# HOMEWORK 3 REPORT

Domenico Tuccillo P38000215

**The code used in this homework has been uploaded here**

## Exercise 1

To evaluate the number of DoFs of the given octacopter let's consider two different situation: fixed on the ground and while flying. If it is fixed to the ground, the DoFs can be computed using Grubler's formula:

- N=8 links(arms) + 1 ground
- J=8 rotational propellers
- $f_i$=1 for each propeller
- m=6 since it is a spatial rigid body

so that $DoFs = 6(9 - 1 - 8) + 8 = 8$. The topology can be written as $S^1 \times S^1 \times S^1 \times S^1 \times S^1 \times S^1 \times S^1 \times S^1 = T^8$ (8 rotating propellers).
The given octocopter, while flying, has a total amount of

$$6(\text{spatial rigid body}) + 8(\text{rotating propellers}) = 14 \text{ DoFs}$$

and the topology of the configuration space can be written as the cartesian product between $\mathbb{R}^3$ (Position in a 3d space), $S^2 \times S^1$ (Orientation in a 3d space), $S^1 \times S^1 \times S^1 \times S^1 \times S^1 \times S^1 \times S^1 \times S^1 = T^8$ (8 rotating propellers), so it is

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

From now on only the flying octacopter will be considered.
However, this is the complete and formal configuration space of the given multirotor, in practice the DoFs given by the rotating propellers are not considered since we don't actually care about the propellers angles, which means that the model-configuration space of interest for the octacopter can be adopted by considering just a subset of the DoFs to describe this model; in such way, excluding the propellers, it has just 6 DoFs, related to the pose in a 3d space, in such way the topology of the model-configuration can be computed as $\mathbb{R}^3 \times S^2 \times S^1$ and the vector of its general coordinates can written as

$$q = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}$$

where it has been selected the RPY convention for the description of the orientation in space.
Regarding octacopter underactuation, if we refer to the complete model, than it is, by definition underactuated since it has more DoFs(14) than actuators(8 propellers). It is convenient

to perform the underactuation analysis on the model-configuration model with just 6 DoFs, in such way since it has more actuators than DoFs, it **may** not be underactuated. Still,the octacopter is underactuated since it has only coplanar propellers, which means that it cannot move in the horizontal direction while being parallel to the ground. Moreover, if the number of propellers is greater than 4, but they are all coplanar, you have a redundancy(which is not related to underactuation) in how to split the velocities among all the propellers, but the system remains underactuated since the $n$ coplanar propellers do not give you the possibility to have more real inputs to the system than 4.

In order to compute the allocation matrix for the given optacopter, the body frame, the

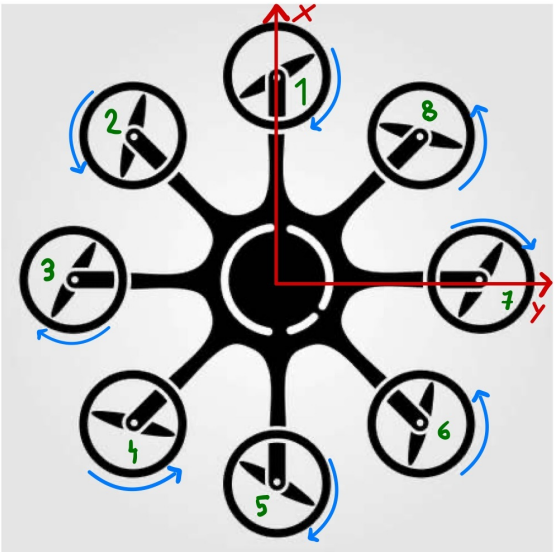



Figure 1: Selected frame and convention

spinning directions of the propellers have been chosen as it is showed in Figure 1; the spinning directions have been chosen in order to guarantee the balance while hovering, by considering the odd propellers spinning clockwise and the even ones spinning counter-clockwise. To compute the allocation matrix it is necessary to analyze how the the thrust of the propellers act on the first two component of the torque, while regarding the third one and the total thrust is easy since for the former the expression is given by the sum of the fan torque $Q_i$ with the sign depending on the verse of rotation(positive if counter-clockwise), and for the latter is given by the sum of all the eight thrusts:

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

$$u_T = \sum_{i=1}^{8} T_i$$

Regarding the $x$ and $y$, it is possible to see that the propellers 1 and 5 do not induce any rotation contribution around the $x$ axis of the body frame and, in the same way, propellers 3 and 7 do not produce any rotation contribution around $y$ axis. Let $l$ be the length of the arm, the remaining control inputs can be computed by observing that:

- **Propellers** 3 **and** 7
  The torque generated by these propellers is directed along the $x$ axis, in particular:

$$\tau_{x,3-7} = l(T_3 - T_7)$$

  Which can be easily proved with the right-hand rule, by looking at Figure 2 you can appreciate how $T_3$ generates a positive torque along the $x$ axis, while $T_7$ generates a negative torque.

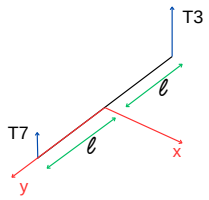



Figure 2: Schematic view

- **Propellers** 1 **and** 5
  The torque generated by these propellers is directed along the $y$ axis, in particular:

$$\tau_{y,1-5} = l(T_1 - T_5)$$

  Which can be easily proved with the right-hand rule, by looking at Figure 3 you can appreciate how $T_1$ generates a positive torque along the $y$ axis, while $T_5$ generates a negative torque.

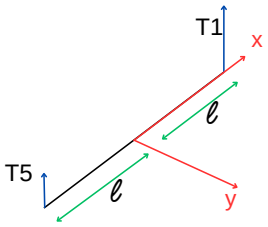



Figure 3: Schematic view

- **Propellers** 2 **and** 6
  The torque generated by these propellers is not completely directed along $x$ or $y$ and it is divided between these two, in order to compute the contribution to both, it is necessary to analyze the projection of the produced torque along the two main axes:

$$\tau_{2,6} = l(T_2 - T_6)$$

This torque is directed along the bisector between $x$ and $y$ axis, so the projection along these two can be computed as:

$$\tau_{x,2-6} = l(T_2 - T_6)\cos\frac{\pi}{4}$$
$$\tau_{y,2-6} = l(T_2 - T_6)\cos\frac{\pi}{4}$$

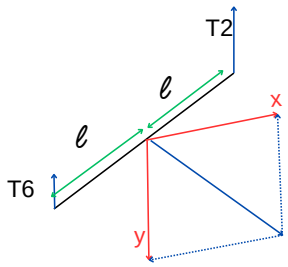



Figure 4: Schematic view

- **Propellers** 4 **and** 8
  The torque generated by these propellers is not completely directed along $x$ or $y$ and it is divided between these two, in order to compute the contribution to both, it is necessary to analyze the projection of the produced torque along the two main axes:

  $$\tau_{4,8} = l(T_4 - T_8)$$

  This torque is directed along the bisector between $x$ and $-y$ axis, so the projection along these two can be computed as:

  $$\tau_{x,4-8} = l(T_4 - T_8)\cos\frac{\pi}{4}$$
  $$\tau_{y,4-8} = -l(T_4 - T_8)\cos\frac{\pi}{4}$$

In such way the torques can be computed as:

$$\tau_x = l(T_3 - T_7) + \frac{\sqrt{2}}{2}l(T_2 - T_6) + \frac{\sqrt{2}}{2}l(T_4 - T_8)$$
$$\tau_y = l(T_1 - T_5) + \frac{\sqrt{2}}{2}l(T_2 - T_6) - \frac{\sqrt{2}}{2}l(T_4 - T_8)$$

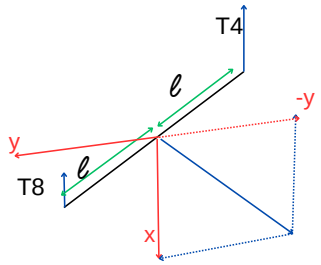



Figure 5: Schematic view

Now it is possible to compute the allocation matrix considering the thrust factor $c_T$ and the drag factor $c_Q$, so that $T_i = c_T\omega_i^2$ and $Q_i = c_Q\omega_i^2$:

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

## Exercise 2

The main differences between the three controllers lie in the dynamic model, the way the stability is achieved, and in their unique features.
The hierarchical controller relies on the RPY dynamic model, which means that it may undergo some representation singularities while reverting the transformation matrix $Q(\eta_b)$ when $\theta = \pm\pi/2$, which means that this controller is not suitable for acrobatic flight.
A fundamental feature of the hierarchical controller is the use of feedback linearization for the control of the angular part, which linearize it and make it easier to control and implement, on the other hand this technique needs perfect knowledge of the model and it is not robust to parameter uncertainties and disturbances, which can also lead to instability if present. It is reasonable to neglect the external disturbances only in case of low speed motion, always supposing a perfect knowledge of the model. Moreover, to employ this controller, the planner must design the trajectory such that $|\mu_z| < g$ since the controller undergoes a singularity when $\mu_z = g$. Finally, it is worth recalling that the planner provides just the desired yaw angle time-law, while the other two angles are computed through the flat output relationships, in order to retrieve the desired angular velocity and acceleration for pitch and roll angles, it is needed to numerically derivate these values, therefore a second-order low pass digital filter should be employed both to reduce noise and compute the time derivatives.

The passivity-based controller overcomes the problem of robustness since it does not rely on a feedback linearization for the angular part; however, it still uses the RPY model, so the

problem of singularity affects also this controller, the same goes for the derivation and filtering of desired roll and pitch. One particular difference between this two controller is that, in the practice, the hierarchical controller is much easier to implement, while the passivity-base needs to be tuned properly since it depends upon a lot of parameters.

The geometric controller overcomes the problem of representation singularities by employing a coordinate-free dynamic model and in such a way the error is computed in the $SO(3)$ space using rotation matrices, unlike the previous controllers which compute the orientation error as difference between measured and desired Euler Angles, moreover the $SO(3)$ space error has a bigger basin of attraction and exponential stability of the closed-loop system can be proved if the initial attitude error is less than 90°, this behavior clearly is one of the main advantages of such controller.
An important difference between this controller and the previous ones and also a huge advantage is that, if there is a big error in the attitude, which results in a big angle between $z_{b,d}$ and $z_b$, the total thrust decreases in magnitude since it is given by the scalar product between $z_{b,d}$ and $z_b$ and this depends upon the cosine of the angle between the two axis, which means that for angles between 0 and $\pi$ it decreases and is also bounded:

$$u_T = -||A|| z_{b,d}^T z_b$$

The passivity-based and the hierarchical don't care about the magnitude of the error and may lead to huge thrust, which is a main problem for the actuators.
However, there is still a limitation on the planner, which is needed to ensure that the planned desired linear acceleration does not exceed the gravity acceleration: $|| - mge_3 + m\ddot{p}_{b,d}|| <$ positive constant.
It partially solves the lack of robustness of the hierarchical since avoid a perfect FBL on the angular part but still relies on some model parameter, however, the bigger basin of attraction of the $SO(3)$ space error makes it more robust than the hierarchical but still suffering from some parametric uncertainties. All the presented controllers share the feature that the angular part (inner loop) has to be faster than the position one(outer loop) in order to achieve a perfect tracking. Moreover, it is possible to implement for all of them an estimator for external disturbance to reject it.
To sum up the main advantages and drawbacks, one respect to the others:

- Hierarchical controller.
  Advantages: easy to implement, very good tracking capability when the parameters are well known;
  Drawbacks: lack of robustness, without external forces estimator it suffers from disturbances, planner limitation on $u_z$, representation singularity, filtering is required,thrust raises when error raises

- Geometric controller.
  Advantages: no representation singularities, self-limitating thrust, bigger basin of attraction, more robust than the hierarchical one
  Drawbacks: planner limitation on linear acceleration, initial attitude must be less than $\pi$, less robust than the passivity-based.

- Passivity-based controller:
  Advantages: more robust than the others.
  Drawbacks: representation singularities, without external forces estimator it suffers from disturbances, planner limitation on $u_z$, filtering is required, parameters tuning can be tough, thrust raises when error raises.

## Exercise 3

The ground effect arises when the UAVs (and UAMs) fly close to the ground, while the ceiling effect arises when approaching some object or surfaces from below. When one of this two effect happens, one must consider that the airflow near the surface induces some velocity and thrust to the propellers. When the ground effect is not negligible, the airflow in the middle of the space between the rotors and the ground hit the ground itself, then it bounces back and hits the multirotor, resulting in an extra lifting force for it. This lifting force tends to straighten up it and block the motion, so in this case one must win the lifting force and more thrust is required to move laterally near the surface and it also makes difficult to grasp objects when only part of the propellers are under the ground effect. However, this effect can be managed to balance the multirotor, since the additional lifting force increases as the distance between a rotor and the ground decreases, so if the rotors are at different distances from the ground, a stabilizing torque is generated and the multirotor is successfully balanced.
Under the ceiling effect the thrust near the ceil increases significantly, pushing the rotor even closer to it; if one does not take care of this effect, crashes may happen. The increased thrust is caused by an increase in the propeller velocity, caused by the vacuum effect, which decreases the propeller drag close to the ceil. Like the ground effect, this can also be managed to save energy, if crashes are avoided, and it results in longer flight time and the possibility to develop more thrust for the same power, which is particularly useful when the task requires a continuous contact with the ceiling.
However it can also be armful if we want to move we must spend more thrust in order to win the vacuum effect, then one must consider that, unlike under ground effect, the ceiling effect doesn't perform a balancing action on the UAV when subject to some perturbation, moreover it tends to crash.
In both cases is applied the method of images which consists of placing a virtual rotor on the opposite side of the surface, at the same distance of the real one to the surface and the ration between the thrust inside the ground/ceiling effect and outside of them for the single rotor can be easily computed, in such way it is possible to study when those effect become negligible or not, however the relationships get more difficult in the case of a multirotor since the interaction of the other rotors must be counted on. For the case of a single rotor, these ratios are as follows:

$$\frac{T_{IGE}}{T_{OGE}} = \frac{1}{1-(\frac{\rho}{4z})^2} \text{ for the ground effect}$$
$$\frac{T_{ICE}}{T_{OCE}} = \frac{1}{1-\frac{1}{k_1}(\frac{\rho}{z+k_2})^2} \text{ for the ceiling effect}$$

Where $k_1$ and $k_2$ are some gains to be tuned, $z$ is the distance from the ground/ceil, $\rho$ is the diameter of the propeller and on the left side there are the ratio of the thrust inside these

effect ($T_{IGE}, T_{ICE}$) and outside of them ($T_{OGE}, T_{OCE}$). In particular, for the ground effect, it can be proven that this effect is negligible when the rotor is more than one diameter off the ground.

## Exercise 4

In order to define the requested estimator of external forces, it is necessary to compute some matrices and vectors from the given workspace, so first $R_b(\eta_b), Q(\eta_b), \dot{Q}(\eta_b), M(\eta_b), C(\eta_b, \dot{\eta}_b), q(\eta_b, \dot{p}_b, \dot{\eta}_b)$ have been computed.
The estimation can be performed with a filtering action in order to reconstruct the external forces, this filter can be computed of different orders, in the simplest case of order 1, the reconstruction can be computed as:

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

Where the integral of the estimator must be computed numerically since the estimator itself is both on the right and left side of the equation. In order to do that it must be discretized, in particular the Backward Euler has been employed, in such way the integral can be computed as:

$$I(k) = I(k) + T_s \left( \begin{bmatrix} \tilde{f}_e(k-1) \\ \tilde{\tau}_e(k-1) \end{bmatrix} + L(k-1) \right)$$

where $T_s$ is the sampling time and $I(k)$ is the value of the integral at time instant $k$. Once this quantity has been evaluated, the estimation can be computed as

$$\begin{bmatrix} \tilde{f}_e(k) \\ \tilde{\tau}_e(k) \end{bmatrix} = k_0 \left( q(k) - I(k) \right)$$

At this point this procedure must be iterated for each time instant. The estimation can be further filtered and achieve higher order of integrals, which can be computed with an iterative procedure as:

$$\gamma_1 = K_1 \left( q - \int_0^t \begin{bmatrix} \tilde{f}_e \\ \tilde{\tau}_e \end{bmatrix} + L\, dt \right)$$
$$\gamma_i = K_i \int_0^t - \begin{bmatrix} \tilde{f}_e \\ \tilde{\tau}_e \end{bmatrix} + \gamma_{i-1}\, dt \quad i = 2, ..., r$$

Where each integral must be computed numerically as seen before; $r$ is the order of the estimator and $K_i$ are computed from the equivalent filter in the Laplace domain, in particular given $c_j$ the coefficient near the j-th power of $s$ of the denominator of the transfer function of the filter, the gains $K_i$ must respect the following relationship:

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

The coefficient on the denominator have been chosen automatically referring to the Butterworth filters normalized( cutoff frequency was fixed to $1rad/s$) of arbitrary order $r$, than the previous relationship must be inverted and the gains $K_i$ can be computed as:

$$K_r = c_{r-1}$$
$$K_j = \frac{c_{j-1}}{\prod_{i=j+1}^{r} K_i} \qquad j = r-1, ..., 1$$

This was done in the matlab function **coeff.m**, which is called in the main of this exercise. where the computation must be performed from $K_r$ to $K_1$ since each gain requires the computation of the next ones. Once retrieved the $K_i$ it is now possible to do the estimation, which was implemented in Simulink,by passing the previous values of the estimation $\gamma_r$ and of the integrals at each step as input for the next steps. There are presented in the next

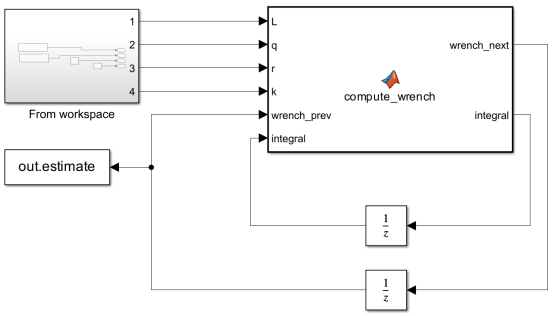
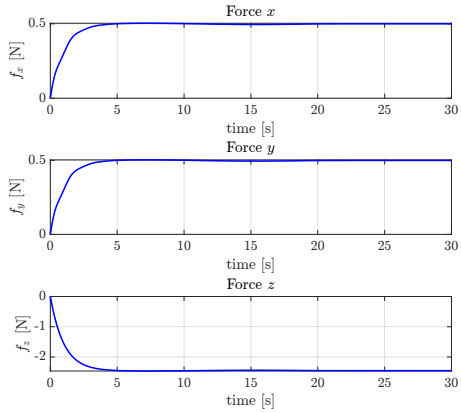
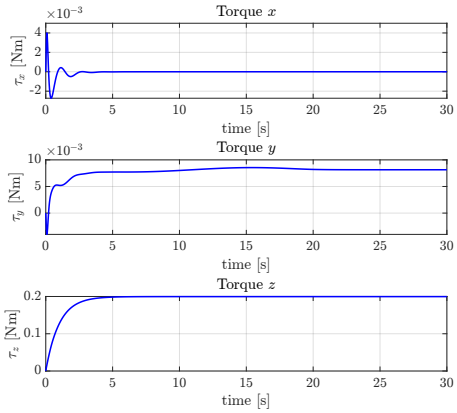



Figure 6: Simulink model

pages the results of the estimations carried out with different filtering order, in such way to appreciate the comparison between them and observe how the performances change by increasing $r$. The goal is to observe constant disturbance forces on the $x$ and $y$ axis of the world frame of $0.5N$ and a torque around the $z$ axis of $0.2Nm$.For this project, it will use the 1% settling time specification, which refers to the time required for the system's response to settle within 1% of the final desired value.

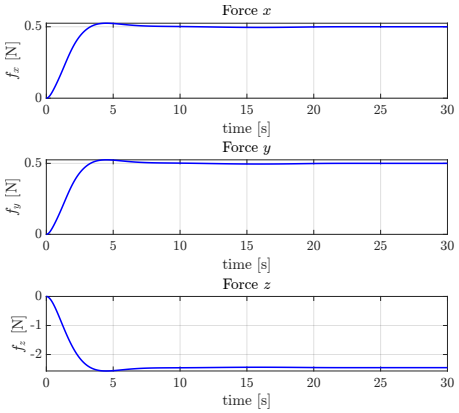



(a) Forces for $r = 1$

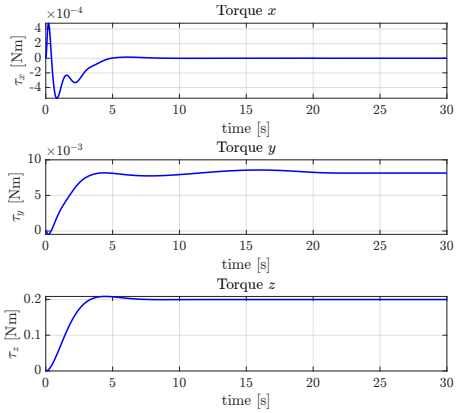



(b) Moments for $r = 1$

Figure 7: Comparison of Forces and Moments for $r = 1$

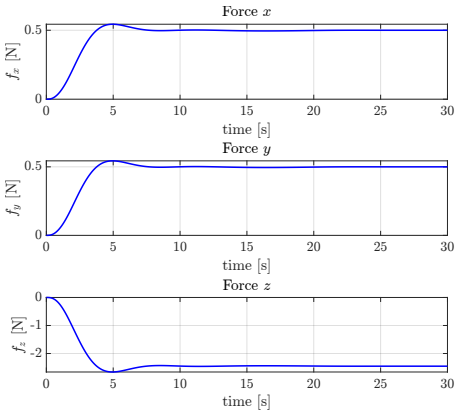



(a) Forces for $r = 2$

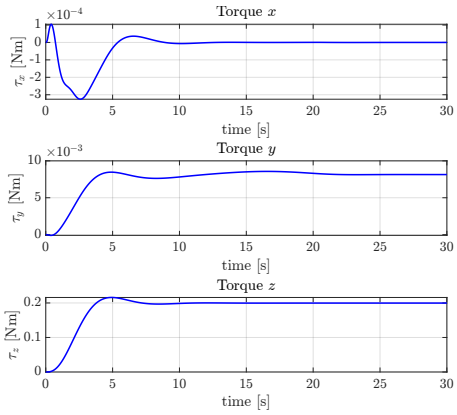



(b) Moments for $r = 2$

Figure 8: Comparison of Forces and Moments for $r = 2$

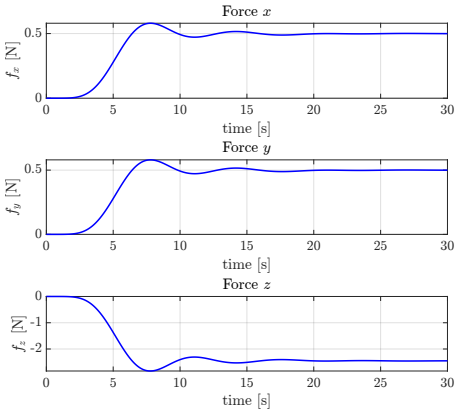



(a) Forces for $r = 3$

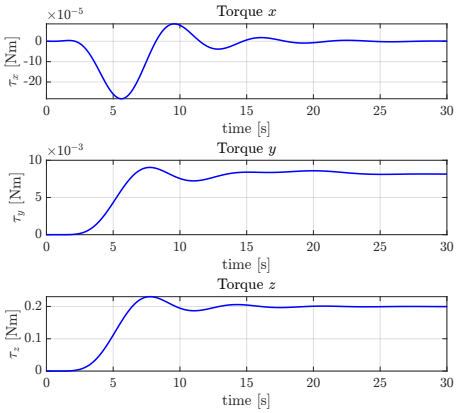



(b) Moments for $r = 3$

Figure 9: Comparison of Forces and Moments for $r = 3$

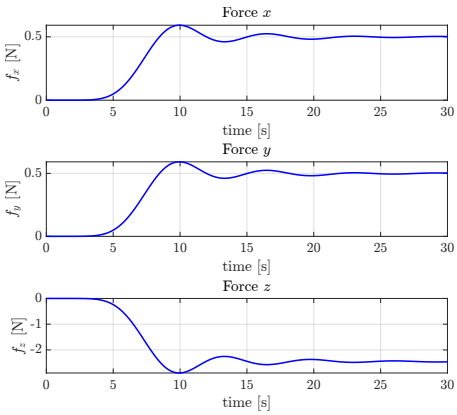



(a) Forces for $r = 7$

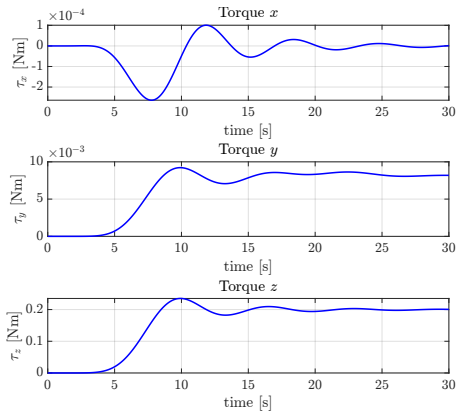



(b) Moments for $r = 7$

Figure 10: Comparison of Forces and Moments for $r = 7$




(a) Forces for $r = 10$




(b) Moments for $r = 10$

Figure 11: Comparison of Forces and Moments for $r = 10$

$r = 1$

- for $f_x$ and $f_y$, the percentage overshoot is zero, while the setting time is 4.2 s.
- for $\tau_z$ the percentage overshoot is zero, while the setting time is 4.9 s

$r = 2$

- for $f_x$ and $f_y$, the percentage overshoot is 5%, while the setting time is 7.5 s.
- for $\tau_z$ the percentage overshoot is 4%, while the setting time is 7.6 s

$r = 3$

- for $f_x$ and $f_y$, the percentage overshoot is 8.7%, while the setting time is 7.5 s.
- for $\tau_z$ the percentage overshoot is 8%, while the setting time is 10 s

$r = 7$

- for $f_x$ and $f_y$, the percentage overshoot is 16%, while the setting time is 19.3 s.
- for $\tau_z$ the percentage overshoot is 15.2%, while the setting time is 15.6 s

$r = 10$

- for $f_x$ and $f_y$, the percentage overshoot is 18%, while the setting time is 21.4 s.
- for $\tau_z$ the percentage overshoot is 17.5%, while the setting time is 21 s

It can be observed how the performances decrease when $r$ increases; the lower is the order the faster and accurate is the estimation, for $r = 1$ the expected values are perfectly matched at steady state(less than $5s$ are required), while for higher order it takes longer and longer to achieve the same result; moreover, oscillatory behaviour and initial delay intensifies increasing the order. In addition, regarding the order of the filter, it was shown that for $r > 15$ the

delay induced is so high that the estimate does not arrive at steady state within $30s$; on a longer time interval even higher orders may actually converge with a sufficiently long settling time; from simulations it turned out that also a $r = 50$ can work with a longer time scale of $150s$, but increasing even more the order, convergence cannot be achieved at all, as happens for $r = 100$, which diverges. The best choice in this case is clearly the first order since it has the best settling time and lack of oscillations, probably in the case of a time-varying disturbance, higher-order filters like 2nd and 3rd, would perform better than the first order. This results are performed with a fixed bandwidth for the choice of the gains, results depend on this parameter and may perform better or worse depending on this choice; the normalized case was chosen in order to obtain the same polynomial coefficients on the denominator of the transfer function to match the ones found online in the butterworth filter table, which I reported below.

| n | Factors of Butterworth Polynomials $B_n(s)$ |
|---|---|
| 1 | $(s+1)$ |
| 2 | $(s^2+1.414214s+1)$ |
| 3 | $(s+1)(s^2+s+1)$ |
| 4 | $(s^2+0.765367s+1)(s^2+1.847759s+1)$ |
| 5 | $(s+1)(s^2+0.618034s+1)(s^2+1.618034s+1)$ |
| 6 | $(s^2+0.517638s+1)(s^2+1.414214s+1)(s^2+1.931852s+1)$ |
| 7 | $(s+1)(s^2+0.445042s+1)(s^2+1.246980s+1)(s^2+1.801938s+1)$ |
| 8 | $(s^2+0.390181s+1)(s^2+1.111140s+1)(s^2+1.662939s+1)(s^2+1.961571s+1)$ |
| 9 | $(s+1)(s^2+0.347296s+1)(s^2+s+1)(s^2+1.532089s+1)(s^2+1.879385s+1)$ |
| 10 | $(s^2+0.312869s+1)(s^2+0.907981s+1)(s^2+1.414214s+1)(s^2+1.782013s+1)(s^2+1.975377s+1)$ |

Figure 12: BW polynomial

Regarding the additional disturbance on the $z$ component of the force, this can be reconduced to an uncertainty in the mass of the UAV; the actual mass can be easily computed as

$$m_{true} = m_{supposed} + \frac{\bar{f}_z}{g} = 1.25Kg$$

where $\bar{f}_z$ is the steady state value of the disturbance, when it should be, and it is true for low orders like I said before, the correct value of the additional forces along that axis. To validate this result I changed the value of the mass in the main file and recomputed the estimation with the real mass and the disturbance along the z component of the force turned out to be null this time, like can also be appreciated in the figure below. Take care that also the mass estimation has different performances depending on the order of the filter, the higher the order, the longer has to last the simulation in order to achieve a perfect estimation, since it depends on the steady state value of $f_z$.

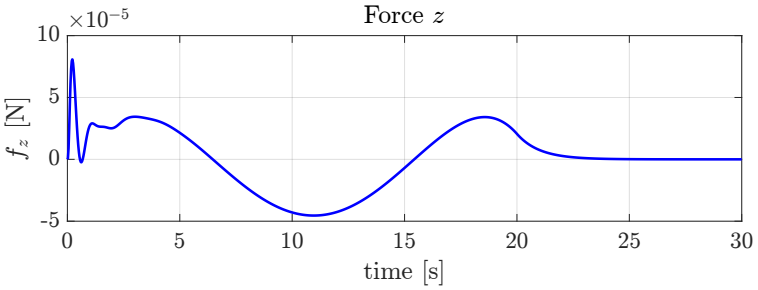



Figure 13: Disturbance force along $z$ axis with $r = 1$

## Exercise 5

The outer loop and inner loop have been completed by writing the error equations and control input laws as requested. For the angular part(inner loop) it was defined the errors $e_R$ and $e_W$ and then, after computing $R_{b,d}$ from the given $z_{b,d}$ and $x_{b,d}$, the torque is computed as:

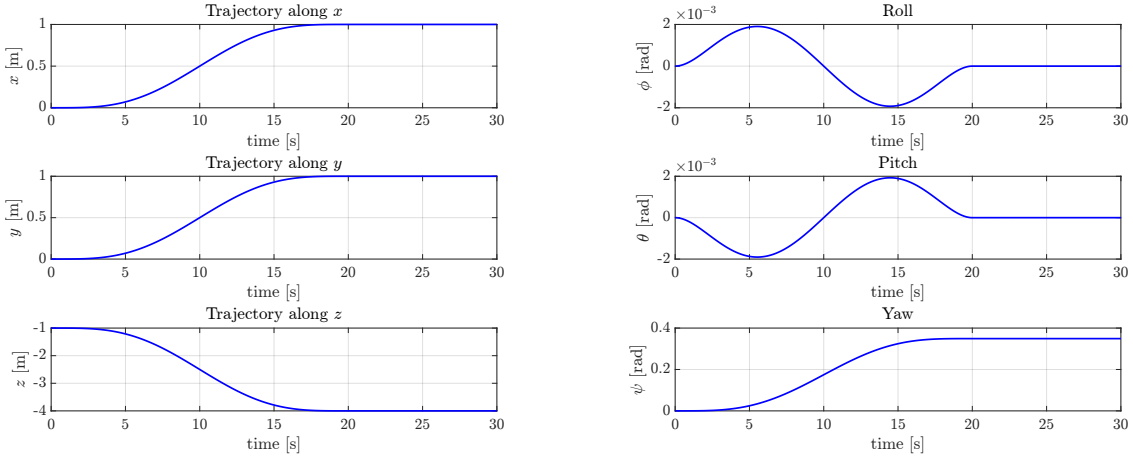
$$\tau = -K_R e_R - K_\omega e_\omega + S(\omega_b^b) I_b \omega_b^b - I_b \left( S(\omega_b^b) R_b^T R_{b,d} \omega_{b,d}^{b,d} - R_b^T R_{b,d} \dot{\omega}_{b,d}^{b,d} \right)$$

While regarding the outer loop, once computed $e_p$ and $\dot{e}_p$ it was computed the total thrust and the $z_{b,d}$ as:

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

Then it has been made the following choices for the control gain in order to make it converge with an error under $10^{-6}$ and making quite smooth position and angular trajectories.

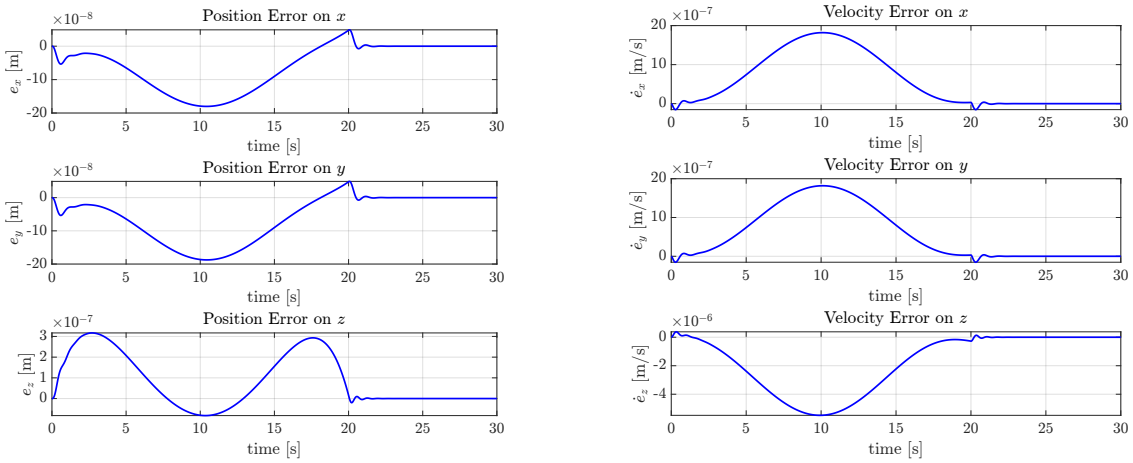
$$K_p = diag(\begin{bmatrix} 50 & 50 & 100 \end{bmatrix}) \quad K_v = diag(\begin{bmatrix} 5 & 5 & 5 \end{bmatrix})$$
$$K_R = diag(\begin{bmatrix} 100 & 100 & 200 \end{bmatrix}) \quad K_W = diag(\begin{bmatrix} 30 & 30 & 5 \end{bmatrix})$$




(a) Position

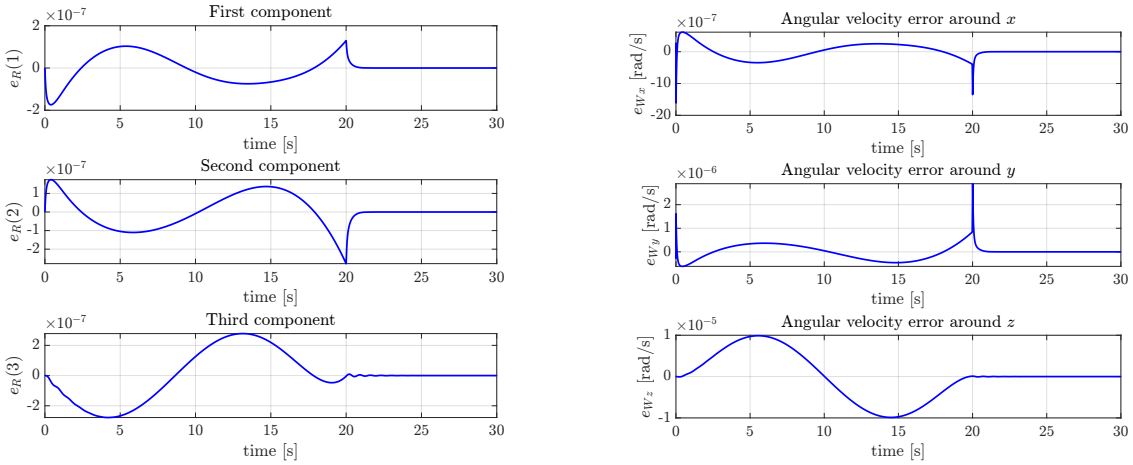
(b) RPY (Roll, Pitch, Yaw)

Figure 14: Position and orientation (RPY)




(a) Linear Position Errors (b) Linear Velocity Errors

Figure 15: Linear position errors and linear velocity errors




(a) Angular Errors (b) Angular velocity Errors

Figure 16: Angular errors and velocities

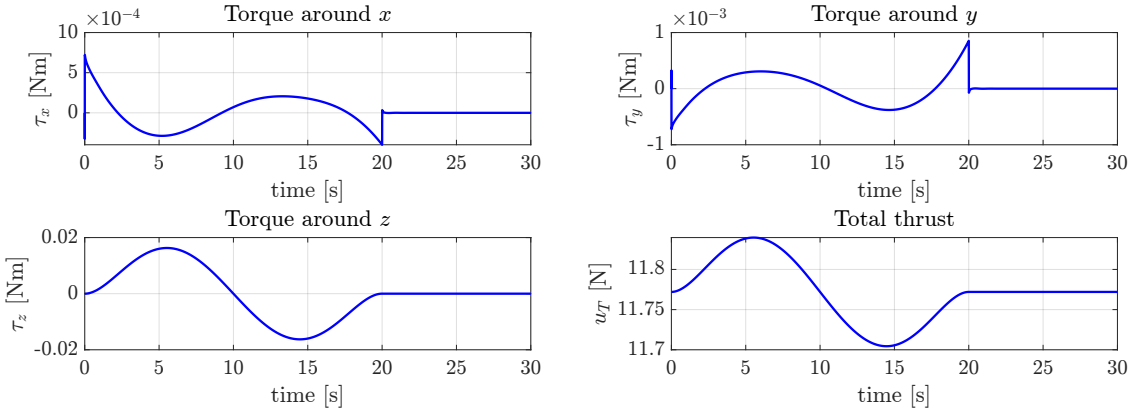
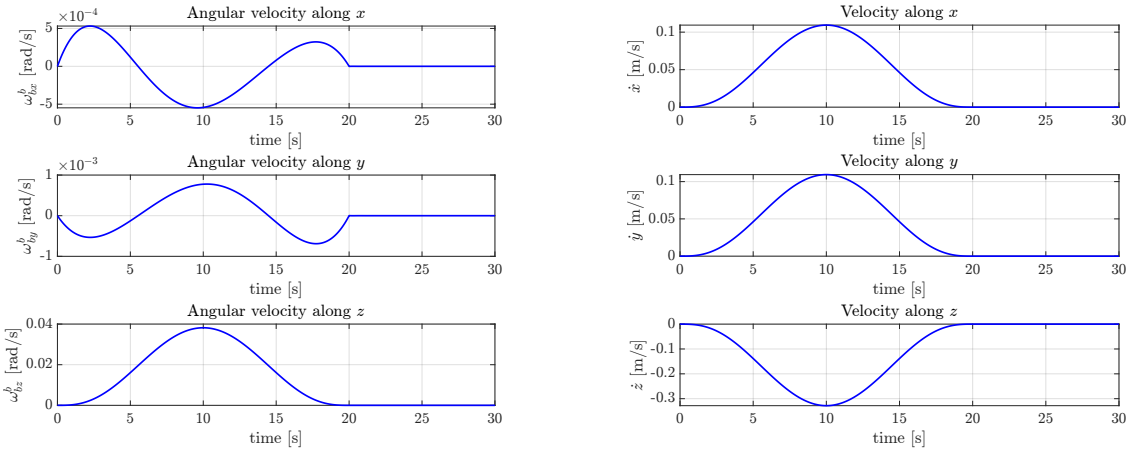


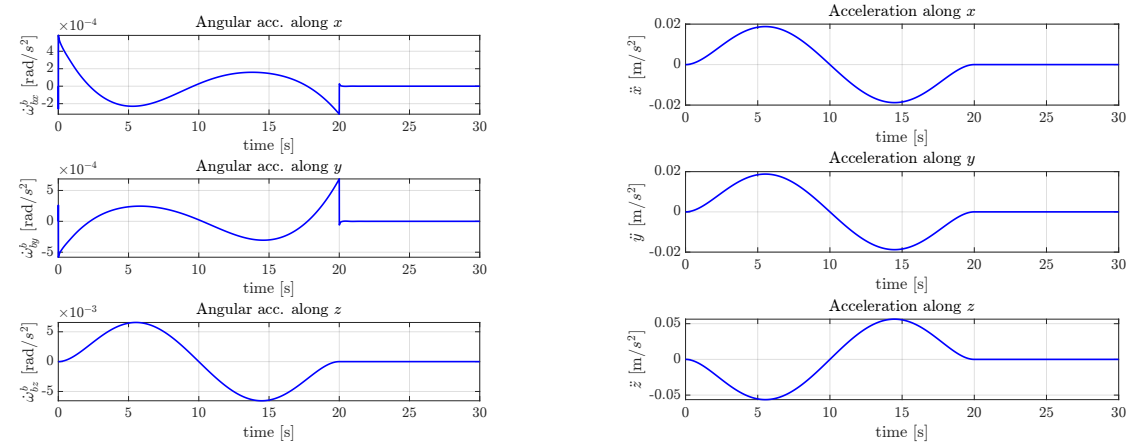

Figure 17: Thrust and torques




(a) Angular velocities (b) Linear velocities

Figure 18: Angular and linear velocities

It can be appreciated that both angular and linear position are tracked smoothly with almost null error (the highest one is the 3rd component of angular velocity error in Figure 16b and has a peak of $10^5 rad/s$). In Figure 14b the RPY angles are plot, these are extracted from the current rotation matrix $R_b$ and, as planned in the callback function provided, the first two angles stay null at steady state and the Yaw goes to 0.35 rad.
Regarding position, it is a point-to-point motion from position $(x, y, z)$ =(0,0,-1) to (1,1,-4), expressed in the world frame.
The control inputs also have a good response ad can be appreciated in 17, just with a small peak before going to steady state. All plots reach steady-state in less than $30s$, despite some oscillatory behaviour that arises before arriving there, but with a low module. Notice that also the accelerations are smooth since the planner employs a 7-th order polynomial, which actually impose a continuous jerk also , see Figure 19. The angular acceleration has a small discontinuity but it's caused by the additional time provided to reach the steady state after the planned trajectory finishes, it it smooth in the first 20s, besides a small transient.




(a) Angular Accelerations (b) Linear accelerations

Figure 19: Angular and linear accelerations