# Homework 3

Alberto Gugliemo

May 27, 2025

# 1 THE OCTOCOPTER WITH COPLANAR PROPELLERS

Given the octocopter in the figures above, write the number of degrees of freedom of the system, providing a formal description of the configuration space. Besides, is the system underactuated? Briefly motivate your answer. Finally, using the equations on the slides, derive the allocation matrix for the drone, considering the top view of the octocopter given in the right image above. [Hint: put the body frame aligned with one arm of the octocopter and label the propellers (counter-)clockwise].

Given the octocopter with coplanar propellers, the number of degrees of freedom of the system depends on the configuration:

## UAV on the Ground (8 DoF)

The octocopter features eight coplanar propellers, each behaving like a revolute joint rotating around its own axis. When the UAV is fixed to the ground, the only degrees of freedom (DoF) are those associated with the propeller rotations, for a total of 8 DoF.

- **Configuration space:** $\mathbb{T}^8 = (S^1)^8$,

where each $S^1$ corresponds to the rotation angle of one propeller.

## UAV in Flight (14 DoF)

In flight, the octocopter must also account for the rigid body's 6 degrees of freedom in three-dimensional space:

- 3 translational DoF along the $x$, $y$, and $z$ axes,
- 3 rotational DoF .

Thus, the total number of DoF becomes $8 + 6 = 14$.

- **Configuration space:** $\mathbb{T}^8 \times \mathbb{R}^3 \times S^2 \times S^1$

where:

- $\mathbb{T}^8$ Includes propeller rotations,
- $\mathbb{R}^3$ models the translation of the drone's center of mass,
- $S^2 \times S^1$ represents the drone's orientation.

An octocopter with coplanar propellers is *underactuated*, meaning it cannot produce independent instantaneous accelerations in all six degrees of freedom of the center of mass through control inputs alone. In this context, we analyze the degrees of freedom only in the *task space*, thus neglecting the individual positions of the propellers.
To relate the control inputs (individual propeller thrusts) to the resulting forces and torques acting on the UAV, we define the allocation matrix $G$. This matrix plays the role of the input matrix in the dynamic model and maps the eight propeller thrusts to the six-dimensional control vector:

$$G \in \mathbb{R}^{6\times 8}$$

The control vector includes:

- Total thrust force $\mathbf{f}^b \in \mathbb{R}^3$
- Total torque $\boldsymbol{\tau}^b \in \mathbb{R}^3$

Both are expressed in the body frame and are obtained as a linear combination of the squared propeller speeds $|w_i|w_i$, via:

$$\begin{bmatrix} \mathbf{f}^b \\ \boldsymbol{\tau}^b \end{bmatrix} = G \cdot \begin{bmatrix} |w_1|w_1 \\ |w_2|w_2 \\ \vdots \\ |w_8|w_8 \end{bmatrix}$$

where $w_i$ is the speed of the $i$-th propeller.
Far from obstacles and considering a quasi-static analysis, we have $f_{p_i} = c_f|\omega_i|\omega_i$:

$$\mathbf{f}^b = \sum_{i=1}^{n} c_f|\omega_i|\omega_i \mathbf{z}_{p_i}^b \tag{1}$$

$\mathbf{z}_{p_i}^b = [0\ 0\ -1]^T$ is the axis that is parallel to the spinning propeller's axis.
$c_f > 0$, thrust constant.

The forces acting on the centre of mass can only be applied along the axis perpendicular to the drone and are equal to the thrust contribution made by each propeller.
The force applied on the $x$ and $y$ axis of the body frame is zero; this is due to the vertical position of the propeller axis.

$$f_z^b = \begin{bmatrix} -c_f & -c_f & -c_f & -c_f & -c_f & -c_f & -c_f & -c_f \end{bmatrix} \begin{bmatrix} |\omega_1|\omega_1 \\ |\omega_2|\omega_2 \\ |\omega_3|\omega_3 \\ |\omega_4|\omega_4 \\ |\omega_5|\omega_5 \\ |\omega_6|\omega_6 \\ |\omega_7|\omega_7 \\ |\omega_8|\omega_8 \end{bmatrix} \tag{2}$$

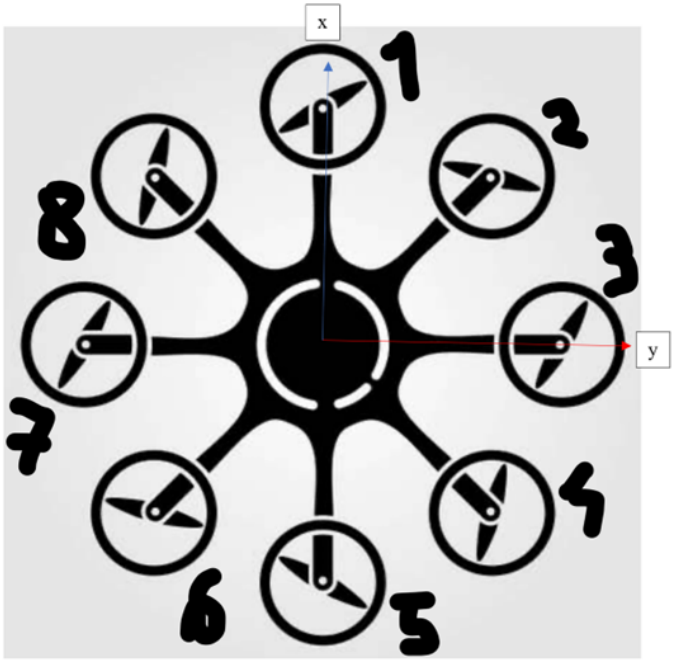


Figure 1: drone view from above with notation

To calculate the total torque $\boldsymbol{\tau}^b$, we refer to the notation in the figure above.
The body frame follows the NED (North-East-Down) convention. For this reason, when observing the drone from above, the $x$ and $y$ axes are "inverted". The propellers are numbered in a clockwise order. The odd-numbered propellers rotate clockwise, while the even-numbered ones rotate counterclockwise.
Having said that, let us now apply the formula:

$$\boldsymbol{\tau}^b = \sum_{i=1}^{n} |\omega_i|\omega_i \left(-k_i c_m \mathbf{z}_{p_i}^b + c_f S(\mathbf{p}_{p_i})\mathbf{z}_{p_i}^b\right) \tag{3}$$

Where:

- $c_m > 0$: drag factor
- $c_f > 0$: thrust constant
- $S(*)$: skew-symmetric operator
- $\mathbf{z}_{p_i}^b = [0\ 0\ -1]^T$: the axis that is parallel to the spinning propeller's axis
- $\mathbf{p}_{p_i} = \left[L\cos\left(\frac{\pi(i-1)}{4}\right),\ L\sin\left(\frac{\pi(i-1)}{4}\right),\ 0\right]^T$: the position of the $i$-th propeller w.r.t. the body frame
- $L$: the distance from the propeller to the origin of the body frame
- $k_i = \begin{cases} 1 & \text{even-numbered propellers (descending chord)} \\ -1 & \text{odd-numbered propellers (ascending chord)} \end{cases}$

The calculations were performed using the MATLAB script `HW3_es1`, and the resulting allocation matrix is:

$$\begin{bmatrix} f_x \\ f_y \\ f_z \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = G \cdot u_w = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -c_f & -c_f & -c_f & -c_f & -c_f & -c_f & -c_f & -c_f \\ 0 & -\frac{c_f\sqrt{2}}{2} & -c_f & -\frac{c_f\sqrt{2}}{2} & 0 & \frac{c_f\sqrt{2}}{2} & c_f & \frac{c_f\sqrt{2}}{2} \\ c_f & \frac{c_f\sqrt{2}}{2} & 0 & -\frac{c_f\sqrt{2}}{2} & -c_f & -\frac{c_f\sqrt{2}}{2} & 0 & \frac{c_f\sqrt{2}}{2} \\ -c_m & c_m & -c_m & c_m & -c_m & c_m & -c_m & c_m \end{bmatrix} \begin{bmatrix} |\omega_1|\omega_1 \\ |\omega_2|\omega_2 \\ |\omega_3|\omega_3 \\ |\omega_4|\omega_4 \\ |\omega_5|\omega_5 \\ ||\omega_6|\omega_6 \\ |\omega_7|\omega_7 \\ |\omega_8|\omega_8 \end{bmatrix}$$

Interpretation of the results:
The torque $\tau_x$ is affected by all propellers except propellers 1 and 5. This is because the moment arm for these propellers is zero, as they lie along the $x$-axis according to the given notation. Propellers 6, 7, and 8 contribute positively to $\tau_x$, while the remaining ones contribute negatively. This behavior is also explained by the direction and magnitude of the moment arms.
Similarly, the torque $\tau_y$ is influenced by all propellers except propellers 3 and 7. The same reasoning applied to $\tau_x$ holds here as well: propellers 3 and 7 lie along the $y$-axis, resulting in a zero moment arm and, therefore, no contribution to $\tau_y$.
Regarding the $\tau_z$ component, we note that, based on the chosen local directions of the propellers, the vector $\mathbf{u}_w$ usually has positive components. If we assume that all the elements $u_{wi}$ have the same magnitude, then the rotational contributions around the $z$-axis cancel out, because the individual effects of the propellers symmetrically balance each other. Therefore, under these conditions, we have: $\tau_z = 0$.
This allocation matrix clearly shows that the drone is underactuated, as the rank of the matrix $G$ is at most 4. In fact, for any given propeller speed, the force components $f_x$ and $f_y$ are always zero, which prevents the system from accelerating in those directions.

# 2 CEILING EFFECT AND GROUND EFFECT

Briefly and qualitatively describe the differences between the ground effect and the ceiling effect.

When a drone flies in close proximity to a surface, whether it's the ground or a ceiling, it experiences notable aerodynamic phenomena known as the ground effect and the ceiling effect. Both are governed by how airflow interacts with nearby boundaries, leading to changes in lift and thrust generation that can significantly impact flight performance, stability, and energy efficiency. The most employed theoretical result for the ground effect is based on the so-called potential aerodynamic assumption, which assumes the fluid is inviscid, incompressible, irrotational, and steady. These simplifications enable analytical insights into how the surrounding air behaves in response to rotor-induced motion near solid surfaces.

Despite their similarities in influencing thrust, ground effect and ceiling effect originate from different physical mechanisms. The ground effect arises when the airflow pushed downward by the rotors encounters the ground and is partially reflected. This interaction thus increases lift and upward thrust without additional power input. The airflow near the ground slows down and spreads laterally, which increases the pressure directly below the rotor disk. In multirotor drones, this effect can also produce stabilizing moments, particularly when some rotors are closer to the ground than others, helping the drone to self-balance in low-altitude hovering; on the other hand, if the surface under the rotors is discontinuous, it can lead to instability.

The ceiling effect occurs when the drone approaches an upper boundary. In this case, airflow is drawn upward into the low-pressure region near the ceiling, effectively reducing resistance and allowing the rotor blades to spin faster in the less dense air. This results in an increase in thrust, sometimes referred to as a "vacuum effect." Although this phenomenon can pose greater risk due to the potential for collision with the overhead surface, it can also offer energy savings and enhanced stability during applications like UAM hovering, provided the drone remains under controlled conditions.

Both effects are highly sensitive to distance from the surface. The ground effect is particularly noticeable when the rotor's height is less than approximately two rotor radii from the ground. As the distance from the surface increases, both effects diminish rapidly and become negligible.

From an energy efficiency standpoint, both ground and ceiling effects result in increased thrust for the same power input, which can be advantageous if managed correctly. However, they also introduce complexity in flight control,especially during maneuvers at low altitudes, since the non-uniform influence on each rotor can lead to asymmetrical forces. This is particularly critical when the drone is tilted during horizontal movement, as the proximity of each rotor to the surface changes dynamically, requiring constant adjustments to maintain balance.

So, ground and ceiling effects are aerodynamic phenomena driven by proximity to solid boundaries, which alter the behavior of the airflow around a drone's rotors. Although their physical origins differ, the ground effect increasing pressure and thrust due to flow reflection, and the ceiling effect enhancing thrust via reduced resistance,both can be harnessed for improved performance if properly controlled.

# 3 EXTERNAL DISTURBANCES

Consider the workspace file attached as ws_homework_3_2025.mat. Within this file, you can find the values of a flight with a quadrotor with the commanded thrust (thrust) and torques (tau), the measured linear velocity (linear_vel), attitude expressed as Euler angles (attitude) and the time derivative of such angles (attitude_vel). The employed quadrotor has a supposed mass of 1.5 kg, and an inertia matrix referred to the body frame equal to diag([1.2416 1.2416 2*1.2416]). Implement yourself the momentum-based estimator of order r to estimate the external disturbances acting on the UAV during the flight. Suppose the sampling time of the estimator is equal to 1 ms. Compare the obtained estimation with the following disturbances applied during the flight:

- 1 N along the x- and y-axis of the world frame;
- -0.4 Nm around the yaw axis.

Compare the estimation results with different values of r. Try to answer the following questions.

- From which value of r the estimation results do not improve too much?
- Compute the real mass of the UAV from the estimated disturbance along the z-axis

The objective of this work is to estimate the external disturbances affecting our drone during the flight phase. The approach is based on the concept of momentum. The idea is to measure the momentum using onboard sensors and compare it with the momentum predicted by our mathematical model. If there is a discrepancy, it means external disturbances are acting on the drone. Momentum is computed as the product of the generalized mass matrix and the generalized velocity vector. By taking its time derivative and substituting the dynamic model (which includes forces and torques, i.e., the wrench), we obtain the following expression:

$$\dot{q} = \begin{bmatrix} mge_3 - u_r R_b e_3 + f_e \\ c^T(\eta_b, \dot{\eta}_b)\dot{\eta}_b + Q^T(\eta_b)\tau^b + \tau_e \end{bmatrix}$$

To simplify the analysis, we aim for a linear relationship between the external wrench and its estimate in the Laplace domain:

$$L\begin{bmatrix} \hat{f}_e \\ \hat{\tau}_e \end{bmatrix} = G(s)L\begin{bmatrix} f_e \\ \tau_e \end{bmatrix}$$

Here, $G(s)$ is a matrix of transfer functions, assumed to be diagonal for simplicity. The system must have zero static gain, to avoid any scaling in the estimation. Since we want to preserve all the signal information, a first-order low-pass may not be the best choice, as it removes high-frequency components that may contain valuable disturbance data. Nevertheless, some filtering is required to reduce measurement noise, so a compromise is usually made.
To compute the estimated wrench, we perform the inverse Laplace transform. However, this becomes more complex as the transfer function order increases.
A key assumption is that both the momentum and the external wrench are zero at the initial time, which holds true if the drone starts from rest (on the ground).
To estimate the external wrench, the following quantities are needed:

- Attitude → from the IMU
- Angular velocity → from the IMU
- Linear velocity → from GPS or onboard sensors
- Control inputs → from the employed controller
- Model parameters → from system modeling

The integral of the estimator is approximated numerically by Forward Euler (Ts=1 ms). Higher-order transfer functions can be used, resulting in nested integrals. We also assume that the drone can perfectly follow the control inputs $u_T$ and $\tau_b$ without transients, which simplifies the wrench estimation.
Below is the formula for a first-order transfer function:

$$\begin{bmatrix}\hat{f}_e\\ \hat{\tau}_e\end{bmatrix} = k_0\left(q - \int_0^t \begin{bmatrix}\hat{f}_e\\ \hat{\tau}_e\end{bmatrix} + \begin{bmatrix} mge_3 - u_T R_b e_3 \\ C^T(\eta_b,\dot{\eta}_b)\dot{\eta}_b + Q^T(\eta_b)\tau^b\end{bmatrix} dt\right)$$

Using the attitude (Euler angles), its derivative (attitude velocity), and the inertia matrix $J$, we can compute:

- The rotation matrix $R_b$
- The $Q$ matrix (mapping Euler rates to angular velocities)
- The $C$ matrix (representing Coriolis and centrifugal effects)
- The $M$ matrix (generalized mass matrix)

In the next steps, we will analyze the system behavior by varying $r$, $k_0$, and $C_0$. These parameters are chosen equal to ensure unit static gain. We choose the $r$ poles of the function all coinciding:

$$G(s) = \left(\frac{k_0}{s + C_0}\right)^r$$

To implement an estimator of order $r$, it is necessary to compute an iterative function to compute :

$$\begin{bmatrix}\hat{f}_e(k+1)\\ \hat{\tau}_e(k+1)\end{bmatrix} = \gamma_r(k+1)$$

Here we use the assumption of the null initial condition.
Since we are solving an inverse Laplace transform with r nested integrals, we need to create a recursive algorithm to compute $\gamma_r$.

$$\gamma_1(k+1) = \gamma_1(k) + K_1(q(k+1) - q(k)) - T_s\left[\left(mge_3 - u^T R_b e_3\right) - \left(C^T\dot{\eta}_b + Q^T\eta_b\tau_b\right) - \begin{bmatrix}\hat{f}_e\\ \hat{\tau}_e\end{bmatrix}\right]$$

$$\gamma_i(k+1) = \gamma_i(k) + T_s K_i\left(-\begin{bmatrix}\hat{f}_e(k)\\ \hat{\tau}_e(k)\end{bmatrix} + \gamma_{i-1}\right), \quad \text{for } i = 2,\ldots,r$$

To study the performance of this estimator, different simulations were carried out, first by varying the order of the estimator $r$ , then by varying $C_0$.
Increasing the order of the estimator results in a slower response, due to the delay introduced by the integrators, and often leads to a more oscillatory estimation. Increasing the $C_0$ parameter helps to speed up the transient response (since the coincident poles have smaller time constant), but it also tends to make the response more oscillatory. In general, increasing the estimator order beyond a certain point does not significantly improve the estimation quality. On the contrary, it introduces delays that worsen the dynamic performance. When pushing the order to very high values, such as $r = 150$ (not shown in the plots), the system tends to lose stability, and the estimation diverges.
It is not possible to answer unambiguously the question for which value of $r$ the estimate does not improve so much, since performance depends not only on $r$ but also on the choice of poles.
Based on these considerations:

- A low-order estimator, for instance of second order, is a good compromise to ensure fast response, even if it may show overshoot.
- A higher-order estimator, like one of order 20, helps to reduce overshoots, but results in slower transients and increased response delay. It would also better reduce high-frequency measurement noise in a real system.

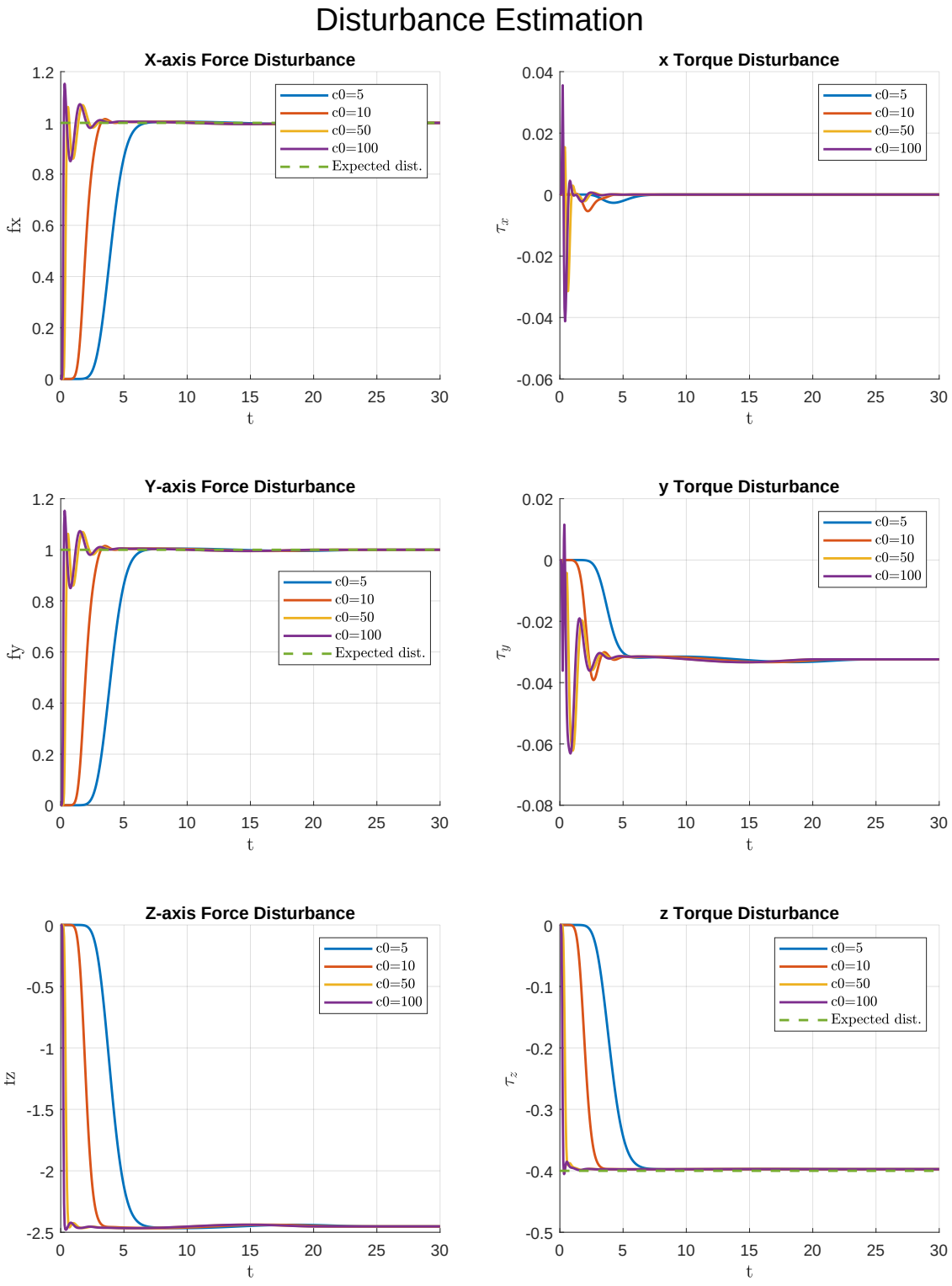


Figure 2: Estimation of disturbances varying r with c0=10

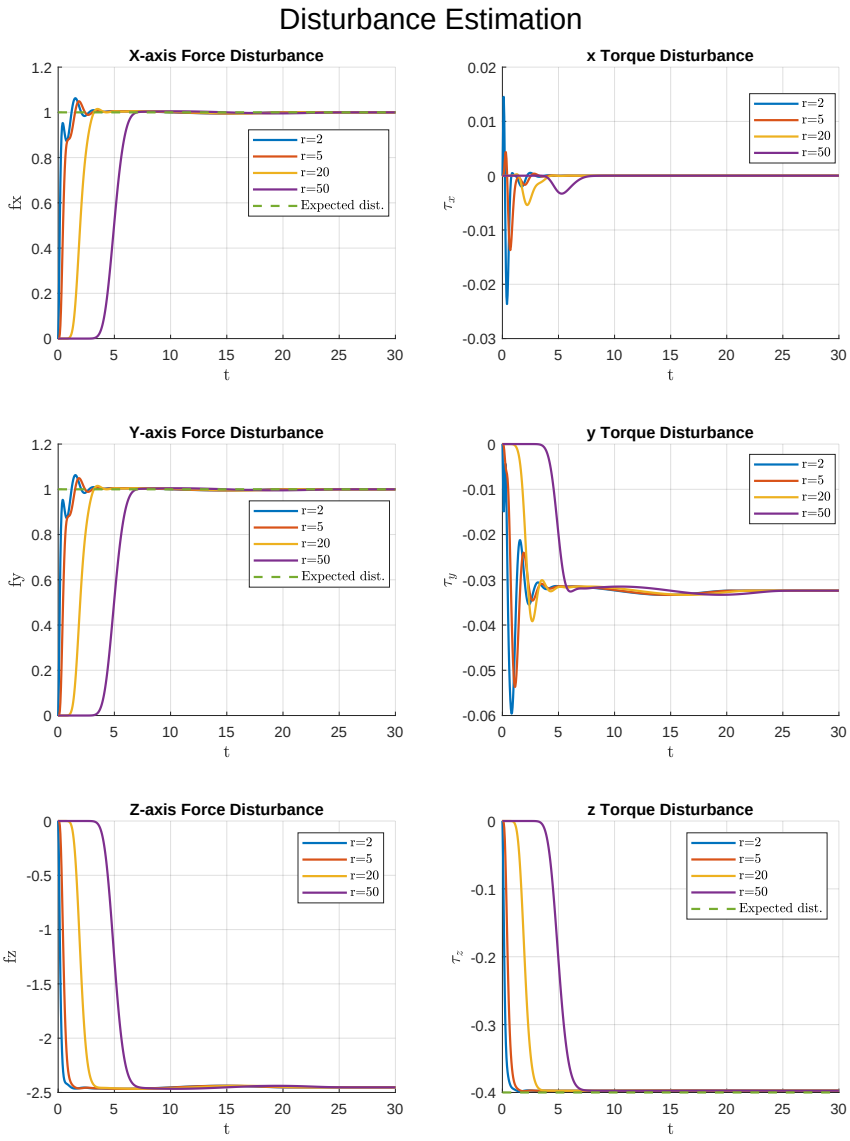


Figure 3: Estimation of disturbances varying c0 with r=20 $r$

Since no external disturbance is applied along the $z$ axis, the uncertainty in the vertical estimation must be due to an uncertainty in the drone mass when the drone is hovering. The estimated disturbance along the $z$-axis can be written as:

$$F_z = \widetilde{m} \cdot g$$

where $\widetilde{m}$ is the mass estimation error and $g$ is the gravitational acceleration.
Given the estimated disturbance (approximately $-2.5\,\mathrm{N}$, the minus sign means an upward force for NED convention, so the mass is overestimated), we can compute the actual mass with respect to the assigned (nominal) mass of $1.5\,\mathrm{kg}$:

$$m_{\text{real}} = \widetilde{m} + m$$

The result shows a difference of $\widetilde{m} = -0.25\,\mathrm{kg}$, meaning that the real mass of the drone is:

$$m_{\text{real}} = 1.5\,\mathrm{kg} - 0.25\,\mathrm{kg} = 1.25\,\mathrm{kg}$$

# 4 GEOMETRIC CONTROLLER

Consider the Simulink file attached as geometric_control_template.slx. Within this file, you can find a template to implement yourself the geometric control. You must fill in the inner and outer loops. Simulate the scheme and report the plots you believe are most interesting. You may add further scopes to the scheme to extract data you believe most interesting to show

A geometric controller is a non-linear controller used for tracking signals. This type of controller is particularly suitable for vehicles such as quadcopters, because it avoids the problems of local representations of orientation, such as Euler angles. It can be divided into two parts: The inner-loop controller is responsible for managing the rotational dynamics of the system, and it relies on information provided by the outer-loop, which handles the translational tracking. The goal of the overall controller is to reduce both the position tracking error and the orientation error to zero, ensuring accurate trajectory tracking.

**Outer-Loop:** The outer-loop controller governs the translational dynamics of the quadcopter. Its primary objective is to ensure that the drone accurately tracks the desired position trajectory by computing the appropriate total thrust and the direction of the thrust vector. Using a PD control law, it evaluates the position and velocity errors to determine the corrective force needed. By combining the desired position, velocity, and acceleration with the current pose and velocity of the drone, the outer loop calculates both the required total thrust and the desired body-frame $z$-axis $\mathbf{z}_{b,d}$. This desired direction is then passed to the inner-loop controller, which handles the drone's orientation to align the actual thrust direction accordingly.

**Inner-Loop:** Once the desired body-frame $z$-axis vector $\mathbf{z}_{b,d}$ is generated by the outer-loop (based on thrust direction), the desired $x$-axis $\mathbf{x}_{b,d}$ provided by the planner may not be orthogonal to it. To construct a proper rotation matrix $R_{b,d}$, we must generate an orthonormal frame. We first calculate $\mathbf{z}_{b,d}$, then $\mathbf{y}_{b,d}$, and finally $\mathbf{x}_{b,d}$ to have a desired orthogonal triplet $R_{b,d}$ which will be used for the inner loop. With this desired orientation $R_{b,d}$ and the current orientation $R_b$ from onboard sensors, we can compute the orientation error in SO(3), as well as the angular velocity error, and finally compute the required control torque $\boldsymbol{\tau}_b$.

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

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

Using a trial and error method, the gains were chosen as follows:

$$K_p = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 100 \end{bmatrix}, \quad K_v = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 100 \end{bmatrix}, \quad K_R = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 50 \end{bmatrix}, \quad K_\omega = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

The controller gains were chosen as diagonal matrices, with a higher weighting applied to the components related to the $z$-axis, since the position and orientation errors were more pronounced along this axis compared to others. Specifically, the gains were increased for position and velocity errors along the $z$-axis to improve control accuracy in that direction. The control system proves effective even with very small errors, on the order of $10^{-6}$ and $10^{-7}$, both in position and orientation.
Regarding the position error on the $z$-axis, there is an initial peak caused by an approximated initial condition of $u_T$ set to 12 N while the drone is stationary. The correct value for $u_T$ should be 11.7 N. This discrepancy arises because the integrator's initial condition was set to 12 N. Nevertheless, the controller quickly compensates the acceleration along the $z$-axis.
The orientation error exhibits a slight oscillatory behavior, but the amplitude of these oscillations is so minimal that it does not significantly affect the control of a real UAV.
Analyzing the control torques $\boldsymbol{\tau}_b$, it can be seen that $\tau_x$ and $\tau_y$ remain very low most of the time, as they are mainly used to orientate and enable motion in the right direction in space. Conversely, $\tau_z$ reaches higher values when a yaw direction change is required (20-degree rotation), then returns to zero once the trajectory becomes stationary and the UAV must stop.
During the hovering phase, $u_T$ remains constant at 11.7 N. From this value, given that the error is practically negligible, the actual mass of the drone can be calculated, resulting in approximately 1.2 kg.

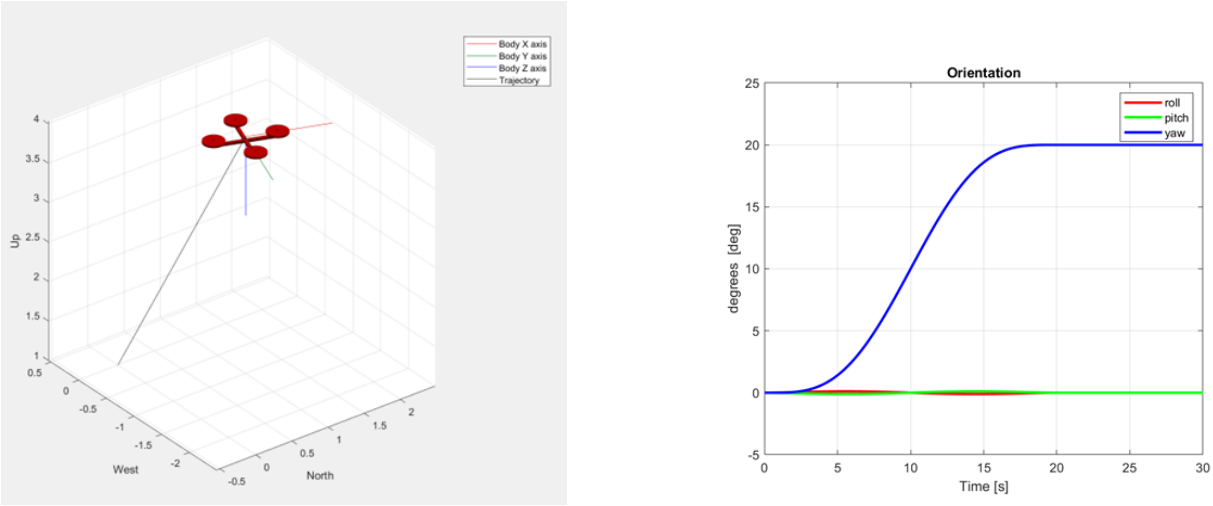


Figure 4: Trajectory and orientation (body frame) of the drone

One of the key features of this geometric controller is that it is fully defined in the configuration space $\mathbb{R}^3 \times \mathrm{SO}(3)$, using rotation matrices. This formulation avoids any singularities typically associated with local attitude representations such as Euler angles.
However, for stability guarantees, two conditions must be verified:

1. The desired linear acceleration must not exceed the gravitational acceleration;
2. The initial attitude error must be less than $90°$.

In the presented case, both conditions are met:

- The desired accelerations remain below 0.06 $\mathrm{m/s^2}$, well within the gravitational acceleration limit.
- The initial attitude error is zero.

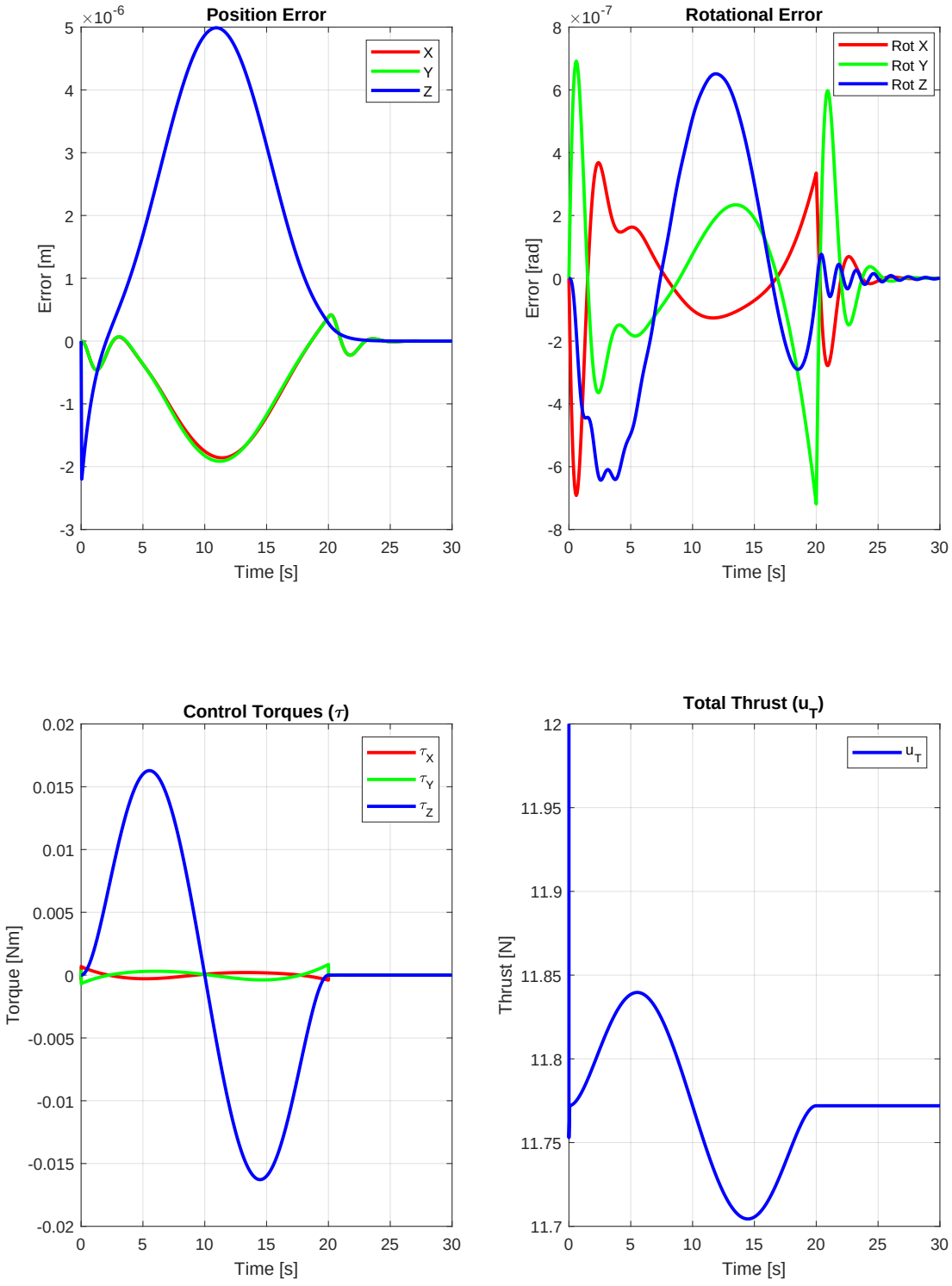


Figure 5: pose error trend and control torque and thrust

# 5 TILTING CONTROL

Consider the Simulink file attached as tilting_control_voliro_template.slx. Within this file, you can find a template to implement yourself the control for the tilting quadrotor using the Voliro approach. You must fill in the controller. Simulate the scheme and report the plots you believe are most interesting. You may add further scopes to the scheme to extract data you believe most interesting to show. Describe also by words what it the trajectory followed by the tilting quadrotor and what it is performing in the last seconds of the trajectory

Voliro's tilt-rotor approach allows the control of a quadrirotor whose propeller rotation axes can be changed, thus separating the magnitude of thrust from its direction. In this way, the drone with 4 propellers can be fully actuated (always depends on the configuration) because the control inputs become 8 and the allocation matrix can be full rank. The goal is to compute the commands to be sent to the propellers (rotation speed and tilt angles) so that the drone follows a desired trajectory in both position and orientation, while respecting the physical dynamics of the system.

1. Error Computation

The first step is to compute the tracking errors with respect to the desired trajectory:

- Position error: difference between the current and desired positions.
- Velocity error: difference between the current and desired linear velocities.
- Orientation error: obtained by comparing the current and desired rotation matrices.
- Angular velocity error: difference between the current and desired angular velocities, expressed in the body frame.

These errors are then used to generate corrective commands.

2. Translational and Rotational Control

The control action is divided into two parts:

- Translational control: generates a desired linear acceleration based on position and velocity errors. It determines where the drone should go and how fast.
- Rotational control: generates a desired angular acceleration (i.e., torque) based on orientation and angular velocity errors. It ensures that the drone rotates correctly to align with the desired orientation.

Both are implemented using PD control laws with appropriate gain matrices. The idea of control is always that of solving the homogeneous solution of the second-degree differential equation. Choosing positive gains for the Cartesio rule, the error tends to zero:

$$\ddot{e} + K_d\dot{e} + K_p e = 0 \quad \text{where} \quad e = \text{state} - \text{desired state}$$

3. System Dynamics

Next, the physical dynamics of the drone are considered. This includes:

- The mass and inertia of the drone.
- The effect of gravity.
- Rotational dynamics, such as Coriolis and centrifugal forces due to angular motion.

These effects are compensated to isolate the control effect.

4. Mapping Forces/Moments to Propeller Commands

The desired linear and angular accelerations, computed by the control system, are combined into a single vector. This vector represents the target accelerations in the drone's body frame. However, the drone does not directly control its accelerations, instead, it commands the propellers. To bridge this gap, an allocation matrix is used. This matrix maps the forces and torques generated by the tiltable propellers to the resulting net force and torque on the drone. Because each propeller can tilt, it can generate thrust in different directions. The allocation matrix accounts for this directional flexibility as well as the physical configuration of the drone.

5. Dynamics Inversion

To determine which commands to send to the propellers to achieve the desired accelerations, the dynamic model is inverted using a pseudoinverse. This allows us to:

- Translate the desired accelerations into actuator commands.
- Handle the system's redundancy (more actuation DoFs than control objectives).
- In our case I didn't use the null space to optimize any parameter

6. Propeller Command Computation

After computing the required forces for each propeller, this information is transformed into actual motor commands: Rotation speed and Tilt angle (computed to orient the thrust in the correct direction).

$$u_{\text{transformed}} = \text{pinv}\left(M^{-1}\begin{bmatrix} R_b & 0 \\ 0 & I_3 \end{bmatrix} G_{\text{static}}\right)\left(-M^{-1}\begin{bmatrix} mge_3 \\ -\text{skew}(\omega_{bb})I_b\omega_{bb} \end{bmatrix} + \begin{bmatrix} -K_p(p-p_{\text{des}}) - K_v(\dot{p}-\dot{p}_{\text{des}}) + a_{\text{des}} \\ -K_R e_R - K_\omega e_\omega + \dot{\omega}_{\text{des}} \end{bmatrix}\right)$$

$$\begin{bmatrix} f_x \\ f_y \\ f_z \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 \\ 0 & 0 & -l & -\frac{c_m}{c_f} & 0 & 0 & l & \frac{c_m}{c_f} \\ l & \frac{c_m}{c_f} & 0 & 0 & -l & -\frac{c_m}{c_f} & 0 & 0 \\ -\frac{c_m}{c_f} & l & \frac{c_m}{c_f} & -l & -\frac{c_m}{c_f} & l & \frac{c_m}{c_f} & -l \end{bmatrix} \begin{bmatrix} u_{v,1} \\ u_{l,1} \\ u_{v,2} \\ u_{l,2} \\ u_{v,3} \\ u_{l,3} \\ u_{v,4} \\ u_{l,4} \end{bmatrix} = G_{static}u(\alpha, u_\omega)$$

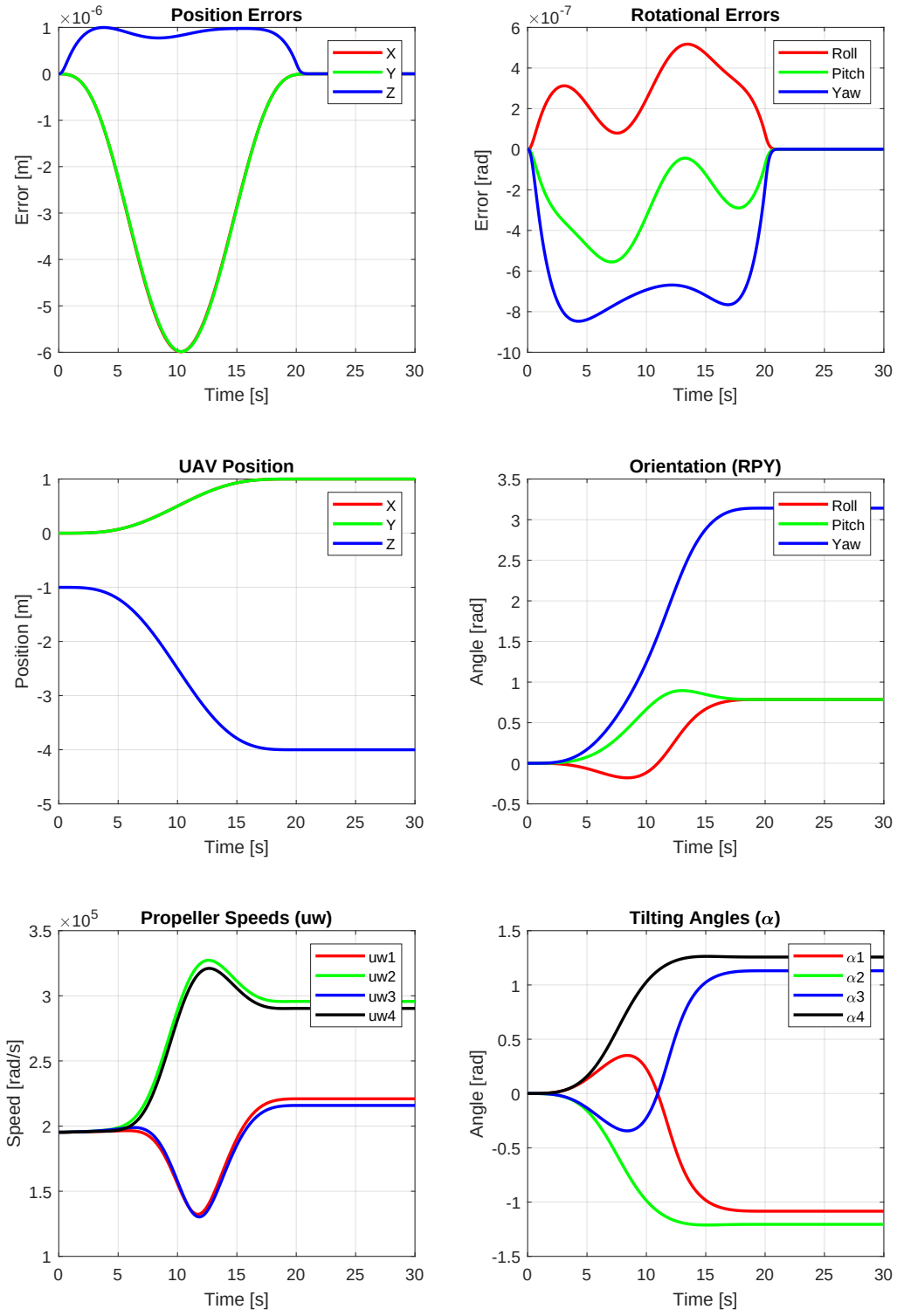


Figure 6: Plots of errors,Pose and control inputs

$$K_p = \begin{bmatrix} 300 & 0 & 0 \\ 0 & 3000 & 0 \\ 0 & 0 & 300 \end{bmatrix}, \quad K_v = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 10 \end{bmatrix}, \quad K_R = \begin{bmatrix} 30 & 0 & 0 \\ 0 & 30 & 0 \\ 0 & 0 & 30 \end{bmatrix}, \quad K_\omega = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix}$$

The position and attitude control gains were selected through a trial-and-error approach. In this specific scenario, the UAV exhibits more pronounced tracking errors along the X and Y axes compared to the Z axis (hence the weights are chosen differently). Nevertheless, both linear and angular errors remain on the order of $10^{-6}$ and $10^{-7}$, ensuring high precision and excellent trajectory tracking performance.

The rotational speed of the individual propellers, as observed from the simulation graphs, is on the order of $10^5$ rad/s. These values are not consistent with those of real drones, whose propellers typically operate within a range of approximately 200 to 1000 rad/s.

The robot follows a trajectory starting from the initial position $(0, 0, -1)$, which, under a North-East-Down (NED) convention, indicates that the drone begins at an altitude of 1 meter above the ground. Initially, all rotor tilting angles are zero, representing a configuration where the propellers are vertically aligned and parallel to each other, as in a standard quadrotor setup.

By applying the Voliro tilt-rotor control scheme, the drone successfully moves to the target position $(1, 1, -4)$, accurately tracing the desired path and rotating 180 degrees around the x-axis of the body frame. One notable observation is that while the initial angular velocities of the four rotors are all roughly 2 rad/s and nearly identical, in the final configuration they are both higher and different from one another. Despite this asymmetry, the robot is able to maintain stability while hovering.

This change is a consequence of the new tilting configuration: in the final state, not all of the generated thrust is directed vertically to counteract gravity, part of it is diverted laterally. This leads to reduced energy efficiency, but allows for enhanced control authority and spatial maneuverability.

Finally, it is worth noting that the final hovering position is accompanied by a rotation of the body frame relative to the base frame. This rotation involves not only the Z axis but also the X and Y axes, indicating that the robot is stationary in space but tilted and not aligned parallel to the ground.