of Motion shown in Section 5 was implemented using Python[2]. In this section,

[2]See github.com/evbernardes/monospinner_simulator

we will analyze some of the implementation details.

7.1. Controller

As shown in Section 6, the desired controller force is given by:

$$\boldsymbol{f}_c = \left(\boldsymbol{K}_p \, \Delta\boldsymbol{\omega} + \boldsymbol{K}_d \, \frac{\mathrm{d}\Delta\boldsymbol{\omega}}{\mathrm{d}t} + \boldsymbol{K}_i \int \Delta\boldsymbol{\omega} dt \right) \times \boldsymbol{e}_z \tag{75}$$

Where:

$$\Delta\boldsymbol{\omega} = \boldsymbol{P} \left(R^T \frac{\boldsymbol{m} \times \bar{\boldsymbol{m}}}{\boldsymbol{m} \cdot \bar{\boldsymbol{m}}} \right) - \boldsymbol{\omega} \tag{76}$$

And the positive definite matrices $\boldsymbol{P}$, $\boldsymbol{K}_p$, $\boldsymbol{K}_d$ and $\boldsymbol{K}_i$ are all adjusted parameters. The full propeller force is:

$$\boldsymbol{f}_p = \boldsymbol{f}_c + k_f \Omega^2 \boldsymbol{e}_z \tag{77}$$

And, noting $\boldsymbol{f}_p = \begin{bmatrix} f_x & f_y & f_z \end{bmatrix}^T$, the rotor control input signal is calculated as follows:

$$\begin{aligned} \beta &= \mathrm{atan2}(f_y, f_x) \\ \alpha &= \mathrm{atan}\left(\frac{|\boldsymbol{f}_c|}{k_f \Omega^2} \right) \\ \dot{\gamma} &= \frac{\Omega}{\sqrt{\cos(\alpha)}} \end{aligned} \tag{78}$$

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

$$\Omega = \sqrt{\frac{mg}{k_f}} \tag{79}$$

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 follows:

$$\begin{aligned} \beta &= \mathrm{atan2}(f_y, f_x) \\ \alpha &= \left\{ \mathrm{atan}\left(\frac{|\boldsymbol{f}_c|}{k_f \Omega^2} \right), \; \alpha_{\max} \right\} \\ \dot{\gamma} &= \Omega \left[\min\left\{ \frac{1}{\sqrt{\cos(\alpha)}}, \; \Lambda \right\} \right] \end{aligned} \tag{80}$$

The propeller thrust force is then calculated as follows:

$$\boldsymbol{r} = \begin{bmatrix} s_\alpha c_\beta \\ s_\alpha s_\beta \\ c_\alpha \end{bmatrix}$$
$$\boldsymbol{f}_p = k_f \dot{\gamma}^2 \boldsymbol{r} \tag{81}$$

7.1.1. Equations of Motion

We use the equations from Eq. 44. As a reminder:

$$\mathbf{D} \begin{bmatrix} \dot{\boldsymbol{\omega}} \\ \dot{\boldsymbol{v}} \end{bmatrix} - \mathrm{ad}_\eta^T \mathbf{D} \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} + \begin{bmatrix} \dot{\gamma} J_{pz} \, \boldsymbol{\omega} \times \boldsymbol{r} \\ 0 \end{bmatrix} = F_{\text{ext}}^B \tag{82}$$

With :

$$\mathbf{D} = \begin{bmatrix} \boldsymbol{J} & 0 \\ 0 & m \, \mathbb{I} \end{bmatrix} + \Delta_m \begin{bmatrix} 0 & -\widehat{\boldsymbol{e}_z} \\ \widehat{\boldsymbol{e}_z} & 0 \end{bmatrix}$$
$$F_{\text{ext}}^B = \begin{bmatrix} f_p \boldsymbol{B}_\tau \boldsymbol{r} + \Delta_m \, g \, \boldsymbol{e}_z \times \boldsymbol{n}_b^E - |\boldsymbol{\omega}| \boldsymbol{K} \boldsymbol{\omega} \\ f_p \boldsymbol{r} - mg \, \boldsymbol{n}_b^E \end{bmatrix} \tag{83}$$

And:

$$m = m_b + m_s + m_p$$
$$\Delta_m = m_s h_S - m_p h_P$$
$$\mathrm{ad}_\eta = \begin{bmatrix} \widehat{\boldsymbol{\omega}} & 0 \\ \widehat{\boldsymbol{v}} & \widehat{\boldsymbol{\omega}} \end{bmatrix} \tag{84}$$

Equation 46 is used to calculate the angular and linear accelerations:

$$\begin{bmatrix} \dot{\boldsymbol{\omega}} \\ \dot{\boldsymbol{v}} \end{bmatrix} = \mathbf{D}^{-1} \left(F_{\text{ext}}^B + \mathrm{ad}_\eta^T \mathbf{D} \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} - \begin{bmatrix} \dot{\gamma} J_{pz} \, \boldsymbol{\omega} \times \boldsymbol{r} \\ 0 \end{bmatrix} \right) \tag{85}$$

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. For the linear and angular velocities:

$$\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \Delta t \, \dot{\boldsymbol{\omega}}_k$$
$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k + \Delta t \, \dot{\boldsymbol{v}}_k \tag{86}$$

with Δt the sampling time of the simulation.

And the new orientation and position can be reconstructed by applying the following equations:

$$\begin{aligned} \dot{q}_k &= \frac{1}{2} q_k \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega}_k \end{bmatrix} \\ q_{k+1} &= q_k + \Delta t \; \dot{q}_k \\ \boldsymbol{s}_{k+1} &= \boldsymbol{s}_k + \Delta t \; R(q_k) \, \boldsymbol{v}_k \end{aligned} \tag{87}$$

Note that an extra normalization step is needed for q.

7.1.2. Parameters

The parameters are all based on a prototype designed at ISM, using 8 anti-torque fins as shown in section 2. The masses and distances are given in Table 1, and the controller parameters for all the following tests are the ones shown in Table 2.

m_b	$0.2100 kg$
m_s	$0.2007 kg$
m_p	$0.0067 kg$
h_P	$0.083 m$
h_S	$0.145 m$
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 1: Masses and distances used during the simulations.

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

Table 2: Controller parameters used during the simulations.

7.1.3. Inputs and parametration

For simplicity, the inputs in the simulation are given as Euler angles in the ZYZ sequence. As seen in [33], the rotation can be decomposed as:

$$R_B^E = R(\theta_3 \boldsymbol{e}_z)\, R(\theta_2 \boldsymbol{e}_z)\, R(\theta_1 \boldsymbol{e}_z) \tag{88}$$

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

$$\begin{aligned} \theta_1 &= \text{atan2}(q_\text{z}, q_\text{r}) - \text{atan2}(-q_\text{x}, q_\text{y}) \\ \theta_2 &= \text{acos}\left(2\, \left(q_\text{r}^2 + q_\text{z}^2\right) - 1\right) \\ \theta_3 &= \text{atan2}(q_\text{z}, q_\text{r}) + \text{atan2}(-q_\text{x}, q_\text{y}) \end{aligned} \tag{89}$$

Equation 88 can be rewritten as:

$$R_B^E = R\left((\theta_1 + \theta_3)\, \boldsymbol{e}_z\right)\; R(\theta_1 \boldsymbol{e}_z)^T\; R(\theta_2 \boldsymbol{e}_z)\, R(\theta_1 \boldsymbol{e}_z) \tag{90}$$

And analyzing Eqs. 55, 89 and 90, we note that:

$$\begin{aligned} \theta_1 + \theta_3 &= 2\,\text{atan2}(q_\text{z}, q_\text{r}) \\ &= \theta_s \end{aligned} \tag{91}$$

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. 55. Moreover, θ_s is uniquely defined when θ_2 is near 0 and arguably more representative of the actual geometrical spin. The input goal orientations to the simulations are given as precessions and nutations:

$$\boldsymbol{\theta} = (\theta_1, \theta_2) \tag{92}$$

So that:

$$\bar{\boldsymbol{n}}_E = \begin{bmatrix} \cos(\theta_1)\sin(\theta_2) \\ \sin(\theta_1)\sin(\theta_2) \\ \cos(\theta_2) \end{bmatrix} \tag{93}$$

And.

$$\bar{\boldsymbol{m}} = \frac{\bar{\boldsymbol{n}}_E + \boldsymbol{e}_z}{|\bar{\boldsymbol{n}}_E + \boldsymbol{e}_z|} = \begin{bmatrix} \cos(\theta_1)\sin(\theta_2/2) \\ \sin(\theta_1)\sin(\theta_2/2) \\ \cos(\theta_2/2) \end{bmatrix} \tag{94}$$

7.2. Simulation 1: orientation tracking without noise

In this scenario, the robot is supposed to hover while changes of large amplitude are imposed on precession and nutation angles (ZYZ sequence). This scenario shows the controllability of the spinning robot in pitch and roll despite the uncontrollable spinning angle. In addition, the controller allows the robot to reach a desired body tilt with a null steady-state error. This first simulation starts with the robot at rest. Then 4 orientation points in Table 3 are tracked in sequence, before going back to the initial orientation. In Fig. 8 **a)** and **b)** we can see how

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

Table 3: Orientation inputs for simulation 1.

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 8 **c)** shows the angular velocities. The component around $\boldsymbol{e}_z$ approaches the theoretical steady-state velocity, while the other components approach zero. Figure 8 **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 7 shows the motor phase. We can note how the phase keeps increasing continuously with a change of rate equivalent to the change of rate of the spin, with inverted sign. This was expected in order to counter the mono-rotor's spin. We can also note that the moment the goal orientation is changed, there is a

discontinuity on the motor input phase, which was also expected, since the robot has a new goal orientation. We note though that even when this phase-shift happens, the change of rate continues to follow the negative of the change of rate of the spin.

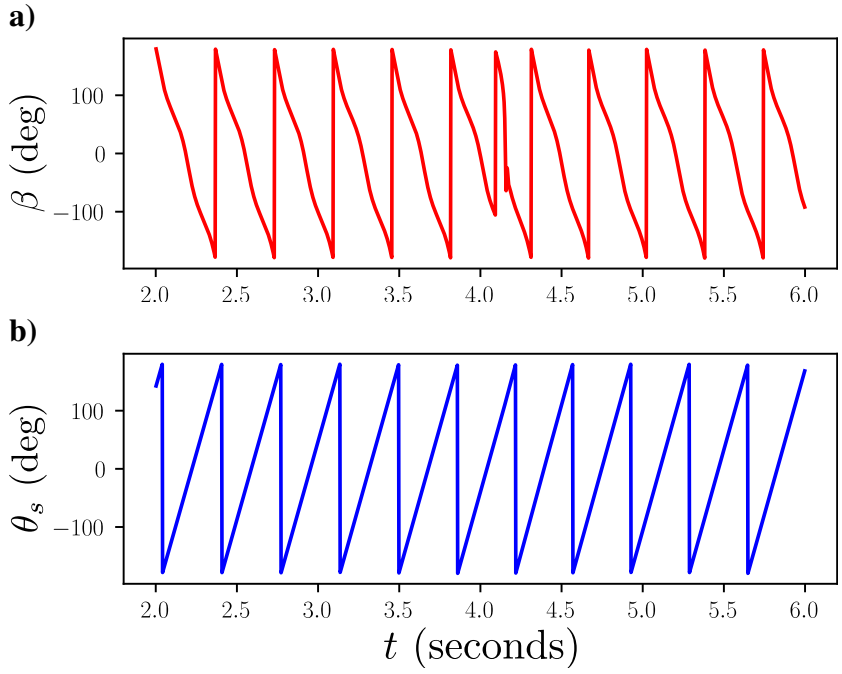

Figure 7: **a)** Motor phase β and **b)** robot's spin θ_s from $t = 2s$ to $t = 6s$. The swashplateless phase increases at the same rate as the spin angle, but in the inverse direction. It is also shown (at $t = 4.1s$) how the motor phase suddenly shifts when the goal orientation changes.

7.3. Simulation 2: vertical stability with random and impulsive noises

In this second scenario, the spinning robot is supposed to reject fast disturbances similar to gust of wind applied to the pitch and roll axes. As shown in Figure 9, the quaternion-based control strategy (see Section 6) allows the robot to reject 90% of the disturbance within about 500ms. This 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 **a)** and **b)** we can see how the controller manages to compensate for these impulses. Figure 9 **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

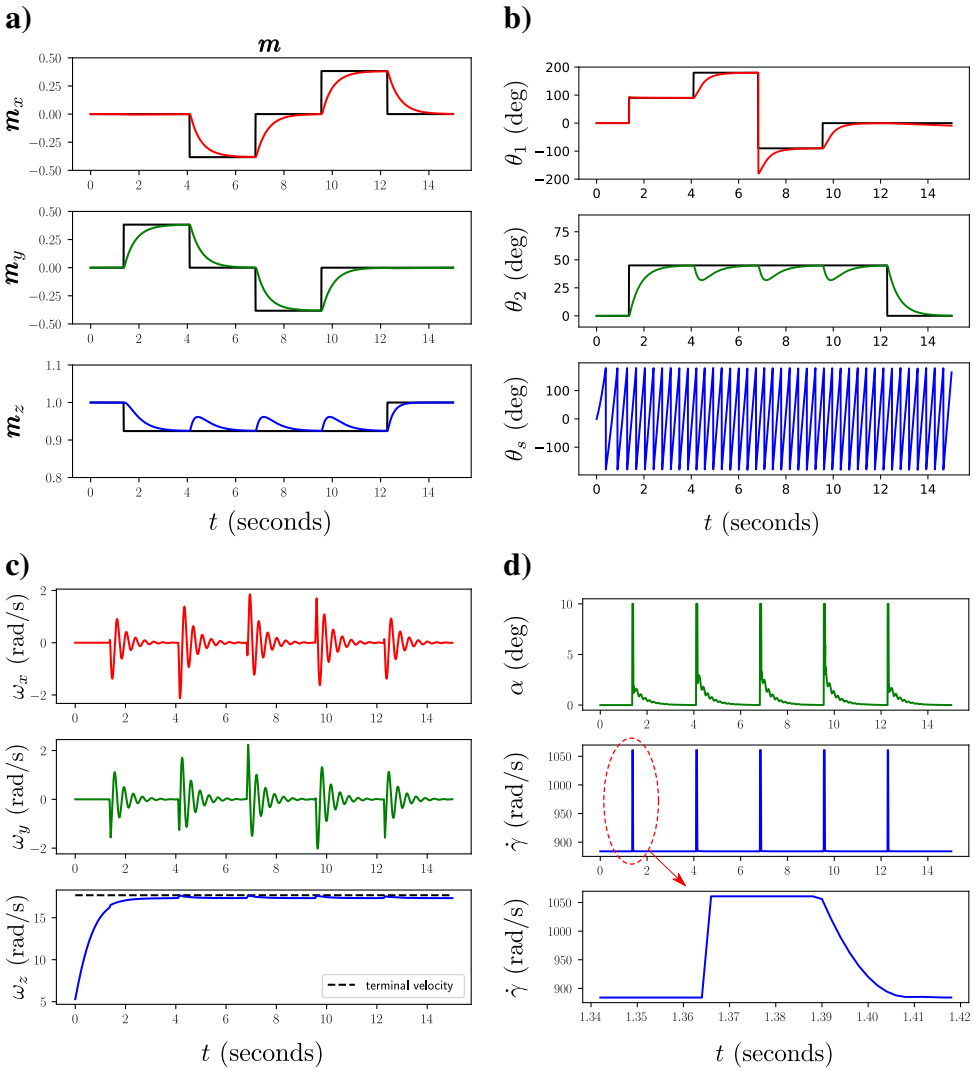

Figure 8: Responses of the simulated mono spinner to changes in attitude angles. **a)** Time course of each component of $\boldsymbol{m}$. **b)** shows the Euler angles representation of the orientation on the ZYZ sequence: θ_1 is the precession (first rotation around the z-axis, representing the direction 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.

usually applied to the orientation estimators, but an additional low pass filter might also be necessary in the controller output.

7.4. Analysis of the middle normal vector's projection

The normal middle vector $\boldsymbol{m}$ also opens up a new opportunity for the visualization of the results. This vector can be separated into two components:

$$\begin{aligned}(\boldsymbol{e}_z \cdot \boldsymbol{m})\boldsymbol{e}_z &= \begin{bmatrix}0 & 0 & \boldsymbol{m}_z\end{bmatrix}^T, \text{ component in the direction of } \boldsymbol{e}_z \\ -\widehat{\boldsymbol{e}_z}^{\,2}\boldsymbol{m} &= \begin{bmatrix}\boldsymbol{m}_x & \boldsymbol{m}_y & 0\end{bmatrix}^T, \text{ component orthogonal to } \boldsymbol{e}_z \end{aligned} \tag{95}$$

By definition, we know that $|\boldsymbol{m}| = 1$ and $\boldsymbol{e}_z \cdot \boldsymbol{m} \geq 0$. The component of $\boldsymbol{m}$ in the z-axis is then completely defined by the other components:

$$\begin{aligned}\boldsymbol{e}_z \cdot \boldsymbol{m} &= +\sqrt{1 - |\widehat{\boldsymbol{e}_z}^{\,2}\boldsymbol{m}|^2} \\ \boldsymbol{m}_z &= +\sqrt{1 - \boldsymbol{m}_x^2 - \boldsymbol{m}_y^2}\end{aligned} \tag{96}$$

Meaning that the orthogonal components $(\boldsymbol{m}_x, \boldsymbol{m}_y)$ completely define the reduced orientation with only two values.

8. Conclusion

Recent studies on unconventional unmanned flying vehicles have shed a new light on their possible advantages, namely reduced cost and better power efficiency. Moreover, the rise of thrust cyclic flapping by torque modulation makes it more accessible to construct even more minimalist flying drones without fully sacrificing their controllability.

In this work, the design of a cylindrical mono-rotor drone using a swashplateless torque modulation system is extensively studied. A comprehensive non-linear dynamic model is studied using a very small amount of simplifications, with the help of the Euler-Poincaré equation. An alternative representation of the drone's attitude is demonstrated and used, showing a novel way to decompose the orientation into two parts: an uncontrollable spin component and a simple vector representing the reduced, controllable part of the orientation. This allowed us to explicitly separate and state the controllable 2 degrees of freedom of the rotation (equivalent to the roll and pitch) from the uncontrollable degree of freedom of the spin. A non-linear reduced attitude controller is then developed, with the help of this decomposition, and an easily implementable strategy is designed based on

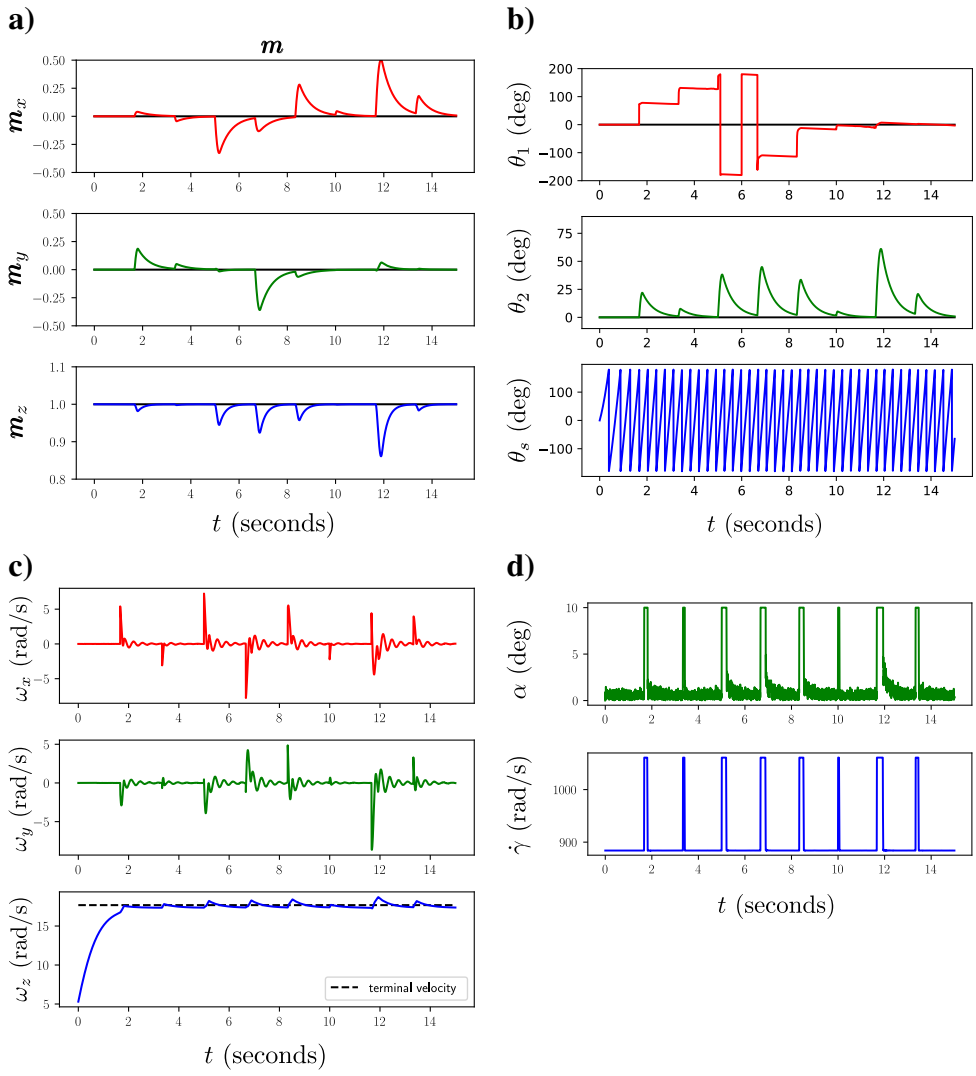

Figure 9: Responses of the simulated mono spinner to perturbations. **a)** Time course of each component of $\boldsymbol{m}$. **b)** Euler angles in ZYZ sequence. **c)** Angular velocity. **d)** Motor inclination, phase and rotation velocity.

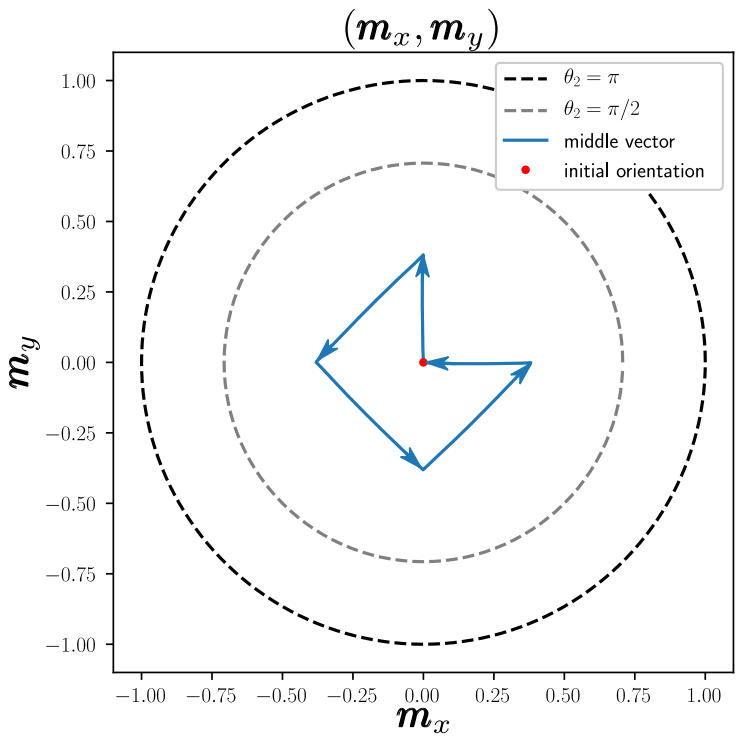

Figure 10: Projection of $\boldsymbol{m}$ in the XY plane for Simulation 1 of the mono-rotor. The dashed circles indicate the possible values of $(\boldsymbol{m}_x, \boldsymbol{m}_y)$ when $\theta_2 = \pi$ (in black) and $\theta_2 = \pi/2$ (in gray).

this controller. The whole model including the controller was fully implemented in Python. The source code of the simulation software is fully available. The model was shown to correctly work even using a simple Euler integration method.

This paper presents a complete theoretical analysis for a spinning single rotor drone, and in the future, a prototype should be built in order to validate the theoretical and simulation results presented here. An important problem that may rise is the orientation estimation problem: having a robot spinning constantly may create a drift in the robot's spin (or yaw) estimation. However, real time attitude estimation based on novel Kalman-based filters have shown to be efficient and robust to the spinning [34]. A full and precise estimation is crucial for the correct use of the control. All these points open full research fields for the development of future autonomous spinning vehicles.

Acknowledgments

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).

Appendix A. Quaternion algebra summary

The 3D orientation can 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:

$$\mathrm{q} = q_\mathrm{r} + q_\mathrm{x} i + q_\mathrm{y} j + q_\mathrm{z} k \tag{A.1}$$

Where $q_\mathrm{r}, q_\mathrm{x}, q_\mathrm{y}, q_\mathrm{z} \in \mathbb{R}$. And all of the properties of quaternions can be derived using: $i^2 = j^2 = k^2 = ijk = -1$. For simplicity, quaternions are written in this work as 4×1 vectors:

$$q = \begin{bmatrix} q_\mathrm{r} \\ q_\mathrm{x} \\ q_\mathrm{y} \\ q_\mathrm{z} \end{bmatrix} = \begin{bmatrix} q_\mathrm{r} \\ \boldsymbol{q} \end{bmatrix} \tag{A.2}$$

Where 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. The Hamilton product between two quaternions in 4-vector form will be written

as[3].

$$q \odot p = \begin{bmatrix} q_\mathrm{r} \\ \boldsymbol{q} \end{bmatrix} \odot \begin{bmatrix} p_\mathrm{r} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} q_\mathrm{r} p_\mathrm{r} - \boldsymbol{q} \cdot \boldsymbol{p} \\ q_\mathrm{r} \boldsymbol{p} + p_\mathrm{r} \boldsymbol{q} + \boldsymbol{q} \times \boldsymbol{p} \end{bmatrix} \tag{A.3}$$

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:

$$q^{-1} = \frac{q^*}{|q|} \tag{A.4}$$

And for any quaternion q:

$$q \odot q^{-1} = q^{-1} \odot q = \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} \tag{A.5}$$

Appendix B. Kinetic energy analysis

In this section, the kinetic energy of the three distinct parts of the mono-rotor will be analyzed.

Appendix B.1. Main body

The body's kinetic energy is trivial:

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

For sake of clarity, we expand Eq. B.1 in the following form:

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

[3] When $\odot$ notation is not used, consider common matrix/vector products.

Appendix B.2. Rotor

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:

$$g_i^j = \begin{bmatrix} R_i^j & \boldsymbol{s}_i^j \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{B.3}$$

Where s_i^j is the position of $\mathcal{F}_i$ in $\mathcal{F}_j$ and R_i^j is the rotation matrix from $\mathcal{F}_i$ to $\mathcal{F}_j$, we define the Ad operator [28]:

$$\mathrm{Ad}_i^j = \begin{bmatrix} R_i^j & \mathbf{0} \\ \widehat{\boldsymbol{s}_i^j} R_i^j & R_i^j \end{bmatrix} \tag{B.4}$$

And the inverse operation is:

$$\mathrm{Ad}_j^i = \begin{bmatrix} \left(R_i^j\right)^T & 0 \\ -\left(R_i^j\right)^T \widehat{\boldsymbol{s}_i^j} & \left(R_i^j\right)^T \end{bmatrix} \tag{B.5}$$

Moreover, by composition of velocities, we can state that:

$$\eta_j^j = \mathrm{Ad}_i^j \, \eta_i^i + \eta_{j/i}^j \tag{B.6}$$

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:

$$\begin{aligned} \eta_P^P &= \left(\mathrm{Ad}_B^P\right) \eta_B^B + \eta_{P/B}^P \\ &= \left(\mathrm{Ad}_P^B\right)^{-1} \eta_B^B + \eta_{P/B}^P \\ \begin{bmatrix} \boldsymbol{\Omega}_p^P \\ \boldsymbol{v}_p^P \end{bmatrix} &= \begin{bmatrix} \left(R_P^B\right)^T & 0 \\ -h_P \left(R_P^B\right)^T \widehat{\boldsymbol{e}_z} & \left(R_P^B\right)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_b^B \\ \boldsymbol{v}_b^B \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Omega}_{P/B}^P \\ \mathbf{0} \end{bmatrix} \end{aligned} \tag{B.7}$$

Which leads to:

$$\begin{aligned} \boldsymbol{\Omega}_p^P &= \left(R_P^B\right)^T \boldsymbol{\omega}_b^B + \boldsymbol{\Omega}_{P/B}^P \\ \boldsymbol{v}_p^P &= \left(R_P^B\right)^T \boldsymbol{v}_b^B - h_P \left(R_P^B\right)^T \widehat{\boldsymbol{e}_z} \boldsymbol{\omega}_b^B \end{aligned} \tag{B.8}$$

And the rotor's kinetic energy is:

$$\begin{aligned} T_P &= \frac{m_p}{2}\left(\boldsymbol{v}_p^P\right)^T \boldsymbol{v}_p^P + \frac{1}{2}\left(\boldsymbol{\Omega}_p^P\right)^T J_p^P\, \boldsymbol{\Omega}_p^P \\ &= \frac{1}{2}\begin{bmatrix}\boldsymbol{\Omega}_p^P \\ \boldsymbol{v}_p^P\end{bmatrix}^T \begin{bmatrix} J_p^P & \mathbf{0} \\ \mathbf{0} & m_p\mathbb{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{\Omega}_p^P \\ \boldsymbol{v}_p^P\end{bmatrix} \end{aligned} \tag{B.9}$$

Introducing Eq. B.7 into Eq. B.9, we can state:

$$T_P = A + B + C + D \tag{B.10}$$

In which:

$$\begin{aligned} A &= \frac{1}{2}\begin{bmatrix}\boldsymbol{\omega}_b^B \\ \boldsymbol{v}_b^B\end{bmatrix}^T \left(\mathrm{Ad}_B^P\right)^T \begin{bmatrix} J_p^P & \mathbf{0} \\ \mathbf{0} & m_p\mathbb{I}\end{bmatrix}\left(\mathrm{Ad}_B^P\right)\begin{bmatrix}\boldsymbol{\omega}_b^B \\ \boldsymbol{v}_b^B\end{bmatrix} \\ &= \frac{1}{2}\begin{bmatrix}\boldsymbol{\omega}_b^B \\ \boldsymbol{v}_b^B\end{bmatrix}^T \begin{bmatrix} R_P^B J_p^P \left(R_P^B\right)^T - m_p h_P^2 \left(\widehat{\boldsymbol{e}}_z\right)^2 & m_p h_P \widehat{\boldsymbol{e}}_z \\ -m_p h_P \widehat{\boldsymbol{e}}_z & m_p\mathbb{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{\omega}_b^B \\ \boldsymbol{v}_b^B\end{bmatrix} \end{aligned} \tag{B.11}$$

The second term is:

$$\begin{aligned} B &= \frac{1}{2}\begin{bmatrix}\boldsymbol{\Omega}_{P/B}^P \\ \mathbf{0}\end{bmatrix}^T \begin{bmatrix} J_p^P & \mathbf{0} \\ \mathbf{0} & m_p\mathbb{I}\end{bmatrix}\begin{bmatrix} \left(R_P^B\right)^T & \mathbf{0} \\ -h_P\left(R_P^B\right)^T \widehat{\boldsymbol{e}}_z & \left(R_P^B\right)^T\end{bmatrix}\begin{bmatrix}\boldsymbol{\omega}_b^B \\ \boldsymbol{v}_b^B\end{bmatrix} \\ &= \frac{1}{2}\left(\boldsymbol{\Omega}_{P/B}^P\right)^T \left(R_P^B J_p^P\right)^T \boldsymbol{\omega}_b^B \end{aligned} \tag{B.12}$$

The third term is:

$$\begin{aligned} C &= \frac{1}{2}\begin{bmatrix}\boldsymbol{\omega}_b^B \\ \boldsymbol{v}_b^B\end{bmatrix}^T \begin{bmatrix} R_P^B & h_P \widehat{\boldsymbol{e}}_z R_P^B \\ \mathbf{0} & R_P^B\end{bmatrix}\begin{bmatrix} J_p^P & \mathbf{0} \\ \mathbf{0} & m_p\mathbb{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{\Omega}_{P/B}^P \\ \mathbf{0}\end{bmatrix} \\ &= \frac{1}{2}\left(\boldsymbol{\omega}_b^B\right)^T \left(R_P^B J_p^P\right)\boldsymbol{\Omega}_{P/B}^P \end{aligned} \tag{B.13}$$

And the fourth and final term is:

$$D = \frac{1}{2}\left(\boldsymbol{\Omega}_{P/B}^P\right)^T J_p^P\, \boldsymbol{\Omega}_{P/B}^P \tag{B.14}$$

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

$$T_P = \frac{1}{2}\begin{bmatrix}\boldsymbol{\omega}_b^B \\ \boldsymbol{v}_b^B \\ \boldsymbol{\Omega}_{P/B}^P\end{bmatrix}^T \begin{bmatrix} J_p^B - m_p h_P^2 \left(\widehat{\boldsymbol{e}}_z\right)^2 & m_p h_P \widehat{\boldsymbol{e}}_z & R_P^B J_p^P \\ -m_p h_P \widehat{\boldsymbol{e}}_z & m_b\mathbb{I} & \mathbf{0} \\ \left(R_P^B J_p^P\right)^T & \mathbf{0} & J_p^P\end{bmatrix}\begin{bmatrix}\boldsymbol{\omega}_b^B \\ \boldsymbol{v}_b^B \\ \boldsymbol{\Omega}_{P/B}^P\end{bmatrix} \tag{B.15}$$

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

Appendix B.3. Power supply

The power supply mass and frame can also be analyzed in the exact same way as the propeller using Eqs. B.4 and B.6, with translation $-h_S \boldsymbol{e}_z$, rotation $R_S^B = \mathbb{I}$ and $\eta_{S/B}^S = \mathbf{0}$:

$$\begin{aligned} \eta_S^S &= \left(\mathrm{Ad}_B^S\right)^{-1} \eta_B^B + \eta_{S/B}^P \\ \begin{bmatrix} \boldsymbol{\Omega}_s^S \\ \boldsymbol{v}_s^S \end{bmatrix} &= \begin{bmatrix} \mathbb{I} & 0 \\ h_S \widehat{\boldsymbol{e}_z} & \mathbb{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_b^B \\ \boldsymbol{v}_b^B \end{bmatrix} \end{aligned} \tag{B.16}$$

Which leads to:

$$\begin{aligned} \boldsymbol{\Omega}_s^S &= \boldsymbol{\omega}_b^B \\ \boldsymbol{v}_s^S &= h_S \widehat{\boldsymbol{e}_z} \boldsymbol{\omega}_b^B + \boldsymbol{v}_b^B \end{aligned} \tag{B.17}$$

And the power supply's kinetic energy is:

$$\begin{aligned} T_S &= \frac{m_p}{2} \left(\boldsymbol{v}_s^S\right)^T \boldsymbol{v}_s^S + \frac{1}{2} \left(\boldsymbol{\Omega}_s^S\right)^T J_s^S \boldsymbol{\Omega}_s^S \\ &= \frac{1}{2} \begin{bmatrix} \boldsymbol{\Omega}_s^S \\ \boldsymbol{v}_s^S \end{bmatrix}^T \begin{bmatrix} J_s^S & \mathbf{0} \\ \mathbf{0} & m_s \mathbb{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Omega}_s^S \\ \boldsymbol{v}_s^S \end{bmatrix} \end{aligned} \tag{B.18}$$

Introducing Eq. B.16 into Eq. B.18, and expanding it as before, we get:

$$\begin{aligned} T_S &= \frac{1}{2} \begin{bmatrix} \boldsymbol{\omega}_b^B \\ \boldsymbol{v}_b^B \end{bmatrix}^T \begin{bmatrix} \mathbb{I} & -h_S \widehat{\boldsymbol{e}_z} \\ \mathbf{0} & \mathbb{I} \end{bmatrix} \begin{bmatrix} J_s^S & \mathbf{0} \\ \mathbf{0} & m_s \mathbb{I} \end{bmatrix} \begin{bmatrix} \mathbb{I} & \mathbf{0} \\ h_S \widehat{\boldsymbol{e}_z} & \mathbb{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_b^B \\ \boldsymbol{v}_b^B \end{bmatrix} \\ &= \frac{1}{2} \begin{bmatrix} \boldsymbol{\omega}_b^B \\ \boldsymbol{v}_b^B \\ \boldsymbol{\Omega}_{P/B}^P \end{bmatrix}^T \begin{bmatrix} J_s^S - m_s h_S^2 \left(\widehat{\boldsymbol{e}_z}\right)^2 & -m_s h_S \widehat{\boldsymbol{e}_z} & \mathbf{0} \\ m_s h_S \widehat{\boldsymbol{e}_z} & m_s \mathbb{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_b^B \\ \boldsymbol{v}_b^B \\ \boldsymbol{\Omega}_{P/B}^P \end{bmatrix}^T \end{aligned} \tag{B.19}$$

Appendix B.4. Total kinetic energy

The expression of the total kinetic is:

$$T = T_B + T_P + T_S \tag{B.20}$$

Defining $\eta = \begin{bmatrix} \boldsymbol{\omega}_b^B \\ \boldsymbol{v}_b^B \end{bmatrix}$ as the inertial twist of the body, $\boldsymbol{\Omega} = \boldsymbol{\Omega}_{P/B}^P$ as the rotor's angular velocity w.r.t. the $\mathcal{F}_B$. The extra inertia component created by the point masses is:

$$\begin{aligned} J_m^B &= (m_s h_S^2 + m_p h_P^2)\left(-\widehat{\boldsymbol{e}_z}^2\right) \\ &= (m_s h_S^2 + m_p h_P^2)\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{B.21}$$

Define $\boldsymbol{J} = J_b^B + J_s^B + J_p^B + J_m^B$ and $m = m_b + m_s + m_p$ the sum of all masses. Inserting Eqs. B.2, B.15 and B.19 into Eq. B.20, and defining $\Delta_m = m_s h_S - m_p h_P$[4], we get:

$$T = \frac{1}{2}\begin{bmatrix} \eta \\ \boldsymbol{\Omega} \end{bmatrix}^T \mathbf{M} \begin{bmatrix} \eta \\ \boldsymbol{\Omega} \end{bmatrix} \tag{B.22}$$

Where:

$$\mathbf{M} = \begin{bmatrix} \boldsymbol{J} & -\Delta_m \widehat{\boldsymbol{e}_z} & R_P^B J_p^P \\ \Delta_m \widehat{\boldsymbol{e}_z} & m\,\mathbb{I} & 0 \\ \left(R_P^B J_p^P\right)^T & 0 & J_p^P \end{bmatrix} \tag{B.23}$$

Appendix C. Quaternion decomposition into middle normal vector and spin angle

In this section, we will derive a novel decomposition of the rotation quaternion into two different components: one representing the uncontrollable spin angle, and the other representing the two remaining controllable degrees of freedom.

Appendix C.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

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

know that:

$$\begin{aligned}\begin{bmatrix}0\\ \boldsymbol{e}_b^E\end{bmatrix} &= q \odot \begin{bmatrix}0\\ \boldsymbol{e}_b^B\end{bmatrix} \odot q^* \\ &= q \odot \begin{bmatrix}0\\ \boldsymbol{e}\end{bmatrix} \odot q^*\end{aligned} \tag{C.1}$$

Or simply $\boldsymbol{e}_b^E = R\boldsymbol{e}_b^B = R\boldsymbol{e}$, where $R = R(q)$, which gives:

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

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

$$\dot{q} = \frac{1}{2} q \odot \begin{bmatrix}0\\ \boldsymbol{\omega}\end{bmatrix} \tag{C.3}$$

Similarly to what was done in [35], but in a more general form, we will suppose the existence of two unit quaternions, q_a, the "reduced attitude quaternion" (also known as the "swing") and q_s, the "spin quaternion" (also known as the "twist"):

$$\begin{aligned}q_a = \begin{bmatrix}a_\mathrm{r}\\ \boldsymbol{a}\end{bmatrix},\ q_s = \begin{bmatrix}s_\mathrm{r}\\ \boldsymbol{s}\end{bmatrix} \\ q = q_a \odot q_s\end{aligned} \tag{C.4}$$

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

$$\begin{aligned}\dot{q}_a &= \frac{1}{2} q_a \odot \begin{bmatrix}0\\ \boldsymbol{\omega}_a\end{bmatrix} \\ \dot{q}_s &= \frac{1}{2} q_s \odot \begin{bmatrix}0\\ \boldsymbol{\omega}_s\end{bmatrix}\end{aligned} \tag{C.5}$$

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

$$|q_a|^2 = |q_s|^2 = 1 \tag{C.6}$$

Appendix C.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 under-determined. A natural choice for this constraint, to assure that no rotation

around $\boldsymbol{e}$ is done with q_a, such that $\boldsymbol{a} \cdot \boldsymbol{e} = 0$. This is a generalization of one of the constraints used in [35]. Using this constraint:

$$q_a \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} = \begin{bmatrix} 0 \\ (a_{\mathrm{r}}\mathbb{I} + \widehat{\boldsymbol{a}})\, \boldsymbol{e} \end{bmatrix} \tag{C.7}$$

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

$$q_a = -\begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \tag{C.8}$$

Morever, since $|q_a| = |\boldsymbol{e}| = 1$, we know that $|\boldsymbol{m}| = 1$. Introducing Eq. C.8 into Eq. C.1:

$$\begin{aligned} \begin{bmatrix} 0 \\ \boldsymbol{e}_b^E \end{bmatrix} &= q \odot \begin{bmatrix} 0 \\ \boldsymbol{e}_b^B \end{bmatrix} \odot q^* \\ &= q_a \odot \begin{bmatrix} 0 \\ \boldsymbol{e}_b^B \end{bmatrix} \odot q_a^* \\ &= -\begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e}_b^B \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \end{aligned} \tag{C.9}$$

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

$$\begin{aligned} -\begin{bmatrix} 0 \\ \boldsymbol{e}_b^E \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} &= -\begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e}_b^B \end{bmatrix} \\ \begin{bmatrix} \boldsymbol{m} \cdot \boldsymbol{e}_b^B \\ \boldsymbol{m} \times \boldsymbol{e}_b^B \end{bmatrix} &= \begin{bmatrix} \boldsymbol{e}_b^E \cdot \boldsymbol{m} \\ \boldsymbol{e}_b^E \times \boldsymbol{m} \end{bmatrix} \end{aligned} \tag{C.10}$$

From Eq. C.10 we know that:

$$\begin{aligned} \boldsymbol{m} \times \boldsymbol{e}_b^B &= \boldsymbol{e}_b^E \times \boldsymbol{m} \\ \boldsymbol{m} \times \left(\boldsymbol{e}_b^E + \boldsymbol{e}_b^B\right) &= \boldsymbol{0} \\ \boldsymbol{m} \times ((R + \mathbb{I})\, \boldsymbol{e}) &= \boldsymbol{0} \end{aligned} \tag{C.11}$$

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

$$\boldsymbol{m} = \lambda\, (R + \mathbb{I})\, \boldsymbol{e} \tag{C.12}$$

Choosing $\lambda > 0$[5] and noting that $|\boldsymbol{m}| = 1$:

$$\lambda = \frac{|\boldsymbol{m}|}{|(R + \mathbb{I})\,\boldsymbol{e}|} = \frac{1}{|(R + \mathbb{I})\,\boldsymbol{e}|} \tag{C.13}$$

And finally:

$$\boldsymbol{m} = \frac{\boldsymbol{e}_b^E + \boldsymbol{e}_b^B}{\left|\boldsymbol{e}_b^E + \boldsymbol{e}_b^B\right|} = \frac{(R + \mathbb{I})\,\boldsymbol{e}}{|(R + \mathbb{I})\,\boldsymbol{e}|} \tag{C.14}$$

And:

$$\begin{aligned} |(R + \mathbb{I})\,\boldsymbol{e}| &= \sqrt{2\,(\boldsymbol{e} \cdot R\boldsymbol{e} + 1)} \\ &= 2\sqrt{1 - |\boldsymbol{e} \times \boldsymbol{q}|^2} \\ &= 2\sqrt{q_{\mathrm{r}}^2 + (\boldsymbol{e} \cdot \boldsymbol{q})^2} \end{aligned} \tag{C.15}$$

Finally, the attitude quaternion q_a can be given as:

$$q_a = -\begin{bmatrix} 0 \\ \boldsymbol{m} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \tag{C.16}$$

Note that this equation has a singularity when $\boldsymbol{e}_b^E = -\boldsymbol{e}_b^B = -\boldsymbol{e}$ ($\boldsymbol{m}$ is undetermined). For $\boldsymbol{e} = \boldsymbol{e}_z$, we can see on Fig. C.11 the domains of both $\boldsymbol{e}_b^E$ and $\boldsymbol{m}$.

Appendix C.3. Spin quaternion

Considering that q_s is the quaternion representing only the rotation that creates the constant spin around $\boldsymbol{e}$:

$$\begin{aligned} q_s \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \odot q_s^* &= \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \\ q_s \odot \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} &= \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \odot q_s \\ \begin{bmatrix} \boldsymbol{e} \cdot \boldsymbol{s} \\ \boldsymbol{e} \times \boldsymbol{s} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{e} \cdot \boldsymbol{s} \\ -\boldsymbol{e} \times \boldsymbol{s} \end{bmatrix} \end{aligned} \tag{C.17}$$

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

$$\boldsymbol{e} \times \boldsymbol{s} = \boldsymbol{0} \tag{C.18}$$

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

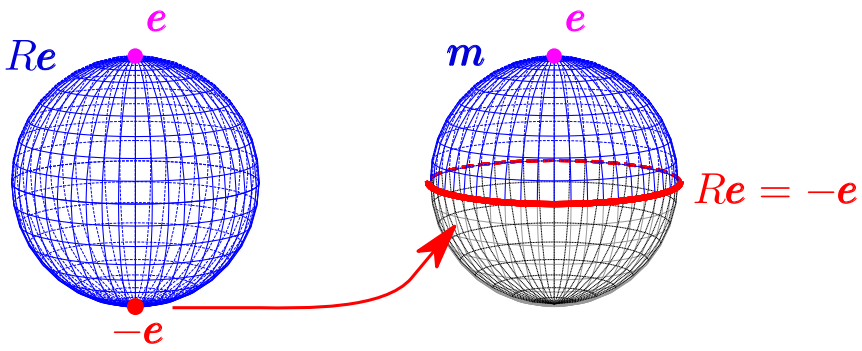

Figure C.11: On the left, the domain (in blue) of $\boldsymbol{e}_b^E$, with a red dot representing the singularity point where $\boldsymbol{e}_b^E = -\boldsymbol{e}$. On the right, the upper hemisphere represents the domain of $\boldsymbol{m}$. We can see that the whole equator between both hemispheres correspond to the case $\boldsymbol{e}_b^E = -\boldsymbol{e}$, which means that $\boldsymbol{m}$ is undefined in this case.

So we know that, for a real scalar s:

$$\boldsymbol{s} = s\boldsymbol{e} \tag{C.19}$$

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

$$q_s = \begin{bmatrix} c \\ s\boldsymbol{e} \end{bmatrix} \tag{C.20}$$

With the constraint that $|q_s|^2 = c^2 + s^2 = 1$. This leads to the natural definition as:

$$\begin{aligned} c &= \cos\theta_s/2 \\ s &= \sin\theta_s/2 \end{aligned} \tag{C.21}$$

But to find these parameters for a given q, we can use C.4 and C.16 to find that:

$$\begin{aligned} q_s &= q_a^* \odot q \\ |(R+\mathbb{I})\,\boldsymbol{e}| \begin{bmatrix} 0 \\ \boldsymbol{e} \end{bmatrix} \odot \begin{bmatrix} c \\ s\boldsymbol{e} \end{bmatrix} &= \begin{bmatrix} 0 \\ (R+\mathbb{I})\,\boldsymbol{e} \end{bmatrix} \odot \begin{bmatrix} q_\mathrm{r} \\ \boldsymbol{q} \end{bmatrix} \\ |(R+\mathbb{I})\,\boldsymbol{e}| \begin{bmatrix} -s \\ c\boldsymbol{e} \end{bmatrix} &= \begin{bmatrix} -\boldsymbol{q}\cdot(R+\mathbb{I})\,\boldsymbol{e} \\ (q_\mathrm{r}\mathbb{I} - \widehat{\boldsymbol{q}})\,(R+\mathbb{I})\,\boldsymbol{e} \end{bmatrix} \end{aligned} \tag{C.22}$$

Remembering Eq. C.2:

$$\begin{aligned} (R+\mathbb{I})\,\boldsymbol{e} &= \left(\mathbb{I} + \mathbb{I} + 2\left(q_\mathrm{r}\widehat{\boldsymbol{q}} + \widehat{\boldsymbol{q}}^2\right)\right)\boldsymbol{e} \\ &= 2\left(\mathbb{I} + q_\mathrm{r}\widehat{\boldsymbol{q}} + \widehat{\boldsymbol{q}}^2\right)\boldsymbol{e} \end{aligned} \tag{C.23}$$

To find s:

$$\begin{aligned} |(R + \mathbb{I})\, \boldsymbol{e}|\, s &= \boldsymbol{q} \cdot (R + \mathbb{I})\, \boldsymbol{e} \\ &= 2\boldsymbol{q} \cdot \boldsymbol{e} \end{aligned} \tag{C.24}$$

And finally:

$$s = \frac{2}{|(R + \mathbb{I})\, \boldsymbol{e}|} \boldsymbol{q} \cdot \boldsymbol{e} \tag{C.25}$$

For c:

$$|(R + \mathbb{I})\, \boldsymbol{e}|\, c\boldsymbol{e} = (q_{\mathrm{r}}\mathbb{I} - \widehat{\boldsymbol{q}})\, (R + \mathbb{I})\, \boldsymbol{e} \tag{C.26}$$

After some tedious algebra, we find that:

$$\begin{aligned} |(R + \mathbb{I})\, \boldsymbol{e}|\, c &= 2\boldsymbol{e}^T \left(q_{\mathrm{r}}\mathbb{I} + 2\widehat{\boldsymbol{q}}\right) \boldsymbol{e} \\ &= 2q_{\mathrm{r}} \end{aligned} \tag{C.27}$$

Which gives:

$$c = \frac{2}{|(R + \mathbb{I})\, \boldsymbol{e}|} q_{\mathrm{r}} \tag{C.28}$$

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

$$\begin{aligned} q_s &= \frac{2}{|(R + \mathbb{I})\, \boldsymbol{e}|} \begin{bmatrix} q_{\mathrm{r}} \\ (\boldsymbol{q} \cdot \boldsymbol{e})\boldsymbol{e} \end{bmatrix} \\ &= \frac{1}{\sqrt{q_{\mathrm{r}}^2 + (\boldsymbol{q} \cdot \boldsymbol{e})^2}} \begin{bmatrix} q_{\mathrm{r}} \\ (\boldsymbol{q} \cdot \boldsymbol{e})\boldsymbol{e} \end{bmatrix} \end{aligned} \tag{C.29}$$

Which gives:

$$\theta_s = 2\ \mathrm{atan2}(\boldsymbol{q} \cdot \boldsymbol{e},\ q_{\mathrm{r}}) \tag{C.30}$$

Appendix C.4. Derivatives

In this section, we will analyze how to decompose the derivative of the full rotation quaternion.

Appendix C.4.1. Derivative of spin quaternion

Analyzing the derivative of q_s:

$$\begin{aligned}\dot{q}_s &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\begin{bmatrix} c \\ s\boldsymbol{e} \end{bmatrix}\right) \\ &= \frac{1}{2}\begin{bmatrix} -s \\ c\boldsymbol{e} \end{bmatrix}\dot{\theta}_s \\ &= \frac{1}{2}\begin{bmatrix} c \\ s\boldsymbol{e} \end{bmatrix} \odot \begin{bmatrix} 0 \\ \dot{\theta}_s\boldsymbol{e} \end{bmatrix}\end{aligned} \tag{C.31}$$

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

$$\dot{q}_s = \frac{1}{2} q_s \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega}_s \end{bmatrix} \tag{C.32}$$

Using Eq. C.30:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}(\tan\theta_s/2) &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\boldsymbol{q}\cdot\boldsymbol{e}}{q_\mathrm{r}}\right) \\ \frac{1}{2c^2}\dot{\theta}_s &= \frac{q_\mathrm{r}\dot{\mathbf{q}}_v - \dot{q}_r\boldsymbol{q}}{q_r^2}\cdot\boldsymbol{e}\end{aligned} \tag{C.33}$$

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:

$$\begin{aligned}\frac{q_r^2}{c^2}\dot{\theta}_s &= \left(q_\mathrm{r}(q_\mathrm{r}\mathbb{I} + \widehat{\boldsymbol{q}})\boldsymbol{\omega} + (\boldsymbol{\omega}\cdot\boldsymbol{q})\boldsymbol{q}\right)\cdot\boldsymbol{e} \\ &= \boldsymbol{\omega}^T\left(\mathbb{I} - q_\mathrm{r}\widehat{\boldsymbol{q}} + \widehat{\boldsymbol{q}}^2\right)\boldsymbol{e} \\ &= \frac{1}{2}\boldsymbol{\omega}^T\left(\mathbb{I} + R\right)^T\boldsymbol{e} \\ &= \frac{1}{2}\boldsymbol{\omega}^T R^T\left(R + \mathbb{I}\right)\boldsymbol{e}\end{aligned} \tag{C.34}$$

And according to Eqs. C.23 and C.28:

$$\begin{aligned}\dot{\theta}_s &= \frac{c^2}{q_\mathrm{r}^2}\boldsymbol{\omega} \cdot \frac{R^T((R+\mathbb{I})\,\boldsymbol{e})}{2} \\ \dot{\theta}_s &= 2\,\frac{((R+\mathbb{I})\,\boldsymbol{e})}{|(R+\mathbb{I})\,\boldsymbol{e}|^2} \cdot R\boldsymbol{\omega} \\ &= \frac{\boldsymbol{m}\cdot R\boldsymbol{\omega}}{\boldsymbol{m}\cdot\boldsymbol{e}}\end{aligned} \tag{C.35}$$

And we can finally state that:

$$\boldsymbol{\omega}_s = \left(\frac{\boldsymbol{m}\cdot R\boldsymbol{\omega}}{\boldsymbol{m}\cdot\boldsymbol{e}}\right)\boldsymbol{e} \tag{C.36}$$

Appendix C.4.2. Derivative of $\boldsymbol{m}$

By definition:

$$\begin{bmatrix}0\\ \boldsymbol{m}\end{bmatrix} = -q\odot q_s^*\odot\begin{bmatrix}0\\ \boldsymbol{e}\end{bmatrix}^* \tag{C.37}$$

Which leads to:

$$\begin{aligned}\begin{bmatrix}0\\ \dot{\boldsymbol{m}}\end{bmatrix} &= -\dot{q}\odot q_s^*\odot\begin{bmatrix}0\\ \boldsymbol{e}\end{bmatrix}^* - q\odot\dot{q}_s{}^*\odot\begin{bmatrix}0\\ \boldsymbol{e}\end{bmatrix}^* \\ &= \frac{1}{2}q\odot\begin{bmatrix}0\\ \boldsymbol{\omega}-\boldsymbol{\omega}_s\end{bmatrix}q^*\odot\begin{bmatrix}0\\ \boldsymbol{m}\end{bmatrix} \\ &= \frac{1}{2}\begin{bmatrix}0\\ R(\boldsymbol{\omega}-\boldsymbol{\omega}_s)\end{bmatrix}\odot\begin{bmatrix}0\\ \boldsymbol{m}\end{bmatrix}\end{aligned} \tag{C.38}$$

Which gives:

$$\begin{aligned}\begin{bmatrix}0\\ \dot{\boldsymbol{m}}\end{bmatrix} &= \frac{1}{2}\begin{bmatrix}0\\ R(\boldsymbol{\omega}-\boldsymbol{\omega}_s)\end{bmatrix}\odot\begin{bmatrix}0\\ \boldsymbol{m}\end{bmatrix} \\ &= \frac{1}{2}\begin{bmatrix}-\boldsymbol{m}\cdot R(\boldsymbol{\omega}-\boldsymbol{\omega}_s)\\ -\boldsymbol{m}\times R(\boldsymbol{\omega}-\boldsymbol{\omega}_s)\end{bmatrix}\end{aligned} \tag{C.39}$$

Which finally gives:

$$\dot{\boldsymbol{m}} = -\frac{1}{2}\boldsymbol{m}\times R\,(\boldsymbol{\omega}-\boldsymbol{\omega}_s) \tag{C.40}$$

Moreover, since $\boldsymbol{m}\cdot R(\boldsymbol{\omega}-\boldsymbol{\omega}_s)=0$, we can also write Eq. C.39 as:

$$\begin{bmatrix}0\\ \dot{\boldsymbol{m}}\end{bmatrix} = -\frac{1}{2}\begin{bmatrix}0\\ \boldsymbol{m}\end{bmatrix}\odot\begin{bmatrix}0\\ R(\boldsymbol{\omega}-\boldsymbol{\omega}_s)\end{bmatrix} \tag{C.41}$$

Appendix C.5. Demonstration of nonlinear attitude controller with a Lyapunov function

Consider the following Lyapunov candidate function:

$$V = V(\boldsymbol{m}) \equiv -\ln\left((\bar{\boldsymbol{m}} \cdot \boldsymbol{m})^2\right) \tag{C.42}$$

We can see that:

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

Differentiating Eq. C.42 w.r.t. time:

$$\dot{V} = -2\frac{\bar{\boldsymbol{m}} \cdot \dot{\boldsymbol{m}}}{\bar{\boldsymbol{m}} \cdot \boldsymbol{m}} \tag{C.43}$$

Introducing Eq. 61 into Eq. C.43 gives:

$$\dot{V} = \frac{\bar{\boldsymbol{m}} \cdot (\boldsymbol{m} \times R\,(\boldsymbol{\omega} - \boldsymbol{\omega}_s))}{\bar{\boldsymbol{m}} \cdot \boldsymbol{m}} \tag{C.44}$$

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:

$$\dot{V} = (\boldsymbol{\omega} - \boldsymbol{\omega}_s) \cdot \left(R^T \frac{\bar{\boldsymbol{m}} \times \boldsymbol{m}}{\bar{\boldsymbol{m}} \cdot \boldsymbol{m}}\right) \tag{C.45}$$

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

$$\boldsymbol{\omega} - \boldsymbol{\omega}_s = -\boldsymbol{P}\left(R^T \frac{\bar{\boldsymbol{m}} \times \boldsymbol{m}}{\bar{\boldsymbol{m}} \cdot \boldsymbol{m}}\right) \tag{C.46}$$

Defining:

$$\boldsymbol{m}' = R^T \frac{\bar{\boldsymbol{m}} \times \boldsymbol{m}}{\bar{\boldsymbol{m}} \cdot \boldsymbol{m}} \tag{C.47}$$

We have:

$$\dot{V} = -\boldsymbol{m}' \cdot (\boldsymbol{P}\boldsymbol{m}') \leq 0 \tag{C.48}$$

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

$$\boldsymbol{\omega} = \boldsymbol{\omega}_s + \boldsymbol{P}\left(R^T \frac{\boldsymbol{m} \times \bar{\boldsymbol{m}}}{\boldsymbol{m} \cdot \bar{\boldsymbol{m}}}\right) \tag{C.49}$$

Which is equivalent to 67 and defines our nonlinear control law. Compared to the Lyapunov function defined in [17], for example, the controller defined here gets more and more aggressive when the robot's orientation error approaches 180°. Moreover, our controller might be more well suited for a desired $\bar{\boldsymbol{m}}$ close to $\boldsymbol{e}_z$, which is the case.

References

[1] V. Baiocchi, D. Dominici, M. Mormile, Uav application in post seismic environment, The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XL-1/W2 (2013) 21–25. doi:10.5194/isprsarchives-XL-1-W2-21-2013.

[2] Y. Qin, W. Xu, A. Lee, F. Zhang, Gemini: A Compact Yet Efficient Bi-Copter UAV for Indoor Applications, IEEE Robotics and Automation Letters 5 (2) (2020) 3213–3220. doi:10.1109/LRA.2020.2974718.

[3] M. Piccoli, M. Yim, Piccolissimo: The smallest micro aerial vehicle, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 3328–3333. doi:10.1109/ICRA.2017.7989378.

[4] M. Piccoli, M. Yim, in: Passive stability of a single actuator micro aerial vehicle, 2014, pp. 5510–5515. doi:10.1109/ICRA.2014.6907669.

[5] S. K. H. Win, L. S. T. Win, D. Sufiyan, S. Foong, Design and control of the first foldable single-actuator rotary wing micro aerial vehicle, Bioinspiration & Biomimetics 16 (6) (2021) 066019. doi:10.1088/1748-3190/ac253a.

[6] S. K. H. Win, L. S. T. Win, D. Sufiyan, G. S. Soh, S. Foong, An agile samara-inspired single-actuator aerial robot capable of autorotation and diving, IEEE Transactions on Robotics 38 (2) (2022) 1033–1046. doi:10.1109/TRO.2021.3091275.

[7] A. Norberg, Autorotation, self-stability, and structure of single-winged fruits and seeds (samaras) with comparative remarks on animal flight, Biological Reviews 48 (4) (1973) 561–596. doi:10.1111/j.1469-185X.1973.tb01569.x.

[8] M. Hedayatpour, M. Mehrandezh, F. 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.

[9] M. W. Mueller, R. D'Andrea, Relaxed hover solutions for multicopters: Application to algorithmic redundancy and novel vehicles, International Journal of Robotics Research 35 (8) (2016) 873–889. doi:10.1177/0278364915596233.

[10] W. Zhang, M. W. Mueller, R. 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.

[11] W. Zhang, M. W. Mueller, R. D'Andrea, Design, modeling and control of a flying vehicle with a single moving part that can be positioned anywhere in space, Mechatronics 61 (October 2018) (2019) 117–130. doi:10.1016/j.mechatronics.2019.06.004.

[12] C. E. Thorne, M. Yim, Design and Analysis of a Gyroscopically Controlled Micro Air Vehicle, Journal of Intelligent & Robotic Systems 65 (1) (2012) 417–435. doi:10.1007/s10846-011-9644-7.

[13] J. Paulos, M. Yim, An underactuated propeller for attitude control in micro air vehicles, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2013, pp. 1374–1379. doi:10.1109/IROS.2013.6696528.

[14] J. Paulos, M. Yim, Cyclic Blade Pitch Control Without a Swashplate for Small Helicopters, Journal of Guidance, Control, and Dynamics 41 (3) (2018) 689–700. doi:10.2514/1.G002683.

[15] J. Paulos, B. Caraher, M. Yim, Emulating a fully actuated aerial vehicle using two actuators, in: 2018 IEEE International Conference on Robotics

and Automation (ICRA), 2018, pp. 7011–7016. doi:10.1109/ICRA.2018.8462975.

[16] Y. Qin, N. Chen, Y. Cai, W. Xu, F. Zhang, Gemini ii: Design, modeling, and control of a compact yet efficient servoless bi-copter, IEEE/ASME Transactions on Mechatronics 27 (6) (2022) 4304–4315. doi:10.1109/TMECH.2022.3153587.

[17] N. Chen, F. Kong, W. Xu, Y. Cai, H. Li, D. He, Y. Qin, F. Zhang, A self-rotating, single-actuated UAV with extended sensor field of view for autonomous navigation, Science Robotics 8 (76) (Mar. 2023). doi:10.1126/scirobotics.ade4538.

[18] S. Sun, G. Cioffi, C. De Visser, D. Scaramuzza, Autonomous Quadrotor Flight despite Rotor Failure with Onboard Vision Sensors: Frames vs. Events, IEEE Robotics and Automation Letters 6 (2) (2021) 580–587. doi:10.1109/LRA.2020.3048875.

[19] R. Mahony, V. Kumar, P. Corke, Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor, IEEE Robotics & Automation Magazine 19 (3) (2012) 20–32. doi:10.1109/MRA.2012.2206474.

[20] M. Faessler, F. Fontana, C. Forster, D. Scaramuzza, Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 1722–1729. doi:10.1109/ICRA.2015.7139420.

[21] D. Falanga, K. Kleber, D. Scaramuzza, Dynamic obstacle avoidance for quadrotors with event cameras, Science Robotics 5 (40) (2020) eaaz9712. doi:10.1126/scirobotics.aaz9712.

[22] H. Poincaré, Sur une forme nouvelle des équations de la mécanique, Comptes rendus de l'Académie des Sciences de Paris 132 (1901) 369–371.

[23] W. Yu, Z. Pan, Dynamical equations of multibody systems on Lie groups, Advances in Mechanical Engineering 7 (3) (2015) 1687814015575959. doi:10.1177/1687814015575959.

[24] F. Boyer, D. Primault, The poincaré-chetayev equations and flexible multibody systems, Journal of Applied Mathematics and Mechanics 69 (6) (2005) 925–942. doi:10.1016/j.jappmathmech.2005.11.015.

[25] F. E. Udwadia, A. D. Schutte, An alternative derivation of the quaternion equations of motion for rigid-body rotational dynamics, Journal of Applied Mechanics, Transactions ASME 77 (4) (2010) 1–4. doi:10.1115/1.4000917.

[26] F. E. Udwadia, A. D. Schutte, Equations of motion for general constrained systems in lagrangian mechanics, Acta Mechanica 213 (1-2) (2010) 111–129. doi:10.1007/s00707-009-0272-2.

[27] F. Boyer, M. Porez, Multibody system dynamics for bio-inspired locomotion: from geometric structures to computational aspects, Bioinspiration & Biomimetics 10 (2) (2015) 025007, publisher: IOP Publishing. doi:10.1088/1748-3190/10/2/025007.

[28] R. M. Murray, S. S. Sastry, L. Zexiang, A Mathematical Introduction to Robotic Manipulation, 1st Edition, CRC Press, Inc., USA, 1994.

[29] H. Sommer, I. Gilitschenski, M. Bloesch, S. Weiss, R. Siegwart, J. Nieto, Why and how to avoid the flipped quaternion multiplication, Aerospace 5 (3) (2018) 1–15. arXiv:1801.07478, doi:10.3390/aerospace5030072.

[30] J. B. Kuipers, Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality, Princeton Univ. Press, Princeton, NJ, 1999.

[31] L. Meier, D. Honegger, M. 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.

[32] D. Brescianini, M. Hehn, R. D'Andrea, Nonlinear Quadrocopter Attitude Control, ETH Zürich Research Collection (2013). doi:10.3929/ethz-a-009970340.

[33] E. Bernardes, S. Viollet, Quaternion to Euler angles conversion: A direct, general and computationally efficient method, PLOS ONE 17 (11) (2022)

e0276302, publisher: Public Library of Science. doi:10.1371/journal.pone.0276302.

[34] P. Solanki, C. C. de Visser, Attitude Estimation of a Quadcopter with one fully damaged rotor using on-board MARG Sensors, AIAA Science and Technology Forum and Exposition, AIAA SciTech Forum 2022 (2022). doi:10.2514/6.2022-0857.

[35] R. G. Valenti, I. Dryanovski, J. Xiao, Keeping a good attitude: A quaternion-based orientation filter for IMUs and MARGs, Sensors (Switzerland) 15 (8) (2015) 19302–19330. doi:10.3390/s150819302.