

Theory and Math Behind RPG Quadrotor Control

Matthias Faessler and Flavio Fontana

March 1, 2018

Contents

1	Introduction	2
2	Nomenclature	2
3	Dynamical Model	3
4	Control	5
4.1	High-Level Control	5
4.1.1	Reference Inputs	5
4.1.2	High-Level Control Law	8
4.2	Low-Level Control	9
4.2.1	Attitude Control	9
4.2.2	Body-Rate Control	9
4.2.3	Iterative Thrust Mixing	10
4.2.4	Saturation with Input Priorities	10
4.3	Delay Compensation	11
4.4	Battery Voltage Compensation	11
5	Trajectory Generation	12
5.1	Computing Relevant Maxima from a Reference State	12
6	Math	15
6.1	Mechanics	15
6.1.1	Euler's first law	15
6.1.2	Euler's second law	15
6.1.3	Differentiating the Angular Momentum	16
6.1.4	Summary	16
6.2	Attitude Representations	17
6.2.1	Rotation Matrices and Euler Angles	17
6.2.2	Quaternions	18
6.2.3	Changing Transformation Representation	19
6.3	Classical Modeling of a Quadrotor	20
6.3.1	Using Euler Angles	20
6.3.2	Using Rotation Matrices	22
6.3.3	Using Quaternions	22
A	Appendices	24
A	Singularities in Reference Inputs	24
B	Zeroth Order Integration of a Unit Quaternion	25
C	Proof of not Measuring Gravity in Flight	25

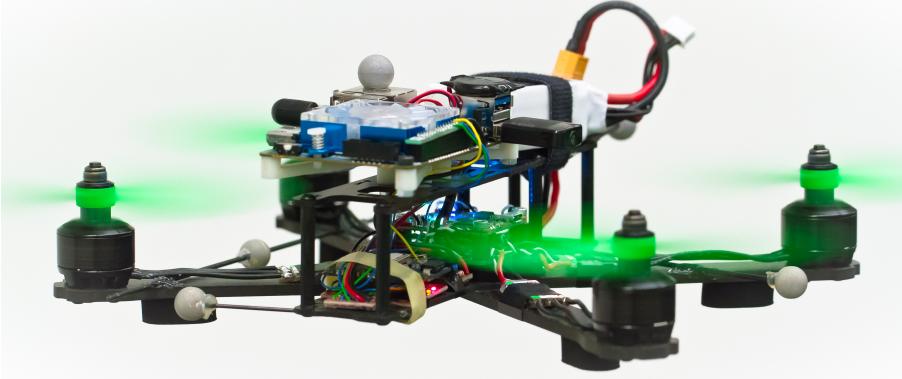


Figure 1: First-person-view racing inspired quadrotor platform used for the experiments presented in [1].

1 Introduction

In this document, we summarized the theory and math behind the quadrotor control algorithms developed at the *Robotics and Perception Group* as they are implemented in our `rpg_quadrotor_control` repository. If you spot any typos or mistakes or want to improve or add something, please open an issue or send us a pull request on https://github.com/uzh-rpg/rpg_quadrotor_control. The control algorithms presented here are published in [1], for which we developed the quadrotor shown in Fig. 1, and in [2].

2 Nomenclature

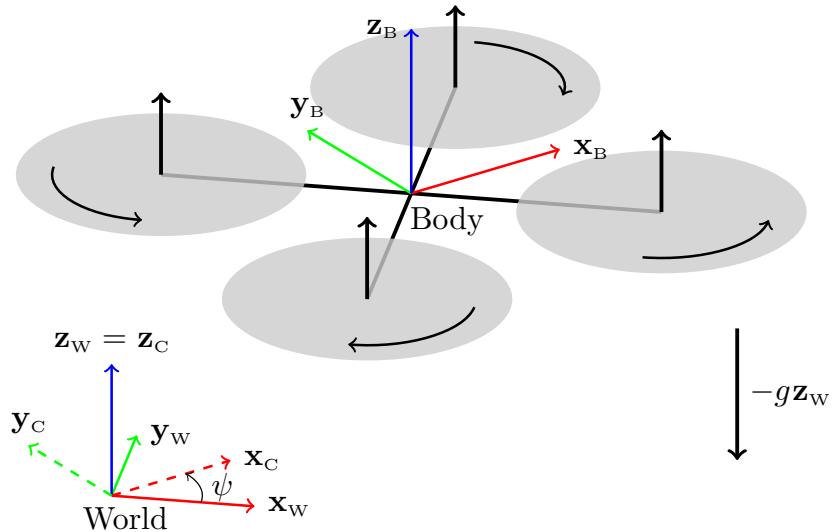


Figure 2: Schematics of the considered quadrotor model with the used coordinate systems and rotor forces.

In this work, we make use of a world frame W with orthonormal basis $\{\mathbf{x}_w, \mathbf{y}_w, \mathbf{z}_w\}$ represented in world coordinates and a body frame B with orthonormal basis $\{\mathbf{x}_B, \mathbf{y}_B, \mathbf{z}_B\}$ also represented in world coordinates. The body frame is fixed to the quadrotor with an origin coinciding with its center of mass as depicted in Fig. 2. The rotor numbering and rotation directions can also be seen in Fig. 2. The quadrotor is subject to a gravitational acceleration g in negative \mathbf{z}_w direction. We denote the position of the quadrotor's center of mass as \mathbf{p} , and its derivatives, velocity, acceleration, jerk, and snap as \mathbf{v} , \mathbf{a} , \mathbf{j} , and \mathbf{s} , respectively. We represent the quadrotor's orientation as a rotation matrix $\mathbf{R} = [\mathbf{x}_B \ \mathbf{y}_B \ \mathbf{z}_B]$ and its body rates (i.e., the angular velocity) as $\boldsymbol{\omega}$ represented in body coordinates. Finally, we denote quantities that can be computed from a reference trajectory as

reference values and quantities that are computed by an outer loop feedback control law and passed to an inner loop controller as *desired values*. A reference trajectory consists of the reference position \mathbf{p}_{ref} , the reference velocity \mathbf{v}_{ref} , the reference acceleration \mathbf{a}_{ref} , the reference jerk \mathbf{j}_{ref} , and the reference snap \mathbf{s}_{ref} , as well as a reference heading ψ_{ref} , a reference heading rate $\dot{\psi}_{\text{ref}}$, and a reference heading acceleration $\ddot{\psi}_{\text{ref}}$.

3 Dynamical Model

We consider a quadrotor that is modeled as a rigid body which is controlled by four single rotor thrusts f_i as illustrated in Fig. 2. By changing these four single rotor thrusts, a three axis torque $\boldsymbol{\eta}$ and a mass normalized collective thrust c can be applied on the quadrotor's body. The relation of the single rotor thrusts to the collective thrust and the body torques can be formulated using the coordinate system of Fig. 2 as

$$\boldsymbol{\eta} = \begin{bmatrix} \frac{\sqrt{2}}{2}l(f_1 - f_2 - f_3 + f_4) \\ \frac{\sqrt{2}}{2}l(-f_1 - f_2 + f_3 + f_4) \\ \kappa_1 f_1 - \kappa_2 f_2 + \kappa_3 f_3 - \kappa_4 f_4 \end{bmatrix}, \quad (1)$$

$$mc = f_1 + f_2 + f_3 + f_4, \quad (2)$$

where l is the quadrotor's arm length, $\kappa_i = \kappa(f_i)$ is a coefficient relating the drag torque and the thrust of a single rotor, and m is the quadrotor's mass. Note that unlike in [3] and [4], we consider the rotor drag torque coefficient κ to be a function of the rotor thrust (cf. Fig 4) and *not a constant*.

We model the single rotor thrust f and drag torque τ as quadratic polynomials of the motor input u as

$$f(u) = k_2^f u^2 + k_1^f u + k_0^f, \quad (3)$$

$$\tau(u) = k_2^\tau u^2 + k_1^\tau u + k_0^\tau, \quad (4)$$

where the coefficients k_j^f and k_j^τ are identified by running a single motor with a propeller on a load cell and measuring the resulting forces and moments. The motor input u corresponds to the command we can send to our electronic speed controllers in the range $[-1, 1]$. We chose to have three coefficients since it approximates the measured values better than modeling the rotor thrust and drag torque with only a quadratic term, as proposed in e.g. [5]. Fig. 3 compares the two methods for fitting the thrust mapping. From (3) and (4), we can compute the rotor drag torque coefficient as

$$\kappa(f) = \frac{\tau \left(u = \frac{-k_1^f + \sqrt{(k_1^f)^2 - 4k_2^f(k_0^f - f)}}{2k_2^f} \right)}{f}. \quad (5)$$

Fig. 4 shows the identified values for the rotor drag torque coefficient and how it varies by about 10% over the entire range of available motor inputs. We consider the dynamical model of a quadrotor with rotor drag developed in [6] with no wind, stiff propellers, and no dependence of the rotor drag on the thrust. Additionally, as in [2], we model the dynamics of the single rotor thrusts as first order systems

$$\dot{f} = \frac{1}{\alpha_{\text{mot}}} (f_{\text{des}} - f) \quad (6)$$

where the time constant α_{mot} is identified from applying step inputs to a single motor with propeller on a load cell as illustrated in Fig. 5. From these load-cell experiments we found that for modern ESCs that actively break the motor when spinning down, the time constant is almost identical when spinning a motor up or down and we therefore do not need to separate those cases. Because of that, also the body torques have first order dynamics with the same time constant as the single

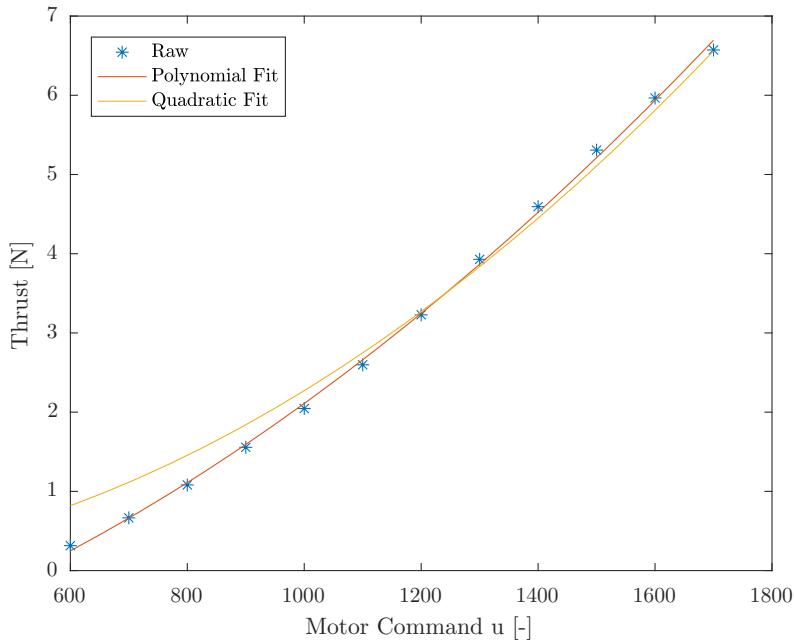


Figure 3: To approximate the thrust mapping for a single motor with propeller, we fit a second order polynomial into raw thrust measurements obtained by running the motor on a load cell. The polynomial fit approximates the measurements much better than a purely quadratic fit of the form $f(u) = k_2^f u^2$. Data is captured with stiff 6x4.5 inch propellers, Cobra CM-2208/20 2000Kv motors, and DYS XSD20A electronic speed controllers which are commanded through the Dshot300 protocol (Setup shown in Fig. 1).

rotor thrusts. According to these models in [6] and [2], the dynamics of the position \mathbf{p} , velocity \mathbf{v} , orientation \mathbf{R} , body rates $\boldsymbol{\omega}$, and body torques $\boldsymbol{\eta}$ can be written as

$$\dot{\mathbf{p}} = \mathbf{v} \quad (7)$$

$$\dot{\mathbf{v}} = -g\mathbf{z}_w + c\mathbf{z}_B - \mathbf{R}\mathbf{D}\mathbf{R}^\top \mathbf{v} \quad (8)$$

$$\dot{\mathbf{R}} = \mathbf{R}\hat{\boldsymbol{\omega}} \quad (9)$$

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1} (\boldsymbol{\eta} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} - \boldsymbol{\tau}_g - \mathbf{A}\mathbf{R}^\top \mathbf{v} - \mathbf{B}\boldsymbol{\omega}) \quad (10)$$

$$\dot{\boldsymbol{\eta}} = \frac{1}{\alpha_{\text{mot}}} (\boldsymbol{\eta}_{\text{des}} - \boldsymbol{\eta}) \quad (11)$$

where c is the mass-normalized collective thrust, $\mathbf{D} = \text{diag}(d_x, d_y, d_z)$ is a constant diagonal matrix formed by the rotor-drag coefficients, $\hat{\boldsymbol{\omega}}$ is a skew-symmetric matrix formed from $\boldsymbol{\omega}$, \mathbf{J} is the quadrotor's inertia matrix, $\boldsymbol{\eta}$ are the torque inputs, $\boldsymbol{\tau}_g$ are gyroscopic torques from the propellers, and \mathbf{A} and \mathbf{B} are constant matrices of the form

$$\mathbf{A} = \begin{bmatrix} 0 & * & 0 \\ * & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \text{ and } \mathbf{B} = \begin{bmatrix} * & 0 & 0 \\ 0 & * & 0 \\ 0 & 0 & * \end{bmatrix}. \quad (12)$$

For the derivations and more details about these terms, please refer to [6]. In this model, we consider the motor dynamics for the rotational states but neglect them for the collective thrust c . We do this because the collective thrust is computed in a high-level position controller, which typically runs at a frequency comparable to the motor dynamics and therefore too slow to consider these dynamics in the control method. In this work, we adopt the thrust model presented in [7]

$$c = c_{\text{cmd}} + k_h v_h^2 \quad (13)$$

where c_{cmd} is the commanded collective thrust input, k_h is a constant, and $v_h = \mathbf{v}^\top(\mathbf{x}_B + \mathbf{y}_B)$. The term $k_h v_h^2$ acts as a quadratic velocity-dependent input disturbance which adds up to the input

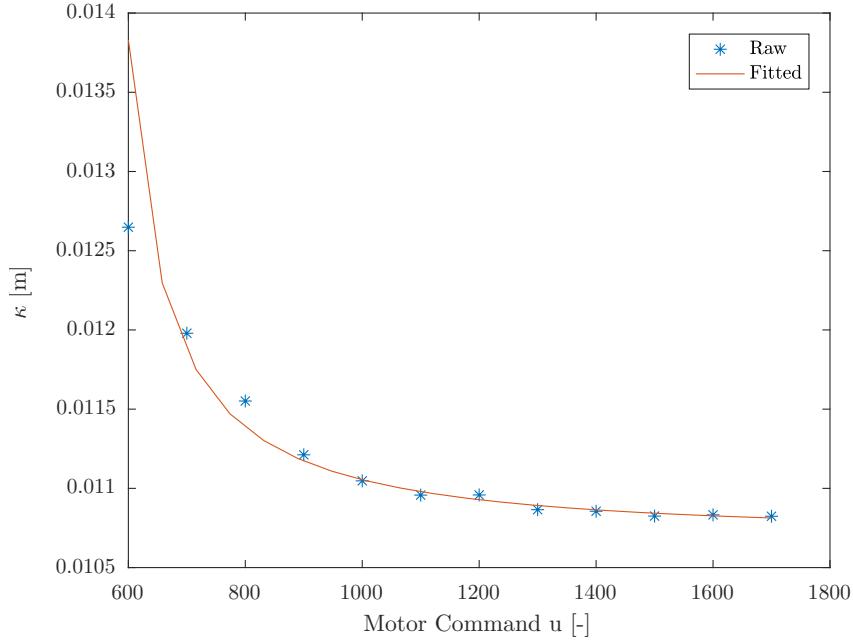


Figure 4: Values for κ estimated from load cell data obtained by dividing drag torque and thrust values for a motor and propeller. The fitted curve is the ratio of the fitted quadratic functions for measured thrust and drag torque in (3) and (4). Data is captured with stiff 6x4.5 inch propellers, Cobra CM-2208/20 2000Kv motors, and DYS XSD20A electronic speed controllers which are commanded through the Dshot300 protocol (Setup shown in Fig. 1).

c_{cmd} . The additional linear velocity-dependent disturbance in the \mathbf{z}_B direction of the thrust model in [7] is lumped by d_z directly in (8) by neglecting its dependency on the rotor speeds. Note that this dynamical model of a quadrotor is a generalization of the common model found, e.g., in [8], in which the linear rotor drag components are typically neglected, i.e., \mathbf{D} , \mathbf{A} and \mathbf{B} are considered null matrices.

4 Control

In this section, we present our proposed quadrotor control scheme, which is split into a high-level part for position control and a low-level part for attitude control. The high-level position control outputs the desired orientation, desired collective thrust command, desired body rates, and the desired angular accelerations. These desired values are then tracked by the low-level attitude controller, which directly commands the four motors. This controller is implemented in our open-source software on https://github.com/uzh-rpg/rpg_quadrotor_control

4.1 High-Level Control

4.1.1 Reference Inputs

To achieve accurate tracking of a reference trajectory, a quadrotor position controller requires reference inputs which it uses as feed-forward terms. In this section, we show how the reference orientation, thrust, body rates, and angular accelerations can be computed from a reference trajectory through the differential flatness property of quadrotor dynamics subject to rotor drag as proved in [1]. For brevity, we omit a ref subscript of the values computed in this section since they are all reference values.

We first compute the reference orientation \mathbf{R} , which is defined by (8) and the reference heading

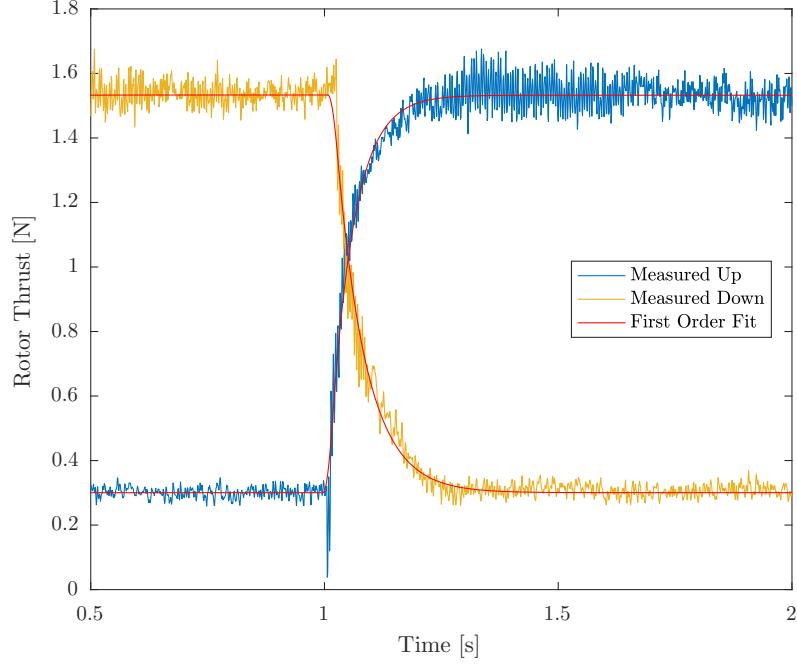


Figure 5: Thrust transients of a single motor with propeller on a load cell for a step up (blue) and a step down (red). A first order system fit for both is overlaid in red. Data is captured with stiff 6x4.5 inch propellers, Cobra CM-2208/20 2000Kv motors, and DYS XSD20A electronic speed controllers which are commanded through the Dshot300 protocol (Setup shown in Fig. 1).

ψ . From (8), we can derive the constraints

$$\mathbf{x}_B^\top \boldsymbol{\alpha} = 0, \text{ with } \boldsymbol{\alpha} = \mathbf{a} + g\mathbf{z}_w + d_x \mathbf{v} \quad (14)$$

$$\mathbf{y}_B^\top \boldsymbol{\beta} = 0, \text{ with } \boldsymbol{\beta} = \mathbf{a} + g\mathbf{z}_w + d_y \mathbf{v}. \quad (15)$$

To enforce a reference heading ψ , we furthermore constrain the projection of the \mathbf{x}_B axis into the $\mathbf{x}_w - \mathbf{y}_w$ plane to be collinear with \mathbf{x}_c (cf. Fig. 2), where

$$\mathbf{x}_c = [\cos(\psi) \quad \sin(\psi) \quad 0]^\top \quad (16)$$

$$\mathbf{y}_c = [-\sin(\psi) \quad \cos(\psi) \quad 0]^\top. \quad (17)$$

From this, (14) and (15), and the constraints that \mathbf{x}_B , \mathbf{y}_B , and \mathbf{z}_B must be orthogonal to each other and of unit length, we can construct \mathbf{R} with

$$\mathbf{x}_B = \frac{\mathbf{y}_c \times \boldsymbol{\alpha}}{\|\mathbf{y}_c \times \boldsymbol{\alpha}\|} \quad (18)$$

$$\mathbf{y}_B = \frac{\boldsymbol{\beta} \times \mathbf{x}_B}{\|\boldsymbol{\beta} \times \mathbf{x}_B\|} \quad (19)$$

$$\mathbf{z}_B = \mathbf{x}_B \times \mathbf{y}_B \quad (20)$$

as

$$\mathbf{R} = [\mathbf{x}_B \quad \mathbf{y}_B \quad \mathbf{z}_B]. \quad (21)$$

One can verify that these vectors are of unit length, perpendicular to each other, and satisfy the constraints (14) - (17). If \mathbf{y}_c is collinear to $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$ is collinear to \mathbf{x}_B , the computation of the orientation is affected by a singularity which we tackle as described in Appendix A.

The collective reference thrust can then be computed as

$$c = \mathbf{z}_B^\top (\mathbf{a} + g\mathbf{z}_W + d_z \mathbf{v}). \quad (22)$$

Then the collective thrust input can be computed as a function of c , \mathbf{R} , and the flat outputs, as

$$c_{\text{cmd}} = c - k_h (\mathbf{v}^\top (\mathbf{x}_B + \mathbf{y}_B))^2. \quad (23)$$

By following [1], we can compute the reference body rates by solving the following linear system of equations

$$\begin{aligned} \omega_y (c - (d_z - d_x) (\mathbf{z}_B^\top \mathbf{v})) - \omega_z (d_x - d_y) (\mathbf{y}_B^\top \mathbf{v}) \\ = \mathbf{x}_B^\top \mathbf{j} + d_x \mathbf{x}_B^\top \mathbf{a} \end{aligned} \quad (24)$$

$$\begin{aligned} \omega_x (c + (d_y - d_z) (\mathbf{z}_B^\top \mathbf{v})) + \omega_z (d_x - d_y) (\mathbf{x}_B^\top \mathbf{v}) \\ = -\mathbf{y}_B^\top \mathbf{j} - d_y \mathbf{y}_B^\top \mathbf{a} \end{aligned} \quad (25)$$

$$\omega_z = \frac{1}{\|\mathbf{y}_C \times \mathbf{z}_B\|} (\dot{\psi} \mathbf{x}_C^\top \mathbf{x}_B + \omega_y \mathbf{y}_C^\top \mathbf{z}_B) \quad (26)$$

for ω_x , ω_y , and ω_z as

$$\omega_x = \frac{-\mathcal{B}_1 \mathcal{C}_2 \mathcal{D}_3 + \mathcal{B}_1 \mathcal{C}_3 \mathcal{D}_2 - \mathcal{B}_3 \mathcal{C}_1 \mathcal{D}_2 + \mathcal{B}_3 \mathcal{C}_2 \mathcal{D}_1}{\mathcal{A}_2 (\mathcal{B}_1 \mathcal{C}_3 - \mathcal{B}_3 \mathcal{C}_1)} \quad (27)$$

$$\omega_y = \frac{-\mathcal{C}_1 \mathcal{D}_3 + \mathcal{C}_3 \mathcal{D}_1}{\mathcal{B}_1 \mathcal{C}_3 - \mathcal{B}_3 \mathcal{C}_1} \quad (28)$$

$$\omega_z = \frac{\mathcal{B}_1 \mathcal{D}_3 - \mathcal{B}_3 \mathcal{D}_1}{\mathcal{B}_1 \mathcal{C}_3 - \mathcal{B}_3 \mathcal{C}_1} \quad (29)$$

where

$$\mathcal{B}_1 = c - (d_z - d_x) (\mathbf{z}_B^\top \mathbf{v}) \quad (30)$$

$$\mathcal{C}_1 = -(d_x - d_y) (\mathbf{y}_B^\top \mathbf{v}) \quad (31)$$

$$\mathcal{D}_1 = \mathbf{x}_B^\top \mathbf{j} + d_x \mathbf{x}_B^\top \mathbf{a} \quad (32)$$

$$\mathcal{A}_2 = c + (d_y - d_z) (\mathbf{z}_B^\top \mathbf{v}) \quad (33)$$

$$\mathcal{C}_2 = (d_x - d_y) (\mathbf{x}_B^\top \mathbf{v}) \quad (34)$$

$$\mathcal{D}_2 = -\mathbf{y}_B^\top \mathbf{j} - d_y \mathbf{y}_B^\top \mathbf{a} \quad (35)$$

$$\mathcal{B}_3 = -\mathbf{y}_C^\top \mathbf{z}_B \quad (36)$$

$$\mathcal{C}_3 = \|\mathbf{y}_C \times \mathbf{z}_B\| \quad (37)$$

$$\mathcal{D}_3 = \dot{\psi} \mathbf{x}_C^\top \mathbf{x}_B. \quad (38)$$

To compute the reference angular accelerations, we take the derivative of the linear system of equations (24)-(26) and solve it for $\dot{\omega}_x$, $\dot{\omega}_y$, and $\dot{\omega}_z$ as

$$\dot{\omega}_x = \frac{-\mathcal{B}_1 \mathcal{C}_2 \mathcal{E}_3 + \mathcal{B}_1 \mathcal{C}_3 \mathcal{E}_2 - \mathcal{B}_3 \mathcal{C}_1 \mathcal{E}_2 + \mathcal{B}_3 \mathcal{C}_2 \mathcal{E}_1}{\mathcal{A}_2 (\mathcal{B}_1 \mathcal{C}_3 - \mathcal{B}_3 \mathcal{C}_1)} \quad (39)$$

$$\dot{\omega}_y = \frac{-\mathcal{C}_1 \mathcal{E}_3 + \mathcal{C}_3 \mathcal{E}_1}{\mathcal{B}_1 \mathcal{C}_3 - \mathcal{B}_3 \mathcal{C}_1} \quad (40)$$

$$\dot{\omega}_z = \frac{\mathcal{B}_1 \mathcal{E}_3 - \mathcal{B}_3 \mathcal{E}_1}{\mathcal{B}_1 \mathcal{C}_3 - \mathcal{B}_3 \mathcal{C}_1} \quad (41)$$

where

$$\mathcal{E}_1 = \mathbf{x}_B^\top \mathbf{s} - 2\dot{c}\omega_y - c\omega_x\omega_z + \mathbf{x}_B^\top \boldsymbol{\xi} \quad (42)$$

$$\mathcal{E}_2 = -\mathbf{y}_B^\top \mathbf{s} - 2\dot{c}\omega_x + c\omega_y\omega_z - \mathbf{y}_B^\top \boldsymbol{\xi} \quad (43)$$

$$\begin{aligned} \mathcal{E}_3 = & \ddot{\psi} \mathbf{x}_C^\top \mathbf{x}_B + 2\dot{\psi}\omega_z \mathbf{x}_C^\top \mathbf{y}_B - 2\dot{\psi}\omega_y \mathbf{x}_C^\top \mathbf{z}_B \\ & - \omega_x\omega_y \mathbf{y}_C^\top \mathbf{y}_B - \omega_x\omega_z \mathbf{y}_C^\top \mathbf{z}_B \end{aligned} \quad (44)$$

$$\begin{aligned} \dot{c} = & \mathbf{z}_B^\top \mathbf{j} + \omega_x(d_y - d_z)(\mathbf{y}_B^\top \mathbf{v}) \\ & + \omega_y(d_z - d_x)(\mathbf{x}_B^\top \mathbf{v}) + d_z \mathbf{z}_B^\top \mathbf{a} \end{aligned} \quad (45)$$

$$\begin{aligned} \boldsymbol{\xi} = & \mathbf{R}(\hat{\omega}^2 \mathbf{D} + \mathbf{D}\hat{\omega}^2 + 2\hat{\omega}\mathbf{D}\hat{\omega}^\top) \mathbf{R}^\top \mathbf{v} \\ & + 2\mathbf{R}(\hat{\omega}\mathbf{D} + \mathbf{D}\hat{\omega}^\top) \mathbf{R}^\top \mathbf{a} + \mathbf{R}\mathbf{D}\mathbf{R}^\top \mathbf{j}. \end{aligned} \quad (46)$$

The computation of the body rates and angular acceleration may be affected by singularities, which we treat as described in Appendix A. More details of these derivations can be found in our technical report [9].

4.1.2 High-Level Control Law

In our high-level position controller, which is adapted from [10], we compute the desired orientation \mathbf{R}_{des} , collective thrust command c_{cmd} , body rates $\boldsymbol{\omega}_{des}$, angular accelerations $\dot{\boldsymbol{\omega}}_{des}$, and angular jerk $\ddot{\boldsymbol{\omega}}_{des}$, which are then applied by the low-level controller described in the next section. As a first step in the position controller, we compute the desired acceleration of the quadrotor's body as

$$\mathbf{a}_{des} = \mathbf{a}_{fb} + \mathbf{a}_{ref} - \mathbf{a}_{rd} + g\mathbf{z}_W \quad (47)$$

where \mathbf{a}_{fb} are the PD feedback-control terms computed from the position and velocity control errors as

$$\mathbf{a}_{fb} = -\mathbf{K}_{pos}(\mathbf{p} - \mathbf{p}_{ref}) - \mathbf{K}_{vel}(\mathbf{v} - \mathbf{v}_{ref}) \quad (48)$$

where \mathbf{K}_{pos} and \mathbf{K}_{vel} are constant diagonal matrices. The reference acceleration \mathbf{a}_{ref} is obtained directly from the reference trajectory and the acceleration due to rotor drag \mathbf{a}_{rd} is computed according to (8) as

$$\mathbf{a}_{rd} = -\mathbf{R}_{ref} \mathbf{D} \mathbf{R}_{ref}^\top \mathbf{v}_{ref}. \quad (49)$$

We compute the desired orientation \mathbf{R}_{des} such that $\mathbf{z}_{B,des} = \mathbf{a}_{des}/\|\mathbf{a}_{des}\|$ and the reference heading ψ_{ref} is respected. This is achieved with

$$\mathbf{z}_{B,des} = \frac{\mathbf{a}_{des}}{\|\mathbf{a}_{des}\|} \quad (50)$$

$$\mathbf{x}_{B,des} = \frac{\mathbf{y}_C \times \mathbf{z}_{B,des}}{\|\mathbf{y}_C \times \mathbf{z}_{B,des}\|} \quad (51)$$

$$\mathbf{y}_{B,des} = \mathbf{z}_{B,des} \times \mathbf{x}_{B,des} \quad (52)$$

as

$$\mathbf{R}_{des} = [\mathbf{x}_{B,des} \quad \mathbf{y}_{B,des} \quad \mathbf{z}_{B,des}]. \quad (53)$$

By projecting the desired accelerations onto the actual body z -axis and considering the thrust model (13), we can then compute the collective thrust input as

$$c_{cmd} = \mathbf{a}_{des}^\top \mathbf{z}_B - k_h(\mathbf{v}^\top (\mathbf{x}_B + \mathbf{y}_B))^2. \quad (54)$$

Finally, we compute the desired body rates, the angular accelerations, and the angular jerk by transforming the reference body rates into the current body frame and take its derivatives as

$$\boldsymbol{\omega}_{des} = \mathbf{R}^\top \mathbf{R}_{ref} \boldsymbol{\omega}_{ref} \quad (55)$$

$$\dot{\boldsymbol{\omega}}_{des} = \mathbf{R}^\top \mathbf{R}_{ref} \dot{\boldsymbol{\omega}}_{ref} - \hat{\boldsymbol{\omega}} \mathbf{R}^\top \mathbf{R}_{ref} \boldsymbol{\omega}_{ref} \quad (56)$$

4.2 Low-Level Control

The low-level controller is responsible for tracking the desired orientation, desired body rates, and the desired angular accelerations as computed by the high-level controller. This can be achieved by controllers such as presented in [11] or [8] but they are often not practical due to the unavailability of an attitude estimate in the low-level control loop. Therefore, we split the low-level controller into an attitude part that runs with the high-level controller and a separate body-rate controller which does not require an attitude estimate. This section introduces the low-level control which we implemented in our `rpg_rotors_interface` except for the attitude controller which is implemented in `position_controller`.

4.2.1 Attitude Control

Our attitude controller is based on quaternion representations of the attitude and is similar to [12]. It can be shown that this control law is globally asymptotically stable and its discrete implementation is robust to measurement noise [12, 13]. This controller is implemented in our `position_controller`.

We first compute the attitude error as an error quaternion as

$$\mathbf{q}_e = \mathbf{q}^{-1} \otimes \mathbf{q}_{\text{des}}. \quad (57)$$

From the elements of the error quaternion we can then directly compute the feedback body rates as

$$\boldsymbol{\omega}_{\text{fb}} = \begin{cases} 2 \cdot \mathbf{K}_{\text{att}} \cdot \mathbf{q}_e & \text{if } \mathbf{q}_{e,w} \geq 0 \\ -2 \cdot \mathbf{K}_{\text{att}} \cdot \mathbf{q}_e & \text{if } \mathbf{q}_{e,w} < 0 \end{cases} \quad (58)$$

where the error quaternion \mathbf{q}_e is defined according to the convention in (126) and $\mathbf{q}_{e,w}$ is its real part and

$$\mathbf{K}_{\text{att}} = \begin{bmatrix} 0 & k_{rp} & 0 & 0 \\ 0 & 0 & k_{rp} & 0 \\ 0 & 0 & 0 & k_y \end{bmatrix} \quad (59)$$

where k_{rp} and k_y are attitude control gains.

4.2.2 Body-Rate Control

In this section, we present a body-rate controller that provides good tracking and disturbance rejection performance by considering the dynamics of the body rates and body torques (10) and (11) as suggested in our work [2]. We achieve this by designing an LQR controller for a dynamical system containing the body rates and body torques as state. The inputs to this controller are the desired body rates $\boldsymbol{\omega}_{\text{des}}$, the feedback body rates $\boldsymbol{\omega}_{\text{fb}}$, and the desired mass normalized collective thrust c_{cmd} , which are given from the high-level position and attitude controller.

Linearizing (10) and (11) around $\boldsymbol{\omega} = \mathbf{0}$ and $\boldsymbol{\eta} = \mathbf{0}$ leads to the system

$$\begin{bmatrix} \dot{\boldsymbol{\omega}} \\ \dot{\boldsymbol{\eta}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{J}^{-1} \\ \mathbf{0} & -\frac{1}{\alpha_{\text{mot}}} \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{\eta} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \frac{1}{\alpha_{\text{mot}}} \mathbf{I}_3 \end{bmatrix} \boldsymbol{\eta}_{\text{des}} \quad (60)$$

which we can use to design an infinite-horizon LQR control law $\mathbf{u} = -\mathbf{K}_{lqr} \mathbf{s}$ that minimizes the cost function

$$\int \mathbf{s}^T \mathbf{Q} \mathbf{s} + \mathbf{u}^T \mathbf{R} \mathbf{u} dt \quad (61)$$

where \mathbf{Q} is a diagonal weight matrix and \mathbf{R} is the identity matrix. The solution to the formulated LQR problem is a gain matrix of the form

$$\mathbf{K}_{lqr} = \begin{bmatrix} k_{\omega_{xy}} & 0 & 0 & k_{\eta_{xy}} & 0 & 0 \\ 0 & k_{\omega_{xy}} & 0 & 0 & k_{\eta_{xy}} & 0 \\ 0 & 0 & k_{\omega_z} & 0 & 0 & k_{\eta_z} \end{bmatrix} \quad (62)$$

which corresponds to a PD controller of the body rates. Additionally, we add feed forward terms such that $\boldsymbol{\omega}_{\text{des}}$ is reached with $\dot{\boldsymbol{\omega}} = \dot{\boldsymbol{\omega}}_{\text{des}}$, resulting in the control policy

$$\boldsymbol{\eta}_{\text{des}} = \mathbf{K}_{lqr} \begin{bmatrix} \boldsymbol{\omega}_{\text{des}} + \boldsymbol{\omega}_{\text{fb}} - \boldsymbol{\omega} \\ \boldsymbol{\eta}_{\text{ref}} - \boldsymbol{\eta} \end{bmatrix} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \mathbf{J}\dot{\boldsymbol{\omega}}_{\text{des}}, \quad (63)$$

with $\boldsymbol{\eta}_{\text{ref}} = \boldsymbol{\omega}_{\text{des}} \times \mathbf{J}\boldsymbol{\omega}_{\text{des}} + \mathbf{J}\dot{\boldsymbol{\omega}}_{\text{des}}$ computed from (10). The vector $\boldsymbol{\omega}$ are the estimated body rates measured by the onboard gyroscopes, and $\boldsymbol{\eta}$ are the estimated body torques obtained by estimating the single rotor thrusts with (6) and using (1) to transform them into body torques. Also note that this estimation can be improved if feedback of the rotor speeds is available. In the controller (63), the term $\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}$ provides feedback linearization, compensating for the coupling terms in the body-rate dynamics, and $\mathbf{J}\dot{\boldsymbol{\omega}}_{\text{des}}$ is a feed-forward term on desired angular accelerations.

4.2.3 Iterative Thrust Mixing

To compute the single rotor thrusts that achieve the desired body torques $\boldsymbol{\eta}_{\text{des}}$ and collective thrust c_{cmd} , which we denote as *mixer inputs*, we have to solve (1) and (2) for f_i . Since we consider the rotor drag torque coefficients to be a function of the rotor thrust, we cannot solve this system of equations directly, but we can do so iteratively, as proposed in our previous work [2].

To initialize the iteration, we start by setting the single rotor thrusts equal such that they achieve the desired collective thrust:

$$f_i = \frac{mc_{\text{cmd}}}{4}. \quad (64)$$

Note that these values are only used to compute the rotor drag torque coefficients in the first iteration. Then, we start the iteration with the following two steps: i) solve (5) to get κ_i , and ii) solve (1) and (2) with $\boldsymbol{\eta}_{\text{des}}$ and c_{cmd} for f_i .

4.2.4 Saturation with Input Priorities

Once we have computed the desired single rotor thrusts, we have to make sure that they lie within the feasible range $[f_{\min}, f_{\max}]$ for each single motor, which we do as proposed in our previous work [2]. Naively, feasible inputs can be achieved by clipping each rotor thrust if its desired value is outside this range. This is a simple and fast procedure with the drawback that none of the desired mixer inputs $\boldsymbol{\eta}_{\text{des}}$ and c_{cmd} is achieved exactly if one of the rotor thrusts is clipped. Nonetheless, not all these mixer inputs are equally important in terms of the quadrotor's ability to stabilize and track a trajectory. Since a quadrotor can only produce a collective thrust in its body upwards direction, it has to be aligned with the desired acceleration for following a trajectory in 3D space. The rotation around the thrust direction is irrelevant for the translational motion of the quadrotor. Therefore, we want to give least priority to achieve the desired yaw torque in case of an input saturation. On the other hand, the quadrotor uses roll and pitch torques to change its thrust direction which enables stabilization and therefore makes them the most important inputs. Furthermore, state of the art control methods for quadrotors (e.g. [8], [3], [10]) are based on the assumption that the orientation of the thrust vector can be changed quickly. For these reasons, in case of an input saturation, we want to give highest priority to applying the desired roll and pitch torques, second highest priority to applying the desired collective thrust, and lowest priority to applying the desired yaw torque.

Yaw-Torque Saturation We achieve this prioritization by a saturation scheme as summarized in Algorithm 1. First, the single rotor thrusts are computed according to Section 4.2.3. If one of the single rotor thrusts exceeds its limits, we try to change the applied yaw torque to avoid saturation, given that the desired yaw torque is above a certain minimum $\eta_{z,\text{assured}}$, which we can optionally impose. Such an assured yaw torque might be desired for applications where we want to guarantee that we always have some control on the heading of a quadrotor. In case of saturation, we do not apply the iterative mixer in order to save time since we are unable to apply the desired yaw torque anyways. To do the yaw-torque saturation, we find the rotor that violates the input limit the most, set it to the corresponding limit and then solve (1) and (2) for the remaining rotor

Algorithm 1 Rotor Thrust Saturation

```

Compute  $f_i$  as detailed in Section 4.2.3
Perform yaw-torque saturation:
if Motor saturated AND  $|\eta_{z,des}| > \eta_{z,assured}$  then
    Find rotor  $j$  that violates thrust limits the most
     $f_j \leftarrow f_{limit}$ 
    Solve (1) and (2) for  $f_{i \setminus j}$  and  $\eta_z$ 
    if  $\text{sign}(\eta_{z,des}) \cdot \eta_z < \eta_{z,assured}$  then
         $\eta_z \leftarrow \text{sign}(\eta_{z,des}) \cdot \eta_{z,assured}$ 
        Solve (1) and (2) for  $f_i$ 
    end if
end if
Perform collective-thrust saturation:
if Motor saturated then
    if NOT(upper AND lower saturation reached) then
        Find rotor  $j$  that violates thrust limits the most
        Shift  $f_i$  equally s.t.  $f_j = f_{limit}$ 
    end if
end if
Enforce single rotor thrust limits by thrust clipping

```

thrusts and the yaw torque. If the resulting yaw torque is still above the value we want to assure, we successfully enforced all the rotor-thrust limits by only changing the applied yaw torque. In other words, in this case, Algorithm 1 guarantees that the quadrotor applies the desired roll and pitch torques and the desired collective thrust, but *not* the desired yaw torque. If the resulting yaw torque is below the value we want to assure, we set it to the assured value $\eta_{z,assured}$ and recompute the rotor thrusts.

Collective-Thrust Saturation If one of the rotor-thrust limits is still violated, we try to change the applied collective thrust to avoid saturation. This is only possible if two rotors do not violate the upper and the lower limit simultaneously, in which case it is impossible to achieve the desired roll and pitch torques by changing the applied collective thrust. If only one limit is violated, we find the rotor that violates the input limit the most, set it to its limit and shift the remaining rotor thrusts by the same amount. In this case, Algorithm 1 ensures that the quadrotor applies the desired roll and pitch torques but *not* the desired collective thrust and *not* the desired yaw torque.

Thrust Clipping At this point, if a rotor still violates its input limits, we have to apply thrust clipping and can therefore not achieve any of the desired mixer inputs precisely.

4.3 Delay Compensation

TODO: about state predictor

4.4 Battery Voltage Compensation

Since we use electronic speed controllers that control the percentage of input voltage, the resulting rotor thrust for a given command depends on the battery voltage. To improve the trajectory tracking performance of our quadrotors, we compensate for this voltage dependency. By performing thrust mapping identifications with different voltages which are controlled by a power supply unit, in Fig. 6, we found that they are related by a constant factor to each other. Furthermore, by letting a quadrotor hover for an entire battery charge while applying the thrust mapping identified at 12 V, we found that this factor, which depicts the ratio of the commanded thrust to the actually produced thrust, is a linear function of the battery voltage (see Fig. 7). The collective thrust

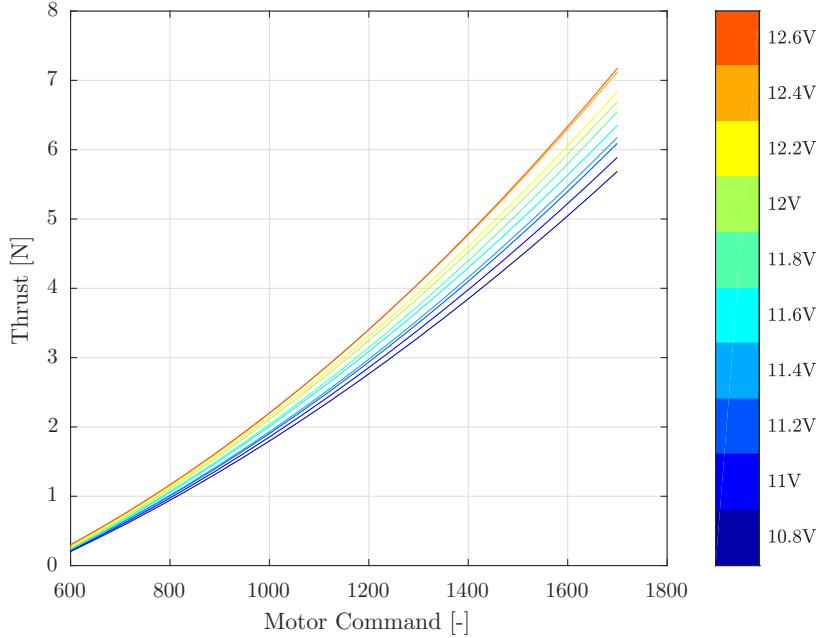


Figure 6: Second order polynomial thrust mapping (c.f. (3)) for different voltages between 10.8 V and 12.6 V identified on an ATI Mini40 load cell.

command that compensates for the varying battery voltage can therefore be obtained as

$$c_{\text{cmd,comp}} = (k_1^v v_{\text{bat}} + k_0^v) c_{\text{cmd}} \quad (65)$$

where v_{bat} is the measured battery voltage and k_1^v and k_0^v are the identified coefficients of the linear function describing the voltage dependent thrust ratio. Compared to [14], our method only needs to measure the battery voltage not the state of charge of the battery. Furthermore, since other influences on the thrust, such as the air density, are also multiplicative, we can identify all of them lumped together with this procedure. The benefits of this procedure are illustrated in Fig. 8.

5 Trajectory Generation

5.1 Computing Relevant Maxima from a Reference State

When designing a trajectory, it is often desired to enforce some limits which need to be computed from a desired state. This section shows how to compute the most relevant values for quadrotors, namely the speed, the collective thrust, and the norm of the roll and pitch rates (especially relevant for vision-based quadrotors) as it is implemented in our polynomial_trajectories library. We assume that from a sample point on a given trajectory, we can get the reference state \mathbf{s}_{des} containing the position, velocity, acceleration, and jerk.

$$\mathbf{s}_{\text{des}} = [\mathbf{R}_{\text{des}} \quad \mathbf{v}_{\text{des}} \quad \mathbf{a}_{\text{des}} \quad \mathbf{j}_{\text{des}}] \quad (66)$$

Speed and Collective Thrust

The speed can simply be obtained by taking the norm of the reference velocity

$$v = \|\mathbf{v}_{\text{des}}\|, \quad (67)$$

and the collective thrust can be computed as

$$c = \|\mathbf{a}_{\text{des}} - \mathbf{g}\|, \quad (68)$$

with $\mathbf{g} = [0 \ 0 \ -g]^{\top}$.

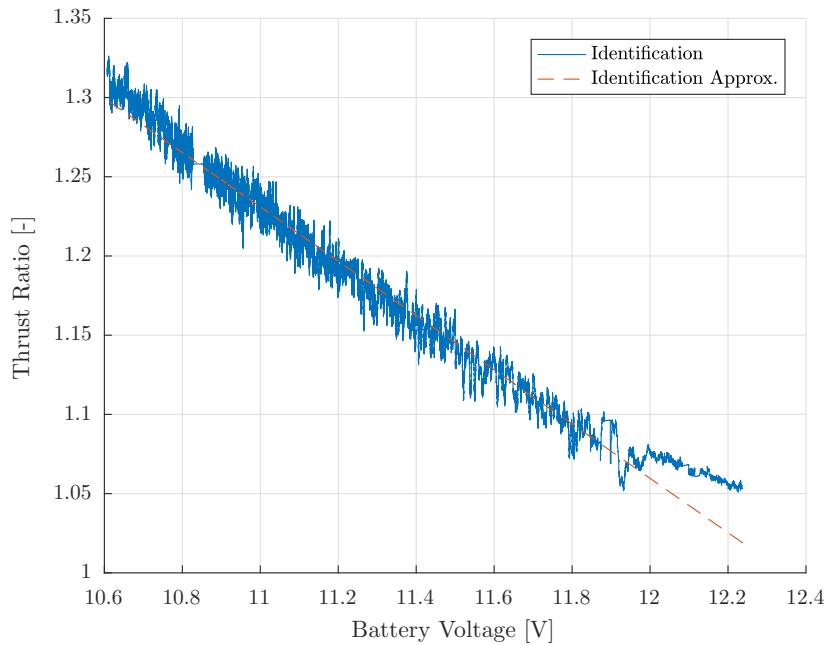


Figure 7: Estimation of thrust correction coefficient.

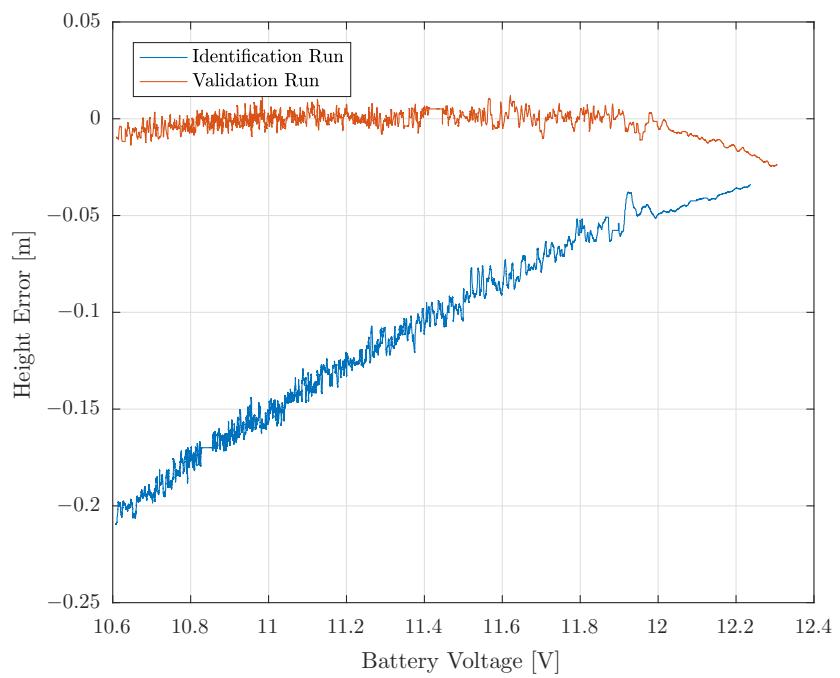


Figure 8: Validation of voltage compensation.

Roll and Pitch Rate Norm

We introduce the desired acceleration vector in world coordinates as

$$\mathbf{f} = \mathbf{a}_{des} - \mathbf{g} = \mathbf{R}_{WB} \cdot \begin{bmatrix} 0 \\ 0 \\ c \end{bmatrix} = \mathbf{R}_{WB} \cdot \mathbf{c}, \quad (69)$$

where \mathbf{g} is defined as above. Note that $\|\mathbf{f}\| = c$ since the rotation of a vector does not change its length. Therefore we can say

$$\mathbf{R}_{WB} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \frac{\mathbf{f}}{\|\mathbf{f}\|} = \dot{\mathbf{f}}. \quad (70)$$

Taking the derivative leads to

$$\mathbf{R}_{WB} \cdot \dot{\boldsymbol{\omega}} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \dot{\mathbf{f}}, \quad (71)$$

$$\dot{\boldsymbol{\omega}} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \mathbf{R}_{WB}^T \cdot \dot{\mathbf{f}}, \quad (72)$$

$$\begin{bmatrix} \omega_y \\ -\omega_x \\ 0 \end{bmatrix} = \mathbf{R}_{WB}^T \cdot \dot{\mathbf{f}}, \quad (73)$$

where $\dot{\mathbf{f}}$ is computed as

$$\dot{\mathbf{f}} = \frac{d}{dt} \left(\frac{\mathbf{f}}{\|\mathbf{f}\|} \right), \quad (74)$$

$$= \frac{\dot{\mathbf{f}} \cdot \|\mathbf{f}\| - \mathbf{f} \cdot \frac{\mathbf{f}^T \cdot \dot{\mathbf{f}}}{\|\mathbf{f}\|}}{\|\mathbf{f}\|^2}, \quad (75)$$

$$= \frac{\dot{\mathbf{f}}}{\|\mathbf{f}\|} - \frac{\mathbf{f} \cdot \mathbf{f}^T \cdot \dot{\mathbf{f}}}{\|\mathbf{f}\|^3}, \quad (76)$$

$$= \frac{\mathbf{j}}{\|\mathbf{f}\|} - \frac{\mathbf{f} \cdot \mathbf{f}^T \cdot \mathbf{j}}{\|\mathbf{f}\|^3}, \quad (77)$$

using the fact that $\dot{\mathbf{f}} = \mathbf{j}$. With all this we can write

$$\begin{bmatrix} \omega_y \\ -\omega_x \\ 0 \end{bmatrix} = \mathbf{R}_{WB}^T \cdot \left(\frac{\mathbf{j}}{\|\mathbf{f}\|} - \frac{\mathbf{f} \cdot \mathbf{f}^T \cdot \mathbf{j}}{\|\mathbf{f}\|^3} \right), \quad (78)$$

$$= \mathbf{R}_{WB}^T \cdot \frac{\mathbf{j}}{c} - \mathbf{R}_{WB}^T \frac{\mathbf{R}_{WB} \cdot \mathbf{c} \cdot \mathbf{c}^T \cdot \mathbf{R}_{WB}^T \cdot \mathbf{j}}{c^3}, \quad (79)$$

$$= \mathbf{R}_{WB}^T \cdot \frac{\mathbf{j}}{c} - \frac{\mathbf{c} \cdot \mathbf{c}^T \cdot \mathbf{R}_{WB}^T \cdot \mathbf{j}}{c^3}. \quad (80)$$

Noting that

$$\mathbf{c} \cdot \mathbf{c}^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & c^2 \end{bmatrix}, \quad (81)$$

and using (118), we get

$$\begin{bmatrix} \omega_y \\ -\omega_x \\ 0 \end{bmatrix} = \begin{bmatrix} (\mathbf{e}_x^B)^T \cdot \frac{\mathbf{j}}{c} \\ (\mathbf{e}_y^B)^T \cdot \frac{\mathbf{j}}{c} \\ (\mathbf{e}_z^B)^T \cdot \frac{\mathbf{j}}{c} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ (\mathbf{e}_z^B)^T \cdot \frac{\mathbf{j}}{c} \end{bmatrix}. \quad (82)$$

Note that $\left\| \mathbf{R}_{WB}^T \cdot \frac{\mathbf{j}}{c} \right\| = \left\| \frac{\mathbf{j}}{c} \right\|$ and that \mathbf{e}_z^B can be retrieved from the desired acceleration as

$$\mathbf{e}_z^B = \frac{\mathbf{a}_{des} - \mathbf{g}}{\|\mathbf{a}_{des} - \mathbf{g}\|}. \quad (83)$$

Therefore, we can compute the norm of the roll and pitch body rates as

$$\|\omega_{x,y}\| = \left\| \begin{bmatrix} \omega_y \\ -\omega_x \\ 0 \end{bmatrix} \right\| = \sqrt{\left\| \frac{\mathbf{j}}{c} \right\|^2 - \left((\mathbf{e}_z^B)^T \cdot \frac{\mathbf{j}}{c} \right)^2} \quad (84)$$

6 Math

6.1 Mechanics

6.1.1 Euler's first law

The linear momentum of a rigid body, \mathbf{p} is defined as

$$\mathbf{p} := m \cdot \mathbf{v}_S, \quad (85)$$

where m is the mass of the body, and \mathbf{v}_S is the velocity of the center of mass. Euler's first law states that the change of linear momentum is equal to the sum of all external forces \mathbf{F} :

$$\dot{\mathbf{p}} = \mathbf{F}, \quad (86)$$

Differentiating (85) and inserting in (86) gives

$$\dot{\mathbf{p}} = m \cdot \mathbf{a}_S = \mathbf{F}, \quad (87)$$

where \mathbf{a}_S is the acceleration of the center of mass.

6.1.2 Euler's second law

The angular momentum of a rigid body with respect to a stationary point O is defined as

$$\mathbf{L}_O := \mathbf{R}_{OP} \times \mathbf{p} + \mathbf{I}_P \cdot \boldsymbol{\Omega} + m \cdot \mathbf{R}_{PS} \times \mathbf{v}_P, \quad (88)$$

where P is an arbitrary point on the body, and \mathbf{I}_P is the second moment of inertia matrix with respect to a rotation around P , $\boldsymbol{\Omega}$ is the angular velocity of the body. Evaluating (88) not for an arbitrary P but at the center of mass $P = S$ we find

$$\mathbf{L}_O = \mathbf{R}_{OS} \times \mathbf{p} + \mathbf{I}_S \cdot \boldsymbol{\Omega}. \quad (89)$$

Euler's second law states that the change of angular momentum is equal to the sum of all external moments \mathbf{M}_O with respect to O :

$$\dot{\mathbf{L}}_O = \mathbf{M}_O. \quad (90)$$

Differentiating (89) and inserting in (90) gives

$$\dot{\mathbf{L}}_O = \mathbf{R}_{OS} \times \dot{\mathbf{p}} + \mathbf{I}_S \cdot \boldsymbol{\Psi} + \boldsymbol{\Omega} \times \mathbf{I}_S \cdot \boldsymbol{\Omega} = \mathbf{M}_O, \quad (91)$$

where $\boldsymbol{\Psi}$ is the angular acceleration of the body. We can further rewrite (91) as

$$\begin{aligned} \dot{\mathbf{L}}_O - \mathbf{R}_{OS} \times \dot{\mathbf{p}} &= \mathbf{I}_S \cdot \boldsymbol{\Psi} + \boldsymbol{\Omega} \times \mathbf{I}_S \cdot \boldsymbol{\Omega} = \mathbf{M}_O - \mathbf{R}_{OS} \times \dot{\mathbf{p}}, \\ &= \mathbf{M}_O + \mathbf{R}_{SO} \times \dot{\mathbf{p}}, \\ &= \mathbf{M}_O + \mathbf{R}_{SO} \times \mathbf{F}, \\ \mathbf{I}_S \cdot \boldsymbol{\Psi} + \boldsymbol{\Omega} \times \mathbf{I}_S \cdot \boldsymbol{\Omega} &= \mathbf{M}_S. \end{aligned} \quad (92)$$

6.1.3 Differentiating the Angular Momentum

This section shows how to differentiate \mathbf{L}_o . First, we restate (89) for the definition of the angular momentum

$$\mathbf{L}_o = \mathbf{R}_{os} \times \mathbf{p} + \mathbf{I}_s \cdot \boldsymbol{\Omega}. \quad (93)$$

Later we will use Euler's differentiation rule, therefore we restate it here. Euler's differentiation rule is used to differentiate in a body fixed coordinate system B

$${}_{B}\dot{\mathbf{c}} = ({}_{B}\mathbf{c})' + {}_W\boldsymbol{\omega}_{WB} \times {}_B\mathbf{c}. \quad (94)$$

Note the special notation for $({}_{B}\mathbf{c})'$ which is just the differentiation over time of ${}_{B}\mathbf{c}$ which is different than the differentiation expressed in the B system ${}_{B}\dot{\mathbf{c}}$. Now, lets start to calculate $\dot{\mathbf{L}}_o$. For better readability we differentiate the two terms in (93) separately:

$$(\mathbf{R}_{os} \times \mathbf{p}) = \dot{\mathbf{R}}_{os} \times \mathbf{p} + \mathbf{R}_{os} \times \dot{\mathbf{p}}, \quad (95)$$

$$= v_s \times \mathbf{p} + \mathbf{R}_{os} \times \dot{\mathbf{p}}, \quad (96)$$

$$= v_s \times m \cdot v_s + \mathbf{R}_{os} \times \dot{\mathbf{p}}, \quad (97)$$

$$= \mathbf{R}_{os} \times \dot{\mathbf{p}}. \quad (98)$$

To differentiate $\mathbf{I}_s \cdot \boldsymbol{\Omega}$ we evaluate it in the body fixed coordinate system B . We do this because the second moment of inertia matrix ${}_{B}\mathbf{I}_s$ is constant in a body fixed frame

$${}_B(\mathbf{I}_s \cdot \boldsymbol{\Omega}) = ({}_{B}\mathbf{I}_{SB}\boldsymbol{\Omega})' + {}_B\boldsymbol{\Omega} \times {}_{B}\mathbf{I}_s \cdot {}_B\boldsymbol{\Omega}, \quad (99)$$

$$= ({}_{B}\mathbf{I}_s) {}^0_B\boldsymbol{\Omega} + {}_{B}\mathbf{I}_s ({}_{B}\boldsymbol{\Omega})' + {}_B\boldsymbol{\Omega} \times {}_{B}\mathbf{I}_s \cdot {}_B\boldsymbol{\Omega}, \quad (100)$$

$$= {}_{B}\mathbf{I}_s ({}_{B}\boldsymbol{\Omega})' + {}_B\boldsymbol{\Omega} \times {}_{B}\mathbf{I}_s \cdot {}_B\boldsymbol{\Omega}. \quad (101)$$

To rewrite $({}_{B}\boldsymbol{\Omega})'$ we differentiate the angular velocity of the body using euler's formula (94)

$${}_{B}\dot{\boldsymbol{\Omega}} = ({}_{B}\boldsymbol{\Omega})' + {}_W\boldsymbol{\omega}_{WB} \times {}_{B}\boldsymbol{\Omega}, \quad (102)$$

$$= ({}_{B}\boldsymbol{\Omega})' + {}_{B}\boldsymbol{\Omega} \times {}_{B}\boldsymbol{\Omega}, \quad (103)$$

$$= ({}_{B}\boldsymbol{\Omega})'. \quad (104)$$

Inserting (104) in (101) we can continue with the differentiation of the second term

$${}_B(\mathbf{I}_s \cdot \boldsymbol{\Omega}) = {}_{B}\mathbf{I}_{SB}\dot{\boldsymbol{\Omega}} + {}_B\boldsymbol{\Omega} \times {}_{B}\mathbf{I}_s \cdot {}_B\boldsymbol{\Omega}. \quad (105)$$

This result is valid in any coordinate system not only the body fixed, therefore we can write (105) as

$$(\mathbf{I}_s \cdot \boldsymbol{\Omega}) = \mathbf{I}_s \dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega} \times \mathbf{I}_s \cdot \boldsymbol{\Omega}. \quad (106)$$

Combining the results (98) and (106) we finally find

$$\dot{\mathbf{L}}_o = \mathbf{R}_{os} \times \dot{\mathbf{p}} + \mathbf{I}_s \dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega} \times \mathbf{I}_s \cdot \boldsymbol{\Omega}, \quad (107)$$

$$= \mathbf{R}_{os} \times \dot{\mathbf{p}} + \mathbf{I}_s \Psi + \boldsymbol{\Omega} \times \mathbf{I}_s \cdot \boldsymbol{\Omega}. \quad (108)$$

Again this result is valid in any coordinate system.

6.1.4 Summary

Note that we can not only evaluate (87) and (92) in the world frame W but in any rotating coordinate systems A and C .

$$m \cdot {}_A\mathbf{a}_s = {}_A\mathbf{F}, \quad (109)$$

$${}_C\mathbf{I}_s \cdot {}_C\Psi + {}_C\boldsymbol{\Omega} \times {}_C\mathbf{I}_s \cdot {}_C\boldsymbol{\Omega} = {}_C\mathbf{M}_s. \quad (110)$$

6.2 Attitude Representations

This is not an introduction to coordinate transformations but a small summary of the important math we need to describe the dynamics of a quadrotor.

6.2.1 Rotation Matrices and Euler Angles

We denote the rotation matrix that converts from system B to W as \mathbf{R}_{WB} and the translation of system B with respect to system W as \mathbf{t}_{WB} , respectively. To convert a vector \mathbf{c} from the body frame B to the world frame W we use

$${}^W\mathbf{c} = \mathbf{R}_{WB} \cdot {}^B\mathbf{c}. \quad (111)$$

Correspondingly, to convert a point \mathbf{p} from the body frame B to the world frame W we use

$${}^W\mathbf{p} = \mathbf{R}_{WB} \cdot {}^B\mathbf{p} + {}^W\mathbf{t}_{WB}. \quad (112)$$

Later on Euler angles can be used. It is really important to keep in mind that with Euler angles the order of each single rotation is important. When you are using formulas with euler angles, always check what convention they are using. Here we will use the $z - y - x$ convention:

1. yaw ψ around the z body axis
2. pitch θ around the new y body axis
3. roll ϕ around the new x body axis

$$\mathbf{R}_{WB} = \mathbf{R}_z(\psi) \mathbf{R}_y(\theta) \mathbf{R}_x(\phi), \quad (113)$$

where

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (114)$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}, \quad (115)$$

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}. \quad (116)$$

Putting it all together we get

$$\mathbf{R}_{WB} = \begin{bmatrix} \cos(\psi) \cos(\theta) & \cos(\psi) \sin(\theta) \sin(\phi) - \sin(\psi) \cos(\phi) & \cos(\psi) \sin(\theta) \cos(\phi) + \sin(\psi) \sin(\phi) \\ \sin(\psi) \cos(\theta) & \sin(\psi) \sin(\theta) \sin(\phi) + \cos(\psi) \sin(\phi) & \sin(\psi) \sin(\theta) \cos(\phi) - \cos(\psi) \sin(\phi) \\ -\sin(\theta) & \cos(\theta) \sin(\phi) & \cos(\theta) \cos(\phi) \end{bmatrix}, \quad (117)$$

$$= [\mathbf{x}_B \quad \mathbf{y}_B \quad \mathbf{z}_B], \quad (118)$$

with \mathbf{x}_B , \mathbf{y}_B and \mathbf{z}_B being the orthogonal basis vectors of the Body coordinate system B represented in world coordinates W as illustrated in Fig. 2. The angular velocity of coordinate system B with respect to the world frame W is denoted as $\boldsymbol{\omega}_{WB}$. For the next step, we define the skew symmetric matrix

$$\hat{\boldsymbol{\omega}}_{WB} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}. \quad (119)$$

The angular velocity is defined as

$${}^W\hat{\boldsymbol{\omega}}_{WB} := \dot{\mathbf{R}}_{WB} \cdot \mathbf{R}_{WB}^T, \quad (120)$$

or in body frame B

$${}_{\text{B}}\hat{\boldsymbol{\omega}}_{\text{WB}} = \mathbf{R}_{\text{BW}} \cdot {}_{\text{W}}\hat{\boldsymbol{\omega}}_{\text{WB}} \cdot \mathbf{R}_{\text{BW}}^T, \quad (121)$$

$$= \mathbf{R}_{\text{BW}} \cdot \dot{\mathbf{R}}_{\text{WB}} \cdot \mathbf{R}_{\text{WB}}^T \cdot \mathbf{R}_{\text{BW}}^T, \quad (122)$$

$$= \mathbf{R}_{\text{WB}}^T \cdot \dot{\mathbf{R}}_{\text{WB}}. \quad (123)$$

Evaluating (120) and (123) for $\boldsymbol{\omega}_{\text{WB}}$ we get

$${}_{\text{W}}\boldsymbol{\omega}_{\text{WB}} = \begin{bmatrix} \dot{\phi} c(\theta) c(\psi) - \dot{\theta} s(\psi) \\ \dot{\phi} c(\theta) s(\psi) + \dot{\theta} c(\psi) \\ -\dot{\phi} s(\theta) + \dot{\psi} \end{bmatrix} = \begin{bmatrix} c(\theta) c(\psi) & -s(\psi) & 0 \\ c(\theta) s(\psi) & c(\psi) & 0 \\ -s(\theta) & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}, \quad (124)$$

and

$${}_{\text{B}}\boldsymbol{\omega}_{\text{WB}} = \begin{bmatrix} \dot{\phi} - s(\theta)\dot{\psi} \\ c(\phi)\dot{\theta} + s(\phi)c(\theta)\dot{\psi} \\ -s(\phi)\dot{\theta} + c(\phi)c(\theta)\dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s(\theta) \\ 0 & c(\phi) & s(\phi)c(\theta) \\ 0 & -s(\phi) & c(\phi)c(\theta) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}. \quad (125)$$

6.2.2 Quaternions

Instead of using Euler angles or rotation matrices, the orientation of a quadrotor can also be represented as a quaternion. We denote

$$\mathbf{q}_{\text{WB}} = [q_w \quad q_x \quad q_y \quad q_z]^T \quad (126)$$

as the quaternion that describes the orientation of coordinate frame B with respect to coordinate frame W , where q_w is the real part of the quaternion. Note that \mathbf{q}_{WB} is what you get from optitrack. The adjoint, norm, and inverse of the quaternion, \mathbf{q} , are

$$\bar{\mathbf{q}} = [q_w \quad -q_x \quad -q_y \quad -q_z]^T, \quad (127)$$

$$\|\mathbf{q}\| = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2}, \quad (128)$$

$$\mathbf{q}^{-1} = \frac{\bar{\mathbf{q}}}{\|\mathbf{q}\|}. \quad (129)$$

Quaternion Multiplication Quaternion multiplication is not commutative. The Quaternion multiplication between quaternions \mathbf{q} and \mathbf{p} is defined as

$$\mathbf{q} \otimes \mathbf{p} = \mathbf{Q}(\mathbf{q}) \cdot \mathbf{p} = \bar{\mathbf{Q}}(\mathbf{p}) \cdot \mathbf{q}, \quad (130)$$

$$\mathbf{p} \otimes \mathbf{q} = \mathbf{Q}(\mathbf{p}) \cdot \mathbf{q} = \bar{\mathbf{Q}}(\mathbf{q}) \cdot \mathbf{p}, \quad (131)$$

$$(132)$$

where

$$\mathbf{Q}(\mathbf{q}) = \begin{bmatrix} q_w & -q_x & -q_y & -q_z \\ q_x & q_w & -q_z & q_y \\ q_y & q_z & q_w & -q_x \\ q_z & -q_y & q_x & q_w \end{bmatrix}, \quad (133)$$

$$\bar{\mathbf{Q}}(\mathbf{q}) = \begin{bmatrix} q_w & -q_x & -q_y & -q_z \\ q_x & q_w & q_z & -q_y \\ q_y & -q_z & q_w & q_x \\ q_z & q_y & -q_x & q_w \end{bmatrix}. \quad (134)$$

We find that

$$\mathbf{Q}(\bar{\mathbf{q}}) = \mathbf{Q}(\mathbf{q})^T, \quad (135)$$

$$\bar{\mathbf{Q}}(\bar{\mathbf{q}}) = \bar{\mathbf{Q}}(\mathbf{q})^T. \quad (136)$$

Rotating a Vector by a Quaternion To rotate a vector \mathbf{v} by a quaternion \mathbf{q} we use the notion $\mathbf{q} \odot \mathbf{v}$. To rotate a vector ${}_B\mathbf{v}$ represented in body coordinates B into world coordinates W we can apply

$$\begin{bmatrix} 0 \\ {}_W\mathbf{v} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{q}_{WB} \odot {}_B\mathbf{v} \end{bmatrix} = \bar{\mathbf{Q}}^\top(\mathbf{q}_{WB}) \cdot \mathbf{Q}(\mathbf{q}_{WB}) \cdot \begin{bmatrix} 0 \\ {}_B\mathbf{v} \end{bmatrix}. \quad (137)$$

This equation will be more clear by looking at the derivations for transforming a quaternion into a rotation matrix as described in Section 6.2.3.

Quaternion Rates to Angular Velocity The time derivative of the unit quaternion is the vector of quaternion rates. The quaternion rates $\dot{\mathbf{q}}$ are related to the angular velocities similar to (120).

$$\begin{bmatrix} 0 \\ {}_W\boldsymbol{\omega}_{WB} \end{bmatrix} = 2\dot{\mathbf{q}}_{WB} \otimes \bar{\mathbf{q}}_{WB}, \quad (138)$$

$$= 2\mathbf{Q}(\dot{\mathbf{q}}_{WB}) \bar{\mathbf{q}}_{WB}, \quad (139)$$

$$= 2\bar{\mathbf{Q}}(\bar{\mathbf{q}}_{WB}) \dot{\mathbf{q}}_{WB}, \quad (140)$$

or in body fixed frame B

$$\begin{bmatrix} 0 \\ {}_W\boldsymbol{\omega}_{WB} \end{bmatrix} = 2\bar{\mathbf{q}}_{WB} \otimes \dot{\mathbf{q}}_{WB}. \quad (141)$$

More compactly, we can write

$${}_W\boldsymbol{\omega}_{WB} = 2\mathbf{W}(\mathbf{q}_{WB}) \dot{\mathbf{q}}_{WB}, \quad (142)$$

$${}_W\boldsymbol{\omega}_{WB} = 2\mathbf{W}'(\mathbf{q}_{WB}) \dot{\mathbf{q}}_{WB}, \quad (143)$$

with $\mathbf{W}(\mathbf{q}_{WB})$ and $\mathbf{W}'(\mathbf{q}_{WB})$ as defined in Section 6.2.3.

6.2.3 Changing Transformation Representation

This section lists the common conversions between different orientation representations.

Rotation Matrix to Euler Angles For this conversion, we denote R_{ij} as the element in row i and column j of \mathbf{R}_{WB} .

$$\phi = \text{atan2}(R_{32}, R_{33}) \quad (144)$$

$$\theta = \text{atan2}(R_{31}, \sqrt{R_{32}^2 + R_{33}^2}) \quad (145)$$

$$\psi = \text{atan2}(R_{21}, R_{11}) \quad (146)$$

Euler Angles to Rotation Matrix

$$\mathbf{R}_{WB} = \begin{bmatrix} c(\psi) c(\theta) & c(\psi) s(\theta) s(\phi) - s(\psi) c(\phi) & c(\psi) s(\theta) c(\phi) + s(\psi) s(\phi) \\ s(\psi) c(\theta) & s(\psi) s(\theta) s(\phi) + c(\psi) c(\phi) & s(\psi) s(\theta) c(\phi) - c(\psi) s(\phi) \\ -s(\theta) & c(\theta) s(\phi) & c(\theta) c(\phi) \end{bmatrix} \quad (147)$$

Quaternion to Euler Angles

$$\phi = \text{atan2}(2q_w q_x + 2q_y q_z, q_w^2 - q_x^2 - q_y^2 + q_z^2) \quad (148)$$

$$\theta = -\text{asin}(2q_x q_z - 2q_w q_y) \quad (149)$$

$$\psi = \text{atan2}(2q_w q_z + 2q_x q_y, q_w^2 + q_x^2 - q_y^2 - q_z^2) \quad (150)$$

Euler Angles to Quaternion

$$\mathbf{q}_{WB} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{\phi}{2}\right) \cos\left(\frac{\theta}{2}\right) \cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\phi}{2}\right) \sin\left(\frac{\theta}{2}\right) \sin\left(\frac{\psi}{2}\right) \\ \sin\left(\frac{\phi}{2}\right) \cos\left(\frac{\theta}{2}\right) \cos\left(\frac{\psi}{2}\right) - \cos\left(\frac{\phi}{2}\right) \sin\left(\frac{\theta}{2}\right) \sin\left(\frac{\psi}{2}\right) \\ \cos\left(\frac{\phi}{2}\right) \sin\left(\frac{\theta}{2}\right) \cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\phi}{2}\right) \cos\left(\frac{\theta}{2}\right) \sin\left(\frac{\psi}{2}\right) \\ \cos\left(\frac{\phi}{2}\right) \cos\left(\frac{\theta}{2}\right) \sin\left(\frac{\psi}{2}\right) - \sin\left(\frac{\phi}{2}\right) \sin\left(\frac{\theta}{2}\right) \cos\left(\frac{\psi}{2}\right) \end{bmatrix} \quad (151)$$

Quaternion to Rotation Matrix Unit quaternions are quaternions with unity norm. Throughout this section, we assume that

$$\|\mathbf{q}\| = 1. \quad (152)$$

To convert a vector \mathbf{c} from the body frame B to the world frame W we use

$$\begin{bmatrix} 0 \\ {}_W \mathbf{c} \end{bmatrix} = \mathbf{q} \cdot \begin{bmatrix} 0 \\ {}_B \mathbf{c} \end{bmatrix} \cdot \mathbf{q}^{-1}, \quad (153)$$

$$= \mathbf{q} \cdot \begin{bmatrix} 0 \\ {}_B \mathbf{c} \end{bmatrix} \cdot \bar{\mathbf{q}}, \quad (154)$$

$$= \bar{\mathbf{Q}}(\mathbf{q})^\top \mathbf{Q}(\mathbf{q}) \begin{bmatrix} 0 \\ {}_B \mathbf{c} \end{bmatrix}, \quad (155)$$

$$= \begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{0} & \mathbf{R}_{WB} \end{bmatrix} \begin{bmatrix} 0 \\ {}_B \mathbf{c} \end{bmatrix}. \quad (156)$$

From (155) and (156), it can be seen that the rotation matrix can be as

$$\mathbf{R}_{WB} = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2, & 2q_x q_y - 2q_w q_z, & 2q_w q_y + 2q_x q_z \\ 2q_w q_z + 2q_x q_y, & q_w^2 - q_x^2 + q_y^2 - q_z^2, & 2q_y q_z - 2q_w q_x \\ 2q_x q_z - 2q_w q_y, & 2q_w q_x + 2q_y q_z, & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix}. \quad (157)$$

To rewrite (155) and (156), we define

$$\mathbf{W}(\mathbf{q}_{WB}) = \begin{bmatrix} -q_x & q_w & -q_z & q_y \\ -q_y & q_z & q_w & -q_x \\ -q_z & -q_y & q_x & q_w \end{bmatrix}, \quad (158)$$

$$\mathbf{W}'(\mathbf{q}_{WB}) = \begin{bmatrix} -q_x & q_w & q_z & -q_y \\ -q_y & -q_z & q_w & q_x \\ -q_z & q_y & -q_x & q_w \end{bmatrix}, \quad (159)$$

which allows us to write (157) as

$$\mathbf{R}_{WB} = \mathbf{W}(\mathbf{q}_{WB}) \mathbf{W}'(\mathbf{q}_{WB})^\top. \quad (160)$$

Note that just as with rotation matrices, sequences of rotations are represented by products of quaternions. That is, for unit quaternions it holds that

$$\mathbf{R}(\mathbf{q} \cdot \mathbf{p}) = \mathbf{R}(\mathbf{q}) \mathbf{R}(\mathbf{p}). \quad (161)$$

Getting a quaternion directly from a rotation matrix is not straight forward but is also not used very often in our applications and if, we let Eigen do it.

6.3 Classical Modeling of a Quadrotor

6.3.1 Using Euler Angles

In this section we use Euler angles and the corresponding rotation matrices as defined in Section 6.2.1, where the rotation matrix $\mathbf{R}_{WB}(\phi, \theta, \psi)$ is a function of the euler angles.

We can evaluate (87) in the world frame as

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \mathbf{R}_{WB}(\phi, \theta, \psi) \begin{bmatrix} 0 \\ 0 \\ c \end{bmatrix}, \quad (162)$$

or in vectorized form

$$\ddot{\mathbf{p}} = {}_W\mathbf{g} + \mathbf{R}_{WB}(\phi, \theta, \psi) \cdot \mathbf{c}. \quad (163)$$

Next, we evaluate (92) in the body coordinate system B

$$\begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2}l(f_1 - f_2 - f_3 + f_4) \\ \frac{\sqrt{2}}{2}l(-f_1 - f_2 + f_3 + f_4) \\ \kappa(f_1 - f_2 + f_3 - f_4) \end{bmatrix}, \quad (164)$$

$$\begin{bmatrix} J_{xx}\dot{p} + q \cdot r(J_{zz} - J_{yy}) \\ J_{yy}\dot{q} + p \cdot r(J_{xx} - J_{zz}) \\ J_{zz}\dot{r} + p \cdot q(J_{yy} - J_{xx}) \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2}l(f_1 - f_2 - f_3 + f_4) \\ \frac{\sqrt{2}}{2}l(-f_1 - f_2 + f_3 + f_4) \\ \kappa(f_1 - f_2 + f_3 - f_4) \end{bmatrix}, \quad (165)$$

or in vectorized form

$$\mathbf{J} \cdot {}_B\dot{\boldsymbol{\omega}}_{WB} + {}_W\boldsymbol{\omega}_{WB} \times \mathbf{J} \cdot {}_W\boldsymbol{\omega}_{WB} = \boldsymbol{\eta}. \quad (166)$$

Now, we want to write the quadrotor dynamics in state space form. First, we chose the state vector as

$$\mathbf{s} = [x \ y \ z \ \dot{x} \ \dot{y} \ \dot{z} \ \phi \ \theta \ \psi \ p \ q \ r]^\top, \quad (167)$$

$$= [{}_W\mathbf{p}_{WB} \ {}_W\mathbf{v}_{WB} \ \phi \ \theta \ \psi \ {}_W\boldsymbol{\omega}_{WB}]^\top. \quad (168)$$

In the following we rewrite (162) and (165) in order to fit the state space form.

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}, \quad (169)$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \mathbf{R}_{WB}(\phi, \theta, \psi) \begin{bmatrix} 0 \\ 0 \\ c \end{bmatrix}, \quad (170)$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & t(\theta)s(\phi) & t(\theta)c(\phi) \\ 0 & c(\phi) & -s(\phi) \\ 0 & \frac{s(\phi)}{c(\theta)} & \frac{c(\phi)}{c(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \quad (171)$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{1}{J_{xx}} \left(\frac{\sqrt{2}}{2}l(f_1 - f_2 - f_3 + f_4) - q \cdot r(J_{zz} - J_{yy}) \right) \\ \frac{1}{J_{yy}} \left(\frac{\sqrt{2}}{2}l(-f_1 - f_2 + f_3 + f_4) - p \cdot r(J_{xx} - J_{zz}) \right) \\ \frac{1}{J_{zz}} (\kappa(f_1 - f_2 + f_3 - f_4) - q \cdot p(J_{yy} - J_{xx})) \end{bmatrix}, \quad (172)$$

or in vectorized form

$${}_W\dot{\mathbf{p}}_{WB} = {}_W\mathbf{v}_{WB}, \quad (173)$$

$${}_W\dot{\mathbf{v}}_{WB} = {}_W\mathbf{g} + \mathbf{R}_{WB}(\phi, \theta, \psi) \cdot \mathbf{c}, \quad (174)$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & t(\theta)s(\phi) & t(\theta)c(\phi) \\ 0 & c(\phi) & -s(\phi) \\ 0 & \frac{s(\phi)}{c(\theta)} & \frac{c(\phi)}{c(\theta)} \end{bmatrix} {}_W\boldsymbol{\omega}_{WB}, \quad (175)$$

$${}_B\dot{\boldsymbol{\omega}}_{WB} = \mathbf{J}^{-1} \cdot (\boldsymbol{\eta} - {}_W\boldsymbol{\omega}_{WB} \times \mathbf{J} \cdot {}_W\boldsymbol{\omega}_{WB}). \quad (176)$$

Finally, the derivation of the Euler angle derivatives in (171) and (175) is shown here. They are a function of the Euler angles and the body rates:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{\phi}(\phi, \theta, \psi, p, q, r) \\ \dot{\theta}(\phi, \theta, \psi, p, q, r) \\ \dot{\psi}(\phi, \theta, \psi, p, q, r) \end{bmatrix}. \quad (177)$$

For this we use (125):

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi) \cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi) \cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}. \quad (178)$$

Inverting (178) finally leads to

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & t(\theta) \sin(\phi) & t(\theta) \cos(\phi) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \quad (179)$$

6.3.2 Using Rotation Matrices

We define \mathbf{R}_{WB} to be the rotation matrix which transforms vectors from B to W , but also describes the orientation of the quadrotor. In this section we will use the vectorized forms of the equations only.

We can evaluate (87) in the world frame as

$$\ddot{\mathbf{p}} = {}_W\mathbf{g} + \mathbf{R}_{WB} \cdot \mathbf{c}. \quad (180)$$

Since we use the same representation of the body rates as in Section 6.3.1, the evaluation of (92) remains the same as in (166).

We chose the state vector as

$$\mathbf{s} = [{}_W\mathbf{p}_{WB} \quad {}_W\mathbf{v}_{WB} \quad \mathbf{R}_{WB} \quad {}_W\boldsymbol{\omega}_{WB}]^\top. \quad (181)$$

In the following, we rewrite (180) and (166) in order to fit the state space form.

$${}_W\dot{\mathbf{p}}_{WB} = {}_W\mathbf{v}_{WB}, \quad (182)$$

$${}_W\dot{\mathbf{v}}_{WB} = {}_W\mathbf{g} + \mathbf{R}_{WB} \cdot \mathbf{c}, \quad (183)$$

$$\dot{\mathbf{R}}_{WB} = \mathbf{R}_{WB} \cdot {}_B\hat{\boldsymbol{\omega}}_{WB}, \quad (184)$$

$${}_B\hat{\boldsymbol{\omega}}_{WB} = \mathbf{J}^{-1} \cdot (\boldsymbol{\eta} - {}_W\boldsymbol{\omega}_{WB} \times \mathbf{J} \cdot {}_W\boldsymbol{\omega}_{WB}). \quad (185)$$

6.3.3 Using Quaternions

In this section, we use a quaternion \mathbf{q}_{WB} , as defined in Section 6.2.2, to represent the orientation of the quadrotor. We will again use the vectorized forms of the equations only.

We can evaluate (87) in the world frame as

$$\ddot{\mathbf{p}} = {}_W\mathbf{g} + \mathbf{q}_{WB} \odot \mathbf{c}. \quad (186)$$

Since we use the same representation of the body rates as in Section 6.3.1, the evaluation of (92) remains the same as in (166).

We chose the state vector as

$$\mathbf{s} = [{}_W\mathbf{p}_{WB} \quad {}_W\mathbf{v}_{WB} \quad \mathbf{q}_{WB} \quad {}_W\boldsymbol{\omega}_{WB}]^\top. \quad (187)$$

In the following, we rewrite (186) and (166) in order to fit the state space form.

$${}_{\text{W}}\dot{\mathbf{p}}_{\text{WB}} = {}_{\text{W}}\mathbf{v}_{\text{WB}}, \quad (188)$$

$${}_{\text{W}}\dot{\mathbf{v}}_{\text{WB}} = {}_{\text{W}}\mathbf{g} + \mathbf{q}_{\text{WB}} \odot \mathbf{c}, \quad (189)$$

$$\dot{\mathbf{q}}_{\text{WB}} = \mathbf{\Lambda}({}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}) \cdot \mathbf{q}_{\text{WB}}, \quad (190)$$

$${}_{\text{B}}\dot{\boldsymbol{\omega}}_{\text{WB}} = \mathbf{J}^{-1} \cdot (\boldsymbol{\eta} - {}_{\text{W}}\boldsymbol{\omega}_{\text{WB}} \times \mathbf{J} \cdot {}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}), \quad (191)$$

where

$$\mathbf{\Lambda}({}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}) = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} = \frac{1}{2} \bar{\mathbf{Q}} \left(\begin{bmatrix} 0 \\ {}_{\text{W}}\boldsymbol{\omega}_{\text{WB}} \end{bmatrix} \right) = \frac{1}{2} \bar{\mathbf{Q}} \left(\begin{bmatrix} 0 \\ p \\ q \\ r \end{bmatrix} \right). \quad (192)$$

We find (190) by using (141)

$$\dot{\mathbf{q}}_{\text{WB}} = \frac{1}{2} \mathbf{q}_{\text{WB}} \cdot \begin{bmatrix} 0 \\ {}_{\text{W}}\boldsymbol{\omega}_{\text{WB}} \end{bmatrix}, \quad (193)$$

$$= \frac{1}{2} \bar{\mathbf{Q}} \left(\begin{bmatrix} 0 \\ {}_{\text{W}}\boldsymbol{\omega}_{\text{WB}} \end{bmatrix} \right) \mathbf{q}_{\text{WB}}. \quad (194)$$

Equation (190) can be integrated as described in Appendix B.

Appendices

A Singularities in Reference Inputs

In this section, we present some practical workarounds for some special cases where the presented math is not valid. These workarounds typically only work if we can assume that these special cases only occur for very short durations at a time.

Case - $\mathbf{y}_C \times \boldsymbol{\alpha} = \mathbf{0}$ This case occurs if either \mathbf{y}_C is aligned with $\boldsymbol{\alpha}$ (defined in (14)) or $\boldsymbol{\alpha} = \mathbf{0}$. In this case, any \mathbf{x}_B that is perpendicular to \mathbf{y}_C satisfies the constraint (14). To overcome this ambiguity, we compute \mathbf{x}_B by projecting the estimated body x -axis into the $\mathbf{x}_C - \mathbf{z}_C$ plane and normalizing it as

$$\mathbf{x}_B = \frac{\mathbf{x}_{B,\text{est}} - (\mathbf{x}_{B,\text{est}}^\top \mathbf{y}_C) \mathbf{y}_C}{\|\mathbf{x}_{B,\text{est}} - (\mathbf{x}_{B,\text{est}}^\top \mathbf{y}_C) \mathbf{y}_C\|}. \quad (195)$$

If the obtained $\|\mathbf{x}_{B,\text{est}} - (\mathbf{x}_{B,\text{est}}^\top \mathbf{y}_C) \mathbf{y}_C\| = \mathbf{0}$, we set $\mathbf{x}_B = \mathbf{x}_C$. Note that this might lead to jumps in the desired orientation. By assuming that this special case only occurs for a short duration it might be better to just remember the last desired orientation that was computed before the special case occurred.

Case - $\boldsymbol{\beta} \times \mathbf{x}_B = \mathbf{0}$ This case occurs if either \mathbf{x}_B is aligned with $\boldsymbol{\beta}$ (defined in (15)) or $\boldsymbol{\beta} = \mathbf{0}$. In this case, any \mathbf{y}_B that is perpendicular to \mathbf{x}_B satisfies the constraint (15). To overcome this ambiguity, we compute \mathbf{y}_B by the cross product of the estimated body z -axis $\mathbf{z}_{B,\text{est}}$ and \mathbf{x}_B and normalizing it as

$$\mathbf{y}_B = \frac{\mathbf{z}_{B,\text{est}} \times \mathbf{x}_B}{\|\mathbf{z}_{B,\text{est}} \times \mathbf{x}_B\|}. \quad (196)$$

If the obtained $\|\mathbf{z}_{B,\text{est}} \times \mathbf{x}_B\| = \mathbf{0}$, we set $\mathbf{y}_B = \mathbf{y}_C$. Note that this might lead to jumps in the desired orientation. By assuming that this special case only occurs for a short duration it might be better to just remember the last desired orientation that was computed before the special case occurred.

Case - Inverted Flight where $\mathbf{z}_W^\top \boldsymbol{\alpha} < 0$ In the case where $\mathbf{z}_W^\top \boldsymbol{\alpha} < 0$, (18) leads to a body x -axis \mathbf{x}_B which has a projection into the $\mathbf{x}_W - \mathbf{y}_W$ plane that points into the $-\mathbf{x}_C$ direction. It is still collinear to \mathbf{x}_C but this causes the actual heading to be off by 180° from the reference heading ψ_{ref} . Enforcing the reference heading during inverted flight by changing the sign of the desired \mathbf{x}_B axis might lead to a jump in the desired orientation. However, to the best of our knowledge, it is not possible to prevent jumps in the orientation for any trajectory. For example, when performing a vertical loop where parts of it are flown upside down, we end up with a continuous orientation of the quadrotor for a reference heading $\psi_{\text{ref}} = 0^\circ$ when allowing the quadrotor to have its actual heading 180° off as long as it flies upside down. When doing the same with a reference heading $\psi_{\text{ref}} = 90^\circ$, we do not end up with a continuous orientation of the quadrotor when applying the same method. In this particular case, changing the sign of \mathbf{x}_B when the quadrotor is inverted, would lead to a continuous orientation over the entire loop. In summary, the presented computation of the desired orientation is not well suited for inverted flights, which require special considerations.

Case - $\mathcal{A}_2 = 0$ or $(\mathcal{B}_1 \mathcal{C}_3 - \mathcal{B}_3 \mathcal{C}_1) = 0$ The solutions of the body rates (27)-(29) and angular accelerations (39)-(41) are obtained by divisions where the denominator is composed of coefficients as defined in (30)-(38). These denominators can become zero which makes the body rates and angular accelerations undefined. Similarly to $\boldsymbol{\alpha} = \mathbf{0}$ and $\boldsymbol{\beta} = \mathbf{0}$, this is the case where the quadrotor is executing ballistic trajectories. In such cases we set $\boldsymbol{\omega} = \mathbf{0}$ and $\dot{\boldsymbol{\omega}} = \mathbf{0}$.

B Zeroth Order Integration of a Unit Quaternion

Simply integrating a quaternion derivative component-wise does not guarantee to result in a unit quaternion and requires normalization. Under the assumption of a discrete time step in the interval $[t, t + \Delta t]$ with constant body rates ${}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}$, a nice solution for the integration exists (from [15]) which does not require to normalize the quaternion. First we restate (190)

$$\dot{\mathbf{q}}_{\text{WB}} = \boldsymbol{\Lambda}({}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}) \cdot \mathbf{q}_{\text{WB}}. \quad (197)$$

In literature we find the solution to this differential equation with starting point $\mathbf{q}_{\text{WB}}(t)$ to be

$$\mathbf{q}_{\text{WB}}(t + \Delta t) = e^{(\boldsymbol{\Lambda}({}_{\text{W}}\boldsymbol{\omega}_{\text{WB}})\Delta t)} \cdot \mathbf{q}_{\text{WB}}(t). \quad (198)$$

The matrix exponential is defined as

$$e^{(\boldsymbol{\Lambda}({}_{\text{W}}\boldsymbol{\omega}_{\text{WB}})\Delta t)} = \mathbf{I}_4 + \boldsymbol{\Lambda}({}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}) \cdot \Delta t + \frac{1}{2!} (\boldsymbol{\Lambda}({}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}) \cdot \Delta t)^2 + \frac{1}{3!} (\boldsymbol{\Lambda}({}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}) \cdot \Delta t)^3 + \dots \quad (199)$$

It can easily be verified that

$$(\boldsymbol{\Lambda}({}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}))^2 = -\frac{1}{4} \|{}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}\|^2 \cdot \mathbf{I}_4, \quad (200)$$

where

$$\|{}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}\| = \sqrt{p^2 + q^2 + r^2}. \quad (201)$$

Using (200) and rearranging the terms, (199) can be written as

$$e^{(\boldsymbol{\Lambda}({}_{\text{W}}\boldsymbol{\omega}_{\text{WB}})\Delta t)} = \mathbf{I}_4 \cdot \left(1 - \frac{(\|{}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}\| \frac{\Delta t}{2})^2}{2!} + \frac{(\|{}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}\| \frac{\Delta t}{2})^4}{4!} - \dots \right), \quad (202)$$

$$\begin{aligned} &+ \frac{2}{\|{}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}\|} \boldsymbol{\Lambda}({}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}) \cdot \left(\frac{(\|{}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}\| \frac{\Delta t}{2})}{1!} - \frac{(\|{}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}\| \frac{\Delta t}{2})^3}{3!} + \dots \right), \\ &= \mathbf{I}_4 \cdot \cos\left(\frac{\|{}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}\| \Delta t}{2}\right) + \frac{2}{\|{}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}\|} \cdot \boldsymbol{\Lambda}({}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}) \cdot \sin\left(\frac{\|{}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}\| \Delta t}{2}\right). \end{aligned} \quad (203)$$

So the state update of the quaternion becomes

$$\mathbf{q}_{\text{WB}}(t + \Delta t) = \left(\mathbf{I}_4 \cdot \cos\left(\frac{\|{}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}\| \Delta t}{2}\right) + \frac{2}{\|{}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}\|} \cdot \boldsymbol{\Lambda}({}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}) \cdot \sin\left(\frac{\|{}_{\text{W}}\boldsymbol{\omega}_{\text{WB}}\| \Delta t}{2}\right) \right) \cdot \mathbf{q}_{\text{WB}}(t). \quad (204)$$

C Proof of not Measuring Gravity in Flight

When a quadrotor is flying, its accelerometer is not measuring gravity but all the other forces acting on the quadrotor (Thrust, Disturbances, ...). In fact, the accelerometer is never measuring gravity. Just by definition, the magnitude of the measured accelerations in hover, or when standing on the ground, is equal to the gravitational acceleration. But note that in these cases we measure $\tilde{\mathbf{a}} = -\mathbf{g}$ which is due to the upwards force acting on the quadrotor in order to have it standing still in a world frame. We demonstrate this using a simple model for an accelerometer. Fig. 9 shows the schematics of a model of an accelerometer in 2D. We model the accelerometer as a mass m_M connected by springs in all three axes to its casing which is rigidly attached to the quadrotor with mass m_q . The springs are modelled to be linear with spring constant k_s . The output of the accelerometer is the estimated acceleration of the quadrotor $\tilde{\mathbf{a}}_q$, which is measured by the deflection \mathbf{d} of the mass m_M . The force of the spring due to its deflection is

$$\mathbf{f}_s = k_s \cdot \mathbf{d} = m_M \cdot \tilde{\mathbf{a}}_q. \quad (205)$$

By establishing the equilibrium of forces for the mass m_M and the quadrotor we get

$$\mathbf{a}_M \cdot m_M = m_M \cdot \mathbf{g} + \mathbf{f}_s, \quad (206)$$

$$\mathbf{a}_q \cdot m_q = \mathbf{f}_{ct} + \mathbf{f}_d + m_q \cdot \mathbf{g} - \mathbf{f}_s, \quad (207)$$

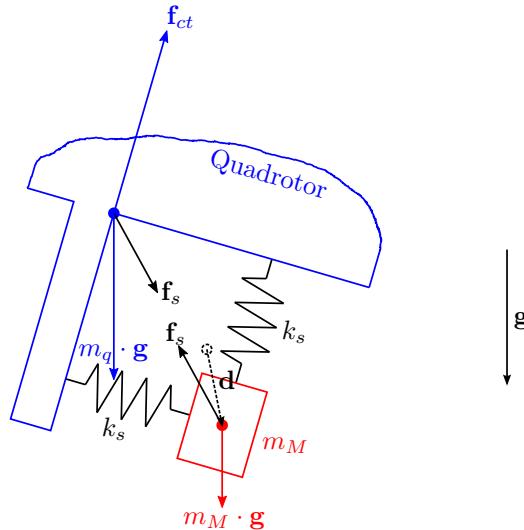


Figure 9: Schematics of the accelerometer model. The mass m_M is connected to the quadrotor by linear springs with spring constant k_s . No dynamical effects of the mass m_M and its deflection \mathbf{d} are considered.

where \mathbf{a}_M and \mathbf{a}_q are the actual accelerations of the mass m_M and the quadrotor represented in the world frame, respectively. The vector $\mathbf{g} = [0 \ 0 \ -g]^\top$ denotes the gravitational acceleration, $\mathbf{f}_{ct} = m_q \cdot \mathbf{c}$ is the force due to the applied collective thrust and \mathbf{f}_d are the forces due to external disturbances. By substituting (205) into (206) and (207), we get

$$\mathbf{a}_M \cdot m_M = m_M \cdot \mathbf{g} + m_M \cdot \tilde{\mathbf{a}}_q, \quad (208)$$

$$\mathbf{a}_q \cdot m_q = \mathbf{f}_{ct} + \mathbf{f}_d + m_q \cdot \mathbf{g} - m_M \cdot \tilde{\mathbf{a}}_q. \quad (209)$$

We neglect any dynamics of the accelerometer mass and assume that for a given acceleration the corresponding deflection is reached instantly. Therefore, we can set $\mathbf{a}_M \equiv \mathbf{a}_q$. Using this and substituting (208) into (209), we get

$$(\tilde{\mathbf{a}}_q + \mathbf{g}) \cdot m_q = \mathbf{f}_{ct} + \mathbf{f}_d + m_q \cdot \mathbf{g} - m_M \cdot \tilde{\mathbf{a}}_q. \quad (210)$$

Since the mass m_M is very tiny, we can assume that $m_M \ll m_q$. Therefore, we can solve (210) for the measured acceleration of the quadrotor $\tilde{\mathbf{a}}_q$ as

$$\tilde{\mathbf{a}}_q = \frac{1}{m_q} \cdot (\mathbf{f}_{ct} + \mathbf{f}_d). \quad (211)$$

This shows that the accelerometer only measures accelerations due to the collective thrust applied on the quadrotor. Because of this fact, it is not possible to estimate the attitude of a quadrotor in flight without drift (even in roll and pitch) by only using IMU measurements. Nonetheless, when the quadrotor has contact to a static object, and therefore is constrained in position, roll and pitch can be estimated without drift by only using IMU measurements.

References

- [1] M. Faessler, A. Franchi, and D. Scaramuzza, “Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories,” *IEEE Robot. Autom. Lett.*, vol. 3, pp. 620–626, Apr. 2018.
- [2] M. Faessler, D. Falanga, and D. Scaramuzza, “Thrust mixing, saturation, and body-rate control for accurate aggressive quadrotor flight,” *IEEE Robot. Autom. Lett.*, vol. 2, pp. 476–482, Apr. 2017.
- [3] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D’Andrea, “A platform for aerial robotics research and demonstration: The Flying Machine Arena,” *J. Mechatronics*, vol. 24, pp. 41–54, Feb. 2014.
- [4] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, “Autonomous, vision-based flight and live dense 3D mapping with a quadrotor MAV,” *J. Field Robot.*, vol. 33, no. 4, pp. 431–450, 2016.
- [5] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, “The GRASP multiple micro UAV testbed,” *IEEE Robot. Autom. Mag.*, vol. 17, pp. 56–65, Sept. 2010.
- [6] J.-M. Kai, G. Allibert, M.-D. Hua, and T. Hamel, “Nonlinear feedback control of quadrotors exploiting first-order drag effects,” in *IFAC World Congress*, vol. 50, pp. 8189–8195, July 2017.
- [7] J. Svacha, K. Mohta, and V. Kumar, “Improving quadrotor trajectory tracking by compensating for aerodynamic effects,” in *IEEE Int. Conf. Unmanned Aircraft Syst. (ICUAS)*, pp. 860–866, June 2017.
- [8] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors,” in *IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 2520–2525, May 2011.
- [9] M. Faessler, A. Franchi, and D. Scaramuzza, “Detailed derivations of ‘Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories’,” *arXiv e-prints*, Dec. 2017.
- [10] M. Faessler, F. Fontana, C. Forster, and D. Scaramuzza, “Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor,” in *IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 1722–1729, 2015.
- [11] T. Lee, M. Leoky, and N. McClamroch, “Geometric tracking control of a quadrotor uav on $\text{se}(3)$,” in *IEEE Conf. Decision Control (CDC)*, pp. 5420–5425, Dec. 2010.
- [12] D. Brescianini, M. Hehn, and R. D’Andrea, “Nonlinear quadrocopter attitude control,” tech. rep., Department of Mechanical and Process Engineering, ETHZ, Oct. 2013.
- [13] C. G. Mayhew, R. G. Sanfelice, and A. R. Teel, “Quaternion-based hybrid control for robust global attitude tracking,” *IEEE Trans. Autom. Control*, vol. 56, pp. 2555–2566, Nov. 2011.
- [14] M. Podhradsky, J. Bone, C. Coopmans, and A. Jensen, “Battery model-based thrust controller for a small, low cost multirotor unmanned aerial vehicles,” in *IEEE Int. Conf. Unmanned Aircraft Syst. (ICUAS)*, pp. 105–113, May 2013.
- [15] N. Trawny and S. I. Roumeliotis, “Indirect Kalman filter for 3D attitude estimation,” Tech. Rep. 2005-002, University of Minnesota, Dept. of Comp. Sci. & Eng., Mar. 2005.