FIELD AND SERVICE ROBOTICS (FSR) – a.y. 2023/2024

Homework N.2

Vincenzo Palomba
P38000180

University of Naples Federico II
Department of Electrical Engineering and Information Technology

Abstract

The report summarizes the completion of the third homework assignment, consisting of five exercises on the aerial robot section. Topics covered include octocopter, the study of different types of controllers, a description of ground and ceiling effect, while the final part focus on developing an estimator for external disturbances and a geometric control on a quadrotor model for a given trajectory.

Exercise 1

Given the *octocopter* in the figure above, it can be noticed that the system the system consists of 8 rotating propellers attached to a central body. Each propeller contributes with 1 DOF to the overall number, for a total amount of 8 DOFs, plus 6 DOFs provided from the rigid body, that can freely move into the space. Therefore, when the drone is constrained to the ground, it has a total of 8 DOFs. However, if the drone is not constrained to the ground, the octocopter has **14 DOFs** in total.

Figure 1: Octocopter

The propellers topology is T^8, while considering that, the UAV is not fixed to the ground and instead it can moves in the 3D space, should be considered also the body topology, that is $R^3 \times S^2 \times S^1$. So, under these assumption, the **Configuration space topology** becomes:

$$T^8 \times R^3 \times S^2 \times S^1 \tag{1}$$

Moreover the system is **underactuated**, as well any kind of co-planar propellers drone. In fact, it doesn't matter how many co-planar Propellers the UAV has, as long as they stay on the same plane, they can only provide

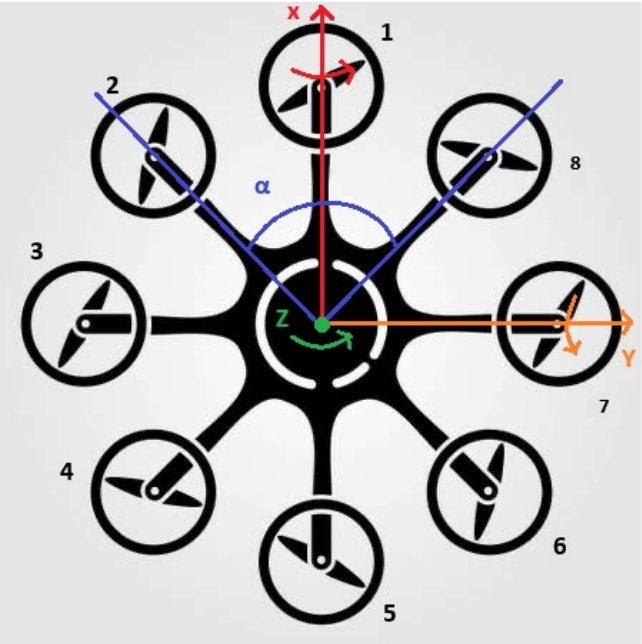

Figure 2: Octocopter scheme

4 real inputs to the system. As we can notice, the allocation matrix is $G_q \in \mathbb{R}^{4\times 8}$ and thus $rank(G_q) < dim(q) = 6$. To get $G_q \in \mathbb{R}^{6\times 8}$ and avoid the underactuation the UAV must have either non co-planar propellers (tilted configurations) or motorised propellers (actively tilting configurations). In our case, the octocopter posseses 8 co-planar propellers (8 actuators), such that the system has redundancy in how to split the velocities among all the propellers ($n > 4$), that as well, do not gives us the possibility to have more than 4 real inputs to the system.

Now let's derive the *allocation matrix* of our octocopter. We can choose indifferently between the NED and ENU body frame conventions, in order to derive the *total Thrust* and *control Torques* relations with the *propeller velocities*. The **NED** frame was chosen, in which the *x-axis* is aligned with the northernmost arm, associated with the first propeller, the *y-axis* and *z-axis* accordingly to the convention. The propellers are labeled counter-clockwise, whether the Torques are considered positive if cause a counter clockwise rotation. Note that, having 8 propellers equally spaced, means that the angle between two consecutive arms is $\alpha = 2\pi/8 = \pi/4$.

The allocation matrix can be written in a compact form:

$$\begin{bmatrix} u_T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = G_q \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \\ \omega_5^2 \\ \omega_6^2 \\ \omega_7^2 \\ \omega_8^2 \end{bmatrix} \tag{2}$$

The forces and torques acting on the system are computed accordingly to the convention in fig 2.

Based on a static approximation, each propeller generate a Thrust proportional to the square of its angular velocity ω_i^2, hence $T_i = c_T\omega_i^2$, and acts along the Z direction of the local coordinate system. So, the Total Thrust u_T is:

$$u_T = \sum_{i=1}^{8} T_i = c_T \sum_{i=1}^{8} \omega_i^2 \tag{3}$$

$$u_T = c_T\left(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2 + \omega_5^2 + \omega_6^2 + \omega_7^2 + \omega_8^2\right) \tag{4}$$

Now, let the torques acting around the X, Y and Z axes of the local coordinate system be denoted as τ_x, τ_y and τ_z, respectively (see 2). Then, we assume that the distances of the octocopter center of mass from the center of mass of each single motor are equal and denoted by l. First of all, τ_x was computed taking into account that the propellers on the west side of the x-axis make a positive contribution to the momentum on the x axis, while the east side ones make a negative contribution. Moreover, considering the symmetry of the system and

choosing α as the angle between the x axes and the second link, we obtain:

$$\tau_x = l\,(T_2 sin(\alpha) + T_3 + T_4 sin(\alpha) - T_6 sin(\alpha) - T_7 - T_8 sin(\alpha)) \tag{5}$$
$$= l\,c_T\,(\omega_2^2\,sin(\alpha) + \omega_3^2 + \omega_4^2\,sin(\alpha) - \omega_6^2\,sin(\alpha) - \omega_7^2 - \omega_8^2\,sin(\alpha)) \tag{6}$$

Then, to compute τ_y, the same procedure was taken, with appropriate consideration. In particular, the north side propellers cause a positive torque, whereas the south ones generate a negative action. Using the same α, yields:

$$\tau_y = l\,(T_8 cos(\alpha) + T_1 + T_2 cos(\alpha) - T_4 cos(\alpha) - T_5 - T_6 cos(\alpha)) \tag{7}$$
$$= l\,c_T\,(\omega_8^2 cos(\alpha) + \omega_1^2 + \omega_2^2 cos(\alpha) - \omega_4^2 cos(\alpha) - \omega_5^2 - \omega_6^2 cos(\alpha)) \tag{8}$$

Finally, the torque around the z-axis is caused by the drag forces of the spinning propellers. The angular motion of any motor included in the design causes a drag moment which is opposite to the direction of the motion according to Newton's third law. Therefore, if we also assume a static approximation of the drag (i.e proportional to the square of the angular velocity, as one can see with the relation $Q_i = c_Q\omega_i^2$) we can model the torque around the Z-axis. Generally, compensate this undesired phenomena, every drone with a symmetrical structure has propellers that rotate alternately clockwise and anticlockwise. Considering the first rotor spinning clockwise, τ_z is computed below:

$$\tau_z = -Q_1 + Q_2 - Q_3 + Q_4 - Q_5 + Q_6 - Q_7 + Q_8 \tag{9}$$
$$= c_Q\,(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2 - \omega_5^2 + \omega_6^2 - \omega_7^2 + \omega_8^2) \tag{10}$$

The system actuation can be finally described by the **Allocation Matrix**, that looks like:

$$G_q = \begin{bmatrix} c_T & c_T & c_T & c_T & c_T & c_T & c_T & c_T \\ 0 & l\,c_T\,sin(\alpha) & l\,c_T & l\,c_T\,sin(\alpha) & 0 & -l\,c_T\,sin(\alpha) & -l\,c_T & -l\,c_T\,sin(\alpha) \\ l\,c_T & l\,c_T\,cos(\alpha) & 0 & -l\,c_T\,cos(\alpha) & -l\,c_T & -l\,c_T cos(\alpha) & 0 & l\,c_T\,cos(\alpha) \\ -c_Q & c_Q & -c_Q & c_Q & -c_Q & c_Q & -c_Q & c_Q \end{bmatrix} \tag{11}$$

and, with further simplifications:

$$G_q = \begin{bmatrix} c_T & c_T & c_T & c_T & c_T & c_T & c_T & c_T \\ 0 & \frac{\sqrt{2}}{2}lc_T & lc_T & \frac{\sqrt{2}}{2}lc_T & 0 & -\frac{\sqrt{2}}{2}lc_T & -lc_T & -\frac{\sqrt{2}}{2}lc_T \\ lc_T & \frac{\sqrt{2}}{2}lc_T & 0 & -\frac{\sqrt{2}}{2}lc_T & -lc_T & -\frac{\sqrt{2}}{2}lc_T & 0 & \frac{\sqrt{2}}{2}lc_T \\ -c_Q & c_Q & -c_Q & c_Q & -c_Q & c_Q & -c_Q & c_Q \end{bmatrix} \tag{12}$$

Finally, one can note that it doesn't matter which angle or frame convention has been chosen, as long as the propeller label are consistent, the allocation matrix remains 12.

Exercise 2: Hierarchical vs Geometric vs Passivity-based controller

There are three primary types of controllers used for the position and attitude tracking of aerial robots: *hierarchical controllers, passivity-based controllers, and geometrical controllers.*

Both Hierarchical and Passivity-Based controller are implemented taking into account the *RPY dynamic model*, that makes use of a minimal representation for the orientation part (i.e Roll-Pitch-Yaw angles). One significant drawback of this approach is the introduction of singularities, which limits the range of motion and complicates control when the system's orientation approaches certain critical angles ($\theta = \pm\pi/2$).

$$\begin{cases} m\ddot{p}_b = mge_3 - u_T R_b e_3 \\ M(\eta_b)\ddot{\eta}_b = -C(\eta_b, \dot{\eta}_b)\dot{\eta}_b + Q^T(\eta_b)\tau^b \end{cases} \tag{13}$$

While hierarchical and passivity-based controllers share some similarities, particularly in the linear part of the control schema, they differ significantly in their approach to the attitude part of the control:

- Hierarchical Controller

  The hierarchical controller is advantageous for its simple interpretation of the control schema in the attitude part and its structured approach in basic maneuvers, leveraging clear separation of angular and linear dynamics. It relies on a partial feedback linearization, which works well when the system parameters are known accurately. However, this reliance can lead to robustness issues if the parameters are uncertain. The control input for the attitude part in hierarchical control is given by:

$$\tau_b = I_b Q(\eta_b)\tilde{\tau} + Q(\eta_b)^{-T} C(\eta_b, \dot{\eta}_b)\dot{\eta}_b \tag{14}$$

where I_b is the inertia matrix, $Q(\eta_b)$ is the transformation matrix, and $C(\eta_b, \dot{\eta}_b)$ represents the Coriolis and centrifugal forces. The hierarchical approach cancels out these forces through the control input, simplifying the control problem.

- Passivity-Based Controller

  In contrast, passivity-based controllers do not rely on feedback linearization, making them more robust to uncertainties in the dynamic model. However, they introduce multiple coupling parameters (e.g., σ and ν) that make tuning more complex (similar approach to the adaptative control seen in the *Siciliano's foundation of robotics book*). The control input for the attitude part in passivity-based control is:

$$\tau_b = Q^{-T}(M(\eta_b)\ddot{\eta}_r + C(\eta_b, \dot{\eta}_b)\dot{\eta}_r - \hat{\tau}_e - D_{ov}\eta - K_{oe}\eta) \tag{15}$$

  where $M(\eta_b)$ is the inertia matrix, $\ddot{\eta}_r$ and $\dot{\eta}_r$ are reference acceleration and velocity, and $\hat{\tau}_e$, D_{ov}, and K_{oe} are disturbance estimations and control gains. This approach does not fully cancel out the Coriolis and inertial terms, requiring more sophisticated tuning.

As pointed out above, both hierarchical and passivity-based controllers share a common approach in the linear part of the control, defining a desired acceleration vector μ_d.

$$\mu_d = -\frac{1}{m} u_T R_b(\eta_{b,d}) e_3 + g e_3 \tag{16}$$

Whether, if we consider the RPY dynamic model with external wrench disturbance:

$$\begin{cases} m\ddot{p}_b = mge_3 - u_T R_b e_3 + f_e \\ M(\eta_b)\ddot{\eta}_b = -C(\eta_b, \dot{\eta}_b)\dot{\eta}_b + Q^T(\eta_b)\tau^b + \tau_e \end{cases} \tag{17}$$

in both controller, μ_d becomes:

$$\mu_d = -\frac{1}{m} u_T R_b(\eta_{b,d}) e_3 + g e_3 + \frac{1}{m}\hat{f}_e \tag{18}$$

This vector, also known as the thruster control, directs the UAV's movement. The controller for μ_d is designed as a PD+ control input, which includes the compensation of the linear feedforward term. μ_d is not the actual control input; it must be converted to μ_t and desired roll and pitch using flat output relationships.

- Geometrical control

  Geometrical control, on the other side, is based on an implicit representation of orientation, avoiding the singularity problem inherent in the RPY (roll-pitch-yaw) dynamic model. Using the coordinate free model:

$$\begin{cases} m\ddot{p}_b = mge_3 - u_T R_b e_3 \\ \dot{R}_b = R_b S(\omega_b^b) \\ I_b\dot{\omega}_b^b = -S(\omega_b^b) I_b \omega_b^b + \tau^b \end{cases} \tag{19}$$

  the inner and outer loop controller relations are:

$$u_T = -(K_p e_p - K_v \dot{e}_p - mge_3 + m\ddot{p}_{b,d})^T R_b e_3 \tag{20}$$

$$z_{b,d} = -\frac{(K_p e_p - K_v \dot{e}_p - mge_3 + m\ddot{p}_{b,d})}{||(K_p e_p - K_v \dot{e}_p - mge_3 + m\ddot{p}_{b,d})||} \tag{21}$$

$$\tau_b = -K_R e_R - K_\omega e_\omega + S(\omega_b^b) I_b \omega_b^b - I_b(S(\omega_b^b) R_b^T R_{b,d}\omega_{b,d}^{b,d} - R_b^T R_{b,d}\dot{\omega}_{b,d}^{b,d}) \tag{22}$$

  Notice how, unlike the hierarchical controller, the outer loop directly give us the Total Thrust u_T, without passing through μ_d, as a result of a PD+ controller, with gravity compensation and a feedforward term, while the vector $z_{b,d}$, coupled with the references, will be used to compute R_b and τ_b.

  This controller is suitable for high-velocity and acrobatic maneuvers, unlike the RPY-based controllers, which are limited to slow movements and hover states, allowing robust performance with exponential stability. However, geometrical control has drawbacks such as difficulty in interpreting the attitude error

due to the computational complexity of the rotation matrix and limited exponential convergence of the attitude error only if the initial error is less than 90°.

Looking at the u_T expression, one can note that contains the scalar product between $z_{b,d}$ and z_b. This means that, if the angle between $z_{b,d}$ and z_b is large (but less than 90 degrees) the thrust u_t is small, because the cosine of a large angle is approximately zero. Hence, for initial attitude errors large in values, the thrust u_T decreases in magnitude, since the dot product $z_{b,d}^T z_b$ decreases, in such a way to self limit the control input.

Stability for all three types of controllers requires addressing the nonlinear interconnection between the position and attitude parts due to the underactuated nature of UAVs with flat propeller configurations. Therefore, in order to prove the stability of the liner part, Lyapunov and perturbation theory is needed. Conversely, to address the stability of the attitude part, the specific methods differ:

- Hierarchical Control

  Stability can be shown by proving that the matrix A_e in the error equation is Hurwitz:

$$\begin{bmatrix} \dot{e}_\eta \\ \ddot{e}_\eta \end{bmatrix} = A_e \begin{bmatrix} e_\eta \\ \dot{e}_\eta \end{bmatrix} \tag{23}$$

- Geometrical Control

  The attitude error equation isn't a simple second-order homogeneus differential equation, in fact e_ω is not the time derivative of e_R. Therefore, as well as in the linear part, it is required Lyapunov theory:

$$I_b \dot{e}_\omega + K_\omega e_\omega + K_R e_R = 0 \tag{24}$$

- Passivity-Based Control

  Also requires Lyapunov theory for stability proof due to the non-trivial relationship between the error terms (i.e v_η is not the time derivative of e_η):

$$M\dot{v}_\eta + (C + D_o)v_\eta + K_o e_\eta = 0 \tag{25}$$

For all controller types, a momentum-based estimator can be used to compensate for external disturbances. This estimator must be faster than the closed control loop to maintain good performance.

Finally, While hierarchical and passivity-based controllers are suitable for scenarios with known system parameters and moderate maneuvering, geometrical controllers offer advantages in robustness and high-velocity scenarios but come with increased computational complexity and tuning challenges.

Exercise 3: Ground effect and Ceiling effect

UAVs and UAMs are usually involved in applications that require flying close to different structures, objects, and obstacles, costraining the airflow produced by the rotor.

- **Ground effect** occurs when a UAV flies at a low altitude, typically within one wingspan or rotor diameter from the ground. This effect **enhances lift** and reduces induced drag due to altered airflow patterns. This is possible because, when a rotor or wing operates close to the ground, the downward airflow is restricted, increasing pressure beneath the UAV and thereby enhancing lift.

  In order to model the effect of a surface like the ground, adopting potential aerodynamic assumptions, the method of images is employed. This method consists in placing a virtual rotor for each propeller on the opposite side of the surface and at the same distance, such that if the rotors are placed at height h, the virtual ones, will be placed under the ground at negative altitude $-h$.

  The ratio between the thrust inside the ground effect $u_{T,ige}$ and the thrust outside the ground effect $u_{T,oge}$ can be derived from the conservation of energy principle and aerodynamic considerations. For a single rotor scenario, it is represented by:

$$\frac{u_{T,ige}}{u_{T,oge}} = \frac{1}{1 - (\frac{\rho}{4z})^2} \tag{26}$$

  We can see that, whenever the rotor is at altitude greater than two times its radius, the lift gain vanishes. It is important to notice that in the case of multiple rotors, the effects can combine and persist at higher altitudes than the previous scenario.

- **Ceiling effect** occurs when a UAV operates in close proximity to an overhead surface, resulting in an **increase in thrust** due to improved propeller efficiency resulting from reduced drag. Consequently, accounting for this effect becomes essential to avoid crashes, given its tendency to attract the vehicle toward obstacles.

  The increased thrust is a consequence of the *vacuum effect*: when a rotor operates near a ceiling, the airflow above it becomes restricted, decreasing drag and allows for more efficient thrust generation.

  In the context of a single rotor, the effect can be approximated by the following formula:

$$\frac{u_{T,ice}}{u_{T,oce}} = \frac{1}{1 - \frac{1}{k_1}(\frac{\rho}{z+k_2})^2} \tag{27}$$

  where $u_{T,ice}$ is the thrust inside the ceiling effect and $u_{T,oge}$ is the thrust outside the ceiling effect, while k_1 and k_2 parameters to estimate.

Ground effect and Ceiling effect, beside the obvious differences (i.e read their name), differs in:

- **Mechanism of Thrust Enhancement**: In the ceiling effect, increased thrust is attributed to the restriction of air above the rotor, reducing friction and allowing for more efficient rotation of propellers. In contrast, ground effect induces more lift due to the accumulation of air expelled at high speed between the floor and the propeller.
- **Thrust Gain Variation**: The gain in thrust varies between the two effects, considering the propellers at the same altitude z.
- **Effect on Stability**: When a hovering multirotor is subject to an attitude perturbation in the presence of ground effect, a stabilizing moment M_{ge} arises due to the differences in rotor-to-ground distances (see Fig. 1 (a)). Conversely, in the ceiling effect, an unstable moment arises, increasing the attitude perturbation (see Fig. 1 (b)).

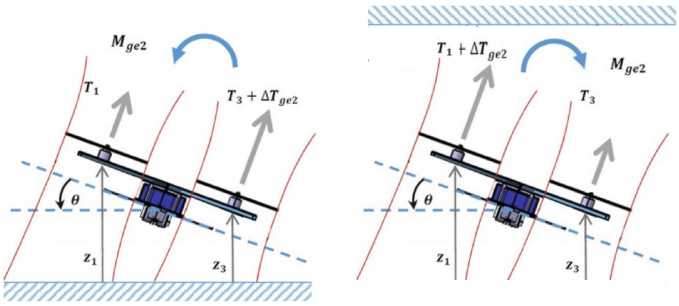

Table 1: Ground effect (a) and Ceiling effect (b) with attitude perturbation

Similarly, in scenarios where only a portion of the rotors are subject to ground or ceiling effect, such as when grasping objects (Fig. 2), since the increment in the rotor thrust depends on the distance of each rotor to the ground, the resulting moment M_{ge} either stabilizes or destabilizes the multirotor, depending on the specific effect at play.

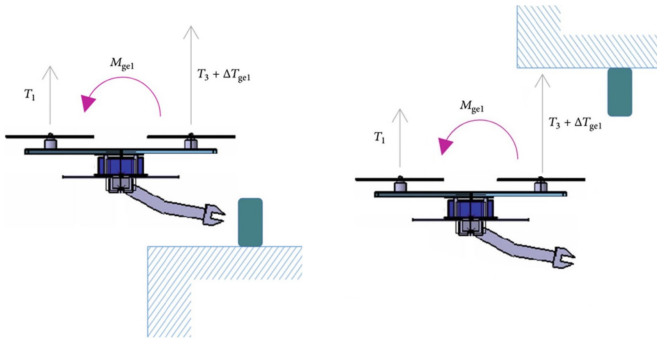

Table 2: Ground effect (a) and Ceiling effect (b) grasping objects

if only a portion of the rotors are subjected to the ground effect, a moment M_{ge} will arise, preventing the grasp of the object. Conversely, if only part of the rotors are subject to the ceiling effect the moment M_{ge} will facilitate the grasp of the object.

Exercise 4: Momentum-based estimator

Based on the measurements of commanded thrust and torques, linear velocity, attitude as Euler angles and their time derivative and considering the nominal model parameters, it is possible to design the **Momentum-based estimator** in order to estimate uncertainty of the model, unmodelled dynamics and disturbances, represented as **external wrench** (i.e $[f_e \quad \tau_e]^T$). Once these terms are estimated, they can be used to compensate for the external wrench in the dynamic model by adjusting the chosen controller. The estimator, as the name suggests, is based on the generalized momentum vector $q \in \mathbb{R}^6$, obtained by the following relation:

$$q = \begin{bmatrix} mI_3 & O_3 \\ O_3 & M(\eta_b) \end{bmatrix} \begin{bmatrix} \dot{p}_b \\ \dot{\eta}_b \end{bmatrix} \tag{28}$$

The goal is to estimate the external wrench $\begin{bmatrix} \mathbf{f_e} \\ \tau_\mathbf{e} \end{bmatrix}$ through its estimation $\begin{bmatrix} \hat{\mathbf{f}}_\mathbf{e} \\ \hat{\tau}_\mathbf{e} \end{bmatrix}$. Moreover, to make it simple, it is desired to have a linear relationship between the external wrench and its estimation in the Laplace domain, through the diagonal matrix transfer function $\mathbf{G(s)} \in \mathbb{C}^{6\times6}$, where:

$$\mathbf{G_i(s)} = \frac{c_0}{s^r + c_{r-1}s^{r-1} + ... + c_1 s + c_0}, \quad i = 1, ..., 6 \tag{29}$$

is the r^{th} order transfer function. knowing that:

$$\prod_{i=j+1}^{r} \mathbf{k_i} = c_j \quad j = 0, ..., r-1 \tag{30}$$

Assuming $\mathbf{G_i(s)}$ has r poles in p. As p increases, estimation will be faster and with more overshoot, or vice versa slower but smoother:

$$\mathbf{G_i(s)} = \frac{k_0^r}{(s+p)^r} \tag{31}$$

$$p = k_0 \tag{32}$$

$$p \geq 0 \tag{33}$$

with $Re(p) > 0$ so the system is *asimptotically stable*, while the choice $k_0 = p$ was made in order to have static gain equal to one, so to do not amplify or dampen the estimation values. The numerator and denominator are raised to the power of r to compute the recursive estimator of r-order. The $\mathbf{K_i}$ values are computed using c_j (i.e., the denominator coefficients of $\mathbf{G(s)}$) as follows:

```
% obtaining ki
Ptmp = 1; % accumulator for prod
for j = 1:r
    k(j) = c(j)/Ptmp;
    Ptmp = Ptmp*k(j);
end
k = flip(k);
```

Having all the measurement from the sensors and considering that the total thrust u_T and the torques τ_b must be the real values and not the commanded ones, the estimation of the external wrench can be made using the **Discrete-time Recursive formula** of the estimator, that can be easily derived from its continuous time form, as demonstrated in the end of the paragraph and the discrete time momentum formula, using a sample time $\mathbf{T_s} = 1ms$.

It yields:

$$q(k+1) = \begin{bmatrix} mI_3 & O_3 \\ O_3 & M(\eta(k+1)) \end{bmatrix} \begin{bmatrix} \dot{p}_b(k+1) \\ \dot{\eta}_b(k+1) \end{bmatrix} \tag{34}$$

$$\gamma_1(k+1) = \gamma_1(k) + k_1 \left(q(k+1) - q(k) - T_s \begin{bmatrix} \hat{f}_e(k) \\ \hat{\tau}_e(k) \end{bmatrix} - T_s \begin{bmatrix} mge_3 - u_T(k)R_b(k)e_3 \\ C^T(\eta_b(k), \dot{\eta}_b(k)) + Q^T(\eta_b(k))\tau^b(k) \end{bmatrix} \right) \tag{35}$$

$$\gamma_i(k+1) = \gamma_i(k) + k_i T_s \left(- \begin{bmatrix} \hat{f}_e(k) \\ \hat{\tau}_e(k) \end{bmatrix} + \gamma_{i-1}(k) \right) \quad i = 2, ..., r \tag{36}$$

In which, γ_i is the i-th order estimation. To estimate the wrench at time $k+1$, values at time k are needed. Initial conditions are given by:

$$\begin{bmatrix} \hat{f}_e(0) \\ \hat{\tau}_e(0) \end{bmatrix} = 0 \quad q(0) = 0 \tag{37}$$

It's crucial to note that those assumptions implies starting with $q = 0$, indicating zero velocity. This signifies that the drone begins from a standstill, implying that our estimator needs to run before takeoff while the drone is on the ground.

The MATLAB implementation relies on a statically allocated three-dimensional array, used to store γ_i at each time iteration:

```
for i = 1:size(t,1)-1
    % compute q
    Rb = attitude(eta(i,1), eta(i,2), eta(i,3));
    [Q, Qdot] = geometricTranform(eta(i,1), eta(i,2), etaDot(i,1), etaDot(i,2));
    M = Q'*Ib*Q;
    C = centrifugalMatrix(etaDot(i,:), Q, Qdot,Ib);

    q(:,i+1) = [m*eye(3), zeros(3,3); zeros(3,3), M]*[pbDot(i+1,:)'; etaDot(i+1,:)'];
    % compute gamma1
    gamma(:,i+1,1) = gamma(:,i,1) + k(1)*(q(:,i+1) - q(:,i) - Ts*extWrench_estim(:,i) - Ts*[(m
        *g*e3 - ut(i)*Rb*e3); C'*etaDot(i,:)' + Q'*taub(i,:)'];
    % loop for j-th gamma
    for j = 2:r
        gamma(:,i+1,j) = gamma(:,i,j) + k(j)*Ts*(-extWrench_estim(:,i) + gamma(:,i,j-1));
    end
    % update the estimation
    extWrench_estim(:,i+1) = gamma(:,i+1,r);
end
```

While the M, R_b, Q and C matrix were computed using MATLAB functions, accordingly to their definitions.

The choice of the dominant pole fell on $p = 50$, as for higher values, the settling time does not improve by about 5 seconds overall, while, on the contrary, the overshoot increases (fig. 3, 4, 5). The results show the external wrench estimated with different poles:

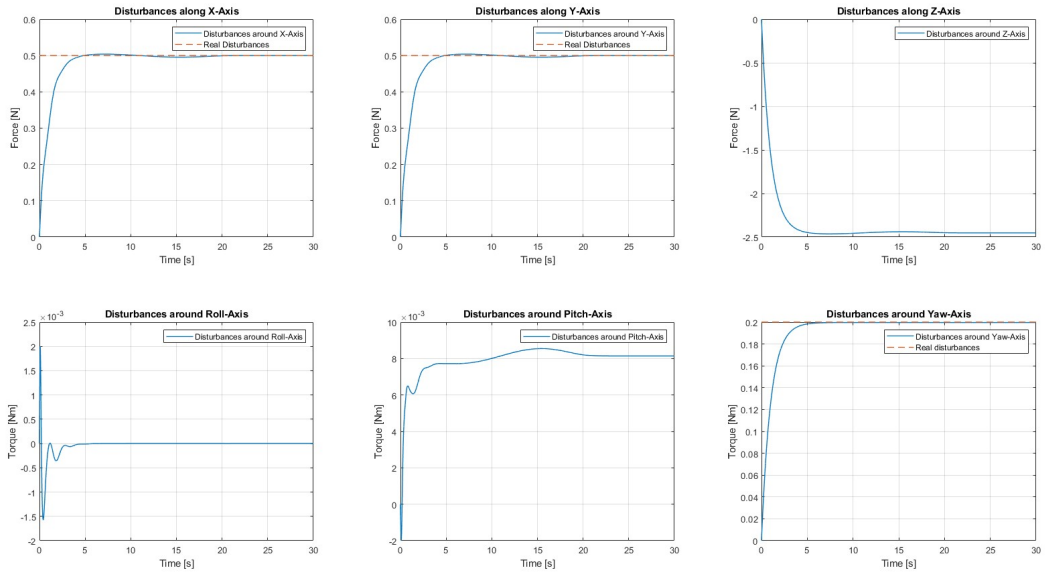

Figure 3: external wrench estimated with $r = 1$ and $p = 1$

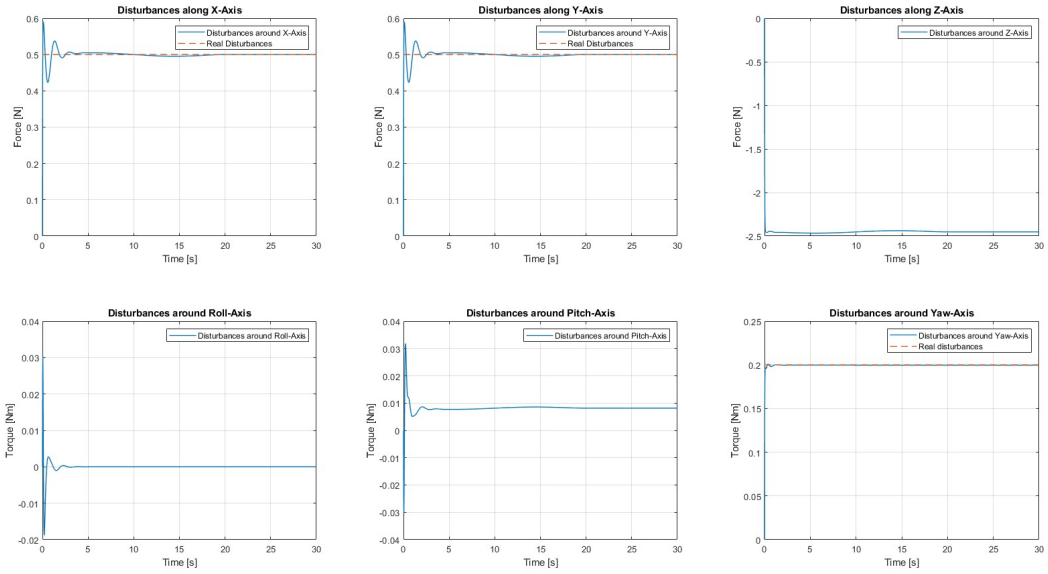

Figure 4: external wrench estimated with $r = 1$ and $p = 50$

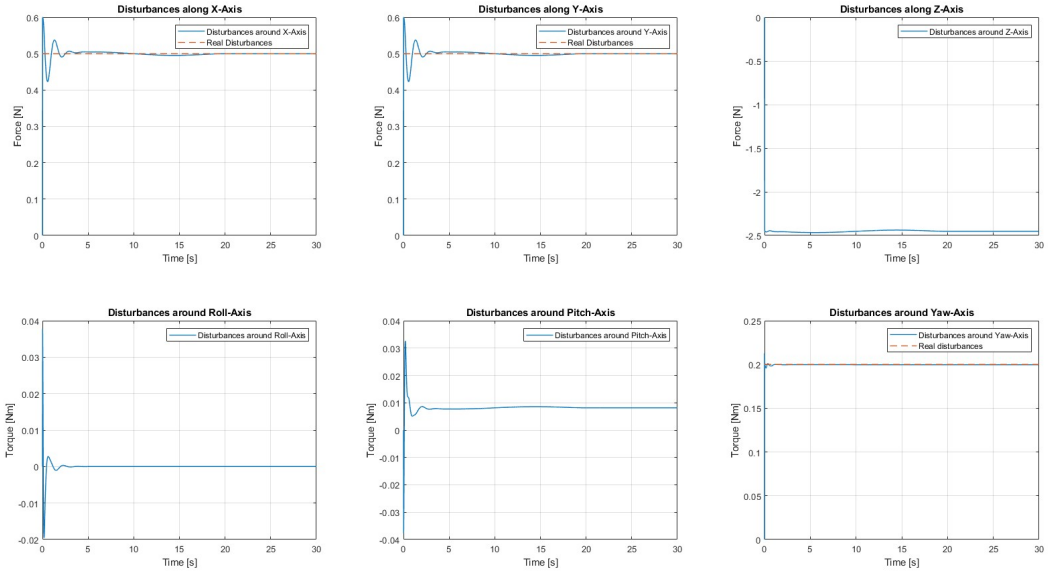

Figure 5: external wrench estimated with $r = 1$ and $p = 500$

The analysis leads to the conclusion that the **optimal estimator order r is 1**. When $p = 50$ is set, it becomes evident that there is minimal difference between the cases where $r = 1$ and $r = 5$, as depicted in Figures 4 and 6. Moreover, the improvement from the **second order onward is not significant** for the estimator. The primary difference is that the higher order $r = 5$ introduces more delay and requires more computation time. Moreover, as the order increases, so does the delay, as shown in Figures 7 and 8. This correlation arises from the increased number of integrators employed for estimation, consequently increasing the delay. However, this approach is not ideal since the estimator should ideally be the fastest system within the control loop. Excessive order increments can lead to instability, as illustrated in Figure 9.

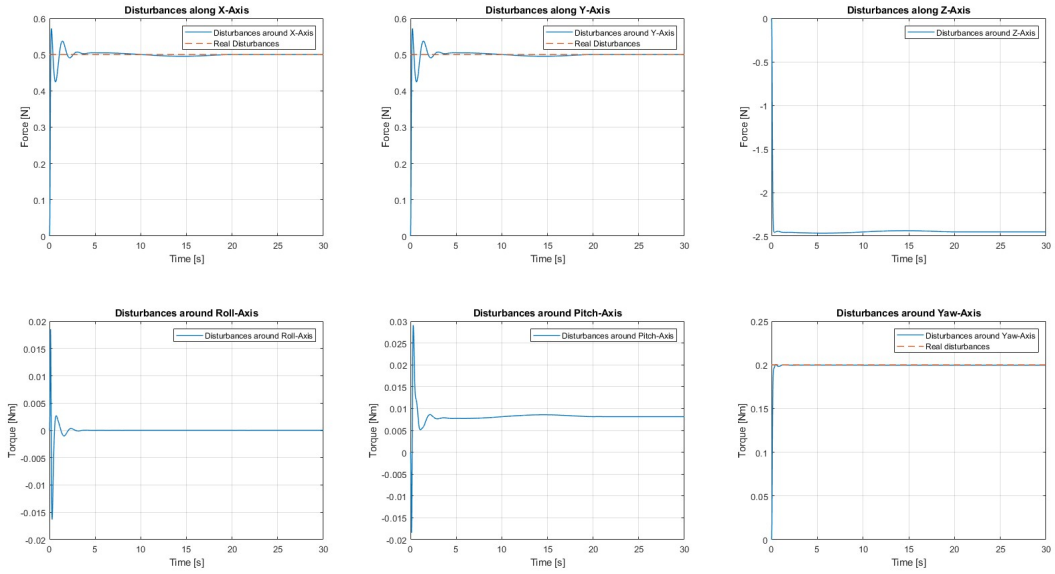

Figure 6: external wrench estimated with $r = 5$ and $p = 50$

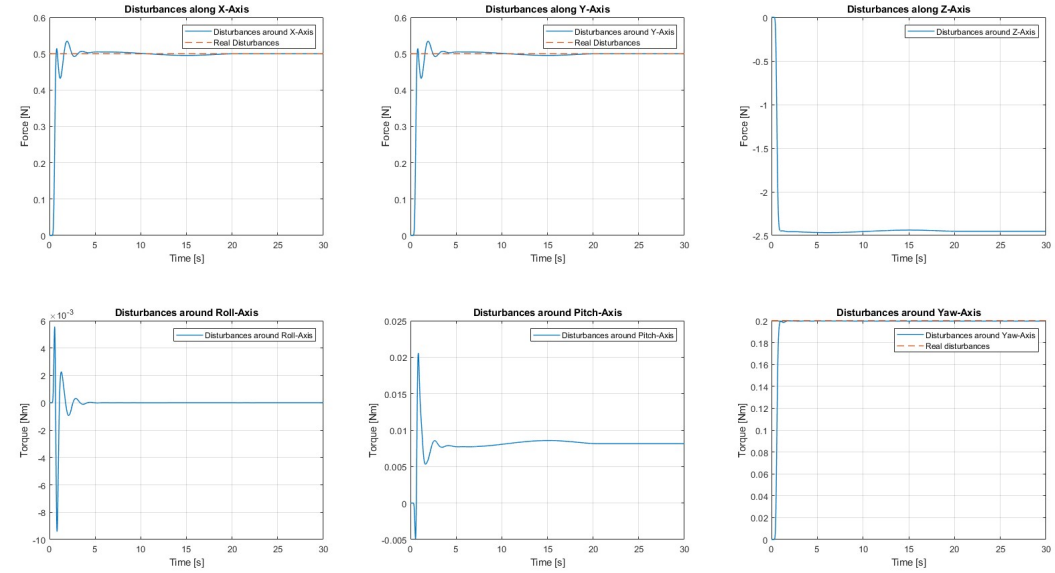

Figure 7: external wrench estimated with $r = 30$ and $p = 50$

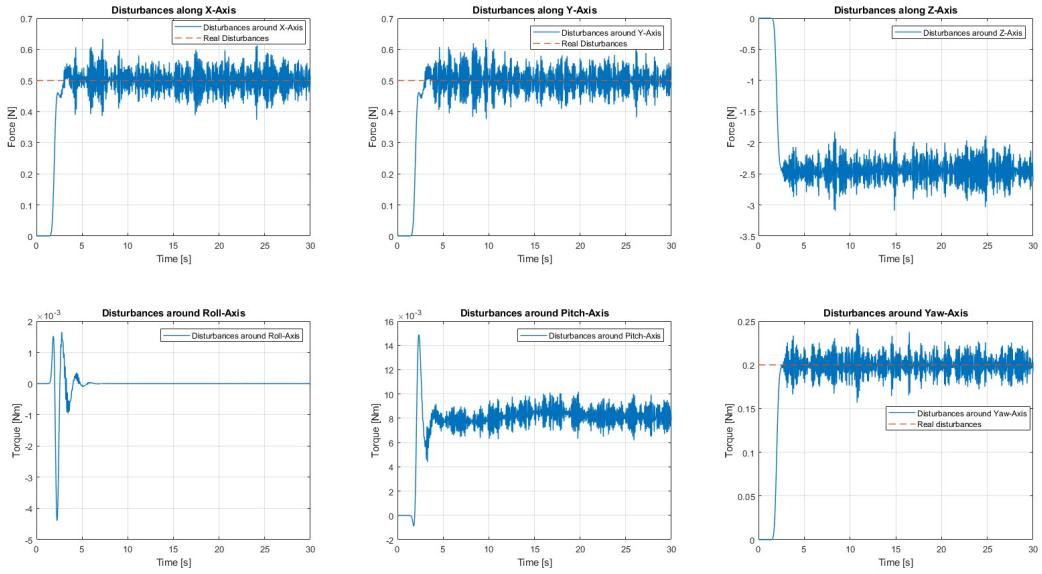

Figure 8: external wrench estimated with $r = 100$ and $p = 50$

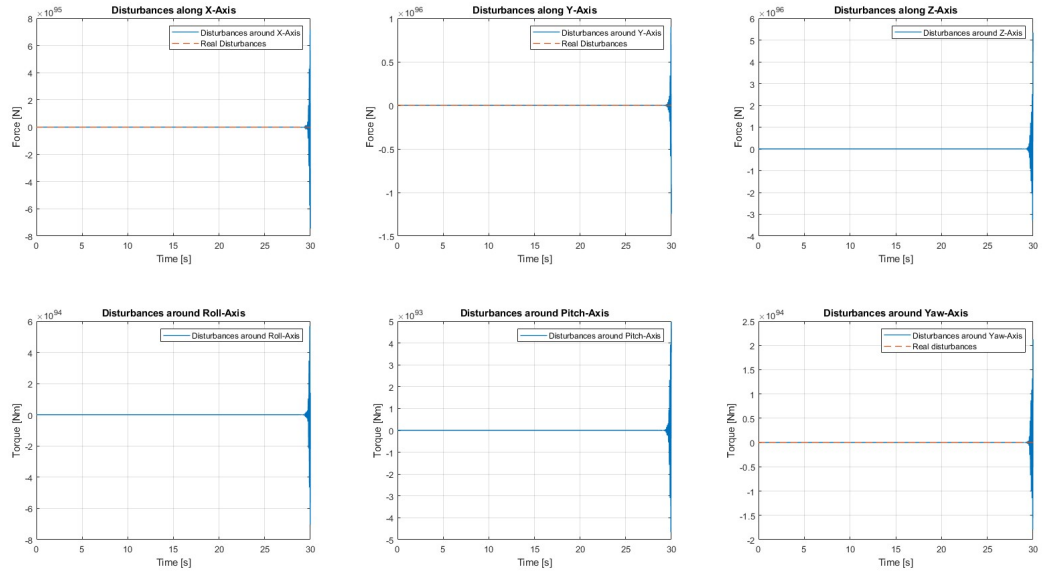

Figure 9: external wrench estimated with $r = 130$ and $p = 50$

The estimation goes to steady state in less than five seconds, giving us the correct value of the constant disturbances applied during the flight on the drone. The **estimated external wrench** are:

$$\hat{f}_e = [0.5 \quad 0.5 \quad -2.45]^T \quad \hat{\tau}_e = [0 \quad 0.0081 \quad 0.2]^T \tag{38}$$

Figure 4 and the results in Equation 38 demonstrate that the estimated **disturbances along the x-axis and y-axis are** $0.5\,N$, while the disturbance **around the yaw-axis is** $0.2\,Nm$ at steady state.

Given that there are no external disturbances along the z-axis, we can assume that any estimated external force along this axis is related to an uncertainty in the model, specifically due to an incorrect estimate of the mass.

To calculate the real mass $\mathbf{m_r}$, we start by considering the equilibrium equation for the UAV. The nominal mass m is given as 1.5 kg. From the estimated force along the z-axis, we determine the estimated mass $\tilde{m}$. Substituting $\hat{f}_z = \tilde{m}g$ into equation 39, we find that the **real mass** $\mathbf{m_r}$ is described by equation 41.

The equilibrium equation is given by:

$$0 = mg - u_T \mathbf{z} + \hat{f}_z \tag{39}$$

where $\hat{f}_z = \tilde{m}g$ and z is the z unit vector.

Rewriting the above equation with the substitution, we get:

$$0 = mg - u_T \mathbf{z} + \tilde{m}g \tag{40}$$

From this, we can derive the real mass $\mathbf{m_r}$, using the relationship:

$$\mathbf{m_r} = m + \tilde{m} \tag{41}$$

Given the values:

$$m = 1.5\,\text{kg}$$

$$\tilde{m} = -0.25\,\text{kg}$$

In which one can note that the nominal mass m was overestimated. Substituting these into the equation for $\mathbf{m_r}$, we get:

$$\mathbf{m_r} = 1.5\,\text{kg} - 0.25\,\text{kg} = 1.25\,\text{kg}$$

Thus, the **real mass $\mathbf{m_r}$ is 1.25 kg**.

0.1 Discrete time estimator formula

The first order estimator, γ_1 will be used as example for the following demonstration:

$$\gamma_1(t) = K_1\left(q(t) - \int_0^t \begin{bmatrix}\hat{f}_e \\ \hat{\tau}_e\end{bmatrix} + \begin{bmatrix} mge_3 - u_T R_b e_3 \\ C^T(\eta_b, \dot{\eta_b}) + Q^T(\eta_b)\tau^b)\end{bmatrix} dt\right) \tag{42}$$

Let's call $a(t)$ the function under the integral:

$$a(t) = \begin{bmatrix}\hat{f}_e \\ \hat{\tau}_e\end{bmatrix} + \begin{bmatrix} mge_3 - u_T R_b e_3 \\ C^T(\eta_b, \dot{\eta_b}) + Q^T(\eta_b)\tau^b)\end{bmatrix} \tag{43}$$

Now, let's differentiate both sides:

$$\frac{d}{dt}\gamma_1 = K_1\left(\frac{d}{dt}q - a(t)\right) \tag{44}$$

Using the Euler numerical method:

$$\frac{\gamma_1(k+1) - \gamma(k)}{T_s} = K_1\left(\frac{q(k+1) - q(k)}{T_s} - a(k)\right) \tag{45}$$

obtaining:

$$\gamma_1(k+1) = \gamma(k) + K_1(q(k+1) - q(k) - T_s a(k)) \tag{46}$$

Exercise 5: Geometric control of a quadrotor

Consider the Simulink file attached as *geometric_control_template.slx*. The scope was to fill in the inner and outer loops of the control scheme in order to implement the *geometric control algorithm*. In this context, the UAV flat output corresponds to the desired yaw and position values. The planner outputs the desired body frame position, linear velocity and linear acceleration, respectively $p_{b,d}, \dot{p_{b,d}}, \ddot{p_{b,d}}$ as well as the desired $x_{b,d}$, that is, in some way, related to the desired yaw. After completing the code writing of the two MATLAB functions, the next step was to tune the control parameters. The process began by selecting the parameters for the inner loop (the angular component) before addressing those of the outer loop. This approach ensures that achieving zero steady-state error in the angular component causes attaining zero steady-state error in the linear component.

Outer-Loop Control

The outer-loop control focuses on tracking the desired position $\mathbf{p}_{b,d}$ and desired yaw $\psi_{\mathbf{d}}$. The control inputs for this loop include the desired position $\mathbf{p}_{b,d}$, velocity $\dot{\mathbf{p}}_{b,d}$, and acceleration $\ddot{\mathbf{p}}_{b,d}$. The outer loop was filled, considering that it is responsible for the total thrust u_T and for the orientation reference $z_{b,d}$, used to build $R_{b,d}$.

1. **Position and Velocity Errors**:
The tracking errors for the translational motion, defined as follows:

$$\mathbf{e}_p = \mathbf{p}_b - \mathbf{p}_{b,d} \tag{47}$$

$$\dot{\mathbf{e}}_p = \dot{\mathbf{p}}_b - \dot{\mathbf{p}}_{b,d} \tag{48}$$

2. **Total Thrust**:
The total thrust u_T is calculated to counteract gravitational forces and to achieve the desired acceleration:

$$u_T = -(-K_p\mathbf{e}_p - K_v\dot{\mathbf{e}}_p - mg\mathbf{e}_3 + m\ddot{\mathbf{p}}_{b,d}) \cdot \mathbf{R}_b\mathbf{e}_3 \tag{49}$$

Here, K_p and K_v are the position and velocity gain matrices, respectively, m is the mass of the quadcopter, g is the acceleration due to gravity, and $\mathbf{R}_b$ is the rotation matrix representing the orientation of the body frame relative to the world frame. The gain matrices are chosen as:

$$K_p = diag\{100, 100, 200\} \quad K_v = diag\{1, 1, 20\} \tag{50}$$

Notice how the z-gain of the controllers is greater: this was done in order to compensate the faster variation on the z-component.

3. **Desired Z-axis of Body Frame**:
The desired z-axis $\mathbf{z}_{b,d}$ of the body frame is computed as:

$$\mathbf{z}_{b,d} = \frac{-K_p\mathbf{e}_p - K_v\dot{\mathbf{e}}_p - mg\mathbf{e}_3 + m\ddot{\mathbf{p}}_{b,d}}{\| - K_p\mathbf{e}_p - K_v\dot{\mathbf{e}}_p - mg\mathbf{e}_3 + m\ddot{\mathbf{p}}_{b,d}\|} \tag{51}$$

Inner-Loop Control (Orientation Control)

1. **Desired X-axis from Yaw**:
The desired x-axis $\mathbf{x}_{b,d}$ is derived from the desired yaw angle ψ_d:

$$\mathbf{x}_{b,d} = \begin{bmatrix} \cos(\psi_d) \\ \sin(\psi_d) \\ 0 \end{bmatrix} \tag{52}$$

2. **Constructing Desired Y-axis and Orthogonalizing X-axis**:
To ensure orthogonality between $\mathbf{x}_{b,d}$ and $\mathbf{z}_{b,d}$, the desired y-axis $\mathbf{y}_{b,d}$ is first constructed:

$$\mathbf{y}_{b,d} = \frac{\mathbf{z}_{b,d} \times \mathbf{x}_{b,d}}{\|\mathbf{z}_{b,d} \times \mathbf{x}_{b,d}\|} \tag{53}$$

Then, the x-axis is reprojected to ensure orthogonality:

$$\mathbf{x}_{b,d} = \mathbf{y}_{b,d} \times \mathbf{z}_{b,d} \tag{54}$$

Notice how every cross product was computed using the notable property of the skew-symmetric matrix:

$$a \times b = S(a) \cdot b \tag{55}$$

3. **Desired Rotation Matrix**:
The desired rotation matrix $\mathbf{R}_{b,d}$ is constructed as:

$$\mathbf{R_{b,d}} = \begin{bmatrix} x_{b,d} & y_{b,d} & z_{b,d} \end{bmatrix} \tag{56}$$

4. **Angular Error and Control Torques**:
The angular error $\mathbf{e}_R$ and angular velocity error $\mathbf{e}_\omega$ are defined as:

$$\mathbf{e}_R = \frac{1}{2}(\mathbf{R}_{b,d}^T \mathbf{R}_b - \mathbf{R}_b^T \mathbf{R}_{b,d})^\vee \tag{57}$$

$$\mathbf{e}_\omega = \boldsymbol{\omega}_b^b - \mathbf{R}_b^T \mathbf{R}_{b,d} \boldsymbol{\omega}_{b,d}^{b,d} \tag{58}$$

Where $\vee$ is the vee operator. It allows the extraction of the vector from the skew-symmetric matrix.
The control torques $\boldsymbol{\tau_b}$ are computed to correct these errors:

$$\boldsymbol{\tau}_b = -K_R \mathbf{e}_R - K_\omega \mathbf{e}_\omega + S(\boldsymbol{\omega}_b^b) I_b \boldsymbol{\omega}_b^b - I_b (S(\boldsymbol{\omega}_b^b) \mathbf{R}_b^T \mathbf{R}_{b,d} \boldsymbol{\omega}_{b,d}^{b,d} - \mathbf{R}_b^T \mathbf{R}_{b,d} \dot{\boldsymbol{\omega}}_{b,d}^{b,d}) \tag{59}$$

where K_R and K_ω are the rotational gain matrices, I_b is the inertia matrix, and $\boldsymbol{S}$ represents the skew-symmetric matrix operator. The gain matrices are chosen as:

$$K_R = diag\{100, 100, 100\} \quad K_w = diag\{10, 10, 10\} \tag{60}$$

The goal, as can be seen by the callback InitFcn in the Simulink file, is to track a trajectory from the initial position $\mathbf{p}_i = [0, 0, 1]$ with yaw $\psi_i = 0$ to the final position $\mathbf{p}_f = [1, 1, 4]$ with yaw $\psi_f = 20°$ over 20 seconds using a geometric path defined by a line and a time law defined by a quintic polynomial.

Simulation Results

Figures 4 (a) and (b) show the position and velocity errors, respectively. The position error is in the order of 10^{-6}, while the velocity error is also small, indicating accurate tracking. Figures 4 (c) and (d) display the orientation and angular velocity errors, respectively. The orientation error is in the order of 10^{-5}, and the angular velocity error is similarly small, demonstrating precise attitude control.

Figure 5 (a) shows that the total thrust u_T initially exceeds the weight of the drone to achieve the required acceleration, then stabilizes to $u_T = mg$ at steady-state. Figure 5 (b) illustrates the control torques τ_b, with the z-component changing due to the yaw reference transitioning from 0 to 20 degrees.

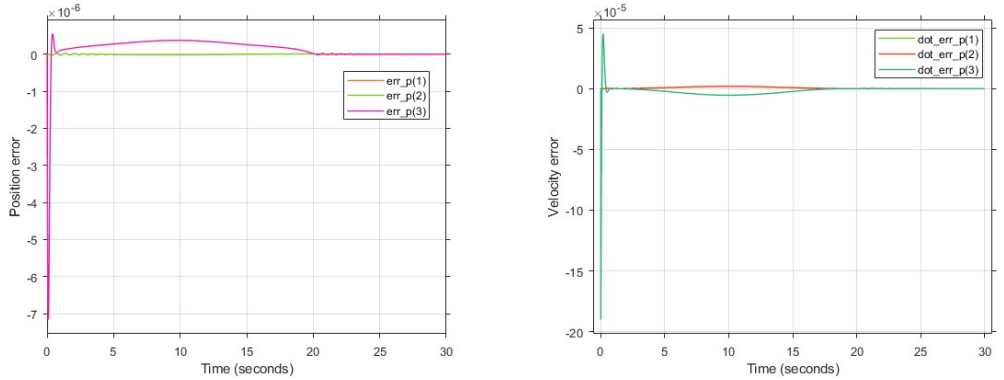

Table 3: Position error (a) and Velocity error (b)

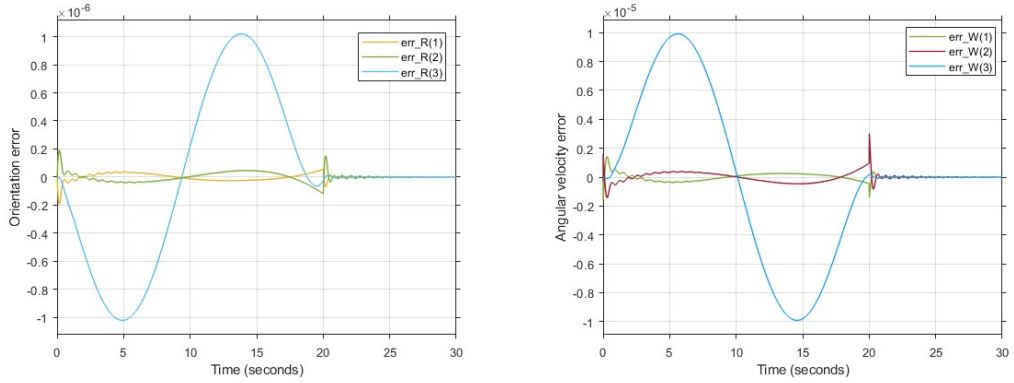

Table 4: Orientation error (c) and Angular Velocity error (d)

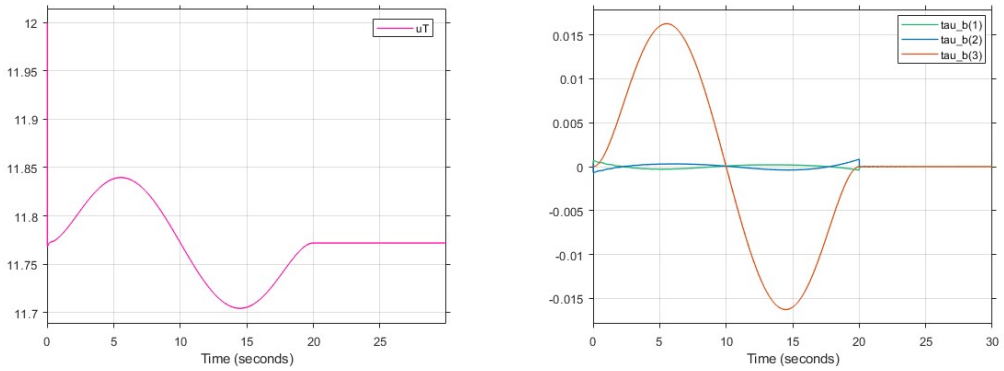

Table 5: u_T (a) and tau_b (b)

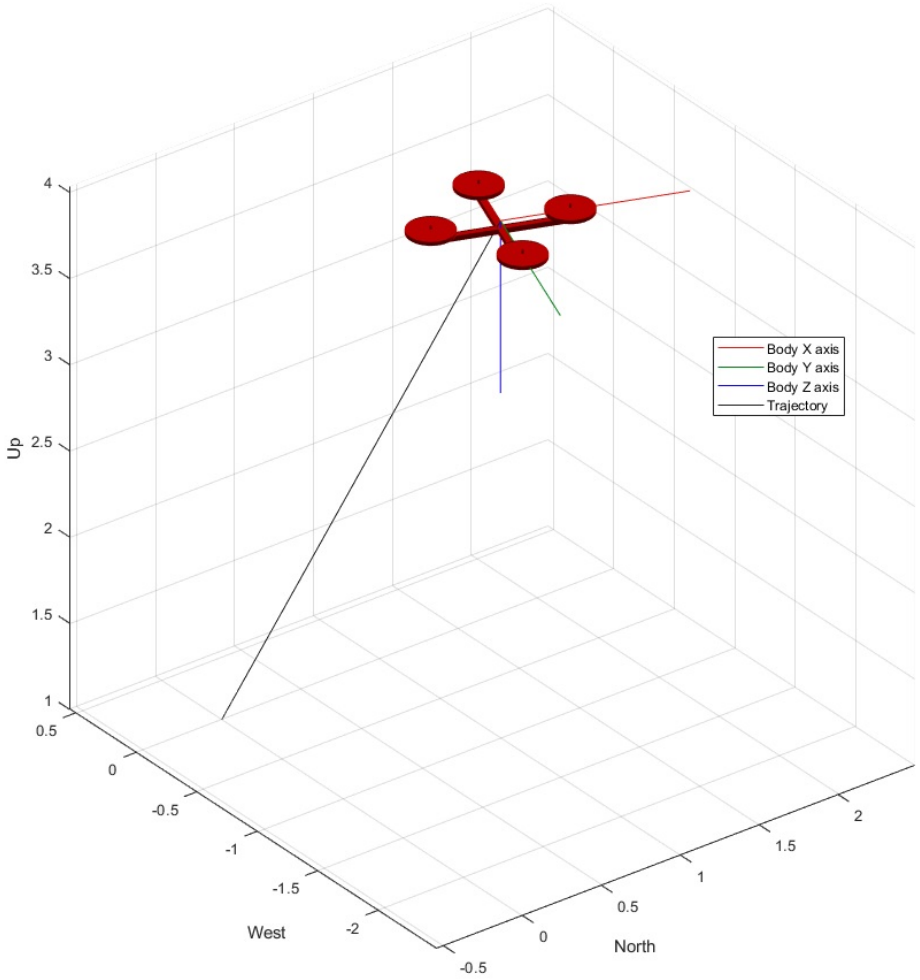

Figure 10: Drone Animation