# Field and Service Robotics: Final Project
# Eyes in the Sky: Advanced Control of Quadrotors for Real-Time Monitoring

Instructor: Fabio Ruggiero

Students: Laura Milone P38000254
Sabrina Sarnataro P38000255

link gitHub: https://github.com/LauraMil01/FinalProjectFSR.git

# Introduction

Modern applications increasingly rely on autonomous aerial vehicles to provide real-time monitoring, inspection, and surveillance in complex environments. Whether it's overseeing a football match, inspecting building exteriors, or observing natural reserves, drones offer the ability to capture dynamic scenes from above with precision and flexibility. A common feature across these scenarios is the need for the UAV to have an elevated perspective that ensures a comprehensive, top-down view of the area of interest. To support this task, the UAV is equipped with onboard sensing equipment. In particular, we assume the presence of a camera rigidly mounted beneath the drone, oriented downward, allowing continuous visual coverage of the environment below.

For this reason, the mission begins with a vertical ascent, enabling the UAV to establish a full visual frame of the scene. At this point it proceeds to move along the perimeter of the area to capture complementary perspectives that enhance the overall monitoring and understanding of the environment. Depending on the shape of the environment, the desired trajectory in the horizontal plane can describe either a square, typical for man-made structures, or a circle, more suited to open or naturally symmetric contexts.

To control the UAV during these maneuvers, four distinct controllers were designed and implemented: two for a co-planar quadrotor, and two for a tilting quadrotor. For the co-planar case, we employed a hierarchical controller, as covered during the course, and an Adaptive Sliding Mode Controller (ASMC), a novel approach inspired by the methodology proposed in the paper [1]. For the tilting quadrotor, we implemented the holocopter-based tilting control strategy, presented in the course, and a Nonlinear Model Predictive Controller (NMPC), introduced as a new contribution, drawing from the paper [2].

A comparative analysis was carried out for each UAV type, evaluating both control strategies over square and circular trajectories, under nominal conditions and in the presence of external disturbances. These tests were essential to assess the performance, robustness, and reliability of each controller in realistic and variable scenarios.

The results of all simulations are provided in the `SimulationVideoMiloneSarnataro.rar` folder, which contains videos illustrating the UAV's behavior. These videos reflect the use of the NED (North-East-Down) convention adopted as the body-frame reference throughout the simulations.

[1] *Adaptive Sliding Mode Control Design for Quadrotor Unmanned Aerial Vehicle*: S. Islam, M. Faraz, R. K. Ashour, G. Cai, J. Dias, L. Seneviratne

[2] *Nonlinear Model Predictive Control of Tiltrotor Quadrotors with Feasible Control Allocation*: Z. Shyan, J. Cristobal, M. Izadi, A. Yazdanshenas, M. Naderi, R. Faieghi

# Planning

We implemented different types of tracking control on two types of quadrotors: a co-planar one (with fixed propellers lying on the same plane) and a tilting one (with variable-inclination propellers, which allows independent control of all position and orientation components). For both UAVs, two desired trajectories are tested: the first consists of a vertical segment along the $z$-axis, followed by a square path in the $x-y$ plane; the second involves a vertical ascent and then a circular motion in the $x-y$ plane, because this allows us to observe the difference between sharper, more abrupt motions, such as those along a square trajectory, and smoother, more continuous motions, like those along a circular path.

**Co-planar UAV**

The co-planar quadrotor is an underactuated system, meaning that it does not have enough independent control inputs to generate direct accelerations along all six degrees of freedom $(x, y, z, \phi, \theta, \psi)$. In particular, accelerations along the $x$ and $y$ axes cannot be produced without tilting the UAV, which creates a coupling between horizontal motion and the roll and pitch angles. The planner, therefore, is designed to provide the desired values for the independent variables required by the controllers, that in our case are $x, y, z$, and the yaw angle $\psi$. The roll and pitch angles are not planned directly; instead, the controller computes them as the angles necessary to achieve the desired accelerations in $x$ and $y$.

- **Square trajectory:** After a vertical ascent phase, the drone executes a closed path along the edges of a square in the x-y plane at constant altitude. The motion is segmented into four linear translations between consecutive waypoints. The transition between each pair of points is generated using $7^{th}$-degree polynomial interpolation, ensuring continuity in position, velocity, acceleration and jerk. This level of smoothness is crucial to produce physically feasible reference inputs for an underactuated system like the quadrotor. The yaw angle evolves smoothly throughout the active portion of the trajectory. Rather than assigning it piecewise, a single $7^{th}$-degree polynomial is used to interpolate the yaw from the initial to the final value. This ensures a continuous and differentiable yaw profile that avoids abrupt rotational commands.
- **Circular trajectory:** Similar in structure to the square case, the circular trajectory starts with a vertical ascent and continues with a planar motion, where the drone follows a full circular loop at constant altitude. In the circular trajectory, the motion in the x-y plane is defined parametrically by an angular variable $\delta(t)$, which determines the position on the circle via trigonometric relationships. The angle $\delta(t)$ itself evolves according to a $7^{th}$-degree polynomial timing law, ensuring that the resulting position $(x(t), y(t))$ has continuous derivatives up to jerk, just like in the square case. Despite the different parametrization, the trajectory is ultimately expressed in terms of smooth Cartesian coordinates. This method allows direct computation of the position, velocity, and acceleration components in the $x-y$ plane from trigonometric relationships. As with the square trajectory, the yaw angle is modulated via a separate smooth polynomial function over the active duration.

**Tilting UAV**

The tilting quadrotor, by contrast, is fully actuated: thanks to the variable inclination of the propellers, it is possible to independently control all six state variables (position and orientation). This enables the planner to directly assign desired values also of roll and pitch independently from the other variables. In the case of the tilting quadrotor, the inclination of the UAV is managed such that the camera mounted underneath the drone constantly keeps the area of interest in its field of view, thus improving observation effectiveness.

- **Square trajectory:** This trajectory shares the same spatial profile as the square-like path used in the co-planar UAV, including the initial ascent and planar motion along the $x$ and $y$ axes. However, the key difference lies in the orientation strategy. Thanks to the tilting mechanism, roll and pitch

are explicitly assigned in each segment to maintain the onboard camera oriented toward the area of interest, that is the interior of the perimeter. The orientation transitions are realized through smooth $7^{th}$-degree polynomials, ensuring dynamic feasibility and continuity up to jerk. The yaw ramps up smoothly during ascent and remains constant afterward.

- **Circular trajectory:** The circular trajectory maintains the same positional profile as in the co-planar case but introduces a significant enhancement in orientation behavior. While the UAV follows the circular path, roll and pitch are continuously adjusted so that the vehicle tilts outward, which results in the onboard camera pointing inward toward the center of the circle at all times. This is achieved by computing the polar angle $\alpha$ of the UAV's position with respect to the circle center and assigning the roll and pitch accordingly. A smooth "ease-in" function ensures a gradual transition from the ascent to the radial tilting regime, avoiding abrupt attitude changes.

Both trajectories leverage the full actuation capabilities of the tilting quadrotor to decouple orientation from translational motion, enabling camera-aware trajectory planning that would not be possible with a conventional underactuated design.

| **Trajectory** | **Segment** | **Position change** | **Yaw $\psi$** | **Roll $\phi$** | **Pitch $\theta$** |
|---|---|---|---|---|---|
| **Co-planar Square** | 1 (0–10 s) | $z$: $-1 \to -4$ | $0^\circ \to ...$ | $0^\circ$ | $0^\circ$ |
| | 2 (10–15 s) | $x$: $0 \to 2$ | $... \to ...$ | - | - |
| | 3 (15–20 s) | $y$: $0 \to 2$ | $... \to ...$ | - | - |
| | 4 (20–25 s) | $x$: $2 \to 0$ | $... \to ...$ | - | - |
| | 5 (25–30 s) | $y$: $2 \to 0$ | $... \to 20^\circ$ | - | - |
| **Co-planar Circle** | 1 (0–10 s) | $z$: $-1 \to -4$ | $0^\circ \to ...$ | $0^\circ$ | $0^\circ$ |
| | 2 (10–30 s) | $360^\circ$ circle, $R = 1$, center $(1, 0)$ | $... \to 20^\circ$ | - | - |
| **Tilting Square** | 1 (0–10 s) | $z$: $-1 \to -4$ | $0^\circ \to 5^\circ$ | $0^\circ$ | $0^\circ$ |
| | 2 (10–20 s) | $x$: $0 \to 2$ | $5^\circ$ | $0^\circ \to -15^\circ$ | $0^\circ$ |
| | 3 (20–30 s) | $y$: $0 \to 2$ | $5^\circ$ | $-15^\circ \to 0^\circ$ | $0^\circ \to 15^\circ$ |
| | 4 (30–40 s) | $x$: $2 \to 0$ | $5^\circ$ | $0^\circ \to -15^\circ$ | $15^\circ \to 0^\circ$ |
| | 5 (40–50 s) | $y$: $2 \to 0$ | $5^\circ$ | $-15^\circ \to 0^\circ$ | $0^\circ \to -15^\circ$ |
| **Tilting Circle** | 1 (0–10 s) | $z$: $-1 \to -4$ | $0^\circ \to 5^\circ$ | $0^\circ$ | $0^\circ$ |
| | 2 (10–50 s) | $360^\circ$ circle, $R \approx 1.41$, center $(1, 1)$ | $5^\circ$ | $-15^\circ \cdot \sin(\alpha)$ | $15^\circ \cdot \cos(\alpha)$ |

Table 1: Detailed segment breakdown for each trajectory

A final dead time of 10 seconds is included in all trajectories to allow observation of the steady-state response.

**Note:** The change of the circle's center and radius was a deliberate design choice aimed at obtaining a trajectory symmetric in the plane. This configuration allows the drone to tilt outward while ensuring that the camera continuously points toward the center of the circle, providing uniform observation conditions along the entire path.

# Hierarchical Controller

The first control strategy that we implement for co-planar quadrotor trajectory tracking and stabilization is the Hierarchical Controller, based on the decoupling of translational and rotational dynamics. This two-loop strategy allows for simplified control design by taking advantage of time-scale separation between fast inner-loop, that manages angular dynamics, and slower outer-loop, that manages translational dynamics.

The UAV is modeled as a standard 6-DOF nonlinear system based on roll-pitch-yaw (RPY) Euler angles, neglecting external disturbances. The dynamic model is given by:

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

where:

- $m \in \mathbb{R}$: mass of the UAV;
- $g \in \mathbb{R}$: gravitational acceleration;
- $e_3 \in \mathbb{R}^3$: unit vector along the inertial vertical axis, $e_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$;
- $u_T \in \mathbb{R}$: total thrust generated by the propellers;
- $R_b \in SO(3)$: rotation matrix from the body frame to the world frame, function of the attitude $\eta_b$;
- $Q(\eta_b) \in \mathbb{R}^{3\times3}$: transformation matrix between RPY derivative and angular velocity;
- $M(\eta_b) = Q(\eta_b)^T I_b Q(\eta_b) \in \mathbb{R}^{3\times3}$: inertia-like matrix in Euler-angle coordinates, where $I_b \in \mathbb{R}^{3\times3}$ is the UAV moment of inertia matrix;
- $C(\eta_b, \dot{\eta}_b) \in \mathbb{R}^{3\times3}$: Coriolis and centrifugal matrix in Euler-angle coordinates;
- $\tau^b \in \mathbb{R}^3$: torque vector expressed in the body frame.

The control objective is to asymptotically stabilize to zero the tracking errors in position, orientation, and their first derivatives. The errors are defined as:

$$e_p = p_b - p_{b,d} = \begin{bmatrix} x - x_d \\ y - y_d \\ z - z_d \end{bmatrix} \qquad \dot{e}_p = \dot{p}_b - \dot{p}_{b,d} = \begin{bmatrix} \dot{x} - \dot{x}_d \\ \dot{y} - \dot{y}_d \\ \dot{z} - \dot{z}_d \end{bmatrix} \qquad e_\eta = \eta_b - \eta_{b,d} = \begin{bmatrix} \phi - \phi_d \\ \theta - \theta_d \\ \psi - \psi_d \end{bmatrix} \qquad \dot{e}_\eta = \dot{\eta}_b - \dot{\eta}_{b,d} = \begin{bmatrix} \dot{\phi} - \dot{\phi}_d \\ \dot{\theta} - \dot{\theta}_d \\ \dot{\psi} - \dot{\psi}_d \end{bmatrix}$$

To achieve this, we must appropriately define the control inputs $u_T$ and $\tau^b$.

- Since the rotational subsystem is fully actuated (with the control input $\tau^b$ we can directly influence all the three rotational components), it can be feedback linearized. We introduce a virtual control input $\tilde{\tau} \in \mathbb{R}^3$, and define the torque input as:

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

  Substituting this into the rotational dynamics yields $\ddot{\eta}_b = \tilde{\tau}$, so we define the virtual control law as:

  $$\tilde{\tau} = -K_e \begin{bmatrix} e_\eta \\ \dot{e}_\eta \end{bmatrix} + \ddot{\eta}_{b,d}$$

  in order to ensure exponential convergence of both $e_\eta$ and $\dot{e}_\eta$ toward zero.

- The translational subsystem is underactuated, since thrust is only available along the body $z$-axis. Therefore, we cannot directly apply feedback linearization on the total subsystem.
  We define a virtual desired acceleration $\mu_d \in \mathbb{R}^3$ as:

$$\mu_d = -K_p \begin{bmatrix} e_p \\ \dot{e}_p \end{bmatrix} + \ddot{p}_{b,d}$$

  which stabilizes the translational tracking errors to zero with exponential behavior.
  The control input $u_T$ is then computed as:

$$u_T = m\sqrt{\mu_x^2 + \mu_y^2 + (\mu_z - g)^2}$$

The gains matrix $K_p$, $K_e \in \mathbb{R}^{3\times 6}$ are designed through an appropriate tuning process. We followed the standard procedure typically used when dealing with a control architecture that includes both an inner and an outer loop. We started by isolating the inner loop, setting the angular references $\phi_d$ and $\theta_d$ to zero, rather than obtaining them from the outer loop. Then, we tuned the inner loop to make the angular errors converge to zero as quickly and smoothly as possible, identifying the optimal gains for the angular controller. Once a stable and responsive inner loop was achieved, we reconnected the outer loop, the position control. At this stage, we tuned the position controller gains to ensure accurate trajectory tracking and to minimize position errors. This method provided an effective and well-balanced control of the overall system.

The Hierarchical Controller was implemented in Simulink within the file `hierarchical_control.slx`, which contains the following key blocks:

- The subsystem **Planner:** it receives as input the variables declared and processed in the `init_function`, where the reference trajectory for the UAV is generated. It provides the desired position, linear velocity, linear acceleration, and the yaw angle $\psi_d$ with its first and second derivatives, all of which are required to ensure accurate trajectory tracking during the simulation.

- The MATLAB Function **outer_loop:** this block handles the translational dynamics. It computes and returns the total thrust $u_T$. Since the planner provides only the desired yaw angle $\psi_d$, but the inner loop requires the full desired attitude, this function also computes and provides the desired roll $\phi_d$ and pitch $\theta_d$ according to the following expressions:

$$\phi_d = \sin^{-1}\left(\frac{m}{u_T}(\mu_y \cos\psi_d - \mu_x \sin\psi_d)\right) \qquad \theta_d = \tan^{-1}\left(\frac{\mu_x \cos\psi_d + \mu_y \sin\psi_d}{\mu_z - g}\right)$$

  Note that singularities in $\theta_d$ occur when $\mu_z = g$, so the function ensures that $|\mu_z| < g$.
  The function also returns the position and linear velocity errors, which are monitored via scopes to evaluate whether they converge to zero as expected from the controller design.

- The subsystem **Inner_loop:** this block handles the rotational dynamics. It computes and returns the torque vector $\tau^b$, the orientation error and its first derivative . These errors are also visualized using scopes to verify the effectiveness of the controller, in particular to check whether they converge to zero as expected.
  As the computation of $\tau_b$ involves the inverse of $Q(\eta)$, which is singular at $\theta = \pm\frac{\pi}{2}$, a control mechanism must be implemented to prevent the system from reaching these singular configurations.
  The function receives as inputs the first and second derivatives of the desired orientation after a numerical differentiation of the desired orientation and its first derivative, both provided by the block itself. However, direct numerical differentiation of a signal is highly sensitive to high-frequency noise, which is often present in real-world sensor data, especially in IMUs. To mitigate this issue, we employed

a first-order discrete low-pass filter after the derivative block. The chosen filter has the following discrete transfer function:

$$H(z) = \frac{b_0}{1 - az^{-1}} = \frac{0.188}{1 - 0.812z^{-1}}$$

which corresponds to a pole at $z = 0.812$, and is designed for a sampling time of $T_s = 0.001\,\text{s}$. The cutoff frequency can be approximated by:

$$f_c \approx \frac{f_s(1-a)}{2\pi} = \frac{1000 \cdot (1 - 0.812)}{2\pi} \approx 30\,\text{Hz}$$

The coefficient $b_0$ is chosen as $b_0 = 1 - a$ to ensure that the filter has a unit gain at low frequencies. This frequency was selected as a trade-off between noise attenuation and dynamic responsiveness, a 30 Hz cutoff effectively preserves the relevant information while suppressing high-frequency noise, which typically arises above 50 Hz in real IMU measurements.

- The subsystem **Quadrotor_dynamics:** it receives the control inputs and, based on the previously described UAV model, computes and returns the current values of position, linear velocity, orientation, and orientation derivative.

Since a PD controller with a feedforward term is applied in both the inner and outer loops, it is standard practice to select the velocity gains as the square root of the corresponding position or orientation gains. So, the controller gains are set as $K_p = \begin{bmatrix} 10I_3 & \sqrt{10}I_3 \end{bmatrix}$, $K_e = \begin{bmatrix} 150I_3 & \sqrt{150}I_3 \end{bmatrix}$ in order to guarantee faster convergence of the inner loop.
In the following, we report the simulation results obtained by applying the hierarchical control strategy to the co-planar quadrotor, for both planned trajectories.

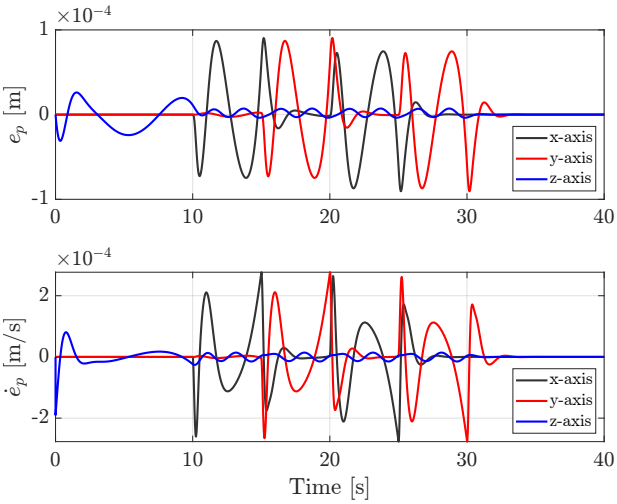


Figure 1: Square trajectory: position and linear velocity errors

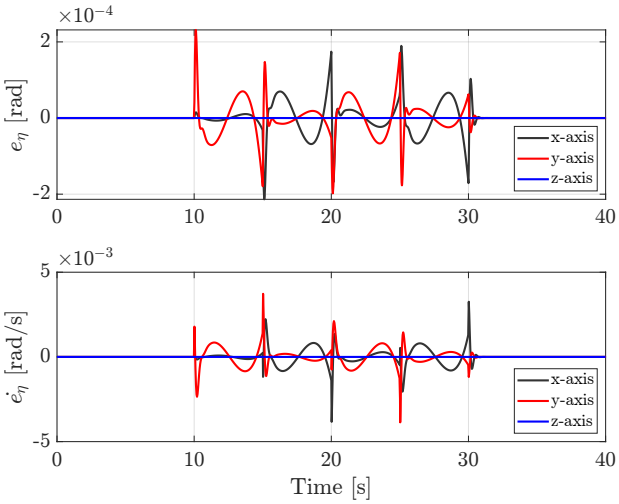


Figure 2: Square trajectory: orientation and orientation derivative errors

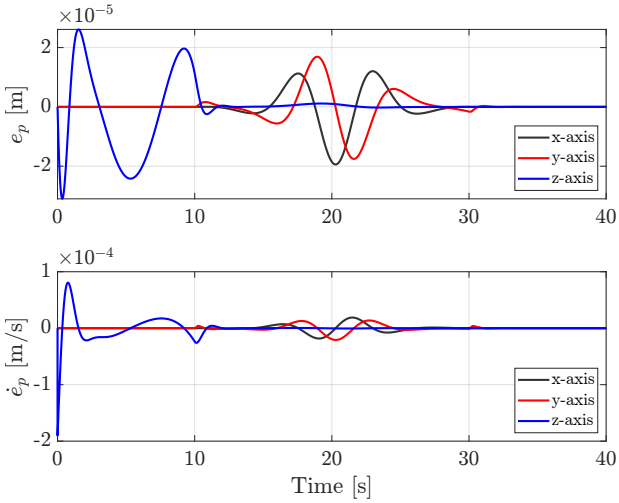


Figure 3: Circular trajectory: position and linear velocity errors

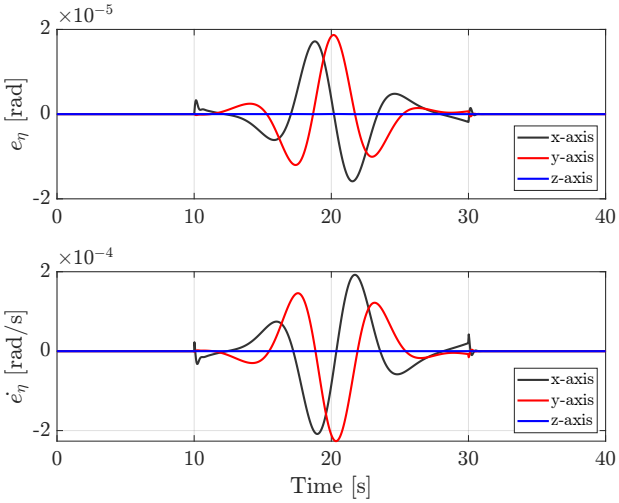


Figure 4: Circular trajectory: orientation and orientation derivative errors

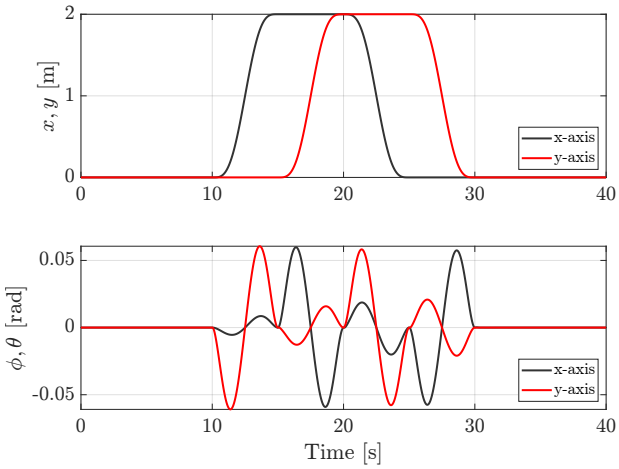


Figure 5: Square trajectory: $x$ and $y$ components of position and orientation

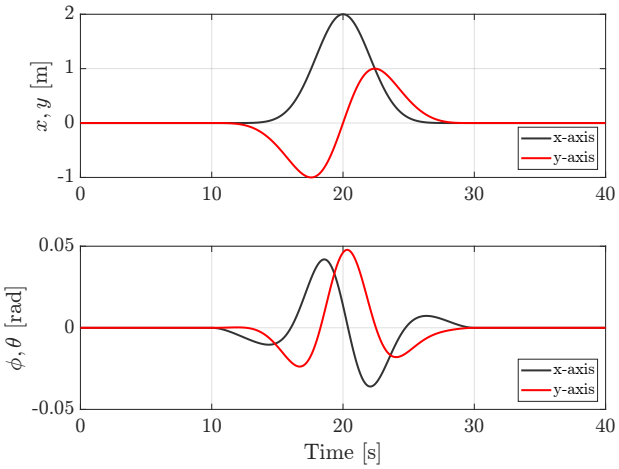


Figure 6: Circular trajectory: $x$ and $y$ components of position and orientation

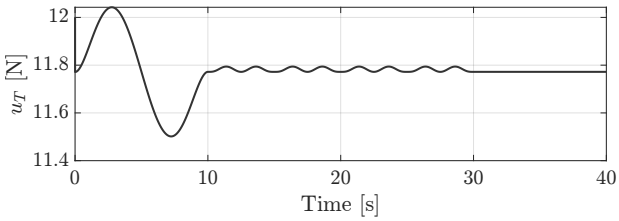


Figure 7: Square trajectory: commanded total thrust

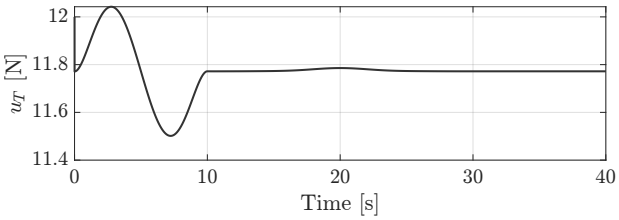


Figure 8: Circular trajectory: commanded total thrust

- **Square trajectory**: As shown in Figures 1 and 2, the tracking errors are very small. Specifically, the position, linear velocity, and orientation errors are on the order of $10^{-4}$, while the $\dot{\eta}$ error is on the order of $10^{-3}$. Moreover, all errors settle to zero between 30 and 40 seconds, i.e., during the dead time,

confirming that the controller meets the performance requirements.

- **Circular trajectory**: As shown in Figures 3 and 4, the tracking errors are even smaller: position and orientation errors are on the order of $10^{-5}$, and angular velocity and $\dot{\eta}$ errors are on the order of $10^{-4}$. As with the square trajectory, all errors converge to zero during the dead time.
- **Square trajectory analysis**: We observe that peaks appear at the beginning of each side of the square trajectory, specifically at 10, 15, 20, and 25 seconds, corresponding to changes in the direction of motion. In particular:
  - During the intervals 10-15 s and 20-25 s, where the UAV performs a translation only along the $x$-axis (keeping $y$ and $z$ constant), we observe oscillations in the position and linear velocity errors along $x$, and almost zero error along $y$, as expected. On the other hand, orientation and its derivative errors exhibit larger values along the $y$-axis, which is consistent with the required dominant pitch rotation (around the $y$-axis) for generating thrust in the $x$ direction.
  - Conversely, during the intervals 15-20 s and 25-30 s, where the UAV moves only along the $y$-axis, the inverse occurs: larger orientation errors appear along the $x$-axis due to the dominant roll motion required for lateral movement, while position and velocity errors are dominant along $y$.
  - The fact that a translation in the horizontal plane requires tilting of the body is evident both from the video `hierarchical_square.mp4` in `SimulationVideoMiloneSarnataro.rar` and from Figures 7. These clearly show that, in order to translate in the $x$-$y$ plane, non-zero roll and pitch angles are required. Among the two, pitch ($\theta$) is the dominant component for translations along $x$, whereas roll ($\phi$) is the dominant component for translations along $y$.
  - Regarding the $z$-axis, position and linear velocity errors show an initial peak in the first 10 seconds, corresponding to the vertical ascent. These errors exhibit an oscillatory behavior with decreasing amplitude and slightly increasing frequency as the UAV executes the square trajectory, until they finally settle during the dead time. However, orientation and angular velocity errors along $z$ remain nearly zero, consistent with the smooth yaw profile planned during the maneuver.
- **Circular trajectory analysis**: The error profiles are smoother compared to the square trajectory, reflecting the continuous nature of the circular path. Notably:
  - Slightly larger errors are observed along the $x$ and $y$ axes during the 10-30 s interval, when the UAV performs the circular motion.
  - As in the square case, a peak in the $z$-axis position and velocity errors is observed during the initial vertical ascent, which then decays rapidly as the UAV transitions into circular flight. Orientation and angular velocity errors along $z$ remain negligible due to the smoothly planned yaw trajectory.
  - As we can see both from the video `hierarchical_circ.mp4` in `SimulationVideoMiloneSarnataro.rar` and from Figures 8, as soon as the UAV begins following the circular path in the $x$-$y$ plane, both roll and pitch become non-zero and exhibit approximately the same amplitude. In this condition, the motion requires simultaneous accelerations along $x$ and $y$, causing both angles to vary continuously over the trajectory.
- **Total thrust analysis**: Regarding the total thrust, for both trajectories it exhibits a certain dynamic evolution during the vertical ascent, after which it settles to the value $mg = 11.772$ N. This occurs because the UAV is hovering and must only compensate for gravity. Moreover, in the square trajectory the thrust shows oscillations before settling, as can also be observed in the $z$ component of the position error, phenomenon that does not affect the circular trajectory.
- **Conclusion**: The hierarchical controller effectively tracks both trajectories, with errors consistently converging to zero during the dead time. Overall, the circular trajectory exhibits smoother behavior and lower tracking errors, typically one order of magnitude smaller than those of the square trajectory.

# Adaptive Sliding Mode Control (ASMC)

Another control strategy that we implement for co-planar quadrotor stabilization and trajectory tracking, also in the presence of disturbances, is the Adaptive Sliding-Mode Controller, that combines the robustness of sliding-mode control with adaptive techniques. Unlike many existing nonlinear schemes, the ASMC does not need prior knowledge of bounds on disturbances or model errors, all unknown effects are handled by real-time adaptive estimates embedded directly in the sliding surface dynamics. In fact, in conventional sliding-mode control, the switching gain must be set higher than the worst-case disturbance, so one has to estimate that bound in advance. In our adaptive scheme, the gain adjusts itself online, eliminating any need for a prior estimate.

The UAV is modeled using a 6-DOF nonlinear system, with translational and rotational dynamics treated separately:

$$\begin{cases} \ddot{P} = \beta U_f - C\dot{P} - \gamma + F_a \\ \ddot{\eta} = MU + \zeta + F_b \end{cases}$$

where:

- $U_f \in \mathbb{R}^3$: total force generated by the propellers;
- $\beta = (mR_t^T)^{-1} \in \mathbb{R}^{3\times3}$: $R_t$ is the rotation matrix between fixed inertial frame and body frame;
- $C = m^{-1}L \in \mathbb{R}^{3\times3}$: $L$ is the aerodynamic drag coefficient matrix for translational motion, with positive diagonal elements;
- $\gamma = \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T$: gravitational acceleration vector;
- $U \in \mathbb{R}^3$: total moment generated by the propellers;
- $M = (IB)^{-1} \in \mathbb{R}^{3\times3}$: $I$ is the inertia matrix of the UAV body and $B$ is the matrix that relates angular velocity to the time derivative of the orientation;
- $\zeta = -B^{-1}\dot{B}(\eta,\dot{\eta})\dot{\eta} - \xi\dot{\eta} - B\dot{\eta} \times BI\dot{\eta} - B\dot{\eta} \times \sum_{i=1}^{4} I_r\omega_i$ where:
  - $-B^{-1}\dot{B}(\eta,\dot{\eta})\dot{\eta}$: contribute given by the relationship between $\omega$ and $\dot{\eta}$
  - $\xi\dot{\eta} = I^{-1}\mathbf{M}\dot{\eta}$: dissipative term, where $\mathbf{M}$ is the aerodynamic drag coefficient matrix for rotational motion, with positive diagonal elements;
  - $B\dot{\eta} \times BI\dot{\eta}$: gyroscopic effect due to the UAV's rigid body rotation;
  - $B\dot{\eta} \times \sum_{i=1}^{4} I_r\omega_i$: gyroscopic effect due to rotor spinning, where $I_r$ is the inertia of the rotor blade and $\omega_i$ is the angular velocity of the $i$-th rotor.
- $F_a, F_b \in \mathbb{R}^3$: unknown external disturbances acting on the translational and rotational dynamics.

Let $P_d$ and $\eta_d$ the desired position and attitude trajectories. The tracking errors are defined as:

$$e_1 = P_d - P \qquad e_2 = \dot{P}_d - \dot{P} \qquad e_3 = \eta_d - \eta \qquad e_4 = \dot{\eta}_d - \dot{\eta}$$

The control objective is to design the inputs $U_f$ and $U$ such that the position and attitude tracking errors $e_1(t)$ and $e_3(t)$ converge to zero as $t \to \infty$, despite the presence of unknown external disturbances.
Sliding Mode Control achieves robustness by enforcing system trajectories to reach and remain on a pre-defined manifold, known as the *sliding surface*, on which the closed-loop dynamics are stable and ideally insensitive to matched disturbances.
**Note**: in rotational dynamics we have to compensate also $\zeta$, so the adaptive law has to estimate the upper bound of $\zeta + F_b$.

In this work, two sliding functions are introduced: one associated with the translational dynamics and the other with the rotational dynamics.
Let us express the system dynamics in terms of the tracking error states:

$$\dot{e}_1 = e_2 \qquad \dot{e}_2 = -\beta U_f + \gamma + C\dot{P} - F_a + \ddot{P}_d \qquad \dot{e}_3 = e_4 \qquad \dot{e}_4 = -MU - \zeta - F_b + \ddot{\eta}_d$$

To stabilize the system, we define the following linear sliding function for the translational dynamics:

$$S_L = e_2 + \lambda_L e_1$$

where $\lambda_L \in \mathbb{R}^{3\times 3}$ is a positive definite diagonal matrix ($\lambda_L = \text{diag}(\lambda_1, \lambda_2, \lambda_3)$).
When the system is on the sliding surface we have:

$$S_L = 0 \quad \Rightarrow \quad e_2 = -\lambda_L e_1$$

Substituting into the equation $\dot{e}_1 = e_2$, we obtain: $\dot{e}_1 = -\lambda_L e_1$. This yields exponentially stable dynamics for $e_1$, since all eigenvalues of $\lambda_L$ are positive. Therefore, if the control law ensures that $S_L \to 0$, the position tracking error $e_1$ will converge to zero, where the rate of convergence depends on $\lambda_L$.
Analogously, for the rotational subsystem, we define the sliding function:

$$S_A = e_4 + \lambda_A e_3$$

with $\lambda_A \in \mathbb{R}^{3\times 3}$ positive definite and diagonal. On the sliding manifold, we have:

$$S_A = 0 \quad \Rightarrow \quad e_4 = -\lambda_A e_3 \quad \Rightarrow \quad \dot{e}_3 = -\lambda_A e_3$$

which again describes an exponentially stable system for the attitude error $e_3$. In this case, the rate of convergence depends on $\lambda_A$.

To ensure finite time convergence to the switching manifold, the control law must enforce $S_L \to 0$, $S_A \to 0$. Let's consider a Lyapunov candidate function that takes into account both the sliding surfaces and the adaptive parameter estimation errors:

$$V(t) = \frac{1}{2} S_L^T S_L + \frac{1}{2} \tilde{\theta}_1^T \Gamma_1^{-1} \tilde{\theta}_1 + \frac{1}{2} S_A^T S_A + \frac{1}{2} \tilde{\theta}_2^T \Gamma_2^{-1} \tilde{\theta}_2$$

where $\theta_1$ and $\theta_2$ are the constant, unknown disturbance bounds on translational and rotational dynamics, respectively. $\tilde{\theta}_1 = \theta_1 - \hat{\theta}_1$, $\tilde{\theta}_2 = \theta_2 - \hat{\theta}_2$ denote the estimation errors of the unknown disturbance bounds.
Taking the time derivative:

$$\dot{V} = S_L^T \dot{S}_L + \tilde{\theta}_1^T \Gamma_1^{-1} \dot{\tilde{\theta}}_1 + S_A^T \dot{S}_A + \tilde{\theta}_2^T \Gamma_2^{-1} \dot{\tilde{\theta}}_2$$

Substituting the translational error dynamics into $\dot{S}_L$:

$$\dot{S}_L = \dot{e_2} + \lambda_L e_2 = -\beta U_f + \gamma + C\dot{P} - F_a + \ddot{P}_d + \lambda_L e_2$$

We define the control input as:

$$U_f = \beta^{-1} \left( \ddot{P}_d + C\dot{P} + \lambda_L e_2 + \gamma + k_L S_L - \hat{\theta}_1 \,\text{sign}(S_L) \right)$$

so that:

$$\dot{S}_L = -k_L S_L + \tilde{\theta}_1 \,\text{sign}(S_L)$$

Similarly, for the rotational dynamics:

$$\dot{S}_A = \dot{e_4} + \lambda_A e_4 = -MU - \zeta - F_b + \ddot{\eta}_d + \lambda_A e_4$$

with control input:

$$U = M^{-1}\left(\ddot{\eta}_d + \lambda_A e_4 + k_A S_A - \hat{\theta}_2 \,\mathrm{sign}(S_A)\right)$$

so we obtain:

$$\dot{S}_A = -k_A S_A + \tilde{\theta}_2 \,\mathrm{sign}(S_A)$$

**Note:** Since the controllers must compensate for the upper bound of the disturbance magnitude, the estimation errors $\tilde{\theta}_1$ and $\tilde{\theta}_2$ explicitly appear in the dynamics of $S_L$ and $S_A$.

Moreover: $\dot{\tilde{\theta}}_1 = \dot{\theta}_1 - \dot{\hat{\theta}}_1 = -\Gamma_1 S_L^T \mathrm{sign}(S_L)$, $\dot{\tilde{\theta}}_2 = \dot{\theta}_2 - \dot{\hat{\theta}}_2 = -\Gamma_2 S_A^T \,\mathrm{sign}(S_A)$.
Now we can compute the Lyapunov derivative that, simplifying the cross terms, is:

$$\dot{V} = -S_L^T k_L S_L - S_A^T k_A S_A = -\|S_L\|^2 k_L - \|S_A\|^2 k_A$$

Since $k_L$ and $k_A$ are positive definite, we conclude: $\lim_{t\to\infty} \|S_L(t)\| = 0$, $\lim_{t\to\infty} \|S_A(t)\| = 0$ as we wanted. This implies that the system is on the sliding surface and the position and the attitude errors converge to zero.

The ASMC was implemented in Simulink within the file `adaptive_sliding_mode_control.slx`, which contains the following key blocks:

- The subsystem **Planner:** as in the hierarchical control architecture, it takes as input the variables declared and computed in the `init_function`, where the reference trajectory for the UAV is generated. The subsystem returns the desired states required for simulation.
- The MATLAB function **compute_control_input:** this block implements the ASMC control law. It computes and returns the control inputs $U_f$ and $U$, as described before. It also gives the position, linear velocity, orientation, and orientation derivative errors, which are monitored via scopes to evaluate whether the objective is achieved. Since the control law leads to first-order dynamics on the position and orientation errors, the convergence to zero of the linear velocity and orientation derivative errors is not directly enforced, but rather occurs as an indirect consequence of the system's stability and the convergence of the primary errors. So we show all the errors for completeness.
  Note that the only desired orientation explicitly planned is the yaw angle $\psi_d$. The desired roll $\phi_d$ and pitch $\theta_d$ angles are computed from $\psi_d$ and the normalized control input $U_f$ as follows:

  $$\phi_d = \sin^{-1}\left(\frac{U_{f_x}}{U_{f_z}}\sin(\psi_d) - \frac{U_{f_y}}{U_{f_z}}\cos(\psi_d)\right) \qquad \theta_d = \sin^{-1}\left(\frac{\frac{U_{f_x}}{U_{f_z}}\cos(\psi_d) + \frac{U_{f_y}}{U_{f_z}}\sin(\psi_d)}{\cos(\psi_d)}\right)$$

  The arguments of the inverse sine functions are saturated to remain within the admissible interval $[-1, 1]$, ensuring numerical stability. The terms $U_{f_x}$ and $U_{f_y}$ are normalized by $U_{f_z}$ because, in practice, the UAV produces thrust only along the body $z$-axis.
- The subsystem **Quadrotor_dynamics:** it receives the control inputs and, based on the previously described UAV model, namely the translational and rotational dynamics equations, computes and returns the current values of position, linear velocity, orientation, and orientation derivative.

In the following, we present the simulation results for the co-planar quadrotor operating under nominal conditions ($F_a = F_b = [0, 0, 0]^T$), using the ASMC control strategy on both planned trajectories. The chosen gain values are: $\lambda_L = \mathrm{diag}\{20,\ 20,\ 50\}$, $k_L = \mathrm{diag}\{20,\ 20,\ 50\}$, $\Gamma_1 = 0.001$, $\Gamma_2 = 0.001$,
$\lambda_A = \mathrm{diag}\{5,\ 5,\ 10\}$, $k_A = \mathrm{diag}\{5,\ 5,\ 10\}$

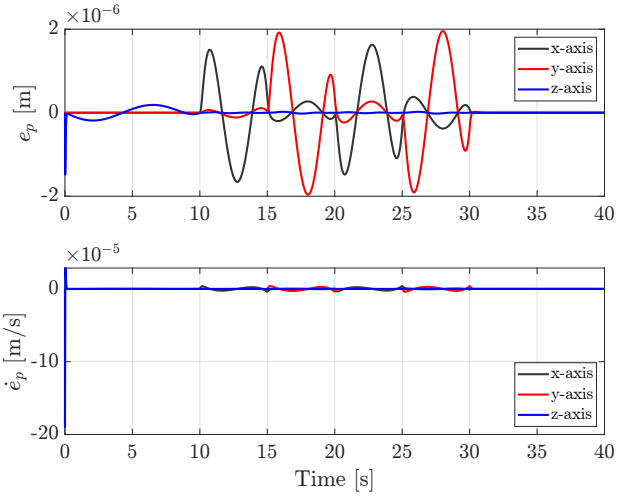


Figure 9: Square trajectory: position and linear velocity errors

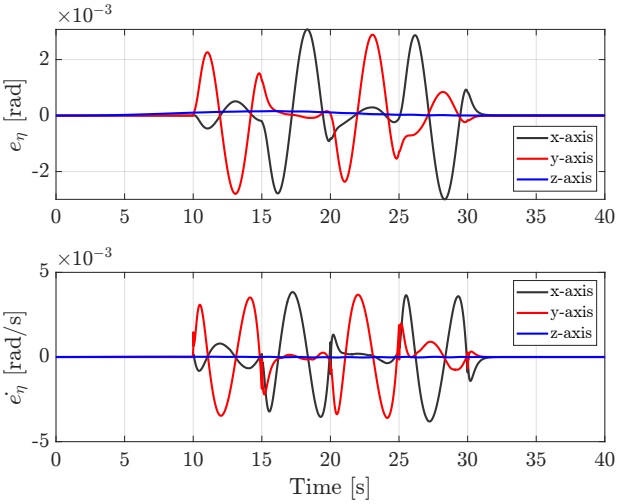


Figure 10: Square trajectory: orientation and orientation derivative errors

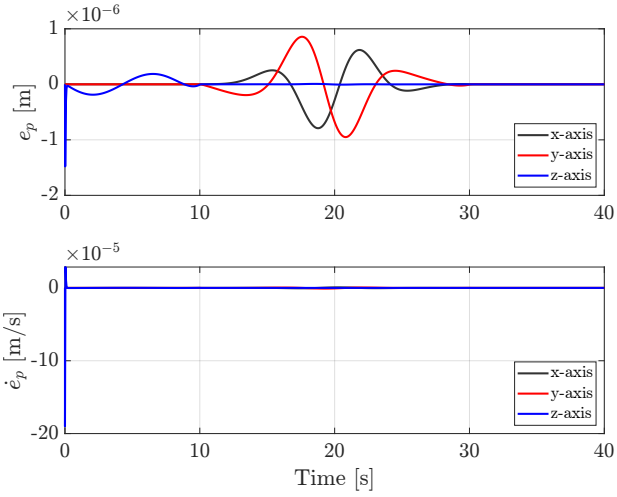


Figure 11: Circular trajectory: position and linear velocity errors

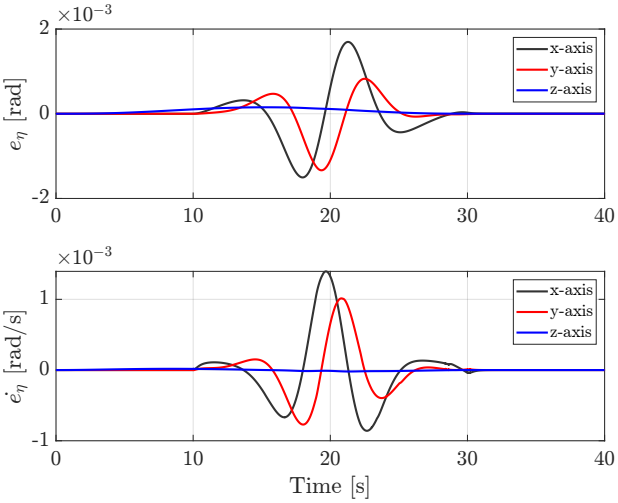


Figure 12: Circular trajectory: orientation and orientation derivative errors

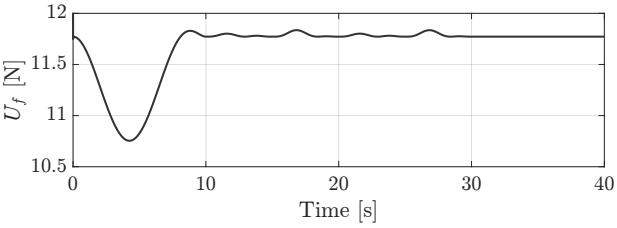


Figure 13: Square trajectory: $z$-component of $U_f$

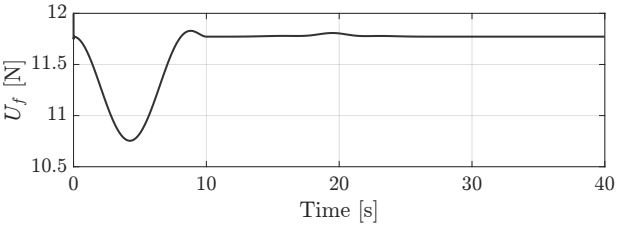


Figure 14: Circular trajectory: $z$-component of $U_f$

- **Error analysis for both trajectories**: For both the square trajectory (Figures 9 and 10) and the circular trajectory (Figures 11 and 12), we observe that the tracking errors are extremely small across all measured quantities. Specifically, the position error is on the order of $10^{-6}$, while the linear velocity

error is generally on the order of $10^{-6}$ with the exception of a noticeable peak along the $z$-axis (on the order of $10^{-4}$) at the beginning of the motion. Orientation and angular velocity errors are on the order of $10^{-3}$. Importantly, all errors progressively decrease and settle to zero between 30 and 40 seconds, during the so-called dead time. This confirms that the controller effectively achieves its primary objective, the convergence of position and orientation errors to zero, and even ensures that the first derivatives of these quantities also converge, although this is not explicitly required by the control objective.

- **Comparison between trajectories:** As expected, the circular trajectory leads to smoother error profiles compared to the square trajectory. This is mainly due to the continuous and curvature, preserving nature of the circular path, which avoids the sudden changes in direction and sharp transitions inherent to the square motion.

- **Comparison with the hierarchical controller:** When comparing the performance of the ASMC with the hierarchical controller, several observations emerge. First, for the square trajectory, we notice that the error peaks are less sharp under the ASMC, and both the position and linear velocity errors are approximately 1 to 2 orders of magnitude smaller than those obtained with the hierarchical control. Conversely, the hierarchical controller achieves better performance in tracking the orientation and its derivative, with errors also reduced by 1 to 2 orders of magnitude compared to the ASMC. This difference reflects the outcome of the simulations: with the chosen gains, the ASMC achieves better performance in controlling position (and consequently linear velocity), while the hierarchical controller results in more accurate tracking of orientation and its derivative. The hierarchical controller is more effective in attitude regulation due to its fast inner-loop design, which provides precise and rapid control of the UAV's orientation.
  Another important consideration is the gain tuning. In the hierarchical controller, the same control gains are applied uniformly across all three axes. This configuration naturally results in smaller errors along the $z$-axis. On the other hand, in the ASMC, achieving similarly low errors along $z$ required assigning higher weights to the gains in that direction. This design choice is particularly significant: minimizing the error along the $z$-axis is crucial during both the square and circular segments of the trajectory to maintain a consistent flight altitude. Any deviation along $z$ could lead to vertical oscillations that negatively affect the UAV's stability and tracking performance.
  Regarding the control action, the same considerations discussed for $u_T$ in the hierarchical control are also valid for the vertical component of $U_f$. The difference is that, in this controller, the oscillations corresponding to the square trajectory (after the vertical ascent) are more damped, which is consistent with the fact that the position error exhibits less abrupt peaks, reduced by two orders of magnitude.

# Co-Planar UAV Controller in Presence of Disturbance

In this section, maintaining the optimal control gains obtained in the tuning process, we evaluate the performance of both the hierarchical controller and the ASMC under the effect of a sinusoidal additive disturbance $d(t) = \sin(0.2t)$ applied to all three components of the translational dynamics. As consequence, the results are shown only for the translational dynamics, since the disturbance affects this subsystem directly. Given the decoupled structure of the controllers design, the rotational dynamics are only marginally influenced.
We report the results only for the circular trajectory, as the behavior observed in the square trajectory is essentially equivalent. This is due to the fact that, although the two reference trajectories induce distinct tracking errors in nominal conditions, the addition of a sinusoidal disturbance in the translational dynamics results in similar error profiles, suggesting that the disturbance dominates the system's response regardless of the path geometry.

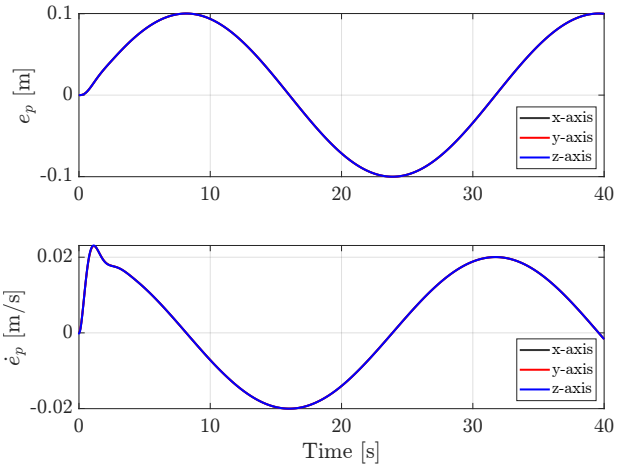


Figure 15: Circular trajectory: $e_p$ and $\dot{e_p}$ of hierarchical control with disturbance

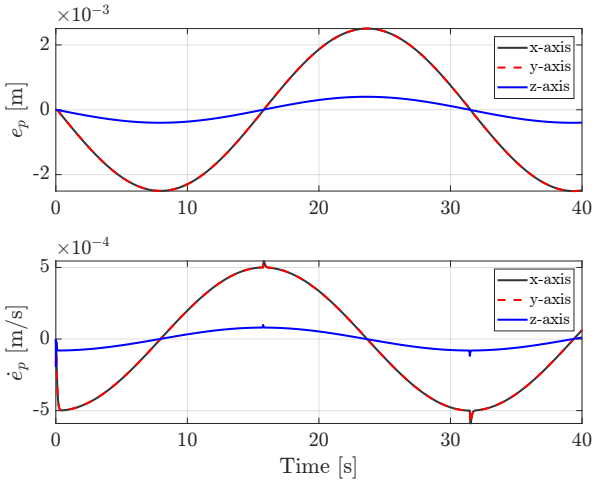


Figure 16: Circular trajectory: $e_p$ and $\dot{e_p}$ of ASMC with disturbance

- **Hierarchical control:** As shown in Figure 15, the hierarchical controller handles the disturbance moderately well. Although it does not incorporate a specific mechanism for disturbance rejection, it achieves a position error on the order of $10^{-1}$ and a linear velocity error on the order of $10^{-2}$. Compared to the nominal case (Figure 3), the position error has increased by four orders of magnitude, and the velocity error by two. Moreover, since the control gains are uniformly applied across all components and the disturbance acts equally along all directions, the system exhibits an equal response in the $x, y$, and $z$ axes.

- **Adaptive Sliding Mode Control:** As shown in Figure 16, the ASMC demonstrates superior robustness to the same disturbance compared to the hierarchical controller; it is evident in the videos `herarchical_circ_dist.mp4` and `adaptive_circ_dist.mp4`: the hierarchical controller shows that the circular trajectory is not perfectly planar, it does not remain at the same height throughout); in contrast, in the ASMC, the videos with and without disturbance are almost identical. In this case, the position error is on the order of $10^{-3}$, and the linear velocity error is on the order of $10^{-4}$. Compared to the nominal case (Figure 11), this corresponds to an increase of three orders of magnitude in the position error, one order less than that observed with the hierarchical controller. Additionally, the smallest errors are observed along the $z$-axis, which is consistent with the higher gain weight assigned to that direction. This improved performance reflects the expected behavior of the ASMC approach. Its adaptive structure estimates the bounds of unknown disturbances in real time and adjusts the control input accordingly.

## Tilting UAV control with Holocopter approach

The first control strategy that we implement to control a tilting quadrotor is the Holocopter allocation approach. The allocation matrix maps the propeller control inputs to the resulting total wrench (forces and torques):

$$\begin{bmatrix} f_x \\ f_y \\ f_z \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} 0 & -c_f sin(\alpha_2) & 0 & c_f sin(\alpha_4) \\ c_f sin(\alpha_1) & 0 & -c_f sin(\alpha_3) & 0 \\ -c_f cos(\alpha_1) & -c_f cos(\alpha_2) & -c_f cos(\alpha_3) & -c_f cos(\alpha_4) \\ 0 & -lc_f cos(\alpha_2) - c_m sin(\alpha_2) & 0 & lc_f cos(\alpha_4) + c_m sin(\alpha_4) \\ lc_f cos(\alpha_1) + c_m sin(\alpha_1) & 0 & -lc_f cos(\alpha_3) - c_m sin(\alpha_3) & 0 \\ lc_f sin(\alpha_1) - c_m cos(\alpha_1) & -lc_f sin(\alpha_2) + c_m cos(\alpha_2) & lc_f sin(\alpha_3) - c_m cos(\alpha_3) & -lc_f sin(\alpha_4) + c_m cos(\alpha_4) \end{bmatrix} \begin{bmatrix} u_{\omega_1} \\ u_{\omega_2} \\ u_{\omega_3} \\ u_{\omega_4} \end{bmatrix}$$

where:

- $c_f \in \mathbb{R}$: thrust coefficient, it maps the squared angular speed of a rotor to the vertical force it generates. Specifically, the thrust produced by the $i^{th}$ propeller is given by $f_i = c_f|\omega_i|\omega_i$, where $\omega_i$ is the angular velocity of the $i^{th}$ propeller.
- $c_m \in \mathbb{R}$: drag factor, it maps the square of the propeller speed to the torque around the propeller spinning axis due to aerodynamic drag. Specifically, the torque produced by the $i^{th}$ propeller is given by $\tau_i = k_i c_m|\omega_i|\omega_i$, where $k_i$ takes into account the sense of rotation of the corresponding propeller.
- $l \in \mathbb{R}$: distance from the center of mass of the UAV to each propeller.
- $\alpha = [\alpha_1, \alpha_2, \alpha_3, \alpha_4]^T \in \mathbb{R}^4$: tilting angles of the 4 propellers, measured with respect to the vertical axis.

This can be written compactly as:

$$\begin{bmatrix} f \\ \tau \end{bmatrix} = \begin{bmatrix} G_{q,f}(\alpha) \\ G_{q,\tau}(\alpha) \end{bmatrix} u_\omega = \sum_{i=1}^{4} g_{q,i}(\alpha) u_{\omega_i}$$

The geometric dynamic model of the tilting UAV is given by:

$$\begin{cases} \begin{bmatrix} mI_3 & O \\ O & I_b \end{bmatrix} \begin{bmatrix} \ddot{p}_b \\ \dot{\boldsymbol{\omega}}_b^b \end{bmatrix} = \begin{bmatrix} mge_3 \\ -S(\omega_b^b) I_b \omega_b^b \end{bmatrix} + \begin{bmatrix} R_b & O \\ O & I_3 \end{bmatrix} \begin{bmatrix} G_{q,f}(\alpha) & O \\ G_{q,\tau}(\alpha) & O \end{bmatrix} \begin{bmatrix} u_\omega \\ u_\alpha \end{bmatrix} \\ \dot{R}_b = R_b S(\omega_b^b) \\ \dot{\alpha} = u_\alpha \end{cases}$$

**Note:** The allocation matrix has rank less than 6, so the system appears to be underactuated. This is because $u_\alpha$ acts at a higher differential level than $u_\omega$, and a direct inversion at acceleration level would involve $u_\omega$ only, causing loss of controllability. To overcome this limitation, we move to a higher differential level where $u_\alpha$ appears explicitly.

The dynamics at jerk level becomes:

$$\begin{cases} \begin{bmatrix} \dddot{p}_b \\ \ddot{\omega}_b^b \end{bmatrix} = b(R_b, \omega_b^b, \dot{\omega}_b^b, \alpha, u_\omega) + A(R_b, \alpha, u_\omega) \begin{bmatrix} u_\nu \\ u_\alpha \end{bmatrix} \\ \dot{R}_b = R_b S(\omega_b^b) \\ \dot{\alpha} = u_\alpha \\ \dot{u}_\omega = u_\nu \end{cases}$$

where:

$$b(R_b, \omega_b^b, \dot{\omega}_b^b, \alpha, u_\omega) = \begin{bmatrix} mI_3 & O \\ O & I_b \end{bmatrix}^{-1} \left( \begin{bmatrix} 0 \\ -S(\dot{\omega}_b^b) I_b \omega_b^b - S(\omega_b^b) I_b \dot{\omega}_b^b \end{bmatrix} + \begin{bmatrix} R_b S(\omega_b^b) & O \\ O & O \end{bmatrix} G_q(\alpha) u_\omega \right)$$

$$A(R_b, \alpha, u_\omega) = \begin{bmatrix} mI_3 & O \\ O & I_b \end{bmatrix}^{-1} \begin{bmatrix} R_b & O \\ O & I_3 \end{bmatrix} \begin{bmatrix} G_q(\alpha) & \sum_{i=1}^{4} \frac{\partial g_{q,i}(\alpha)}{\partial \alpha} u_{\omega_i} \end{bmatrix}$$

The input matrix $A(R_b, \alpha, u_\omega) \in \mathbb{R}^{6\times 8}$ is now full-rank as long as $u_\omega \neq 0$. In this case, feedback linearization can be applied to design the controller.
We introduce a virtual control input $v$ related to the real control inputs in this way:

$$\begin{bmatrix} u_\nu \\ u_\alpha \end{bmatrix} = A^\dagger (v - b)$$

Substituting into the jerk level dynamics we obtain:

$$\begin{bmatrix} \dddot{p}_b \\ \dddot{\boldsymbol{\omega}}_b^b \end{bmatrix} = v \qquad v = \begin{bmatrix} -k_{pl} e_p - k_{vl} \dot{e}_p - k_{al} \ddot{e}_p + \dddot{p}_{b,d} \\ -k_{pa} e_R - k_{va} e_\omega - k_{aa} e_a + \dddot{\omega}_{b,d}^b \end{bmatrix}$$

where $v$ is chosen in this way in order to give an exponential decreasing behavior to the tracking errors on position, velocity and acceleration:

$$e_p = p_b - p_{b,d} \qquad \dot{e_p} = \dot{p_b} - \dot{p_{b,d}} \qquad \ddot{e_p} = \ddot{p_b} - \ddot{p_{b,d}}$$

$$e_R = \frac{1}{2}(R_{b,d}^T R_b - R_b^T R_{b,d})^\vee \qquad e_\omega = \omega_b^b - R_b^T R_{b,d} \omega_{b,d}^b \qquad e_a = \dot{\omega}_b^b - R_b^T R_{b,d} \dot{\omega}_{b,d}^b$$

The Tilting control with Holocopter approach was implemented in Simulink within the file `tilting_holocopter_approach.slx`, which contains the following key blocks:

- The subsystem **Planner:** as already described in the previous controllers, this block takes as input the variables declared and computed in the `init_function`, where the reference trajectory for the UAV is generated. The subsystem returns the desired states required for simulation. Specifically, unlike in co-planar UAV controllers where the planner only provides the desired yaw angle and the desired roll and pitch angles must be computed in subsequent blocks, in this tilting controller the full desired orientation is expressed through the rotation matrix $R_{b,d}$, given as output by the planner.

- The MATLAB Function **compute_control_input:** this block implements the control law by computing and returning the control inputs $u_\omega$ and $u_\alpha$. It also returns the errors in position, linear velocity, linear acceleration, orientation, angular velocity, and angular acceleration. These errors are monitored through scopes to evaluate whether they converge to zero as expected.

- The subsystem **Quadrotor_dynamics:** this block receives the control inputs and, through the MATLAB Function `allocation_matrix`, transforms them into standard control inputs, namely the forces $f^b$ and torques $\tau^b$ applied to the UAV.
  Subsequently, using the translational and rotational dynamics equations, the MATLAB Function `dynamics` computes the current values of position, linear velocity and acceleration, orientation (through the rotation matrix), as well as angular velocity and angular acceleration. These values are returned by the subsystem.

In the following, we present the simulation results for the tilting quadrotor using the Holocopter approach as the control strategy on both planned trajectories. The chosen gain values are: $k_{pl} = \text{diag}\{50, 50, 100\}$, $k_{vl} = \text{diag}\{30, 30, 100\}$, $k_{al} = \text{diag}\{10, 10, 100\}$, $k_{pa} = \text{diag}\{4, 4, 4\}$, $k_{va} = \text{diag}\{3.15, 3.15, 3.15\}$, $k_{aa} = \text{diag}\{1.75, 1.75, 1.75\}$.

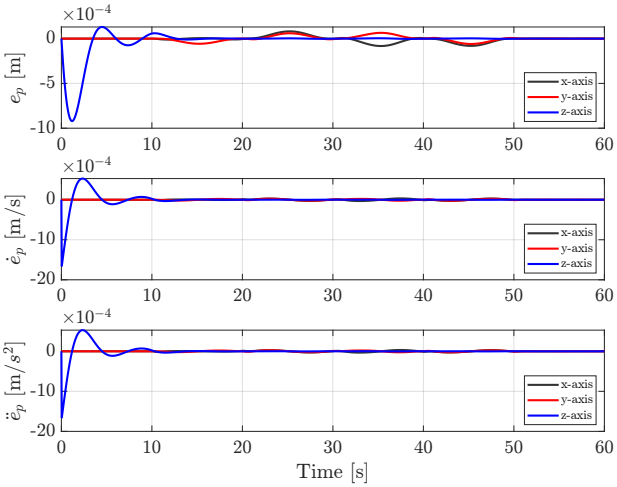


Figure 17: Square trajectory: position, linear velocity and linear acceleration errors

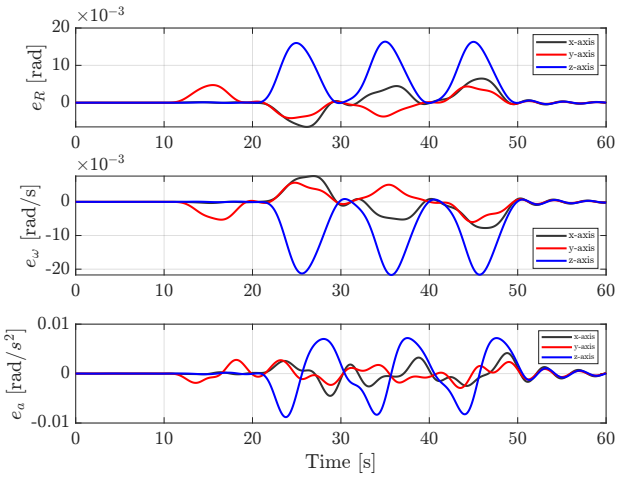


Figure 18: Square trajectory: orientation, angular velocity and angular acceleration errors

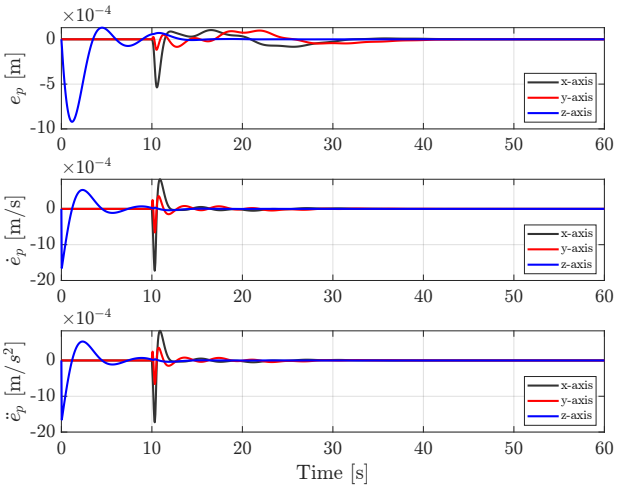


Figure 19: Circular trajectory: position, linear velocity and linear acceleration errors

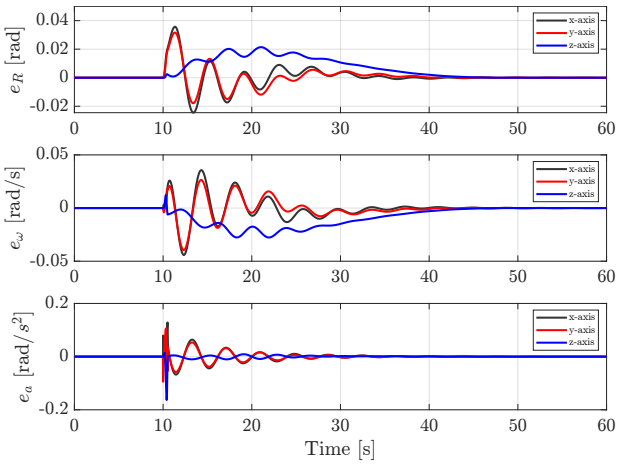


Figure 20: Circular trajectory: orientation, angular velocity and angular acceleration errors

- **Square trajectory**: As shown in Figures 17 and 18, the tracking errors are very small. Specifically, except for the initial peaks along the $z$-axis, the position, linear velocity, and linear acceleration errors are on the order of $10^{-4}$. The orientation and angular velocity errors are on the order of $10^{-3}$, with the exception of the $z$ component, which is one order of magnitude larger. The angular acceleration error is on the order of $10^{-2}$. Moreover, all linear errors converge to zero between 50 and 60 seconds, i.e., during the dead time, while angular errors are still settling (suggesting that a slightly longer dead time would be needed to observe full convergence), confirming that the controller satisfies the performance requirements.
- **Circular trajectory**: As shown in Figures 3 and 4, the tracking errors are even smaller. Specifically, the position, linear velocity, and linear acceleration errors are all on the order of $10^{-4}$, except for the initial peaks along the $z$-axis and the peaks around $t = 10\,\text{s}$ along the $x$-axis. The orientation and angular velocity errors are on the order of $10^{-2}$, and the angular acceleration error reaches up to the order of $10^{-1}$. All errors converge to zero before the dead time: linear errors stabilize around 30

seconds, while angular errors settle around 40 seconds, again confirming that the controller meets the design requirements.

- **Square trajectory analysis**: As expected, during the first 10 seconds (the vertical ascent phase), the position, linear velocity, and linear acceleration errors are larger along the $z$-axis, showing an initial peak that then decreases as the UAV performs the square path. Along the $x$ and $y$ axes, these errors exhibit more oscillatory behavior during the square segments.
  As for orientation, angular velocity, and angular acceleration errors, they remain nearly zero during the vertical ascent phase, but increase significantly once the UAV enters the square trajectory. In the first side of the square (10–20 s), only a roll motion (around the $x$-axis) is commanded. In this case, the controller must compensate by introducing a small pitch motion (around the $y$-axis) to maintain lateral stability. As a result, angular errors appear mainly around the $y$-axis, while errors around the other axes remain negligible. In the subsequent sides (20–30, 30–40, 40–50 s), where both roll and pitch are actively commanded, angular errors appear along both $x$ and $y$ as expected. Additionally, due to the non-commutative nature of 3D rotations, the simultaneous variation of roll and pitch produces coupling effects that also result in angular errors around the $z$-axis (yaw), even though no yaw rotation is commanded. These yaw errors tend to be minimal at the endpoints of each side and reach a maximum near the midpoint, corresponding to the points of highest rotational activity.

- **Circular trajectory analysis**: The linear error profiles are overall similar to those in the square trajectory. However, at the transition between the vertical ascent and the beginning of the circular path (around $t = 10\,\text{s}$), we observe significant peaks in the linear errors along $x$ and $y$, especially along x. This difference arises because, unlike the square where each motion segment is isolated (one axis at a time), the circular path requires simultaneous control of all translational and rotational degrees of freedom starting from $t = 10\,\text{s}$. As a result, the controller must manage a more complex dynamic interaction, particularly after a segment that involved only motion along and around the $z$-axis.
  Regarding angular errors, they remain negligible during the vertical segment, similar to the square case. They begin increasing at the start of the circular motion (10 s). The angular errors along $x$ and $y$ exhibit a transient increase, followed by damped oscillations around zero. The error along $z$ oscillates around a nonzero value, in fact, as we can see in the orientation error, it first increases and then gradually converges to zero.

- **Conclusion:** Unlike the co-planar case, the circular trajectory in this configuration does not exhibit a significantly smoother error profile compared to the square trajectory. This is mainly due to the increased complexity of the motion involved. In the tilting setting, starting from $t = 10\,\text{s}$, the circular path requires simultaneous translation along both $x$ and $y$ axes and coordinated rotations around both the roll and pitch axes. This combined maneuver leads to stronger dynamic coupling between translational and rotational subsystems. As a result, the controller must handle multiple interactions concurrently, which increases the difficulty of maintaining smooth and precise tracking. Nevertheless, the tracking errors tend to settle more quickly in the circular trajectory compared to the square one. This behavior can be attributed to the fact that the circular path is handled as a single continuous arc, allowing the controller to manage the trajectory in a globally coordinated manner. In contrast, the square trajectory is composed of multiple independent segments, each corresponding to a straight-line motion along a specific axis. These discontinuities between segments introduce additional transient dynamics that the controller must compensate for at every corner, resulting in slower error convergence overall.

# Non Linear Model Predictive Control (NMPC)

Another control strategy that we implement for tilting quadrotor trajectory tracking is the NMPC, that computes the control inputs by solving a single optimization problem at each control step.
The dynamic model of the UAV includes both translational and rotational dynamics, expressed respectively in the world and body frames:

$$\begin{cases} \ddot{p_b} = g + \frac{1}{m}\left(R_b(f_P^b - A_T \mathrm{v}_b^b) + f_D\right) \\ \dot{\omega}_b^b = J^{-1}\left(-\omega_b^b \times J\omega_b^b + \tau_P^b - A_R\omega_b^b + \tau_D\right) \end{cases}$$

where:

- $f_P^b \in \mathbb{R}^3$: total force vector in the body frame;
- $f_D \in \mathbb{R}^3$: external disturbance force;
- $A_T \in \mathbb{R}^{3\times3}$: translational aerodynamic drag matrix;
- $J \in \mathbb{R}^{3\times3}$: UAV inertia matrix;
- $\tau_P^b \in \mathbb{R}^3$: torque vector in the body frame;
- $\tau_D \in \mathbb{R}^3$: external disturbance torque;
- $A_R \in \mathbb{R}^{3\times3}$: rotational aerodynamic drag matrix.

The system state is defined as: $x = \begin{bmatrix} p_b^T & \eta_b^T & v_b^{bT} & \omega_b^{bT} \end{bmatrix}^T \in \mathbb{R}^{12}$ and the desired state is defined analogously: $x_d = \begin{bmatrix} p_{b,d}^T & \eta_{b,d}^T & v_{b,d}^{bT} & \omega_{b,d}^{bT} \end{bmatrix}^T \in \mathbb{R}^{12}$.

The NMPC solves, at each time step, the following finite-horizon ($N$) optimal control problem:

$$\min_v \sum_{i=k}^{k+N-1} \ell(x(i), v(i)) + \Phi(x(k+N))$$

subject to:

- System dynamics: $x(i+1) = f(x(i), v(i))$;
- State and input constraints: $x(i) \in \mathcal{X},\ v(i) \in \mathcal{V}$;
- Initial condition: $x(0) = x_0$.

The cost function is defined as: $\ell(x, v) = (x - x_d)^T Q_x (x - x_d) + v^T Q_v v$, with terminal cost: $\Phi(x(k+N)) = 0$.

Note that the attitude error $e_R$ and angular velocity error $e_\omega$ are not computed as simple differences, but as defined in the tilting controller framework.
The solution of the optimization is the virtual wrench: $v = \begin{bmatrix} f_P^{bT} & \tau_P^{bT} \end{bmatrix}^T \in \mathbb{R}^6$. However, the real control input applied to the UAV is: $u = \begin{bmatrix} u_{\omega_1} & u_{\omega_2} & u_{\omega_3} & u_{\omega_4} & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \end{bmatrix}^T \in \mathbb{R}^8$, which includes the square of the propeller speeds and the tilt angles of the four propellers.
To implement the virtual wrench $v$ using the actual input $u$, a pseudo-inverse-based allocation strategy is used. Let's consider an intermediate vector $u' = \begin{bmatrix} f_{P1x} & f_{P1z} & f_{P2x} & f_{P2z} & f_{P3y} & f_{P3z} & f_{P4y} & f_{P4z} \end{bmatrix}^T \in \mathbb{R}^8$ that includes the non-zero thrust components of each rotor: rotors 1 and 2 generate thrust in the $x-z$ plane, while rotors 3 and 4 act in the $y-z$ plane. It is related to the wrench in this way:

$$v = Bu' \quad \Rightarrow \quad u' = B^\dagger v$$

where $B \in \mathbb{R}^{6\times 8}$ is the control effectiveness matrix:

$$B = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & l & 0 & -l & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -l & 0 & l \\ -l & -\frac{c_m}{c_f} & l & -\frac{c_m}{c_f} & l & \frac{c_m}{c_f} & -l & \frac{c_m}{c_f} \end{bmatrix}$$

with $l, c_f, c_m$ defined as in the previous controller.
The real control inputs are recovered from $u'$ as follows:

$$u_{\omega_i} = \frac{30}{\pi}\sqrt{\frac{|f_{P_i}|}{c_f}} \qquad \alpha_i = \tan^{-1}\left(\frac{f_{P_i,z}}{\sqrt{f_{P_i,x}^2 + f_{P_i,y}^2}}\right)$$

A key feature of this NMPC formulation is that actuator constraints are explicitly handled within the optimization problem. This is done by enforcing $u_{\min}(k) \leq h(v) \leq u_{\max}(k)$, where $h : \mathbb{R}^6 \to \mathbb{R}^8$ is the nonlinear mapping from the virtual wrench $v$ to the real actuator commands $u$.

**Note:** Although the physical UAV is actuated via motor speeds $u_{\omega_i}$ and tilt angles $\alpha_i$, in simulation we bypass the low-level actuator dynamics by directly applying the desired wrench $v = [f_P^{bT}, \tau_P^{bT}]^T$ to the system model. This simplification is acceptable for high-level control validation, as it allows us to test the performance of the NMPC without simulating motor dynamics, saturations, or delays. Nevertheless, we still incorporate the mapping between the wrench $v$ and the actual control inputs $u_{\omega_i}, \alpha_i$ when formulating input constraints. In real world, constraints are applied to the real control inputs, not directly to the wrench, and this relationship ensures that the computed wrench remains physically feasible.

The NMPC was implemented in Simulink within the file `tilting_NMPC.slx`, which contains:

- The subsystem **Planner:** this block receives as input the variables declared and computed in the `init_function`, where the reference trajectory for the UAV is generated. It returns the desired state vector $x_d$, as previously described, required for the simulation.
- The block **Nonlinear MPC Controller:** this is a built-in MATLAB block that solves the optimal control problem at each time step, using the current state, the reference trajectory, and the solution from the previous iteration. The block internally calls the MATLAB script `mainMPC.m`, which sets the prediction horizon, defines the number of state, output, and input variables, specifies the cost function weights, and includes the following defined functions: `droneDynamics` that defines the UAV's nonlinear dynamics; `CostFunction` that defines the stage cost, which in this case penalizes the tracking error between the state and the reference, and the control effort; `OutputFcn`, that returns the outputs variables, that in this case are coincident with the state variables; `myConstraintsFcn`, that defines the actuator constraints.
- The subsystem **Quadrotor_dynamics:** this block receives the outputs of the optimization problem and, based on them, computes and provides the current values of position, orientation (exploiting the relationship between $\dot{\eta}$ and $\omega$ provides by the matrix $H(\eta)$), linear velocity, and angular velocity.
- The MATLAB Function **compute_errors:** this function computes the position, orientation, linear velocity, and angular velocity errors. These values are displayed using scopes to evaluate the effectiveness of the optimization process, particularly with respect to the minimization of the tracking errors.

In the following, we present the simulation results for the tilting quadrotor operating under nominal conditions, using the NMPC controller on both planned trajectories. The weight matrices used in the cost function

are reported below: $Q_p = \text{diag}\{10^5,\ 10^5,\ 10^6\}$, $Q_\eta = \text{diag}\{10^{-4},\ 10^{-4},\ 10^2\}$, $Q_{\dot{p}} = \text{diag}\{10^{-2},\ 10^{-2},\ 10^{-2}\}$, $Q_\omega = \text{diag}\{10^{-2},\ 10^{-2},\ 10^{-2}\}$, $Q_{\text{input}} = \text{diag}\{0.01,\ 0.01,\ 0.01,\ 0.01,\ 0.01,\ 0.01\}$.

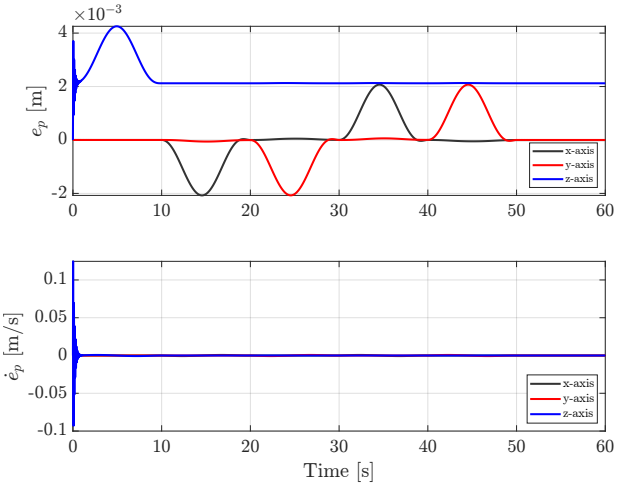


Figure 21: Square trajectory: position and linear velocity errors

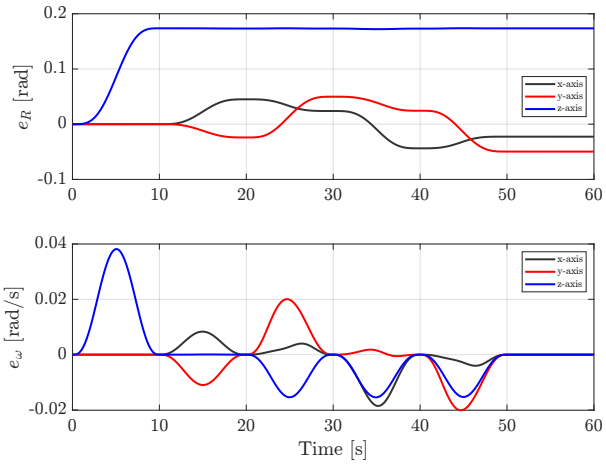


Figure 22: Square trajectory: orientation and angular velocity errors

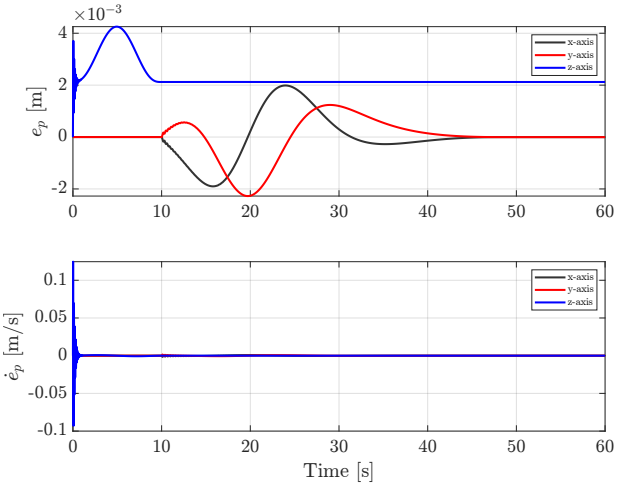


Figure 23: Circular trajectory: position and linear velocity errors

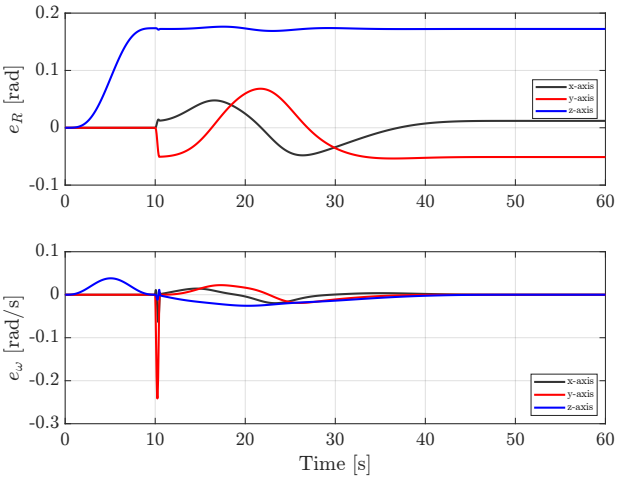


Figure 24: Circular trajectory: orientation and angular velocity errors

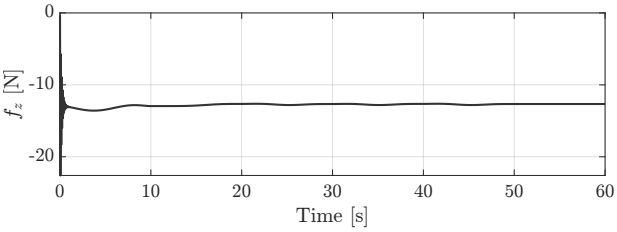


Figure 25: Square trajectory: $z$ component of $f_P$

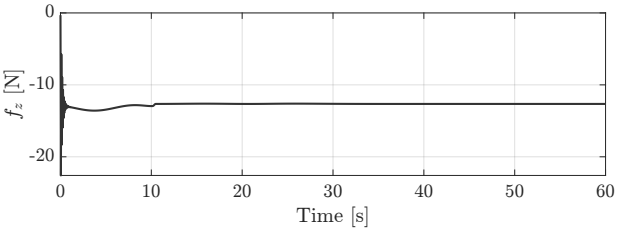


Figure 26: Circular trajectory: $z$ component of $f_P$

- **Error analysis for both trajectories:** For both the square trajectory (Figures 21 and 22) and the circular trajectory (Figures 23 and 24), we observe that the tracking errors are reasonably small. Specifically, the position error is on the order of $10^{-3}$, while the linear velocity error is generally on the order of $10^{-3}$, with the exception of a larger initial peak (two order of magnitude higher) along the $z$-axis. The orientation error is on the order of $10^{-1}$ for $z$ component and $10^{-2}$ for $x$ and $y$, and the angular velocity errors are around $10^{-2}$, except for a peak along the $x$ and $y$ axes in the circular trajectory at the transition between the vertical and planar segments (around $t = 10\,\text{s}$). We also observe that linear velocity and angular velocity errors tend to converge to zero during the dead time (50–60 s). However, the situation is different for position and orientation errors. In both trajectories, a similar steady-state behavior emerges: the orientation error around the $z$-axis stabilizes around approximately 0.17 rad, while the $x$ and $y$ components settle to small values close to zero. For the position error, the $x$ and $y$ components converge to zero, but the $z$ component stabilizes around approximately 0.004 m.

- **Comparison between trajectories:** Regarding the position error, in the square trajectory each axis is affected only during the corresponding motion segment: along $z$ in the initial ascent, along $x$ during segments 10–20 s and 30–40 s, and along $y$ during segments 20–30 s and 40–50 s. In each case, the error peaks at the midpoint of the segment and decreases towards the endpoints. In contrast, for the circular trajectory, the $z$-axis behavior is similar, but, as expected, the $x$ and $y$ components of the position error are different to zero throughout the entire circular motion.
  As for orientation errors, the $z$ component shows a similar trend in both trajectories, instead for the $x$ and $y$ components, that starts to increase after 10 s, in the square trajectory, converge more slowly and show smaller amplitude, whereas in the circular trajectory they are more pronounced.
  For the linear velocity errors, we observe high-frequency oscillations on $z$ component at the beginning of the simulation. This behavior is primarily due to the weight configuration within the NMPC cost function: relatively low weights were assigned to the control inputs in order to avoid excessive penalization of the control effort. This allows the controller to apply stronger control actions when necessary, as we can see in Figures 25 and 26. This effect is evident in the $z$ component because, to compensate for the offset between the desired and current position values, it was necessary to assign a higher weight to $z$. However, a common side effect of such aggressive control inputs is the appearance of high-frequency oscillations, particularly in the velocity profiles. To mitigate this, a careful tuning process was carried out to find an appropriate balance between minimizing tracking errors and limiting the magnitude of the control effort, ensuring a reasonable trade-off between tracking accuracy and smoothness of the control signal.

- **Comparison with the Holocopter approach:** When comparing the performance of the NMPC with the Holocopter-based controller, several key differences emerge. First, a fundamental distinction lies in the convergence behavior: while the Holocopter approach ensures full convergence of the tracking errors to zero, the NMPC minimizes the errors according to a cost function. As a result, with the chosen weights, the NMPC achieves small but non-zero steady-state errors, as we can expected.
  Another notable difference appears in the circular trajectory, particularly at the transition around 10 s. In the Holocopter approach, this point generates distinct peaks in both linear and angular errors, due to the abrupt increase in control effort required by the onset of coupled motions. In contrast, the NMPC manages this transition more smoothly: all peaks are significantly reduced, except for a noticeable one in angular velocity, while the others are almost completely suppressed.
  In summary, while the Holocopter approach achieves better accuracy in terms of error magnitude and guarantees asymptotic convergence, the NMPC exhibits a smoother and more gradual error evolution, especially during dynamic transitions.

## Tilting UAV Controller in Presence of disturbance

As in the case of the UAV co-planar controllers, we assess the performance of both the Holocopter approach and the NMPC under the effect of a sinusoidal additive disturbance $d(t) = \sin(0.2t)$, applied uniformly to all three components of the translational dynamics. The control gains previously tuned for each strategy are kept unchanged for this analysis. For the same reason as in earlier sections, we focus exclusively on the translational dynamics and, in the opposite of the co-planar UAV, we report results only for the square trajectory, as the behavior observed in the circular trajectory is qualitatively similar.

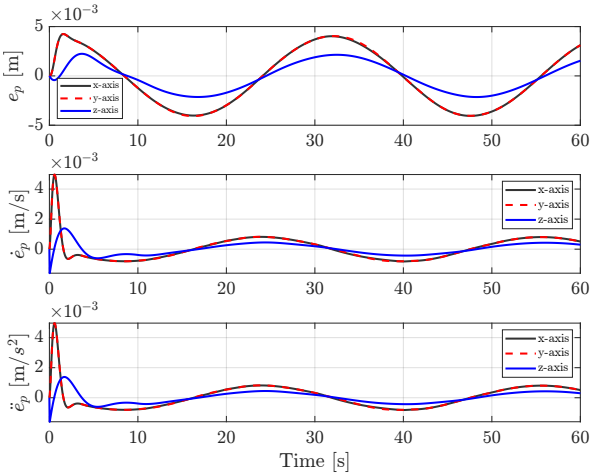


Figure 27: Square trajectory: $e_p, \dot{e}_p$ and $\ddot{e}_p$ of Holocopter approach with disturbance

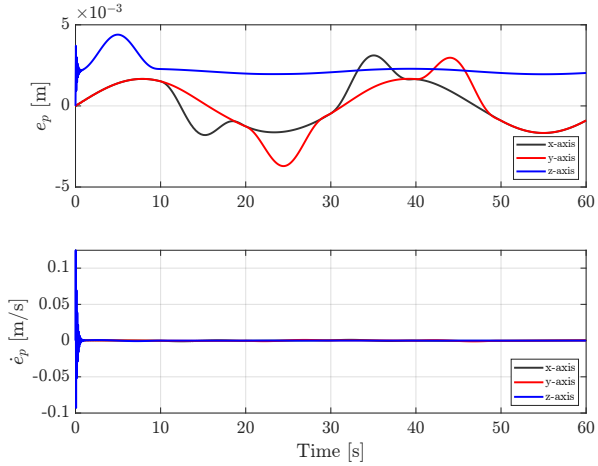


Figure 28: Square trajectory: $e_p$ and $\dot{e}_p$ of NMPC with disturbance

- **Holocopter Approach:** As shown in Figure 27, the Holocopter control strategy handles the disturbance reasonably well. Although it does not include a dedicated mechanism for disturbance rejection, it maintains position, linear velocity, and linear acceleration errors on the order of $10^{-3}$. Compared to the nominal case (Figure 17), this represents an increase of approximately one order of magnitude in all linear error components. Due to the higher gain weights assigned to the $z$-direction, we observe better disturbance rejection along this axis, particularly in the position error. For the linear velocity and acceleration, however, the $z$-component exhibits only a smaller initial peak, while during the rest of the simulation the amplitude remains comparable across all three directions. This suggests that the effect of the gain asymmetry is more pronounced in steady-state position tracking.
- **NMPC:** As shown in Figure 28, the NMPC demonstrates greater robustness to the same disturbance compared to the Holocopter approach. In this case, both position and linear velocity errors remain on the same order of magnitude as in the nominal scenario (Figure 21). The main visible difference lies in the position error behavior. Specifically, the $x$ and $y$ components exhibit slightly larger amplitudes compared to the nominal case and no longer stay at zero during phases of no corresponding motion. Instead, they tend to oscillate following a sinusoidal pattern. The $z$-component, by contrast, remains practically unchanged from the undisturbed case. This superior disturbance rejection performance is consistent with the nature of the NMPC. It optimizes future control actions over a moving horizon based on the current measured state. Even if disturbances are not explicitly modeled, their effect is reflected in the updated state, allowing the controller to adjust the entire future trajectory rather than only correcting the instantaneous error. This gives NMPC more effective disturbance compensation compared to the Holocopter tilting control.

Overall, the videos `holocopter_square_dist` and `MPC_square_dist` show that both approaches react fairly well to disturbances.

# Conclusion

All the implemented control strategies, after an accurate tuning process, demonstrated good performance on both the planned square and circular trajectories. In the following, we present separate considerations for the controllers designed for co-planar UAVs and for tilting UAVs, highlighting the operational conditions under which one controller may be preferred over another.

Regarding the **Co-planar UAV** controllers (Hierarchical Controller and the Adaptive Sliding Mode Controller), which were developed for controlling co-planar UAVs, both exhibit better performance on the circular trajectory, which is characterized by smoother motion.
In nominal condition, if the priority is achieving **high orientation accuracy**, for instance during visual inspection tasks where the onboard camera must remain consistently aligned with a fixed target, the Hierarchical Controller is preferable. Its inner loop, dedicated to attitude control, operates faster than the outer loop acting on position errors, providing better angular dynamics management. Conversely, if the priority is **position accuracy**, as in precise payload delivery scenarios, it is preferable to choose the ASMC, since with the selected gains it provides better performance.
However, in scenarios with **external disturbances**, especially sinusoidal ones (e.g., periodic turbulence caused by industrial machinery or wind turbine blades), the ASMC is more suitable. Its robustness is due to the adaptive terms included in its structure, which compensate for disturbances even without precise knowledge of their bounds.

Regarding the **Tilting UAV** controllers (controller that use Holocopter approach and the Nonlinear Model Predictive Controller), both them perform better on the circular trajectory than on the square one, thanks to the smoother motion that favors stabilization.
In nominal conditions, if the priority is achieving **exact convergence of position and orientation errors** (including their derivatives), for example in applications where absolute precision is required, the Holocopter approach is preferable. This method is designed to cancel errors and guarantee accurate trajectory tracking.
Instead, if the priority is handling **sharp changes in orientation** along the trajectory, such as the critical transition between the vertical ascent and the circular path, the NMPC is more advantageous. Thanks to its predictive nature, it mitigates the effect of these abrupt variations on the linear dynamics, stabilizing the system more effectively.
In the presence of **external disturbances**, particularly periodic or sinusoidal ones, the NMPC proves to be more robust than the Holocopter approach. This robustness arises from its predictive structure, which forecasts the system's future evolution and compensates for disturbances within its prediction horizon.

**Note:** Control of the position along the $z$-axis is generally more delicate than for the other components. In this application, precise altitude control becomes critical after completing the vertical ascent, when the UAV is expected to have a planar motion. At this stage, it is essential to properly compensate for the effect of gravity to maintain the achieved altitude; otherwise, any residual vertical drift would affect horizontal trajectory tracking and overall stability. For this reason, the control gains associated with the $z$ component are typically set higher than those of the lateral axes, providing a faster and more robust response to disturbances and modeling uncertainties that impact altitude regulation.

**Note:** The dynamic models adopted in this work are slightly different, both for the controllers of the co-planar UAVs and for those of the tilting UAVs. This was an intentional choice to remain consistent with the course material used for the controllers studied during the lectures and with the references adopted for the newly implemented controllers. Although the physical system being controlled is the same, it can be modeled in different ways depending on the assumptions made, for example, regarding effects not strictly related to the dynamics or parameter modeling.