Field and Service Robotics: Technical Report

Fabio Ruggiero

2024/2025

Students:

Gianmarco Corrado P38000287

Andrea Colapinto P38000309

Table of Contents

Supplementary Material

GitHub Repository

Full MATLAB/Simulink source code and documentation can be found at the following link:

`github.com/colandrea02/NMPC_for_tilting_quadrotor`

Simulation Videos

Videos showing simulations in a 3D environment can be accessed here:

`Google Drive Folder - Simulation Results`

Team Contributions

All parts of the project were developed collaboratively by both team members. Andrea focused primarily on the Simulink implementation, while Gianmarco contributed mainly to the MATLAB scripting and function development. The design, testing, and debugging phases were carried out jointly.

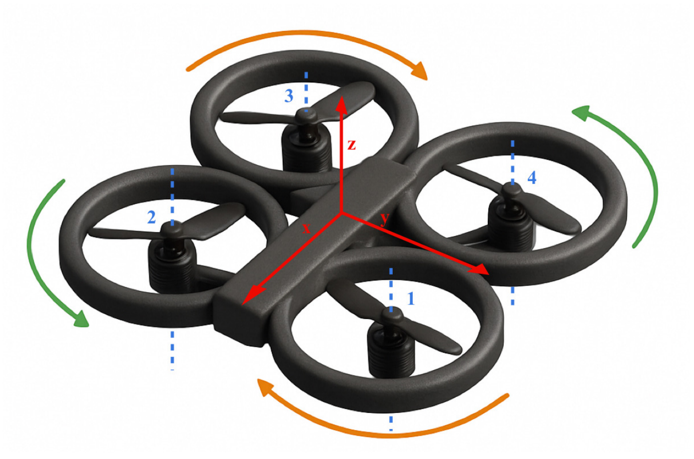


Figure 1: Tilting quadrotor model.

Abstract

This work focuses on the development and simulation of an NMPC[1] for a tilting quadrotor, operating in a structured indoor environment. This project was chosen due to the complex control challenges made possible by the tilting mechanism, and for the opportunity to explore advanced model-based control strategies in a realistic scenario.

The UAV (Unmanned Aerial Vehicle) is equipped with four rotors arranged in a cross configuration, as shown in Figure 1, where the direction of rotation for each rotor is also indicated. The alternating clockwise and counterclockwise rotation pattern ensures torque balance around the vertical (yaw) axis. Each rotor features an independent tilting mechanism, allowing an adjustment of the thrust direction. This additional degree of freedom improves maneuverability, which is beneficial during aggressive maneuvers and navigation through confined spaces. However, it also increases the complexity of the system dynamics and requires advanced control techniques to handle the nonlinear behavior and the input constraints.

Given the nonlinear nature of the UAV dynamics, a nonlinear model predictive control strategy has been adopted. Unlike classical linear MPC approaches, which rely on local linearization and may lose accuracy during dynamic maneuvers, the NMPC incorporates the full nonlinear model directly into the control problem formulation, providing better performance in complex scenarios.

The proposed control framework enables accurate trajectory tracking while taking into account aerodynamic effects such as ground and ceiling interactions, which significantly influence thrust efficiency in proximity to such surfaces.

The mission involves navigating a cluttered 3D indoor environment (see Figure 2), passing through a narrow opening, and reaching a target location where the UAV is required to hover precisely.

[1]Nonlinear Model Predictive Control

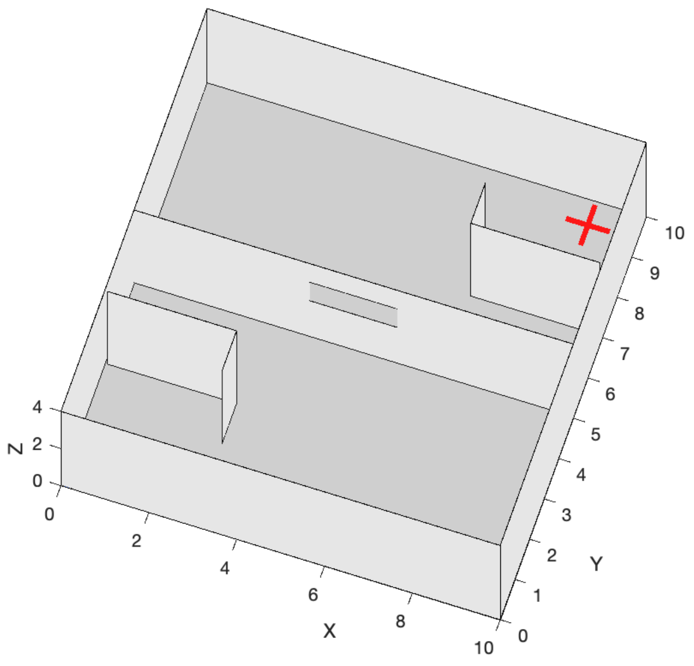


Figure 2: Simulation environment.

To achieve these objectives, the dynamic model is described in Chapter 1, which includes rigid-body dynamics, actuator behavior, and environmental interaction terms related to ground and ceiling proximity.
A full-state NMPC controller is then implemented in a MATLAB/Simulink environment, as detailed in Chapter 2. In the end, Chapter 3 shows the various simulation results.
The implementation includes:

- a full nonlinear dynamic model of the UAV with tilt-rotor actuation,
- polynomial-based 3D trajectory planning,
- integration of aerodynamic effects (ground and ceiling), external disturbances and noises,
- a Simulink-based NMPC controller for computation of optimal control inputs.

1 System Modeling

This section focuses on the development of the dynamic model of the tilting quadrotor that has been previously introduced. Such dynamic model plays a fundamental role in predicting the behavior of the system and serves as the basis for the design of control algorithms.
For algorithmic requirements, the orientation of the aerial platform is represented using a unit quaternion instead of traditional rotation matrices oe Euler angles. This choice deletes singularities and also consents to represent the orientation in a column vector.

1.1 Model Dynamics

The dynamic model adopted in this work is based on the formulation proposed in [1], and has the following structure:

$$\begin{cases} \dot{\boldsymbol{p}} = \boldsymbol{v} \\ \dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \\ \begin{bmatrix} m\boldsymbol{I}_3 & 0 \\ 0 & \boldsymbol{J} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{v}} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} -m[\boldsymbol{\omega}]_\times & 0 \\ 0 & [\boldsymbol{J}\boldsymbol{\omega}]_\times \end{bmatrix} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix} + \begin{bmatrix} m\boldsymbol{g} \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{bmatrix} + \begin{bmatrix} \Delta\boldsymbol{f} \\ \Delta\boldsymbol{\tau} \end{bmatrix} \end{cases} . \tag{1}$$

In this equations,

- $\boldsymbol{p} \in \mathbb{R}^3$, $\boldsymbol{v} \in \mathbb{R}^3$, and $\boldsymbol{\omega} \in \mathbb{R}^3$ denote respectively the UAV position, linear velocity, and angular velocity in the inertial frame.
- $\boldsymbol{q} \in \mathbb{R}^4$ is the unit quaternion representing the UAV attitude.
- $m \in \mathbb{R}$ is the UAV mass.
- $\boldsymbol{J} \in \mathbb{R}^{3\times3}$ is the inertia matrix.
- $\boldsymbol{g} = \begin{bmatrix} 0 & 0 & -g \end{bmatrix}^T$ is the gravity vector, with $g = 9.81\,\mathrm{m/s^2}$.
- $\boldsymbol{f} \in \mathbb{R}^3$ and $\boldsymbol{\tau} \in \mathbb{R}^3$ are the total force and torque vectors generated by the propellers and acting on the UAV.
- $\Delta\boldsymbol{f} \in \mathbb{R}^3$ and $\Delta\boldsymbol{\tau} \in \mathbb{R}^3$ are disturbances acting on the system.
- $[\cdot]_\times$ denotes the skew-symmetric operator.
- $\otimes$ denotes the quaternion multiplication.

To maintain consistency with the reference paper, the choice was made to use an ENU (East-North-Up) coordinate frame, which differs from the more common NED (North-East-Down) convention typically adopted in UAV literature.

The state of the system can be described by a vector $x \in \mathbb{R}^{13}$, defined as

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{v} \\ \boldsymbol{q} \\ \boldsymbol{\omega} \end{bmatrix}. \tag{2}$$

The evolution of the states is forced by an input vector $u \in \mathbb{R}^6$, defined as the vector of propellers induced forces and torques, as specified below:

$$u = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{bmatrix}. \tag{3}$$

However, the actual actuator commands are not directly the ones specified in (3), but rather the rotor angular velocities and the tilting angles for each propeller. Defining

- $\omega_{r,i} \in \mathbb{R}$ the angular velocity of propeller i,
- α_i the tilting angle of propeller i,
- $u_{\omega,i} = |\omega_{r,i}|\omega_{r,i} \in \mathbb{R}$ a controllable input related to the rotor velocity,
- $u_\omega = \begin{bmatrix} u_{\omega_1} & u_{\omega_2} & u_{\omega_3} & u_{\omega_4} \end{bmatrix}^T$,
- $s_i = \sin(\alpha_i), \quad c_i = \cos(\alpha_i)$,
- $c_f, c_m \in \mathbb{R}$ thrust constant and drag factor respectively,
- $l \in \mathbb{R}$ the arm length,

it is possible to state

$$\begin{bmatrix} f_x \\ f_y \\ f_z \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} 0 & -c_f s_2 & 0 & c_f s_4 \\ c_f s_1 & 0 & -c_f s_3 & 0 \\ -c_f c_1 & -c_f c_2 & -c_f c_3 & -c_f c_4 \\ 0 & -lc_f c_2 - c_m s_2 & 0 & lc_f c_4 + c_m s_4 \\ lc_f c_1 + c_m s_1 & 0 & -lc_f c_3 - c_m s_3 & 0 \\ lc_f s_1 - c_m c_1 & -lc_f s_2 + c_m c_2 & lc_f s_3 - c_m c_3 & -lc_f s_4 + c_m c_4 \end{bmatrix} \begin{bmatrix} u_{\omega 1} \\ u_{\omega 2} \\ u_{\omega 3} \\ u_{\omega 4} \end{bmatrix} = \begin{bmatrix} G_{q_f}(\alpha) \\ G_{q_\tau}(\alpha) \end{bmatrix} u_\omega. \tag{4}$$

where $\begin{bmatrix} G_{q_f}(\alpha) \\ G_{q_\tau}(\alpha) \end{bmatrix} = G_q(\alpha) \in \mathbb{R}^{6\times 4}$ is the allocation matrix of the system.

1.2 Aerodynamic Effects

When an UAV moves near boundaries such as the ground or a ceiling, the airflow generated by the propellers is altered, causing variations in the thrust. The ground effect increases thrust at low altitudes, improving stability and landing safety, while the ceiling effect acts when the UAV flies close to a ceiling, once again increasing thrust and pushing the robot even closer to the ceiling. This causes a reduction in the controllability of the system.
These phenomena are modeled through correction factors applied to thrusts. Define

- T_{IGE} thrust inside the ground effect,
- T_{OGE} thrust outside the ground effect,
- T_{ICE} thrust inside the ceiling effect,
- T_{OCE} thrust inside the ceiling effect,
- ρ the radius of the propeller,
- k_1, k_2 constants to tune.

The ground effect can be defined through the equation

$$\frac{T_{IGE}}{T_{OGE}} = \frac{1}{1 - (\frac{\rho}{4z})^2}, \tag{5}$$

whereas the ceiling effect follows the equation

$$\frac{T_{ICE}}{T_{OCE}} = \frac{1}{1 - \frac{1}{k_1}(\frac{\rho}{z+k_2})^2}. \tag{6}$$

While z in the ground effect is the robot altitude, in this case is the distance from the ceil. Both effects are considered only in the proximity of the corresponding surface, specifically when $z \leq 2\rho$.
Another aerodynamic effect accounted for is the wind: supposing to have a constant or time-varying wind in the room, its effect can be modeled as a disturbance in the control action Δf acting on the UAV.
In this work, all these effects are considered in order to have a more precise behavior, but also to study the controller's response to unmodeled dynamics. In fact, the controller is designed basing on the nominal UAV model in (1), which includes a disturbance term that needs to be rejected by the controller in order to achieve a good tracking. In this way, we can determine the robustness of the controller against such disturbances.

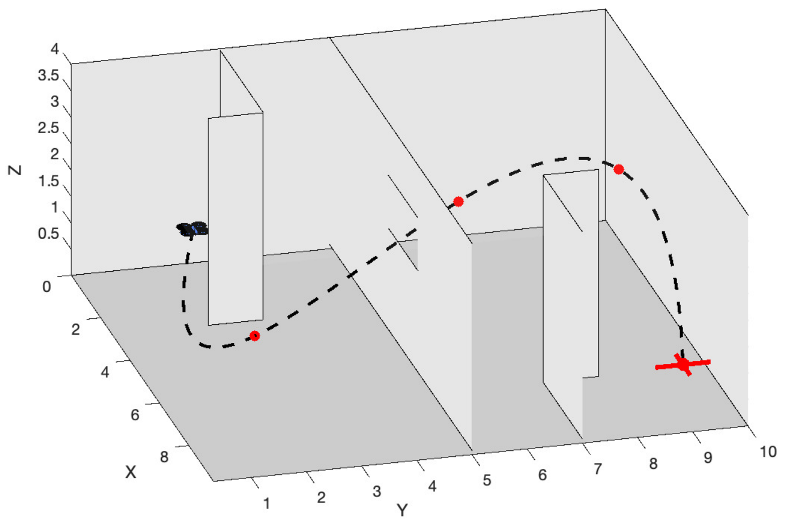


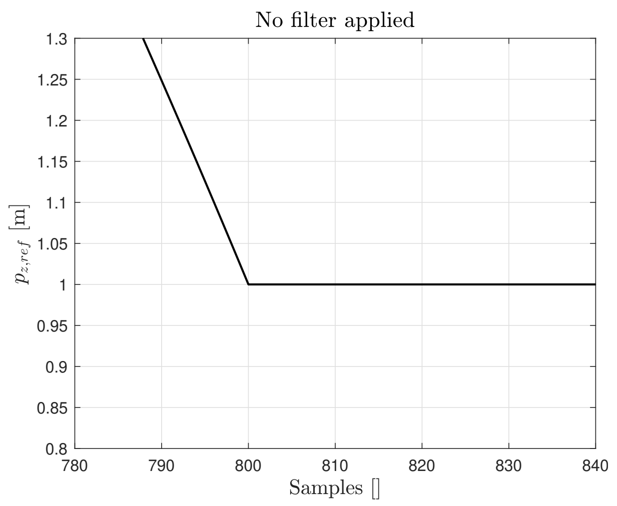
Figure 3: Generated trajectory.

1.3 Reference Trajectory

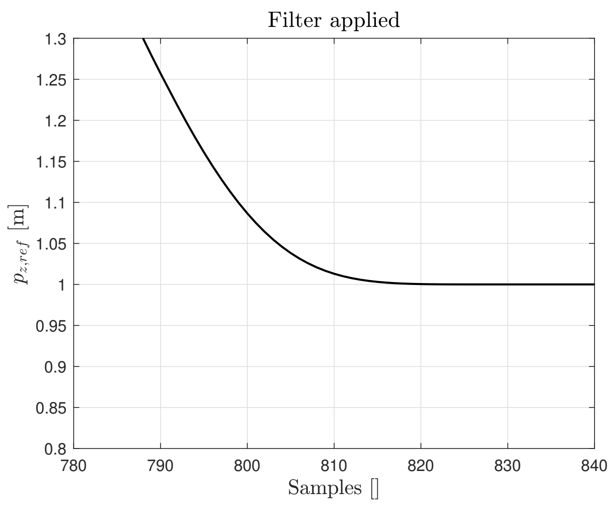
The objective of the vehicle is to navigate within an indoor environment, starting from a specified initial position and reaching a defined final point. To achieve this, the drone must follow a path that passes through a certain set of intermediate three-dimensional waypoints that are interpolated using splines. Referring to Figure 3, the waypoints are

- Initial point: $\begin{bmatrix}2.0 & 2.0 & 1.5\end{bmatrix}$,
- Surpass first wall and descend to test ground effect: $\begin{bmatrix}4.3 & 2.5 & 0.35\end{bmatrix}$,
- Pass through the central window: $\begin{bmatrix}5.0 & 6.0 & 2.8\end{bmatrix}$,
- Reach the far end of the room and ascend to test ceiling effect: $\begin{bmatrix}6.5 & 8.5 & 3.75\end{bmatrix}$,
- Final target point: $\begin{bmatrix}9.0 & 9.0 & 1.0\end{bmatrix}$.

By interpolating these waypoints, a desired position for the vehicle was defined at each simulation step.
The desired orientation is set to the identity quaternion (i.e., no rotation, chosen for simplicity) throughout the simulation. The desired position and orientation jointly define the reference pose that the controller must track. By computing the numerical derivatives of these reference signals, it is also possible to provide the corresponding reference linear and angular velocities.



(a) No filter applied.



(b) Filter applied.

Figure 4: Zoomed reference trajectory on p_z.

After 20 s of simulation, a dead interval is added. This interval is characterized by constant desired positions along all axes, which implies zero desired velocities and accelerations.
However, the direct introduction of this interval to the interpolated trajectory introduces abrupt transitions in the desired references, possibly causing problems in the actuators. For this reason, it has been decided to apply a low-pass filter to smooth the desired trajectory. In particular, a moving average filter was applied on the desired position signal, limited in the time interval where the abrupt variation occurs. The filter uses a window of $N = 15$ samples, and it is applied four times to improve its performance and emphasize the smoothing without excessively deforming the original trajectory.
Figure 4 shows the effect of the filter on p_z. As shown in Figure 4b, the filtered reference is significantly smoother, resulting in a slower and safer stop with respect to 4a.

2 Nonlinear Model Predictive Control

The Nonlinear Model Predictive Control (NMPC) is an advanced control strategy that uses a dynamic model of the system to predict its future behavior over a finite time horizon. At each time step, NMPC solves an optimization problem to determine the control inputs that minimize a cost function while satisfying constraints on states and inputs. It is also capable of tracking reference signals under disturbances, which makes it suitable for complex scenarios such as the control of an UAV flight. In this section, the controller's behavior and how it was used in the simulations will be analyzed.

2.1 Controller structure

An NMPC controller computes the optimal control action u^* at each time step using a model-based prediction of the states and an optimization, based on the problem constraints. Specifically, given a certain state x, the controller uses the dynamics of the system to predict the future trajectory for a specific time horizon. Next, it chooses an optimal sequence of control actions that minimizes a cost function, and after checking that the system constraints are not violated, applies only the first control action of the computed sequence to the system. This algorithm grants that the controller always adapts in case of disturbances. The cost function J can be defined as

$$J = \sum_{k=0}^{N-1} \left[(x_k - x_k^{\text{ref}})^T Q (x_k - x_k^{\text{ref}}) + u_k^T R u_k \right] + (x_N - x_N^{\text{ref}})^T P (x_N - x_N^{\text{ref}}), \tag{7}$$

where

- N is the prediction horizon,
- x_k and x_k^{ref} are the actual state and the reference state at step k,
- u_k is the control input at step k,
- Q, R and P are weight matrices to be properly tuned.

The first term in (7) penalizes the deviation of the predicted state from the reference trajectory, ensuring accurate tracking. The second term penalizes the control effort to avoid excessive commands, while the final term reduces the tracking error at the end of the prediction horizon. By appropriately tuning Q, R, and P, it is possible to achieve satisfying tracking performances. In particular, higher values of Q and P will prioritize an accurate but aggressive tracking, while higher values of R will lead to smoother commands but a less precise tracking. Another important characteristic of an NMPC controller is its ability to handle state and input constraints automatically. If certain variables are limited in specific bounds, the controller takes these limitations into account and adjusts its computations accordingly. This property will be exploited in the simulations presented in this work.

2.2 MATLAB Implementation

In the MATLAB environment, an NMPC controller can be defined thanks to the Model Predictive Control Toolbox, which provides a set of structures and functions that ease the implementation of this controller.
A controller object can be created with the function *nlmpc*, which takes as input the number of model states n_x, the number of the model outputs n_y and inputs n_u. Since in the examined case the planner provides a reference trajectory for each state in (2), one has

$$y = h(x) = x, \tag{8}$$

and thus

$$n_y = n_x = 13, \qquad n_u = 6. \tag{9}$$

Moreover, it is necessary to define a sampling time, in this case $T_s = 0.025\,\mathrm{s}$ to avoid computationally heavy simulations, and a prediction horizon, here chosen as 5. This means that at each time step the controller uses the five next steps of the trajectory to retrieve the optimal control input. The controller is then configured with a state function, in this case the one in 1, and an output function, which is in 8.
Here, the choice of representing the vehicle orientation with a unit quaternion becomes even more clear: the controller needs a state function of the form $\dot{x} = f(x, u)$, and the quaternion is the only choice that does not suffer from representation discontinuities or singularities and can be fit in a $n \times 1$ state vector.
The controller also requires to be informed about the model parameters. Such parameters are the ones appearing in the dynamic model of the vehicle and in the allocation matrix. They are defined as follows:

$$\begin{gathered} m = 1.32 \text{ kg}, \\ J = \begin{bmatrix} 0.0154 & 0 & 0 \\ 0 & 0.0154 & 0 \\ 0 & 0 & 0.0263 \end{bmatrix} \text{kg} \cdot \text{m}^2, \\ c_f = 1.65776 \cdot 10^{-5} \text{N}/(\text{rad/s})^2, \\ c_m = 2.1792 \cdot 10^{-5} \text{N} \cdot \text{m}/(\text{rad/s})^2, \\ l = 0.45 \text{ m}. \end{gathered} \tag{10}$$

The next thing to do is to define state constraints. For this problem, only position constraints are defined: given the dimensions of the room in Figure 2, the following constraints are set:

$$\begin{aligned} 0 \le p_x &\le 10, \\ 0 \le p_y &\le 10, \\ 0 \le p_z &\le 4. \end{aligned} \tag{11}$$

Regarding the weight matrices, the final step weight matrix P is not considered, so only $Q \in \mathbb{R}^{1\times n_x}$ and $R \in \mathbb{R}^{1\times n_u}$ are defined. These are row vectors, where each element weights a specific state or control input in the cost function. In this case, more importance is given to the pose, namely, position and orientation, especially on the x-y plane, compared to linear and angular velocities. To ensure good tracking performance, the elements of Q are chosen significantly higher than those in R.
Moreover, the weight associated with the z-axis position (which is 10) is deliberately lower than those of x and y (set to 40). This choice is made to make vertical deviations more noticeable in the simulation, especially to highlight the effects of ground and ceiling interactions more clearly from a visual standpoint.
The resulting matrices are:

$$\begin{aligned} Q &= \begin{bmatrix} 40 & 40 & 10 & 10 & 10 & 10 & 15 & 15 & 15 & 15 & 5 & 5 & 5 \end{bmatrix}, \\ R &= \begin{bmatrix} 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix}. \end{aligned} \tag{12}$$

The initial conditions both in the state and in the control input are also defined. In the state, all the initial conditions are null, except in the position, where

$$\boldsymbol{p}_0 = \begin{bmatrix} 2.0 & 2.0 & 1.5 \end{bmatrix}^T \tag{13}$$

which corresponds to the first waypoint. For the control input, the initial condition is

$$u_0 = \begin{bmatrix} -m\boldsymbol{g}^T & 0 & 0 & 0 \end{bmatrix}^T \tag{14}$$

which represents the initial wrench that must be applied to the drone to make it hover in its initial position and prevent it from falling.

2.3 Simulink Implementation

Once the controller object is fully defined in MATLAB, a Nonlinear MPC Controller block must be added in Simulink and linked to the previously defined controller object. The controller takes four inputs:

- x: current state vector, obtained from the model output,
- ref: reference vector, generated by the planner,

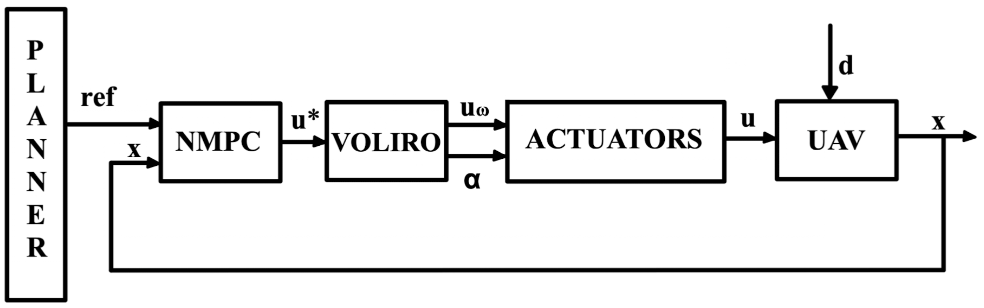


Figure 5: Diagram of control logic.

- *last_mv*: control input applied at the previous time step,
- *params*: bus containing all parameters defined in (10).

The output is a vector MV (Manipulated Variables) containing the optimal control action, minimizing the cost function in (7).
The actuator commands are obtained using the Voliro approach: the idea is to perform a variable transformation decomposing the value of $u_{\omega,i}$ in a vertical component $u_{v,i}$ and a lateral component $u_{l,i}$, defined as

$$u_{v,i} = c_f u_{\omega,i} c_i, \qquad u_{l,i} = c_f u_{\omega,i} s_i, \qquad \forall i = 1...4. \tag{15}$$

The relationship between the wrench and this set of coordinates is

$$\begin{bmatrix} f_x \\ f_y \\ f_z \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 \\ 0 & 0 & -l & -c_m/c_f & 0 & 0 & l & c_m/c_f \\ l & c_m/c_f & 0 & 0 & -l & -c_m/c_f & 0 & 0 \\ -c_m/c_f & l & c_m/c_f & -l & -c_m/c_f & l & c_m/c_f & -l \end{bmatrix} \begin{bmatrix} u_{v,1} \\ u_{l,1} \\ u_{v,2} \\ u_{l,2} \\ u_{v,3} \\ u_{l,3} \\ u_{v,4} \\ u_{l,4} \end{bmatrix}. \tag{16}$$

In matrix form, the previous equation becomes

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{bmatrix} = \boldsymbol{G}_{q,static} \boldsymbol{u}(\alpha, u_\omega). \tag{17}$$

Hence, the allocation matrix $G_{q,\text{static}}$ is now constant. It is sufficient to invert such matrix to obtain $\boldsymbol{u}(\alpha, u_\omega)$ from the wrench vector, provided by the controller. Inverting (15), it is possible to compute the actuator commands through

$$u_{\omega,i} = \frac{1}{c_f}\sqrt{u_{l,i}^2 + u_{v,i}^2}, \qquad \alpha_i = atan2(u_{l,i}, u_{v,i}) \qquad \forall i = 1...4. \tag{18}$$

The computed commands are then sent to the robot, and by applying (4), and (5) and (6) when the vehicle is close to the ground or the ceiling, the actual wrench coming from the propellers and acting on the UAV is obtained and provided as input to the dynamic model in (1). The coefficients k_1 and k_2 appearing in (6) are set to

$$k_1 = 6.924, \qquad k_2 = 0.03782, \tag{19}$$

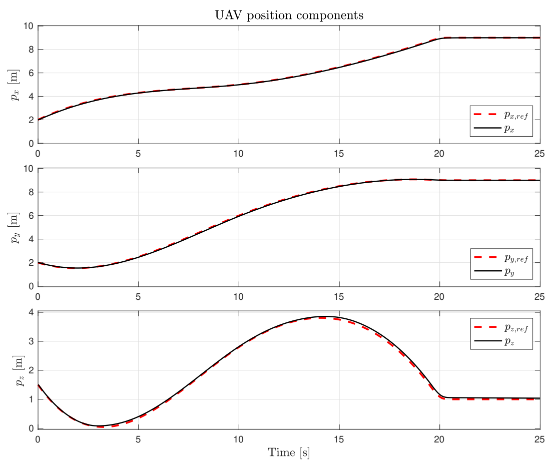
as suggested in [2].
In the end, a disturbance to the drone is applied. This disturbance can either be constant in the xy-plane, which represents a constant wind in the room, in which case

$$\boldsymbol{d} = \begin{bmatrix} 2.5 & 2.5 & 0 \end{bmatrix}^T \text{ N}, \tag{20}$$

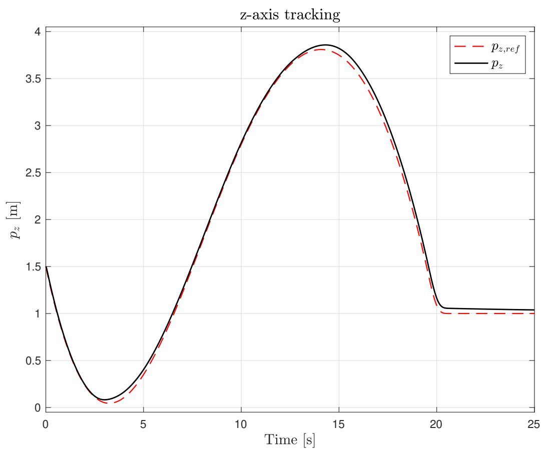
or stochastic, modeled as a Gaussian random variable with

$$\mu = 0, \qquad \sigma = 0.2. \tag{21}$$

This kind of disturbance emulates a random gust of wind, where the values of μ and σ were selected to represent moderate turbulence while ensuring a realistic disturbance magnitude. Figure 5 summarizes the control logic: the reference and the state are used by the controller to compute the optimal control action, expressed as a wrench. Then, exploiting the Voliro approach, this wrench is converted into the desired actuator commands and sent to the actuators. They generate the actual wrench which, combined with external disturbances, is applied to the UAV, resulting in the evolution of the state vector.



(a) Trajectory tracking along x, y, z-axes.



(b) Trajectory tracking along z-axis.

Figure 6: Position tracking.

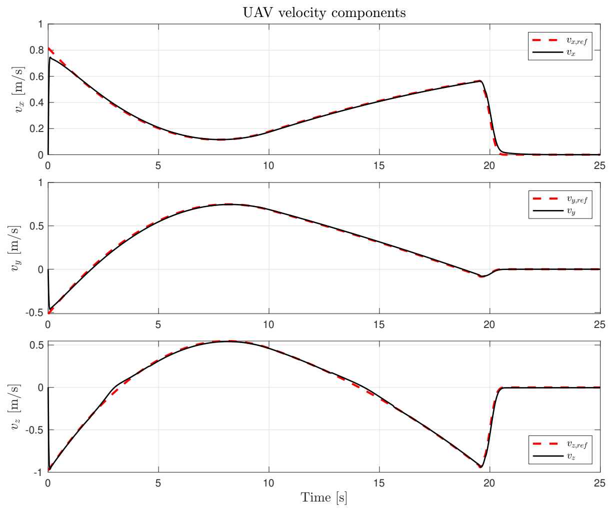
3 Simulation Results

After completing the Simulink model, the controller was applied in several scenarios to test its performance and robustness. In this chapter, the results of the simulations are presented, focusing on the accuracy in tracking the reference trajectory, and the response of the NMPC to various types of perturbations.

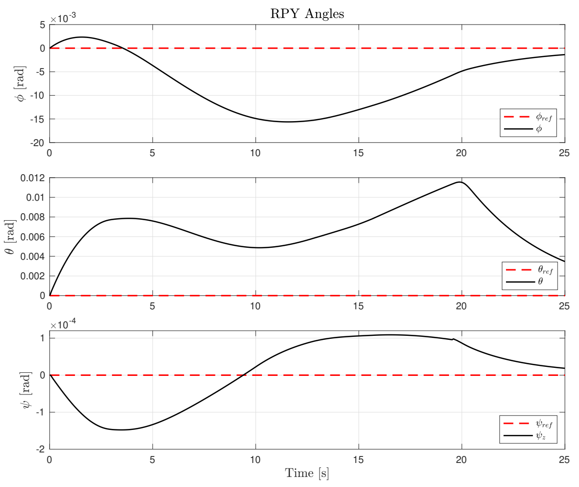
This section focuses on the analysis of the time evolution of the system states and their respective references, the comparison between commanded and actual forces and torques, and the actuator inputs. The results are examined both in nominal conditions and in the presence of external disturbances.

In Figure 6 the position tracking performance of the controller in the absence of disturbances is illustrated. As it can be seen, the desired trajectory is correctly tracked by the controller across all components. Figure 6b shows the tracking along the z-axis only, which allows for a clearer observation of the effects of ground and ceiling interactions. In fact, given our settings, they should be noticeable when $z < 0.28$ m or $z > 3.72$ m, and this happens around 3 s and 15 s, respectively. In this instants of simulation, the thrust increases, and the position of the vehicle along the z-axis increases as well. Nevertheless, the controller demonstrates a satisfactory disturbance rejection capability, since the deviation of the trajectory from the reference is corrected in a few seconds or less.

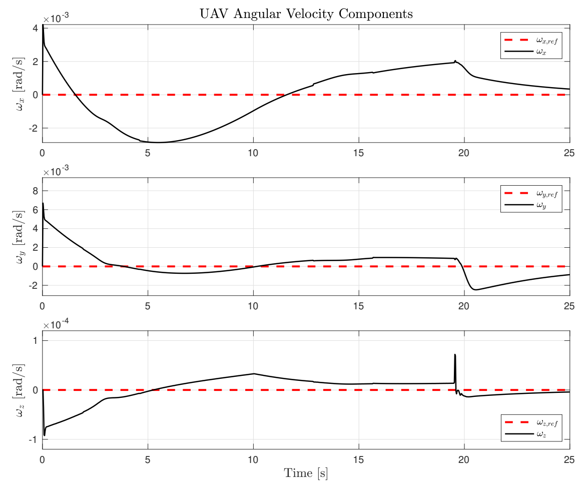
Figure 7 summarizes the tracking performance in terms of velocity, orientation, and angular velocity. In particular, Figure 7a shows the velocity components. The reference is accurately followed, with noticeable deviations on z-axis caused by ground and ceiling interactions. These appear during the same time intervals discussed earlier, where the velocity momentarily exceeds the reference before settling back. At the end of the trajectory, the dead interval is clearly visible, during which all velocity components converge to zero.



(a) Velocity tracking components.



(b) Orientation tracking (Euler angles).



(c) Angular velocity tracking.

Figure 7: Tracking performance in velocity, orientation, and angular velocity.

Figure 7b displays the orientation tracking in terms of Euler angles, obtained by converting the quaternion representation. Euler angles were chosen for their clearer interpretability, and are expressed in the XYZ convention. The controller performs well here too, as all references are set to zero and the actual angles remain close to the origin throughout the simulation.

Finally, Figure 7c shows the angular velocity components. As in the orientation case, all references are null. The angular velocities remain sufficiently small, indicating that rotational motion is effectively suppressed and the controller maintains stability during the entire maneuver.

Figure 8 reports the norm of the position error. Although it remains very small, approximately 8 cm at the beginning of the dead interval and decreasing over time, it does not

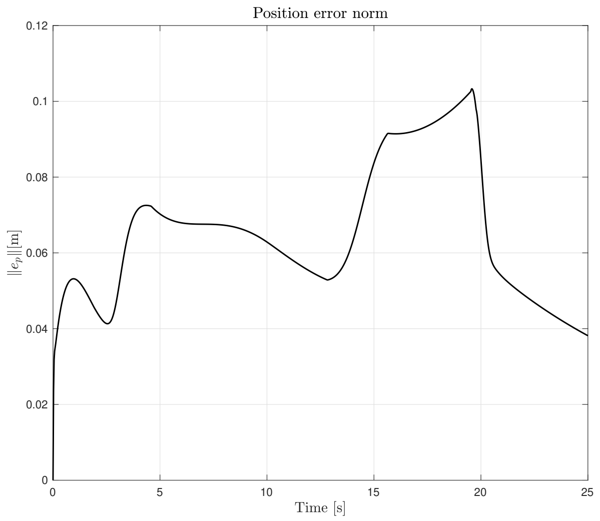


Figure 8: Position error norm.

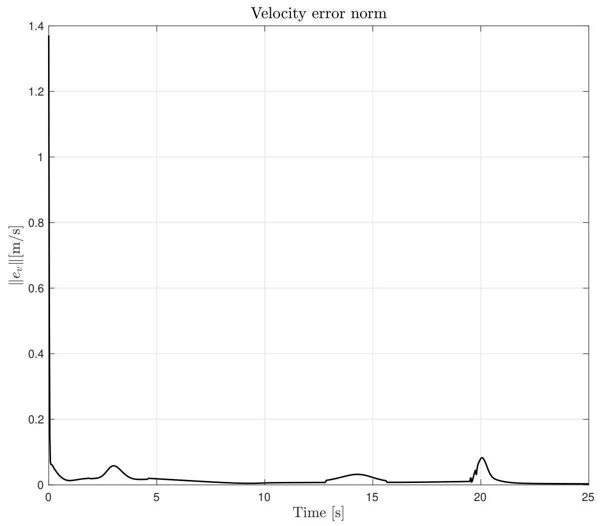


Figure 9: Velocity error norm.

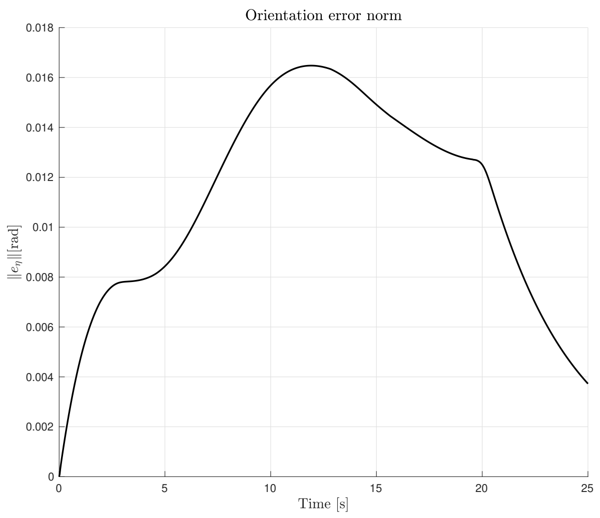


Figure 10: Orientation error norm.

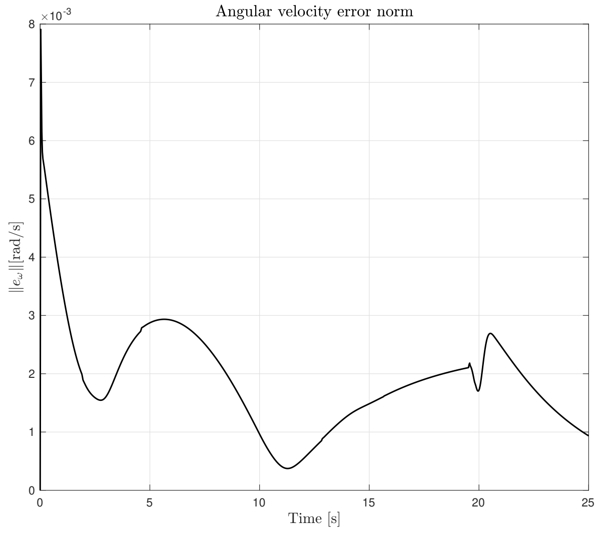
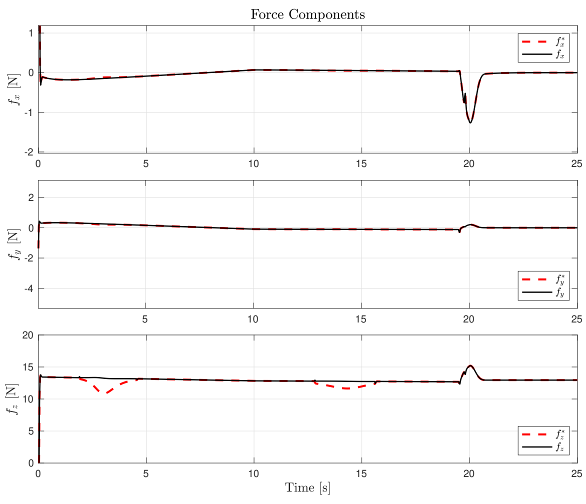


Figure 11: Angular velocity error norm.

converge to zero, or at least not instantaneously. This is because in general, NMPC does not guarantee zero steady-state error unless specific integral terms or disturbance observers, such as Kalman filters, are introduced into the control scheme. Since this implementation of the control scheme is a simple one, the error is not null, but still remains low.
Figure 9 confirms the overall effectiveness of the controller: the norm of the linear velocity error remains consistently low over the entire simulation and tends to zero. Slight increases in the error are observed during phases where the UAV interacts with the ground or ceiling, or when it comes to a stop at a fixed waypoint. Figure 10 illustrates the orientation error norm, which stays small throughout the simulation and progressively decreases towards the end. Similarly, Figure 11 reports the norm of the angular velocity error, which remains low and exhibits no significant peaks. These results further confirm the high accuracy of the


(a) Optimal and actual forces.

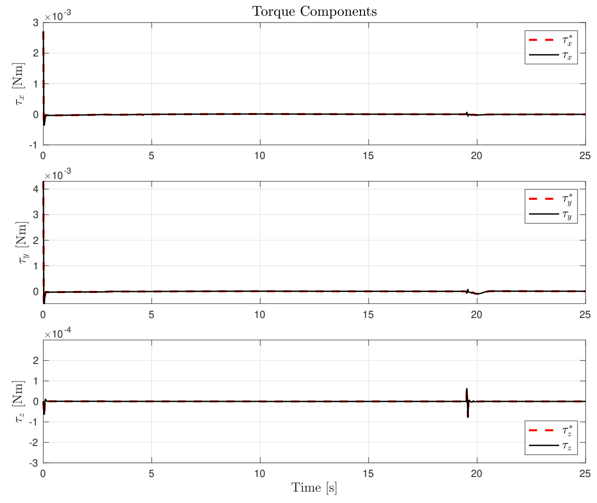

(b) Optimal and actual torques.

Figure 12: Control inputs.

tracking performance.

As previously stated, the controller provides an optimal control input, which is different from the actual one. Figure 12 presents a comparison between the optimal and actual forces and torques.

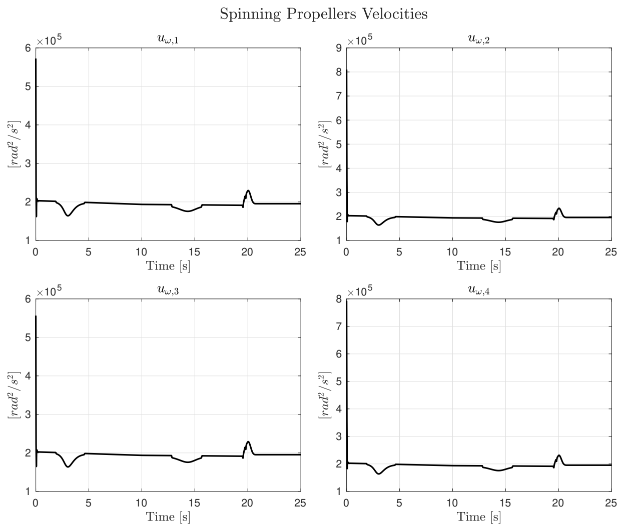
The desired actuator commands exhibit good fidelity: the force components along the x- and y-axes remain very small, while along the z-axis we have an almost constant force, with some oscillations in the optimal force due to the correction of aerodynamic effects.

Regarding the torques, the values are even smaller. Their magnitudes remain low, and as before, the optimal torques are almost null, as the UAV is not required to tilt. A small oscillation is present in τ_z before 20 s, corresponding to the stopping phase of the vehicle.

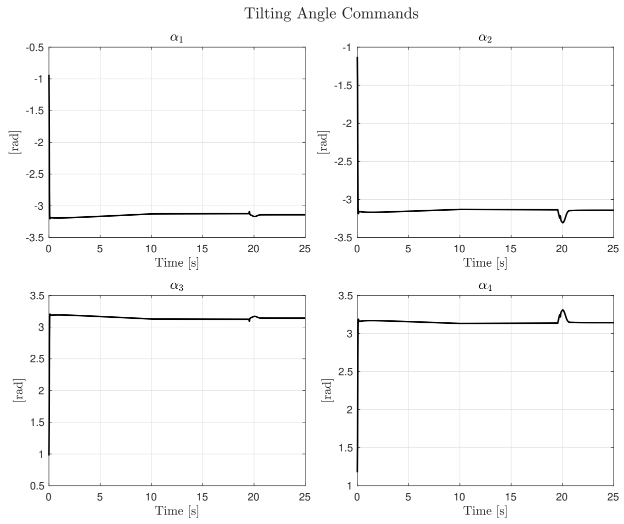
Figure 13a shows the evolution of the spinning commands u_ω. Their high magnitude is justified by the fact that these variables correspond to the square of the actual rotor angular velocities $\omega_{r,i}$. Their behavior remains nearly constant, overall, with variations occurring during the ground effect, ceiling effect, and stopping phase.

Regarding the tilting angle commands, Figure 13b illustrates their behavior. The angles rapidly settle to steady-state values around $-\pi$ for propellers 1 and 2, and π for propellers 3 and 4. As we expected, the opposite propellers in Figure 1 share the same tilting angle. The fact that they tend to $\pm\pi$, is a consequence of the constant orientation of the vehicle. Now, noticeable oscillations are present only during the stopping phase of the drone.

In this video we can observe the complete 3D simulation in the nominal case. The vehicle follows the desired trajectory without changing its attitude and stops on the target at the other side of the room. As a safety warning, when the robot gets close to a surface, in particular when the distance to the surface is less than a safety distance, its mesh turns red. Also, as shown in this video, another safety measurement was applied: if the robot hits any surface, the simulation automatically stops.



(a) Spinning propellers velocities.



(b) Tilting angle commands.

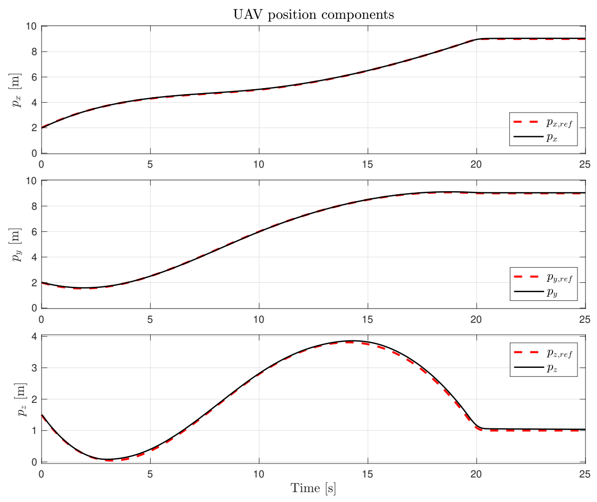
Figure 13: Control inputs for the drone.

3.1 Robustness Analysis

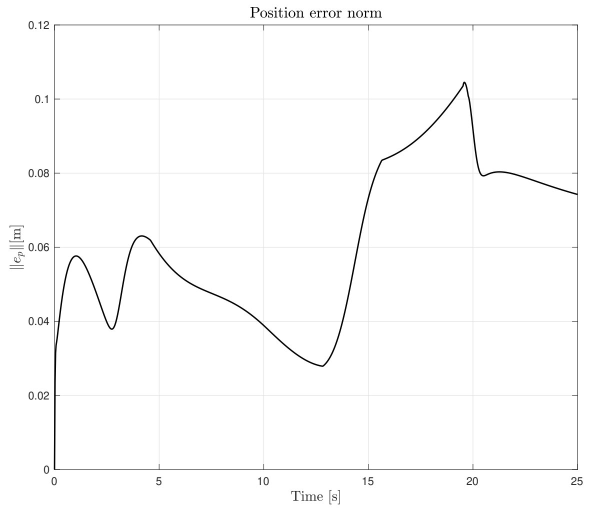
This section analyzes the robustness of the controller with respect to external forces. While the effects of ground and ceiling interactions on tracking performance have already been discussed, the focus now shifts to the influence of other aerodynamic factors. Specifically, a constant wind-like disturbance is first considered, as introduced in (20), followed by the analysis of a stochastic noise, as described in (21).

3.1.1 Constant Disturbance

In case of a constant disturbance applied to the UAV as an external force, the resulting position tracking performance is presented in Figure 14.
It can be noticed that the tracking remains highly accurate, as there are no evident errors in following the reference. This is confirmed by the error norm plot, where the error, although being slightly higher than in in the non-disturbed case in Figure 8, remains low, as it peaks at 10 cm, and at the end of the simulation the residual error is equal to 8 cm and still decreasing in time. This is made possible thanks to the prevision of the trajectory by the controller.
The same can be said regarding the velocity tracking: the plots in Figure 15 show an almost identical velocity tracking with respect to the non-disturbed case in Figures 7a and 9. For this reason, we can conclude that the NMPC effectively rejects dynamic disturbances, maintaining the desired velocity profile despite the presence of external forces.

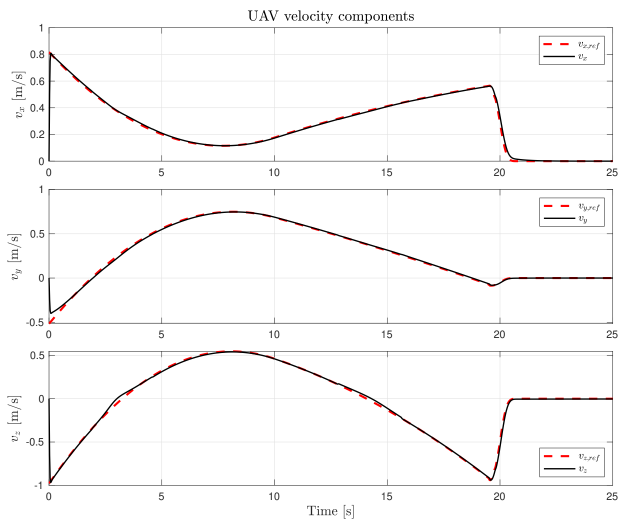


(a) Disturbed position tracking.

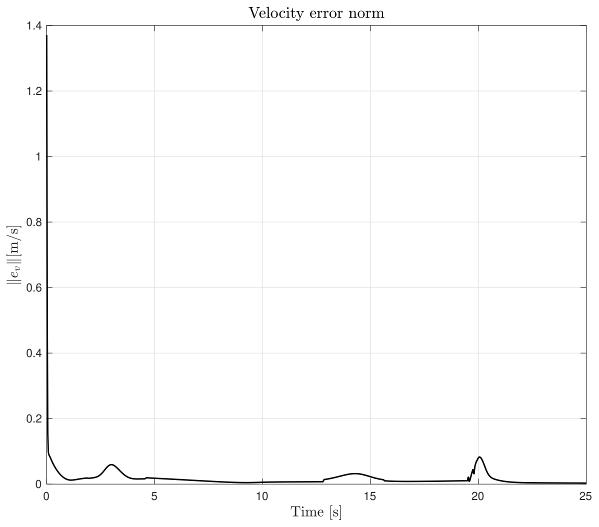


(b) Disturbed position error norm.

Figure 14: Disturbed tracking - Position.



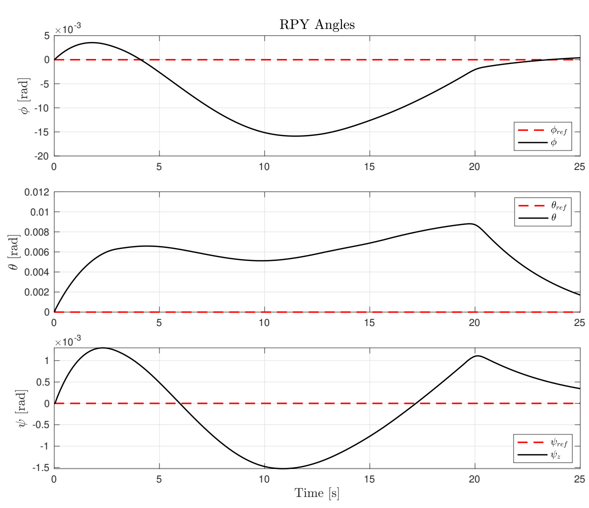
(a) Disturbed velocity tracking.



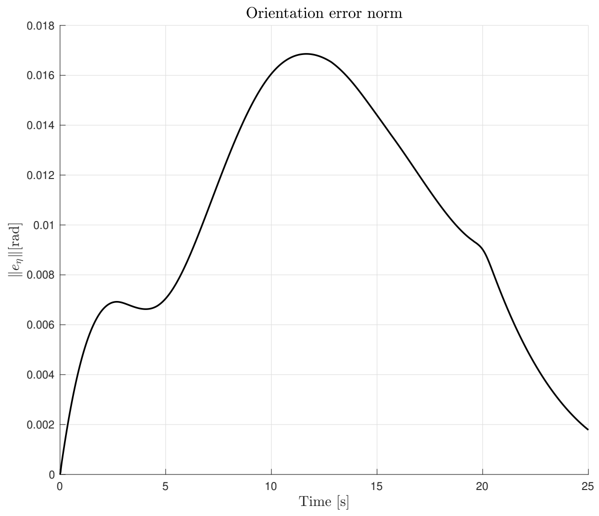
(b) Disturbed velocity error norm.

Figure 15: Disturbed tracking - Velocity.

Figure 16 illustrates the orientation tracking and the corresponding error norm. A clear difference can be observed when compared to the undisturbed case: in Figure 16a, the yaw angle exhibits an opposite behavior with respect to that shown in Figure 7b. Initially, it increases and then decreases, becoming negative. This behavior is the result of the NMPC strategy, which relies on a prediction horizon to anticipate the system's response. In the presence of external disturbances, the controller anticipates that strictly following the original yaw trajectory would result in higher overall tracking errors. Consequently, it modifies the yaw angle to minimize the cost function over the horizon, leading to a better compromise. Moreover, this adaptation comes with a moderate increase in the order of magnitude of the

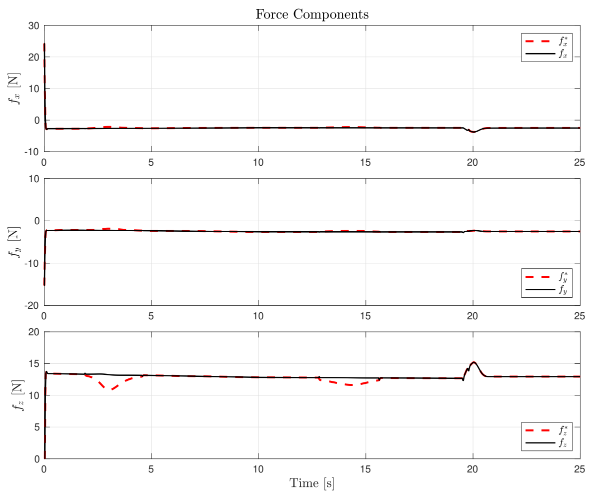


(a) Disturbed orientation tracking.

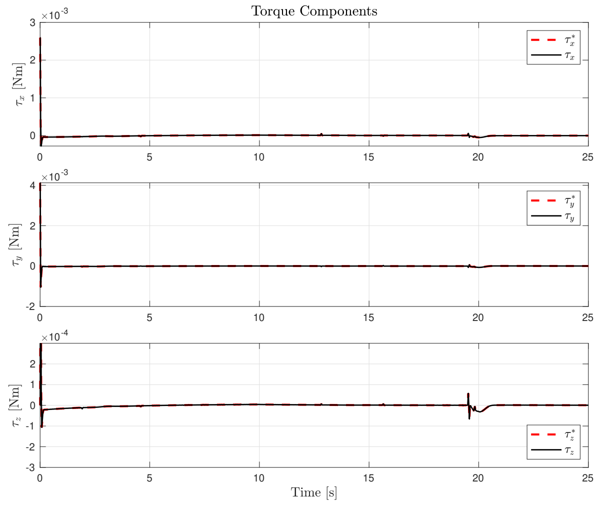


(b) Disturbed orientation error norm.

Figure 16: Disturbed tracking - Orientation.



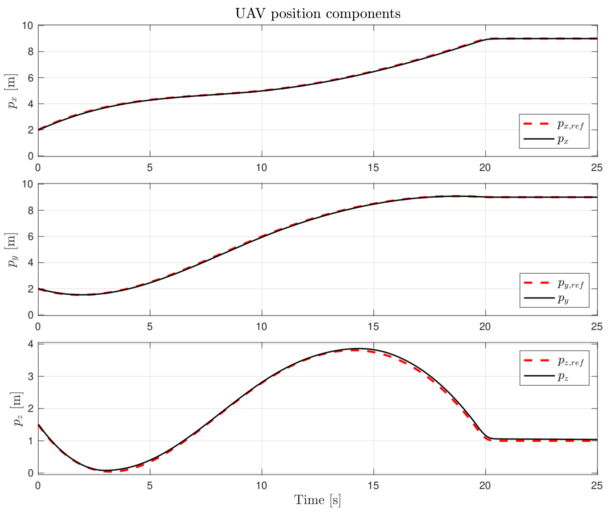
(a) Disturbed forces.



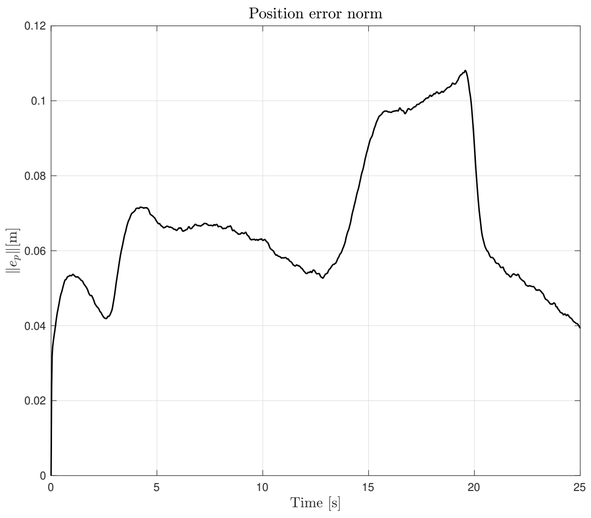
(b) Disturbed torques.

Figure 17: Disturbed control inputs.

yaw trajectory, though the values remain acceptably low. Finally, Figure 16b shows that the orientation error norm is actually lower than in the undisturbed case: at the end of the simulation, the error decreases to about 0.002 rad, compared to 0.004 rad previously.
Another effect of the constant disturb can be observed in Figure 17, where the control inputs are shown. In Figure 17b, the torques remain almost unchanged compared to the non-disturbed case. However, a major difference is visible in Figure 17a, where we can observe that the forces on x-axis and y-axis both tend to -2.5 N, which is the opposite value of the disturb on such axes. This behavior confirms that the controller is informed about the variation of the state from the desired trajectory, and automatically outputs a desired

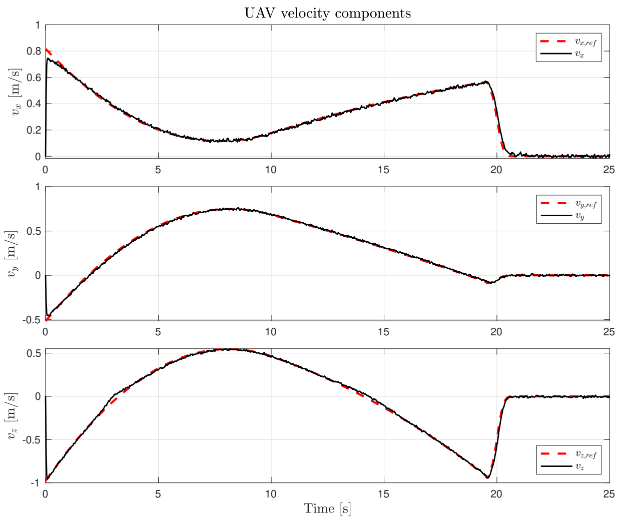


(a) Noisy position tracking.

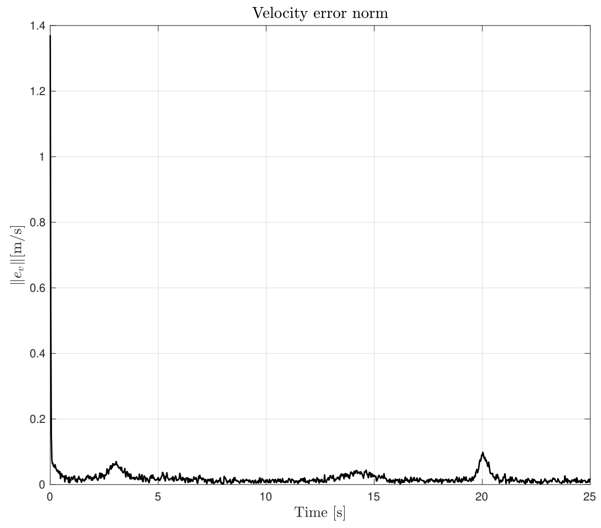


(b) Noisy position error norm.

Figure 18: Noisy tracking - Position.



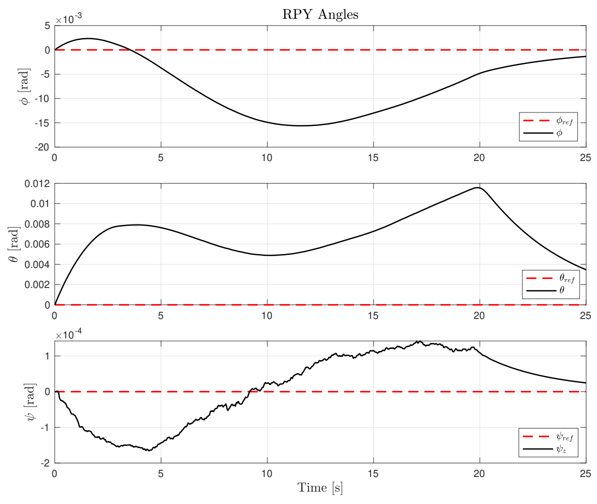
(a) Noisy velocity tracking.



(b) Noisy velocity error norm.

Figure 19: Noisy tracking - Velocity.

force to contrast the disturbance and make the UAV follow the trajectory.
After analyzing these plots, we can conclude that the overall behavior of the controller is robust against constant disturbances. As a further confirm of this, here we have the video containing the final 3D simulation. The path is the same followed in the previous case, even if there is a constant disturbance.


(a) Noisy orientation tracking.

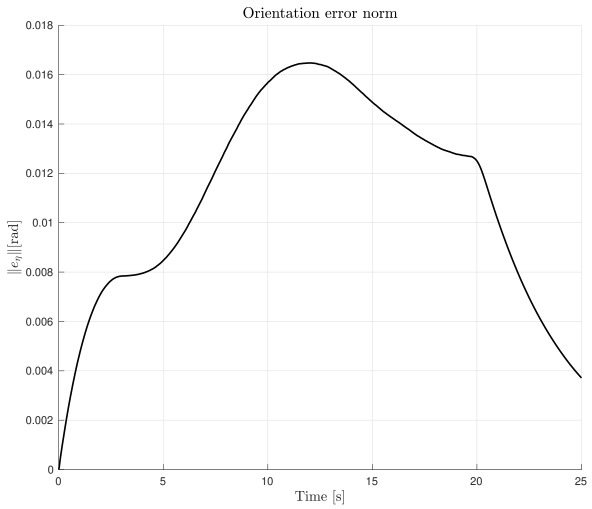

(b) Noisy orientation error norm.

Figure 20: Noisy tracking - Orientation.

3.1.2 Stochastic Noise

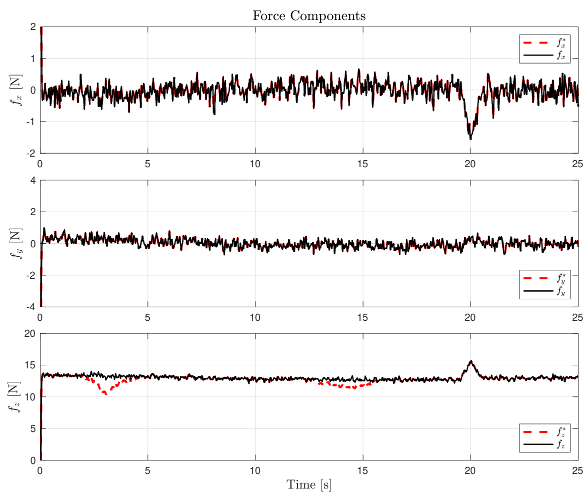
As anticipated, the controller's robustness is also tested under the effect of stochastic noise. The random disturbance is applied independently on the three spatial axes, simulating unpredictable environmental effects.

Figure 18 shows the position tracking performance in this scenario: Figure 18a once again exhibits good tracking performances over all the simulation, whereas Figure 18b portraits a low behavior of the error norm. This time, the behavior is more oscillating, due to the presence of uncertainties, but the norm still values a few centimeters (4 cm in this case) with a descending trend at the end of the simulation.

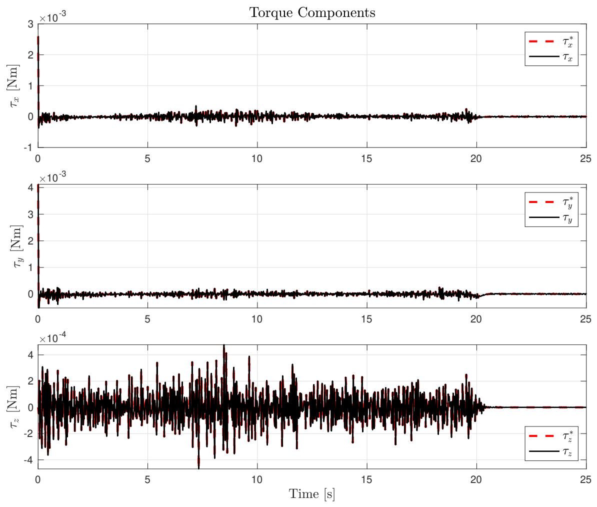
Figure 19 presents the velocity tracking and corresponding error norm. As observed in Figure 19a, the velocity tracking shows good accuracy, but with a much more evident noise in the response on all axes, proving that the dynamic behavior of the UAV is more sensitive to high-frequency perturbations than low-frequency ones, as it was shown in Figure 15a. The error norm in 19b is also satisfying, since such error approaches the zero, but also here, we can see a noisy and oscillating behavior, confirming our previous result about the controller's sensitivity.

In Figure 20, it is possible to notice the behavior of the Euler angles: the roll and pitch angles in Figure 20a have the same smooth trend, whereas the yaw angle has a more oscillating behavior, that is much more similar in shape and magnitude to the original yaw in 7b. Also the error norm in Figure 20b is more similar to the one in the first case, having the same steady-state value at 0.004 rad.

Finally, Figure 21 shows both the optimal forces and torques computed by the controller and the actual ones applied to the UAV. In this case, it is evident that the commanded controls themselves appear noisy. This behavior is expected, as the controller detects the



(a) Noisy forces.



(b) Noisy torques.

Figure 21: Noisy control inputs.

state deviations caused by high-frequency disturbances and proactively adjusts the control inputs to keep the UAV on the desired trajectory.

In conclusion, the NMPC controller demonstrates robust performance not only under constant disturbances but also in the presence of random noise, maintaining accurate tracking even in a highly perturbed environment.

In this video we can see the drone animation in this third case: the path is smooth, and the robot correctly follows all the waypoints, once again confirming the correctness of the algorithms.

Conclusions

This project focused on the design and simulation of a nonlinear model predictive controller for a tilting quadrotor. We started by defining the problem, which is an indoor three-dimensional navigation, and the dynamics of the robot. Aerodynamical effects such as wind, ground effect and ceiling effect were considered, not only to make the simulation more realistic, but also to test the disturbance rejection capabilities of the examined controller.
The implementation of the controller required a dynamic model of the UAV that represents its orientation using quaternion-based notation.
The results showed how the prediction capability of the NMPC controller was essential firstly to minimize a cost related to errors and control inputs, but also to check eventual detachments from the reference trajectory, being able to supply a control action always capable of rejecting any kind of disturbance.
This project allowed us to better understand concepts such as the Voliro approach, which was used to give desired commands to the actuators, or the NMPC controller itself, that we never had the opportunity to study so far.
Future developments could concern the integration of optimal state observers, such as Kalman filters, or adaptive elements to further reduce steady-state errors and improve performance in more complex environments, or also to implement a better obstacle avoidance or a battery handler, even on a real hardware.

References

[1] K. B. Maximilian Brunner and R. Siegwart, "Trajectory tracking nonlinear model predictive control for an overactuated mav," May–August 2020.

[2] P. J. Sanchez-Cuevas, P. Ramon-Soria, B. Arrue, A. Ollero, and G. Heredia, "Robotic system for inspection by contact of bridge beams using uavs," *Sensors*, vol. 19, no. 2, p. 305, 2019.