DIETI UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL'INFORMAZIONE

Automation and Robotics Engineering

Field and Service Robotics Course

Report – Homework 3

Student: Salvatore Granata - P38000219

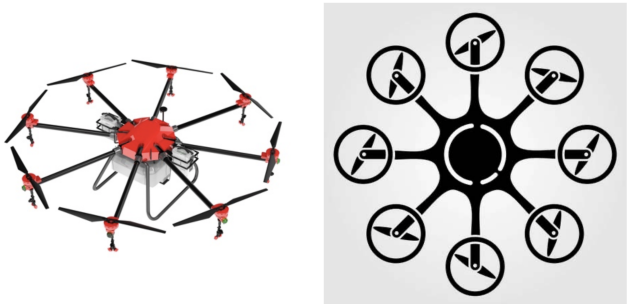
Abstract

This report documents the development of Homework 3, written in LaTeX, where all contents and code were created and written by Salvatore Granata with conceptual organization improvements made thanks to AI tools. It shows the solutions to the proposed exercises concerning application of theoretical knowledge related about aerial robotics, such as drone configuration, allocation matrix, types of control concerning UAV and their application.

The GitHub repository cointaining the report and the all the code sources of the developed homework can be found at the following link: https://github.com/Salvatoregr/homework3_FSR.git.

Contents

1 Exercise 1 - Octocopter



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

1.1 Degrees of Freedom and Configuration Space

The computation of the degrees of freedom involves the consideration of 2 different situations, in which it's possible to apply the Grübler's Formula.

Body Fixed on the ground

- $m = 6$ because it's a spatial congiuration.
- 8 rotating propellers (1 link plus 1 rotational joint each).
- 1 ground to be considered.

$$DoF = m(N - 1 - J) + \sum_{j=1}^{J} f_i = 6(8 + 1 - 1 - 8) + 8 = 8 \text{ degrees of freedom}$$

Flying body

If the drone is flying, 6 more DoFs need to be added given by the fact that the robot is moving in the space.

$$DoF = 8 + 6 = 14 \text{ degrees of freedom}$$

1.2 Underactuation and Fully-actuation

The robot, being in a *flat* configuration, is always *underactuated*, because there are 8 propellers, but they are all coplanar (all displaced at the same configuration) and by varying them all simultaneously, it's possible only to lift up and down, while changing the rotation of each propeller, the result is a change of inclination. Hence it's not possible to obtain an acceleration that is directed in any direction without rotating the drone. So it's not possible to express an acceleration in any direction. It means that the allocation matrix G is not full rank and the system is underactuated.

1.3 Allocation Matrix

The aim of the control of an UAV is the generation over the time the right control input, expressed as a force f and a torque τ, to achieve the goal. Although, it's important to underline that the real system inputs are the spinning velocity for each propeller ω_i . So in the end, the main target

to move the UAV within the space is to apply a rotation velocity to each propeller.

For this reason, since the controller can only generate torques and forces as references, there's the need to have a matrix that expresses this mapping. This matrix is called *allocation* matrix.

So, the first thing to do is to set the right references to the octocopter model and compute the controller forces and torques.

A UAV is made of propellers, whereas each one produces a thrust $T_i > 0$. In the case of an octocopter, $i = 1, ..., 8$, the total thrust is:

$$u_T = \sum_{i=1}^{6} T_i \rightarrow u_T = T_1 + T_2 + T_3 + T_4 + T_5 + T_6 + T_7 + T_8$$

This total thrust is directed along the z_b-axis of the body frame to compensate for gravity and control the vertical motion. The horizontal movements are then controlled by directing the total thrust vector $u_T z_b$ in the appropriate direction (thrust vectoring control), thereby resulting in longitudinal and lateral force components.

In a general case, for an UAV, the control force for model based purposes is f_x, f_y, f_z, but for a flat configuration (including the octocopter's one), it's sufficient only to consider the third element $[0\ 0\ -u_T]$. So it's possible to define the thrust force as $f^b = [0\ 0\ -u_T]^T$. Regarding the torques instead, it's possible to define the three components as $[\tau_x, \tau_y, \tau_z]$ In this case it's possible to actuate every τ along each axis, because there's the possibility to give different velocity to each of the propeller.

With reference to the figure, the torques generated by the single propellers thrust are:

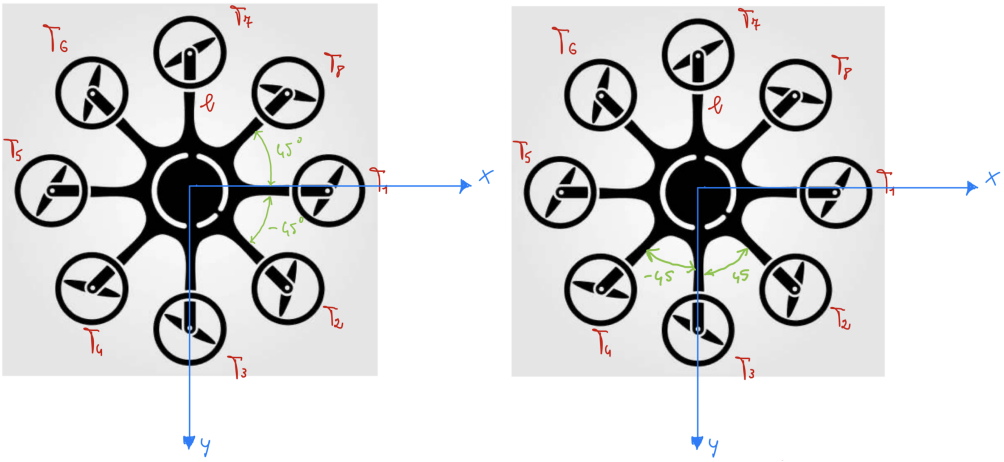
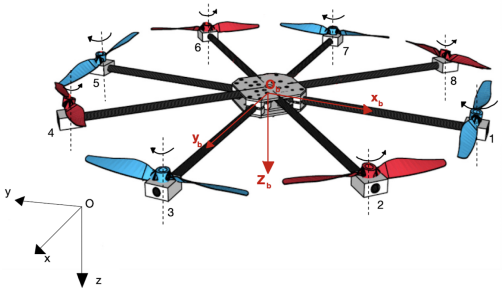


Figure: on the left the reference system for τ_x on the right the reference system for τ_y

$$\tau_x = l(T_7 - T3) + lT_2 \sin(-45°) + lT_6 sin(135°) + lT_4 \sin(-135°) + lT_8 \sin(45°)$$
$$\tau_x = l(T_7 - T_3) + l\frac{\sqrt{2}}{2}(T_6 - T_2) + l\frac{\sqrt{2}}{2}(T_8 - T_4)$$

$$\tau_y = l(T_1 - T_5) + lT_2 \sin(45°) + lT_6 \sin(-135°) + lT_8 sin(135°) + lT_4 sin(-45°)$$
$$\tau_y = l(T_1 - T_5) + l\frac{\sqrt{2}}{2}(T_2 - T_6) + l\frac{\sqrt{2}}{2}(T_8 - T_4)$$

The torque along the z-axis is given by the composition of all the fan torques Q_i due to air drag:



$$\tau_z = -Q_1 + Q_2 - Q_3 + Q_4 - Q_5 + Q_6 - Q_7 + Q_8$$

In the end, the number of control inputs is only 4, which are $\tau_x \tau_y \tau_z u_T$.

$$\begin{aligned} u_T &= T_1 + T_2 + T_3 + T_4 + T_5 + T_6 + T_7 + T_8 \\ \tau_x &= l(T_7 - T_3) + l\frac{\sqrt{2}}{2}(T_6 - T_2) + l\frac{\sqrt{2}}{2}(T_8 - T_4) \\ \tau_y &= l(T_1 - T_5) + l\frac{\sqrt{2}}{2}(T_2 - T_6) + l\frac{\sqrt{2}}{2}(T_8 - T_4) \\ \tau_z &= -Q_1 + Q_2 - Q_3 + Q_4 - Q_5 + Q_6 - Q_7 + Q_8 \end{aligned}$$

For model purposes, these 4 virtual controls which are way easier to handle.

There's now the necessity to connect the control inputs with the real inputs of the system. To this purpose, it's possible to neglect the parasitic forces and moments induced by control inputs coupling, rotors dynamics, gyroscopic effects, and other small aerodynamic effects.
Still, since the relationship of the thrust of each motor with the spinning of each propeller can be very tricky, some approximations can be done: the UAV is far from obstacles, flying very slow and we're in quasi-static conditions.
Relying on these hypothesis, the relationship between the single propellers thrust and the velocity is approximated by the following formulas:

$$T_i = c_T \omega_i^2 \qquad Q_i = c_Q \omega_i^2$$

with $c_T > 0$ as a thrust constant, and $c_Q > 0$ as a drag factor.
Substituting these formulas to the previous control parameters, it's now possible to define the allocation matrix as the following:

where:

$$B = \begin{bmatrix} c_T & c_T & c_T & c_T & c_T & c_T & c_T & c_T \\ 0 & -l\frac{\sqrt{2}}{2}c_T & -lc_T & -l\frac{\sqrt{2}}{2}c_T & 0 & l\frac{\sqrt{2}}{2}c_T & lc_T & l\frac{\sqrt{2}}{2}c_T \\ lc_T & l\frac{\sqrt{2}}{2}c_T & 0 & -l\frac{\sqrt{2}}{2}c_T & -lc_T & -l\frac{\sqrt{2}}{2}c_T & 0 & l\frac{\sqrt{2}}{2}c_T \\ -c_Q & c_Q & -c_Q & c_Q & -c_Q & c_Q & -c_Q & c_Q \end{bmatrix}$$

So:

$$\begin{bmatrix} u_T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} c_T & c_T & c_T & c_T & c_T & c_T & c_T & c_T \\ 0 & -l\frac{\sqrt{2}}{2}c_T & -lc_T & -l\frac{\sqrt{2}}{2}c_T & 0 & l\frac{\sqrt{2}}{2}c_T & lc_T & l\frac{\sqrt{2}}{2}c_T \\ lc_T & l\frac{\sqrt{2}}{2}c_T & 0 & -l\frac{\sqrt{2}}{2}c_T & -lc_T & -l\frac{\sqrt{2}}{2}c_T & 0 & l\frac{\sqrt{2}}{2}c_T \\ -c_Q & c_Q & -c_Q & c_Q & -c_Q & c_Q & -c_Q & c_Q \end{bmatrix} \begin{bmatrix} {\omega_1}^2 \\ {\omega_2}^2 \\ {\omega_3}^2 \\ {\omega_4}^2 \\ {\omega_5}^2 \\ {\omega_6}^2 \\ {\omega_7}^2 \\ {\omega_8}^2 \end{bmatrix}$$

2 Exercise 2 - Hierarchical - Geometric - Passivity-based controllers

Briefly and qualitatively describe the main differences between the hierarchical controller, the geometric controller and the passivity-based controller presented in the course. What are the main advantages and drawbacks of each control method?

2.1 Controllers Introduction

The aim of the 3 controllers is to bring the UAV from an initial pose made of a position and an attitude, to a final pose. The dynamic model of an UAV can be assimilated to a mass flying in an environment. Hence, 3 dynamical model have been analyzed for the development of the controller strategy: the dynamic model in the body frame, the dynamic model in the world frame (coordinate-free) and the dynamic model using the Euler angles (RPY).

2.2 Hierarchical Control

Hierarchical Control relies on the RPY (roll, pitch, yaw) dynamic model and uses a two-loop structure where the outer loop handles position control, and the inner loop manages attitude control. It applies a *partial feedback linearization* directly on the UAV dynamic model for the angular part, creating two linear sub-systems (linear and angular) interconnected by a non-linear term. The main features can be briefly summarized in:

- based on the RPY dynamic model.
- uses partial feedback linearization on the angular dynamics
- two-loop control structure: outer loop for position and inner loop for attitude.

2.2.1 Advantages

- The use of RPY angles makes the approach more intuitive and understandable.
- Asymptotic Stability can be proven using Lyapunov theory and perturbation theory, ensuring errors decrease to zero over time.

2.2.2 Drawbacks

- The control suffers from representation singularities when the pitch angle is $\pm\frac{\pi}{2}$ due to the minimum representation of attitude.
- The control is less robust to model uncertainties because of potential imperfections in feedback linearization.
- The hierarchical control requires a careful tuning to avoid singularities and maintain stability, especially regarding the conditions where the total thrust is never zero and vertical acceleration does not exceed gravity.

2.3 Geometric Control

Geometric Control is based on a coordinate-free dynamic model using rotation matrices in *SO(3)*. Unlike the Hierarchical control, it constructs a dynamic model of the errors directly in SO(3) and applies feedback linearization on this error model. This control uses a two-loop structure as well, where the outer loop computes the total thrust vector, and the inner loop computes the torques. The main features can be briefly summarized in:

- It uses a coordinate-free dynamic model with rotation matrices in SO(3) to avoid singularities.
- It constructs an error dynamic model in SO(3) for the feedback linearization.
- Two-loop control structure is maintained: outer loop for thrust vector control, inner loop for attitude control.

2.3.1 Advantages

- The geometric control avoids singularities associated with RPY representation by using a rotation matrix.
- Operating in SO(3) using rotation matrices, geometric control offers a larger basin of attraction compared to traditional coordinate-based approaches. This means that the system can maintain stability over a wider range of initial conditions, enhancing control robustness.
- It's very versatile, suitable for dynamic and acrobatic maneuvers due to better handling of large initial attitude errors.

2.3.2 Drawbacks

- This control in SO(3) is less intuitive compared to RPY angles.
- The exponential stability can be proven only for initial attitude errors less than 90°.

2.4 Passivity-based Control

Passivity-Based Control also uses the RPY dynamic model but avoids the feedback linearization. It defines indeed a set of particular errors with coupling terms, making the closed-loop system resemble a mass-damping-spring system with programmable stiffness and damping. This structure enhances robustness to uncertainties. The main features can be briefly summarized in:

- Based on the RPY dynamic model.
- Avoids complete feedback linearization by adding a coupling term σ.
- Resembles a mass-damping-spring system with programmable parameters.

2.4.1 Advantages

- It's very robust to model uncertainties compared to other methods because it doesn't use the partial feedback linearization approach.
- Thanks to the passivity relationship, with an estimator it's possible to estimate the external disturbances.

2.4.2 Drawbacks

- It's very complicated to tune due to the additional coupling term and requires careful parameter adjustment.
- The application of this control is suited for stationary and hovering operations with small movement angles.
- Asymptotic stability can be proven only if the compensation is done perfectly. This means that if $\tilde{f}$ and $\tilde{\tau}$ go to zero, then it's possible to prove asymptotical stability, otherwise not.

2.5 Comparison

Regarding robustness, Geometric Control and Passivity-Based Control are more robust than Hierarchical Control. Geometric Control achieves robustness through its error definition in SO(3), which avoids singularities and ensures stability despite uncertainties. Passivity-Based Control, on the other hand, derives its robustness from its mass-damping-spring system representation, which provides a more flexible and resilient structure against disturbances. However, Hierarchical Control stands out as the most intuitive approach because it uses RPY (roll, pitch, yaw) angles. This makes it easier to understand and implement. Geometric Control, with its rotation matrix-based approach, is less intuitive and requires a deeper understanding of advanced mathematical concepts.

Thanks to its robustness and avoidance of singularities, Geometric Control is the most versatile. It is ideal for dynamic and acrobatic maneuvers, making it suitable for applications like racing and aerobatics. In contrast, Hierarchical Control and Passivity-Based Control are better suited for stationary or hovering operations, where small movement angles are more common. A significant advantage of Geometric Control is its ability to avoid the singularities that affect both Hierarchical Control and Passivity-Based Control, since both of these rely on the RPY model, which can encounter issues when the pitch angle approaches $\pm\frac{\pi}{2}$. Geometric Control's use of the rotation matrix in SO(3) eliminates these singularity problems, providing smoother and more reliable control.

Passivity-Based Control is more complex to tune due to the need for precise parameter adjustment to maintain stability and performance while Hierarchical and Geometric Controls are easier to tune but still require careful attention to avoid conditions that can lead to singularities or excessive vertical acceleration.

All three controllers can implement an estimator for external perturbations, particularly for the passivity-based control, for which the passive relationship between the error ν_η and the estimation error $\tilde{\tau}$ can be demonstrated, which increases its ability to effectively handle external perturbations. The two-loop structure in all controllers is based on the *time-scale separation*, where the angular dynamics are faster than the linear dynamics. This design aims to bring errors to zero through proportional actions, pseudo-derivative actions, and feedforward compensation terms. Once the angular errors are minimized, the interconnection term, which is the non-linear coupling term, also reduces to zero, subsequently minimizing the linear errors. For Hierarchical Control, Lyapunov and perturbation theories verify asymptotic stability, while for Passivity-Based Controls this is true if and only if the compensation is done perfectly. For both controls, the error remains bounded during the transient phase and eventually go to zero. If the errors persist, an integral action can be applied without losing asymptotic stability. In Geometric Control instead, exponential stability can be demonstrated for initial attitude errors less than $\frac{\pi}{2}$, and integral actions can be added to eliminate constant errors due to the interconnection term, maintaining stability as proven by Lyapunov theory.

3 Exercise 3 - Differences between Ground Effect and Ceiling Effect

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

3.1 Ground Effect

Ground effect refers to the phenomena experienced by UAVs when flying close to the ground.

The theoretical framework is based on potential aerodynamics, assuming the fluid is inviscid, incompressible, irrotational, and steady. To model the ground effect, *the method of images* is employed, where virtual rotors are placed on the opposite side of the surface at the same distance. When UAVs fly, propellers create a downwash effect, generating a repulsive force that lifts the UAV up. This effect can be simulated by mirrored propellers placed under the ground, pushing the UAV from below. The aim is to understand what is the heigh where the mirrored propellers don't affect the UAV anymore and this effect can be considered vanished.

Starting the analysis with a single rotor UAV, the power preserving theory holds, where the product between the induced velocity and the thrust of the rotor inside the ground effect (v_{IGE} and T_{IGE}) is equal to the induced velocity and the thrust of the rotor outside the ground effect (v_{OGE} and T_{OGE}), which means that going up and down, the power of the propellers should always be preserved.

From here it's possible to show that:

$$\frac{T_{IGE}}{T_{OGE}} = \frac{1}{1-(\frac{\rho}{4z})^2} \qquad \frac{z}{\rho} > 2 \rightarrow z > 2\rho$$

where ρ is the radius of the propeller and z is the heigh from the ground. Therefore, this formula shows that the ground effect is negligible when the rotor is more than one diameter above the ground.

For multirotor UAVs with multiple propellers (quadcopter, hexacopter, etc...), the situation is even more critical since the influence of adjacent propellers must be considered.

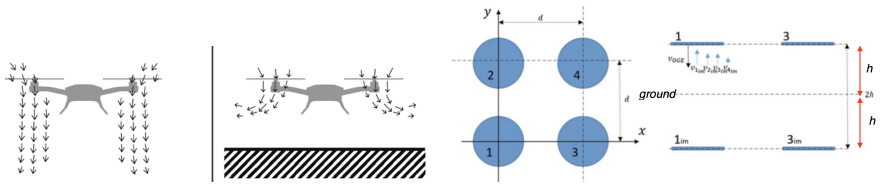


Figure: Ground effect for a multirotor UAV

When the propellers spin, the fluid that bounces on the ground influences the propeller itself but may also influences even the other propellers. So in this case, the distance between the propellers should be taken into account in the previous ratio formula:

$$\frac{u_{T,IGE}}{u_{T,OGE}} = \frac{1}{1-\left(\frac{\rho}{4z}\right)^2 - \frac{z\rho^2}{\sqrt{(d^2+4z^2)^3}} - \frac{z\rho^2}{2\sqrt{(2d^2+4z^2)^3}}}$$

As it's possible to see, the bigger is d and the smaller is this ratio, so the 2 additional terms become negligible and it's again the case of single propeller.

The situation presented before was in the case of perfect hovering of the UAV. In scenarios where UAV moves very close to the surface, auto-hovering effects are observed due to differential ground effect on individual propellers. This because the propellers on one side (further from the ground) suffers less than ground effect. In the limit case it may happen that they don't suffer from the ground effect, while the closer ones yes. In that case, the closer side is pushed up, while the further one isn't and the observed effect is an auto-alignment. This effect may seem good as it doesn't lead to unstable or weird situations, but it increases the effort to move toward a direction, requiring additional torque to counteract this effect!

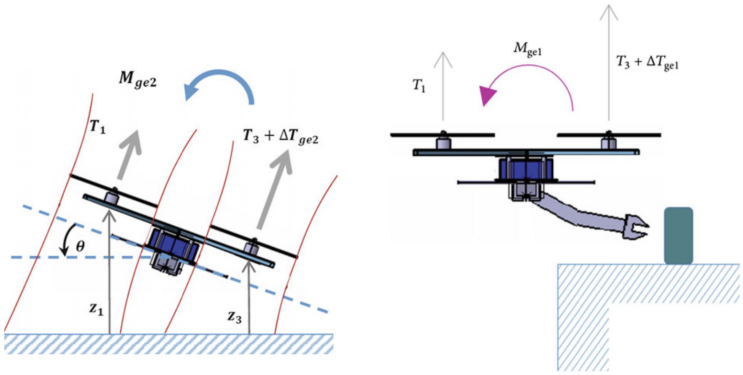
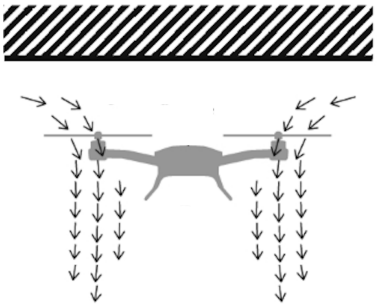


Figure: Ground effect acting on moving UAV situations

In situations where only a part of the rotors are subject to ground effect, asymmetrical aerodynamic effects occur, leading to rotational forces that affect the stability and maneuverability of the UAV. In the case of the right figure, the effect is the generation of a moment that pushes the drone far from the object placed on the ground.

3.2 Ceiling Effect

Ceiling effect arises when an UAV gets close to objects from below, so when the UAV is close not to the ground, but to the ceil.



When a drone flies close to the ceiling, the restricted upward airflow increases the pressure between the rotors and the ceiling, generating an upward force that pushes the drone closer to the ceiling. This effect is the opposite of the ground effect and should be remembered to avoid crashes. The increased thrust results from higher propeller velocity, as the reduced drag near the ceiling allows the propellers to spin faster. This effect can provide more thrust for the same power, leading to lower energy consumption and longer flight times if crashes are avoided. To counter the ceiling effect and prevent crashes, it is important to slow the propellers down.

4 Exercise 4 - UAV External Disturbances Estimation through r-th order Momentum-Based Estimator

Consider the workspace file attached as *ws_homework_3_2024.mat*. Within this file, you can find the values of a flight with a quadrotor with the commanded thrust (*thrust*) and torques (*tau*), the measured linear velocity (*linear_vel*), attitude expressed as Euler angles (*attitude*) and the time derivative of such angles (*attitude_vel*). The employed quadrotor has a supposed mass of *1.5 kg*, and an inertia matrix referred to the body frame equal to *diag([1.2416 1.2416 2*1.2416])*. Implement yourself the momentum-based estimator of order r to estimate the external disturbances acting on the UAV during the flight. Suppose the sampling time of the estimator is equal to *1 ms*. Compare the obtained estimation with the following disturbances applied during the flight:

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

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

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

4.1 Estimator Implementation

Refining control strategies for UAVs requires careful consideration of all unaccounted factors as disturbances. These disturbances, integrated into the dynamic model as external wrenches (f_e and τ_e), are split into linear and angular components. Estimators play an important role in estimating and mitigating these disturbances, enabling the effective application of conventional control schemes.
The estimator is based on the concept of physical momentum, computed as the product of mass and velocity[1].

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

where I and O are identity and zero matrices of proper dimensions.

Given the dynamic model of the system, momentum can be computed to estimate the wrench, but sensors measurements also allow to measure it. If there's a difference from the computation and the measurements, then the discrepancy is given by some terms that are denoted as external disturbances, and can eventually be compensated.
So the aim of the estimator is to estimate the external disturbance through that process, estimating the external wrench $\begin{bmatrix} f_e \\ \tau_e \end{bmatrix}$ through an estimation called $\begin{bmatrix} \hat{f}_e \\ \hat{\tau}_e \end{bmatrix}$.

This estimation is done by means of filters, implemented in the Laplace domain as a transfer function $G(s)$, chosen in different ways and of any order.

$$G_i(s) = \frac{c_0}{s^r + c_{r-1}s^{r-1} + \cdots + c_1 s + c_0}, i = 1, \cdots, 6$$

This is one possible filter to that can be used. The parameters c_j can be chosen in different ways (as will be seen during the development of the exercise), but an easy way could be by using a

[1]The presented computation are developed for the UAV dynamic model with roll,pitch and yaw angles, but it wouldn't change anything in case of dynamic model with the rotation matrices.

Butterworth table, while the static gain of the transfer function is suggested to be equal to one (unit gain). This filter will give a linear relationship between the external wrench and its estimation in the Laplace domain. Then, in order to use it in a real system, it's possible to get a time version of the estimator.

Hence, 2 types of filters have been implemented (one using the *Butterworth* Matlab function and one self-implementing the transfer function). The MATLAB[2] implementation of the the filter is based on the time derivative of the momentum:

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

In order to compute the wrench, some quantities have to be computed with the studied formula, such as R_b, C^T, Q^T. From here, switching to time domain and exploiting the previous computations, it's possible to retrieve the estimated wrench $\gamma = \begin{bmatrix} \hat{f}_e \\ \hat{\tau}_e \end{bmatrix}$ for any order of the used filter.

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

$$\gamma_i(t) = K_i \int_0^t - \begin{bmatrix} \hat{f}_e \\ \hat{\tau}_e \end{bmatrix} + \gamma_{i-1}\, dt, \qquad i = 2, \cdots, r$$

$$\prod_{i=j+1}^{r} K_i = c_j, \qquad j = 0, \cdots, r-1$$

As it's possible to notice, to correctly estimate the external wrench, some quantities have to be measured, such as the attitude and the angular velocity (with an IMU), the linear velocity (with a GPS, or an onboard sensor), inputs and the model parameters.

4.2 Results with different filter orders

With a first order ($r = 1$) Butterworth Filter, the estimations can be represented with the plots[3]:

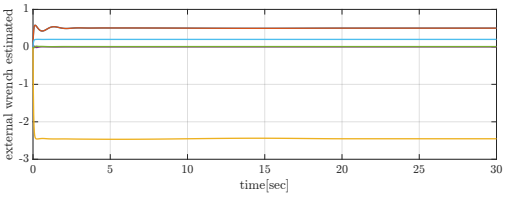


Figure: Plot of the eximated wrench $[f_e \ \tau_e]^T$

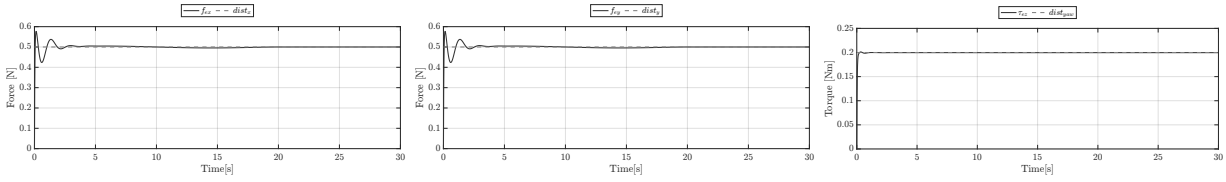
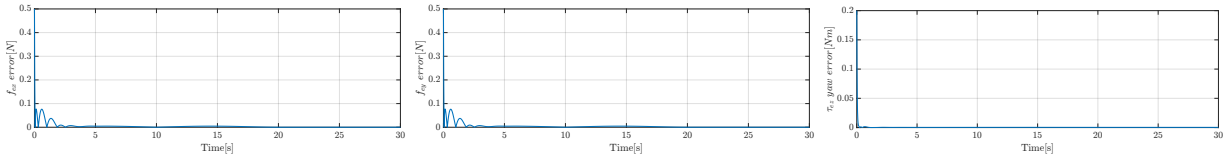


Figure: Plot of the estimation with the disturbances applied during the flight



[2]The logic and implementation of the code were done by Salvatore Granata. The order, organization, and graphics were improved through the use of the given functions and AI ChatGPT tool.

[3]All the single plots of the other vectors of the wrench have not been included, because not necessary to understand the performance of the code, but they can be found and seen in the GitHub repository. Also, the upload of the plots is made in high-resolution (they can be zoomed without any quality loss), but bigger version are upload on the GitHub repository

Figure: Plot of the estimation error with the disturbances applied during the flight

As it's possible to see, the disturbances on x and y axis are well-estimated to be equal to $0.5N$, same happens for the disturbance around the yaw axis of $0.2N$.

Increasing the order of the filter, the expected result is the introduction of delays. In particular, with the selected filter, the results remains almost the same, while after a 5th order the results don't improve too much.
Choosing a very high order ($r = 20$) indeed, the estimation become much worse:

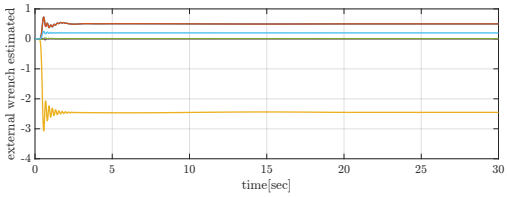


Figure: Plot of the eximated wrench $[f_e\ \tau_e]^T$

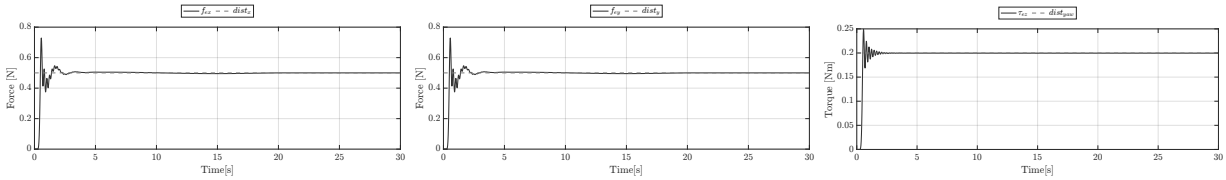


Figure: Plot of the estimation with the disturbances applied during the flight

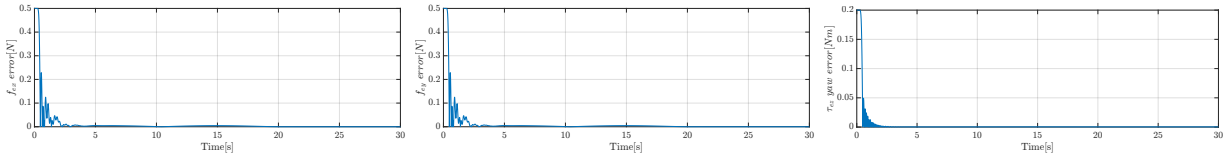


Figure: Plot of the estimation error with the disturbances applied during the flight

Which eventually, choosing a too high order, may lead into a wrong estimation of the external disturbances.

It's also possible not to rely on the Butterworth filter but use instead a *custom transfer function* as a filter to compute the parameters used to estimate the external wrench [4].

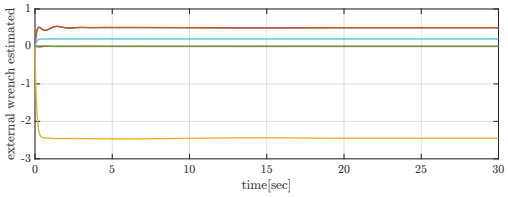


Figure: Plot of the eximated wrench $[f_e\ \tau_e]^T$

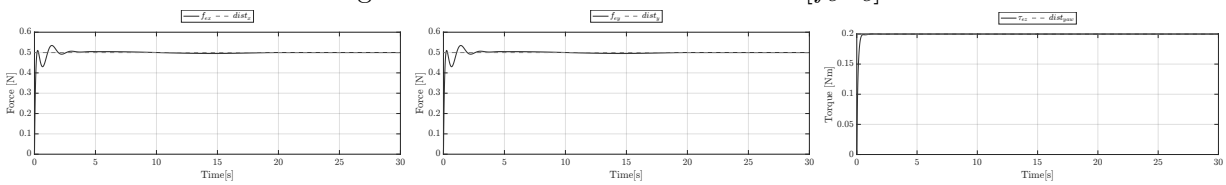


Figure: Plot of the estimation with the disturbances applied during the flight

[4]For this purpose, 2 Matlab scripts have been developed

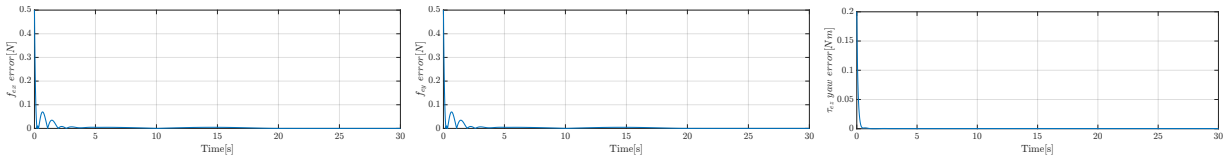


Figure: Plot of the estimation error with the disturbances applied during the flight

In particular, the obtained results show a slightly better tracking capability (less demanding) even with higher order filters (which suffers only of a noticeable delay, but don't produce oscillations of the estimated wrench). In this case, results don't improve too much after a 20th order of the estimation filter. Choosing a very high order ($r = 20$) indeed:

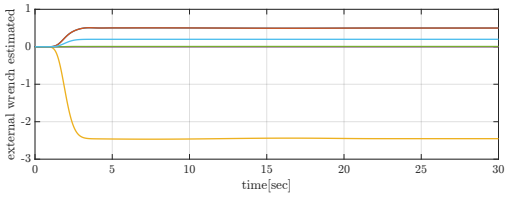


Figure: Plot of the eximated wrench $[f_e\ \tau_e]^T$

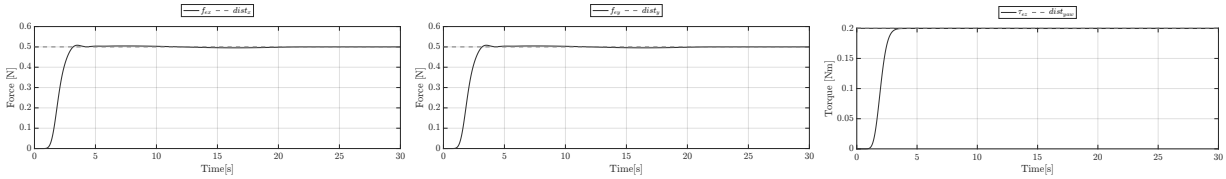


Figure: Plot of the estimation with the disturbances applied during the flight

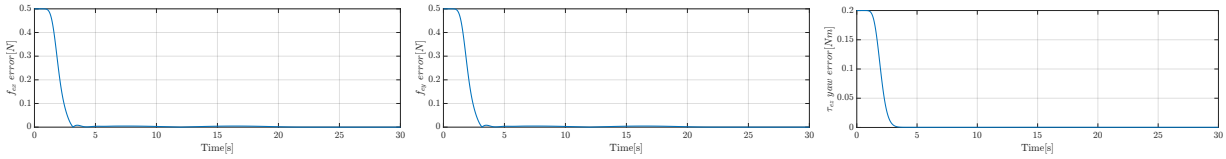


Figure: Plot of the estimation error with the disturbances applied during the flight

4.3 Computation of the real mass

The computation of the real mass of the UAV from the estimation of the disturbance along the z-axis can be carried out in 2 different ways, supposing that the drone is in hovering condition:

- Considering $u_d = -\frac{1}{m}u_T R_b(\eta_b)e_3 + ge_3 + \frac{1}{m}\hat{f}_e$, it's possible to compute the mass as:

$$m_{real} = \frac{f_z}{g} + m$$

- Another way to compute it is by applying the flat output relationship, from which the mass is computed as:

$$m_{real} = \frac{u_{T_z}}{g}$$

Both the formulas lead to the same result, where the real mass of the UAV computed from the disturbance along the z-axis is of 1.25 Kg.
This can be proven showing that, changing the mass to the computed value, the error along the z axis is now minimized:

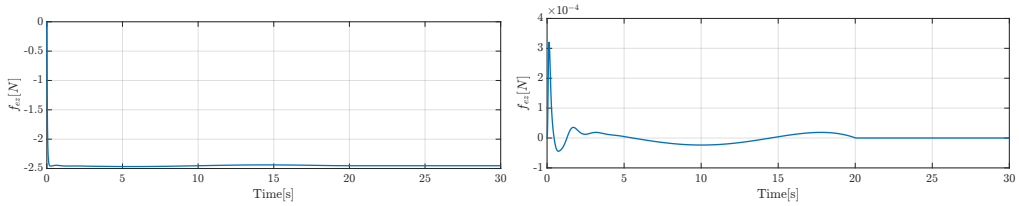


Figure: Estimated disturbance along the z-axis before and after real mass computation.

5 Exercise 5 - UAV Geometric Control

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

5.1 Geometric Control Implementation

Geometric control for a UAV is based on coordinate-free quadcopter dynamic model, which operates by utilizing rotation matrices and concepts from SO(3) space to manage both position and attitude.

Similar to hierarchical control, geometric control also employs an inner and outer loop.

5.1.1 Outer Loop

The outer loop, which handles translational motion, consists of a PD controller plus gravity compensation and an acceleration feedforward term.

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

From which $z_{b,d}$ can be planned as:

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

where:

$$e_p = p_b - p_{b,d}, \qquad \dot{e}_p = \dot{p}_b - \dot{p}_{b,d}$$

The controller for the linear part is a PD + gravity compensation control for the linear part, projected along the current direction of the z_b axis of the body frame, plus an acceleration feedforward term. With that last multiplication, it's like this control is projected along the 3^{rd} axis of R_b (so only along z_b).

5.1.2 Inner Loop

The inner loop is responsible for angular control, includes a proportional term for rotation matrix error, a proportional term for angular velocity error, compensation for dynamic model terms, and a compensation for desired angular velocity and acceleration.

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

where:

$$\bullet y_{b,d} = \frac{S(z_{b,d}) x_{b,d}}{||S(z_{b,d}) x_{b,d}||} \qquad \bullet R_{b,d} = [S(y_{b,d}) z_{b,d} \quad y_{b,d} \quad z_{b,d}]$$
$$\bullet e_R = \frac{1}{2} (R_{b,d}^T R_b - R_b^T R_{b,d})^\wedge \qquad \bullet e_\omega = \omega_b^b - R_b^T R_{b,d} \omega_{b,d}^{b,d}$$

5.2 Simulink Implementation and Plots

Once Implemented the upper relationships, after an accurate tuning, the following control gains have been chosen:

$$K_p = \begin{bmatrix} 355 & 0 & 0 \\ 0 & 355 & 0 \\ 0 & 0 & 565 \end{bmatrix} \qquad K_v = \begin{bmatrix} 27 & 0 & 0 \\ 0 & 27 & 0 \\ 0 & 0 & 15 \end{bmatrix}$$
$$K_R = \begin{bmatrix} 300 & 0 & 0 \\ 0 & 300 & 0 \\ 0 & 0 & 500 \end{bmatrix} \qquad K_\omega = \begin{bmatrix} 25 & 0 & 0 \\ 0 & 25 & 0 \\ 0 & 0 & 20 \end{bmatrix}$$

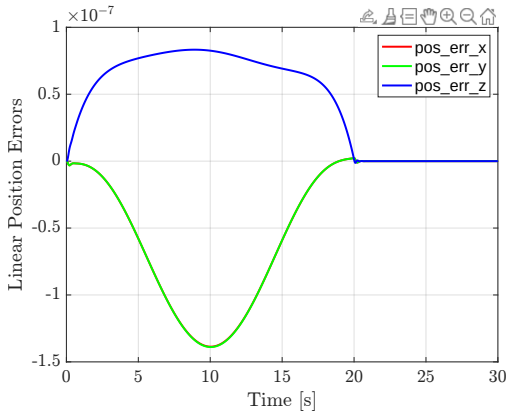


Figure: Plot of the position error e_p and linear velocity error $\dot{e}_p$

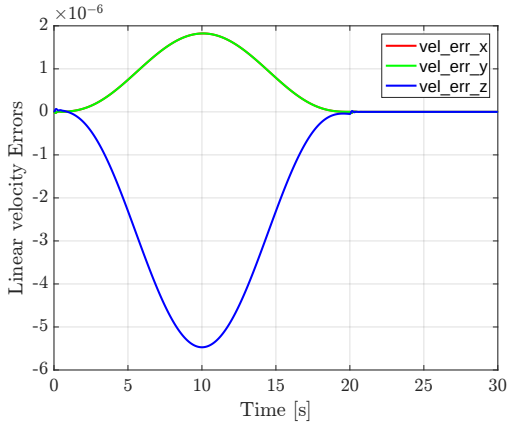


Figure: Plot of the linear velocity error $\dot{e}_p$

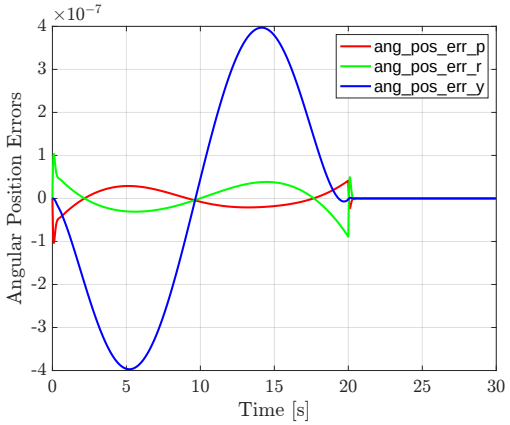


Figure: Plot of the attitude error e_R

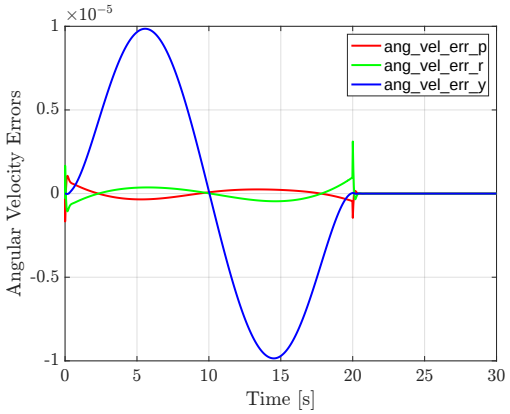


Figure: Plot of the angular velocity error e_ω

While the computed inputs results:

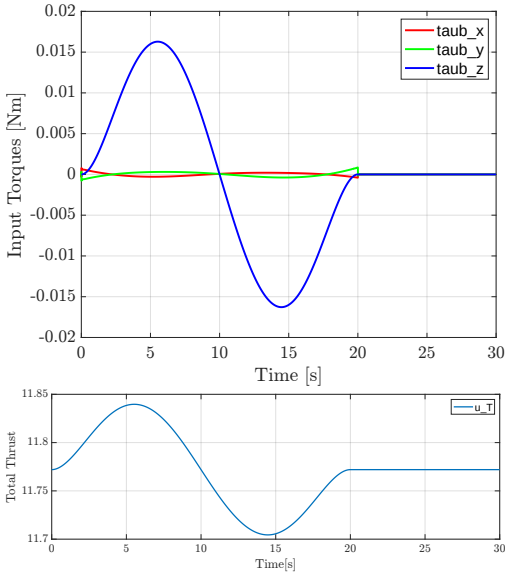


Figure: Plots of torque and thrust inputs of the system

As it's possible to see, this choice of gains, leads to errors of the order less than 10^{-7} for both linear and angular position errors, for the entire duration of the simulation, which permits to achieve a satisful result. Also, thanks to time-scale separation, it's possible to notice that the angular dynamic is slightly faster than the linear one.