Homework 3 FSR

Antonio Manzoni P38000234

1 Exercise 1

In this exercise we have an octocopter with propellers that are all coplanar.

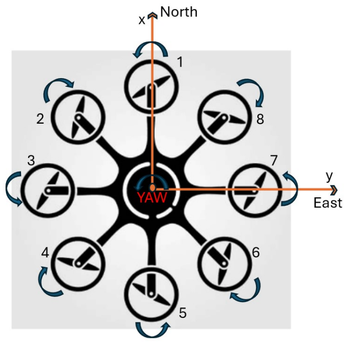


Figure 1: Scheme of the octocopter

1.1 DOFs and underactuation

To fulfill the request of calculate the number of DOFs of the system we must use the Gruebler's formula.

$$DOFs = m(N - 1 - J) + \sum_{i=1}^{J}(f_i) \tag{1}$$

While the system is fixed on the ground it only has 8 degrees of freedom provided by the propellers because $m = 6$, $N = 9$, $J = 8$ and $\sum_{i=1}^{J}(f_i) = 8$, instead when the drone is flying the degrees of freedom became $6 + 8 = 14$, since it has the 3 translations and the 3 rotations of a spatial rigid body. Since the propellers provide full rotations we can retrieve the configuration space easily, in fact we can say that when the body is fixed on the ground the configuration space is equal to T^8, instead when it flies the configuration space becames $T^8 \times \mathbb{R}^3 \times S^2 \times S^1$. The drone is in a flat configuration and we can say that the system has $DOFs = 6$ while flying and it is provided with 8 actuators. We can spoil that the allocation matrix G_q belongs to $\mathbb{R}^{4\times n}$ space, where n is the number of propellers. Even if the number of the propellers is > 4 they are all co-planar, so we have redundancy in how to split the velocities among all the propellers, but the system is still **underactuated** since the n co-planar propellers do not give the possibility to have more control inputs to the system than 4

1.2 Allocation matrix

Now referring to Figure 1 it is possible to compute the allocation matrix for this drone. I've chosen the North-East-Down (NED) convention for the body frame where x axis is pointed to the North, y axis is pointed to the East and the z axis is pointed down. As requested the propellers are numbered following the counter-clockwise verse starting from the x axis. The allocation matrix is a tool that allows us to map the real inputs of the drone which are the velocities of the propellers ω_i^2 to the real inputs which are instead the total thrust u_T and the control torques τ_x , τ_y and τ_z in such way:

$$\begin{bmatrix} u_T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = G_q \begin{bmatrix} \omega_1^2 \\ \vdots \\ \omega_8^2 \end{bmatrix}$$

Now it is possible to define formally those quantities:

$$u_T = \sum_{i=1}^{n} T_i \tag{2}$$

$$\tau_x = lT_2 sin(45°) + lT_3 + lT_4 sin(45°) - lT_6 sin(45°) - lT_7 - lT_8 sin(45°) \tag{3}$$

$$\tau_y = lT_1 + lT_2 cos(45°) - lT_4 cos(45°) - lT_5 - lT_6 cos(45°) + lT_8 cos(45°) \tag{4}$$

$$\tau_z = -Q_1 + Q_2 - Q_3 + Q_4 - Q_5 + Q_6 - Q_7 + Q_8 \tag{5}$$

where: $Q_i = c_Q \omega_i^2$ with $c_Q > 0$ and $T_i = c_T \omega_i^2$ with $c_T > 0$. Making the appropriate substitutions and computing the values of the sines and cosines we can retrive the allocation matrix from the equations (2), (3), (4), (5):

$$G_q = \begin{pmatrix} c_T & c_T & c_T & c_T & c_T & c_T & c_T & c_T \\ 0 & \sigma_1 & c_T\,l & \sigma_1 & 0 & -\sigma_1 & -c_T\,l & -\sigma_1 \\ c_T\,l & \sigma_1 & 0 & -\sigma_1 & -c_T\,l & -\sigma_1 & 0 & \sigma_1 \\ -c_Q & c_Q & -c_Q & c_Q & -c_Q & c_Q & -c_Q & c_Q \end{pmatrix}$$

where: $\sigma_1 = \frac{\sqrt{2}\,c_T\,l}{2}$

2 Exercise 2

During the course have been presented various types of controllers for the drones and in particular they are the hierarchical controller, the geometric controller and the passivity-based controller. Those controllers are different from each other but they share a common feature which belongs to the separation of the linear dynamics from the angular dynamics of the UAV's model. Another note to do is that every controller can be done both with the consideration of external disturbances and also without. Briefly I'd describe the particularities of those controllers with their advantages and drawbacks.

- **Hierarchical controller**: This controller is based on the RPY model of the drone and it takes the name of hierarchical because it generates two control loops: the inner one which is the faster one and takes care of attitude control and the outer one takes which takes care of position control (the property in which the angular part is faster than the linear one is called time-scale separation).In particular, if we focus on the angular part of the model we can notice that the system is fully actuated; this fact allows us to apply the feedback linearization, but this one is only partial because the linear part is underactuated. A great feature of this controller is the simplicity of the tracking error for the attitude which is designed as follows:

$$\begin{bmatrix} e_\phi \\ e_\theta \\ e_\psi \end{bmatrix} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} - \begin{bmatrix} \phi_d \\ \theta_d \\ \psi_d \end{bmatrix}$$

where, due the fact we are considering the NED frame, the errors are defined as the current values minus the desired ones. Note that this error has no geometric meaning but it works in practice. The problem is that the planner gives us only the desired yaw and not also the desired pitch and roll, so we need to retrieve them in another way. It is possible to compute the virtual desired linear acceleration considering the desired rotation matrix as follows:

$$\mu_d = \begin{bmatrix} \mu_x \\ \mu_y \\ \mu_z \end{bmatrix} = \frac{1}{m} u_T R_b(\eta_{b,d}) e_3 + g e_3 \in R^3 \tag{6}$$

This equation is really helpful because it allows us to retrieve u_T (total thrust) and the desired values of roll and pitch; in particular to compute the desired roll u_T should never be equal to 0 and to calculate the desired pitch the third component of μ_d should never be equal in module to the gravity. This condition occurs only when I turn off everything and so the total thrust is equal to 0. The linear errors are defined like the angular part as the current values minus the desired ones. Considering the closed loop control, the objective is now to bring to 0 the error thanks to the inputs mu_d and $\tilde{\tau}$, in particular we can notice that the angular part goes asymptotically to 0, while to show that the linear part is stable we need to recall the pertubation theory. It is surely possible to say that the linear part is bounded cause its pertubation term is composed by sines and cosines. The main advantages of this controller are:

1. The fact that consider a feedback linearization for the angular part.
2. Easy definition of the error using the RPY convention, which is understable also for people who are not into control theory.
3. Possibility to implement PD controllers with acceleration feedforward.

The main drawbacks are:

1. The RPY convention suffers from singularities in which $\theta \neq \pm\frac{\pi}{2}$, so the controller cannot be employed for instance in acrobatic flights.
2. Needs to add robustness due the fact that uses the feedback linearization, and needs to increase tracking accuracy, for instance by adding an integral action;
3. In order to retrieve the desired angular velocity and acceleration, we must numerically derivate these values.

- **Geometric controller**: The geometric controller is based on the coordinate-free UAV dynamic model, so the attitude is not described by a minimum representation but in a geometric way thanks to the use of the rotation matrix. The job of the controller is to construct an error that tells how the rotation matrix is distant from another one in the space of the rotation matrices. For the linear part the error is defined as the current position minus the desired one. For the angular part the errors are defined directly in $SO(3)$. Like for the hierarchical controller, also this one implement an outer loop that acts on the linear errors:

$$e_p = p_b - p_{b,d}, \tag{7}$$

$$\dot{e}p = \dot{p}_b - \dot{p}b, d \tag{8}$$

and an inner loop that works on the angular errors:

$$e_R = \frac{1}{2}\left(R_{b,d}^T R_b - R_b^T R_{b,d}\right)^V \tag{9}$$

$$\omega_e = \omega_b - R_b^T R_{b,d} \omega_{b,d} \tag{10}$$

To design the closed loop we need to do the assumption that:

$$\|-\mathbf{mge_3} + \mathbf{m\ddot{p}_{b,d}}\| < \text{positive constant} \tag{11}$$

which ensures that the planned desired linear acceleration does not exceed the gravity acceleration.

For the outer loop control we have:

$$u_T = -\left(K_p e_p + K_v \dot{e}_p - mge_3 + m\ddot{p}_{b,d}\right)^T R_b e_3 \tag{12}$$

$$Z_{b,d} = -\frac{K_p e_p + K_v \dot{e}p - mge_3 + m\ddot{p}_{b,d}}{\|K_p e_p + K_v \dot{e}_p - mge_3 + m\ddot{p}_{b,d}\|} \tag{13}$$

which is a PID + gravity compensation for the linear part projection along the current direction of the z axis of the body frame. while for the inner loop control we have:

$$\tau_b = -K_R e_R - K_\omega e_\omega + (S(\omega_b))^I \omega_b - I\left(S(\omega_b) R_b^T R_{b,d}\omega_{b,d} - R_b^T R_{b,d}\dot{\omega}_{b,d}\right) \tag{14}$$

which is a feedback linearization with some compensation terms. This loop control is implemented after retrieved $R_{b,d}$. The main advantages of this controller are:

1. Avoiding minimum representations consent to prevent representation singularities, so it can be used for instance for acrobatic flights.
2. As the hierarchical controller uses a feedback linearization for the angular part which simply things.
3. Possibility to implement PID controllers with acceleration feedforward;
4. Exponential stability of the closed-loop system can be proven if the initial attitude error is less than 90°.
5. Has an autoregulation effect of the thrust given to the fact that the error between $z_{b,d}$ and z_b goes inside a cosine.

The main drawbacks are:

1. difficulty to read the angular error;
2. needs to add robustness due the fact that uses the feedback linearization;
3. to compute the error for the angular part we need to know the Vee operator.

- **Passivity-based controller**: Like the hierarchical controller it is based on the RPY convention. A particularity of this controller is that avoids the partial feedback linearization for the angular part. In this case the dynamic model is presented with an explicit compensation of the external wrench. We need to define for the inner loop the following quantities:

$$\dot{\eta}_r = \dot{\eta}_{b,d} - \sigma e_\eta \tag{15}$$

$$\ddot{\eta}_r = \ddot{\eta}_{b,d} - \nu e_\eta \tag{16}$$

$$e_\eta = \eta_b - \eta_{b,d} \tag{17}$$

$$\dot{e}_\eta = \dot{\eta}_b - \dot{\eta}_{b,d} \tag{18}$$

$$v_\eta = \dot{e}_\eta + \sigma e_\eta \tag{19}$$

Where σ and ν are both positive coupling parameters. The computation of μ_d is exactly the same to the one of the hierarchical controller and also $\ddot{p}_b$ is the same but this time I need to take into account the external forces.

If we look the closed-loop equations we can do some considerations:

1. Both closed-loop equations can be seen as mass-damping-spring systems with programmable stiffness, partially-programmable damping and given mass and inertia.
2. The right side of the angular closed-loop equation acts like an external disturbance of the mass-damping-spring system of the left side and ,in particular, this disturbance is given by the estimation error of the angular part.
3. It can be proven that these equations establish a passive relationship between $\tilde{\tau}$ and v_η.
4. Boundness of the errors can be proven through Lyapunov and perturbation theories.

The main advantages of this controller are:

1. it is more robust than the previous ones due the fact that avoids the feedback linearization.
2. The dynamic model can be seen as a mass-damping-spring system with programmable parameters which have also a physical meaning.

The main drawbacks are:

1. The coupling factor, σ, should be properly tuned since it affects the tracking results dramatically, in fact the robot may vibrate for small values of this parameter, while larger values improve tracking.
2. It suffers from representation singularities.

3 Exercise 3

The UAVs and UAMs are usually involved in applications that require flying close to different structures, objects, and obstacles in which the aerodynamics effects are relevant. Among those effects the most common ones are the ground effect and the ceiling effect. Both the effects are related to the radius of the propeller ρ, to the distance on the z axis of the propeller to the ground (ceiling) and ,for the multi-rotor systems, to the distance between the propellers.

- **Ground effect**: This effect is based on the potential aerodynamic assumption that considers the fluid as:

    1. Inviscid
    2. Incompressible
    3. Irrotational
    4. Steady

    and it arises when the drone is close to the ground, for instance when it is taking off or landing. In this situation, it occurs a downwash effect where the airflows rebounds on the surface and increasing the air pressure beneath the drone. This increase in pressure leads to an improvement in lift, reducing the power required to maintain flight. So we can consider that there are a sort of virtual propellers located under the ground at the same height in the opposite side to the height where are located the drone's propellers. It is possible to show from the power preserving theory and from aerodynamic considerations that this effect is negligible when the drone flies at an height that is bigger than the diameter of the rotor.

- **Ceiling effect**: This effect can be seen as the right opposite of the previous one. It occurs when a drone flies close to the ceiling of a building or another structure. In these situations, the air above the drone may be compressed between the drone itself and the ceiling, increasing the air pressure above the drone. So, it can be seen as a vaacum effect in which the drone is attracted to the surface provoking a speed up of the rotors that may involve into crashes. To avoid this problem is necessary to slow down the propellers when the drone is close to the surface. If we take into account this effect it is possible to develop more thrust for the same spent power; this results in less energy consumption and more flight time.

4 Exercise 4

The goal is to implement an estimator-based controller with a sampling time of 1ms. The first thing to do in this exercise is to load the provided file in which is possible to find the values of a flight with a quadrotor with the commanded thrust (thrust) and torques (tau), the measured linear velocity, attitude expressed as Euler angles and the time derivative of such angles. The drone has a supposed mass equal to 1.5kg. Also the inertia matrix is provided. The next step is to extract the signal values from the ws file which are useful to retrieve all the quantities needed to the estimator. To define the transfer function I've used an analogic chebyshev filter type 1 in low pass configuration. The filter takes as inputs the order of the estimator, the peak to peak ripple expressed in decibels and the passband edge frequency. After that it is possible to calculate the K_i coefficients from the c_j coefficients provided by the filter.

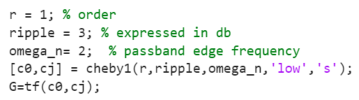
$$\prod_{i=j+1}^{r} K_i = c_j \quad j = 0, ..., r-1 \tag{20}$$

Finally it is possible to design the estimator and thanks to it we can also retrieve the real mass of the drone. Now I'll show some tests tuning the parameters of the filter in order to reach the following values:

1. A disturbance force of 0.5N along x axis.
2. A disturbance force of 0.5N along y axis.

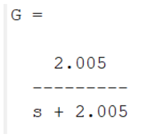
3. A disturbance torque of 0.2 Nm around yaw axis.

- Test 1



```
r = 1; % order
ripple = 3; % expressed in db
omega_n= 2;  % passband edge frequency
[c0,cj] = cheby1(r,ripple,omega_n,'low','s');
G=tf(c0,cj);
```

Figure 2: Chebyshev parameters



```
G =

   2.005
 ---------
 s + 2.005
```

Figure 3: Transfer function

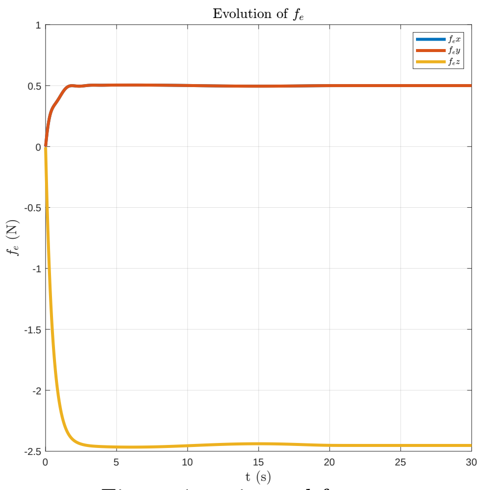


Figure 4: estimated forces

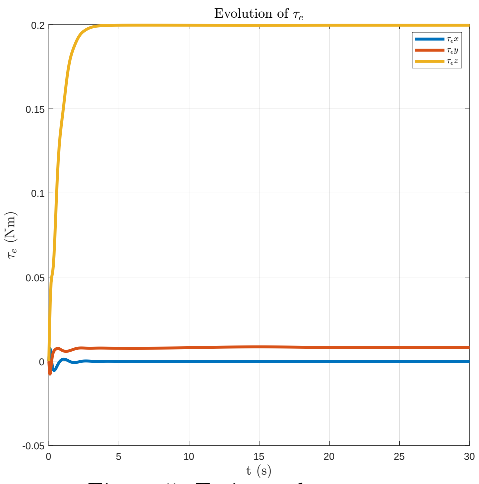
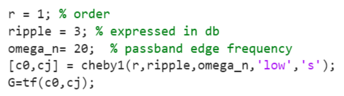


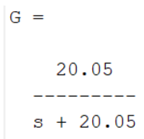
Figure 5: Estimated torques

- Test 2



```
r = 1; % order
ripple = 3; % expressed in db
omega_n= 20;  % passband edge frequency
[c0,cj] = cheby1(r,ripple,omega_n,'low','s');
G=tf(c0,cj);
```

Figure 6: Chebyshev parameters



```
G =

   20.05
 ---------
 s + 20.05
```

Figure 7: Transfer function

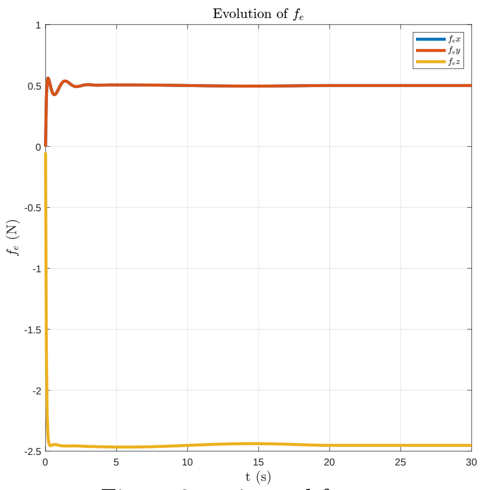


Figure 8: estimated forces

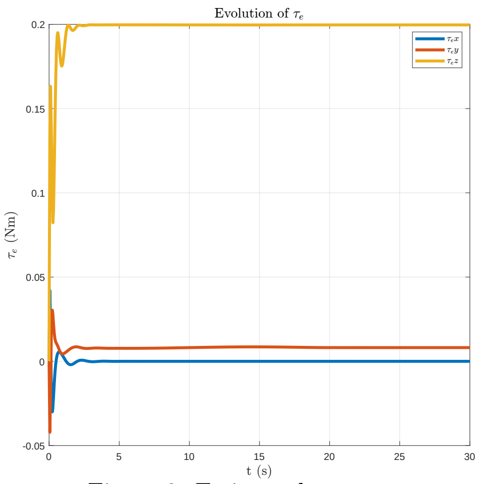
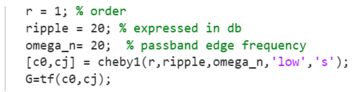


Figure 9: Estimated torques

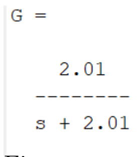
In this test I've sped up the controller increasing the bandwidth, but as we can see an oscillatory behavior appears.

- Test 3



```
r = 1; % order
ripple = 20; % expressed in db
omega_n= 20;  % passband edge frequency
[c0,cj] = cheby1(r,ripple,omega_n,'low','s');
G=tf(c0,cj);
```

Figure 10: Chebyshev parameters



```
G =

    2.01
  --------
  s + 2.01
```

Figure 11: Transfer function

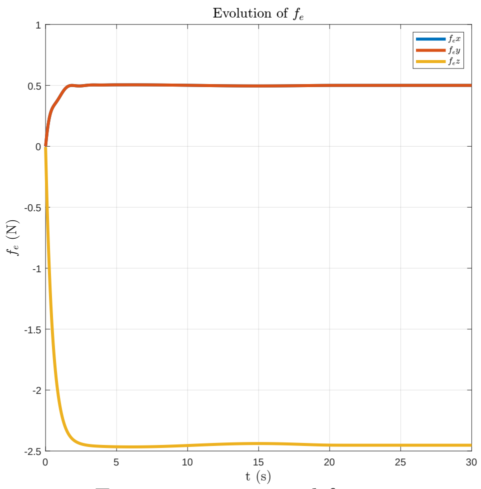


Figure 12: estimated forces

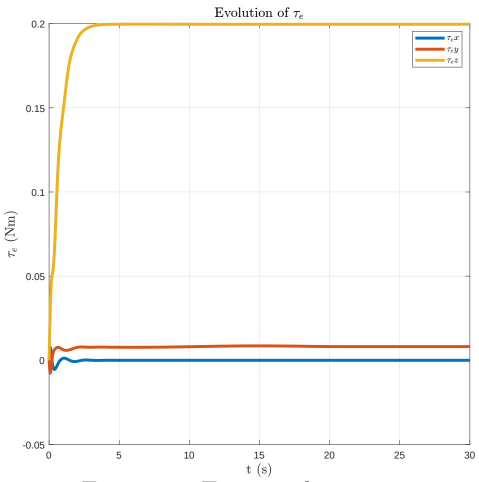
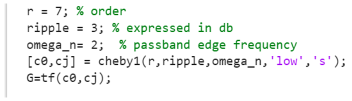


Figure 13: Estimated torques

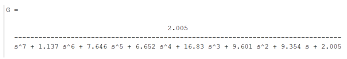
With the same bandwidth of the previous case I've increased the ripple and the result is that the curve is more smooth than before but the controller is slower.

- Test 4



```
r = 7; % order
ripple = 3; % expressed in db
omega_n= 2;  % passband edge frequency
[c0,cj] = cheby1(r,ripple,omega_n,'low','s');
G=tf(c0,cj);
```

Figure 14: Chebyshev parameters



```
G =

                                        2.005
  ----------------------------------------------------------------------------
  s^7 + 1.137 s^6 + 7.646 s^5 + 6.652 s^4 + 16.83 s^3 + 9.601 s^2 + 9.354 s + 2.005
```

Figure 15: Transfer function

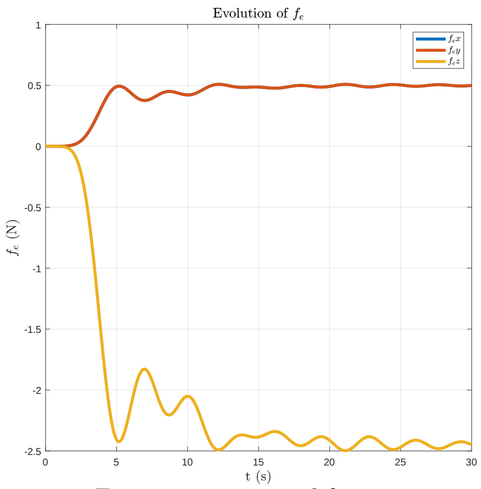


Figure 16: estimated forces

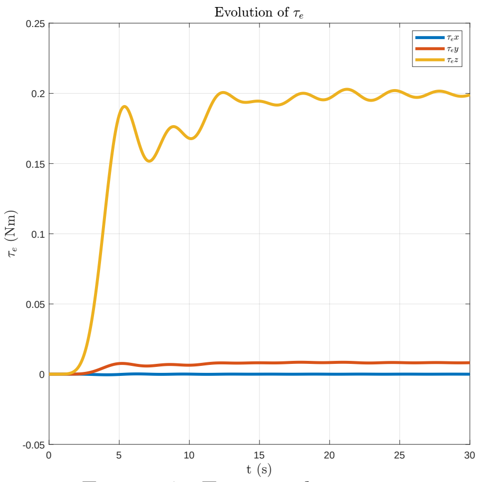
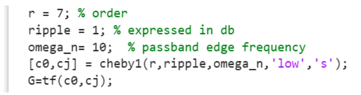


Figure 17: Estimated torques

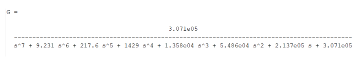
In this test I've use the same ripple and bandwidth values as the first test but increasing the order of the filter. The result shows us that the controller has more oscillations and converges later than before.

- Test 5



```
r = 7; % order
ripple = 1; % expressed in db
omega_n= 10;  % passband edge frequency
[c0,cj] = cheby1(r,ripple,omega_n,'low','s');
G=tf(c0,cj);
```

Figure 18: Chebyshev parameters



```
G =

                                      3.071e05
  ----------------------------------------------------------------------------------
  s^7 + 9.231 s^6 + 217.6 s^5 + 1429 s^4 + 1.358e04 s^3 + 5.486e04 s^2 + 2.137e05 s + 3.071e05
```

Figure 19: Transfer function

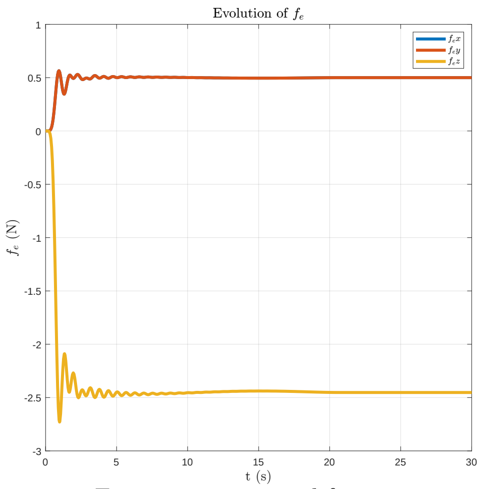


Figure 20: estimated forces

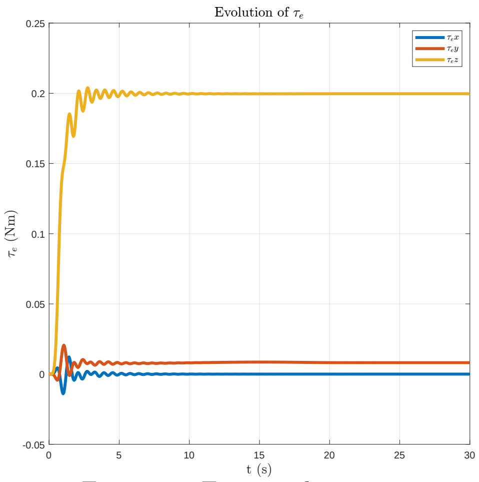


Figure 21: Estimated torques

Increasing the value of the bandwidth and decreasing the ripple I obtain a faster controller but with a larger number of oscillation.

- Test 6

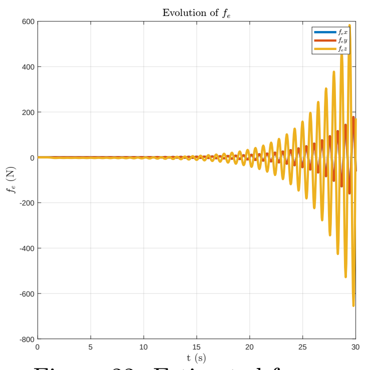


Figure 22: Estimated forces

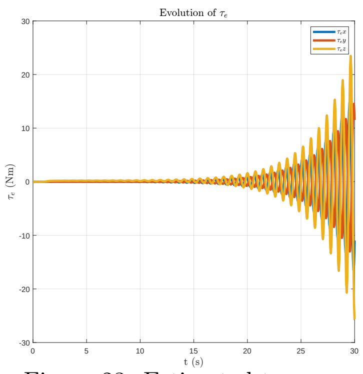
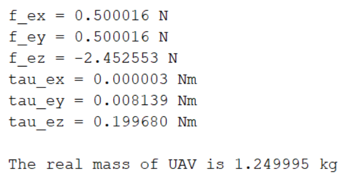


Figure 23: Estimated torques

In this last test I've shown that the controller, with the same values of ripple and bandwidth but with an order of the estimator equal to **13**, starts to not converge to the set values. It's important to note that the values of $\hat{f}_x$and $\hat{f}_y$ are identical, and so the curves are overlapping. As last thing it is possible, as said before, to calculate the real mass of the UAV from $\hat{f}_{ez}$ in such way:

$$m_r = m_e + \frac{\hat{f}_{ez}}{g}$$



```
f_ex = 0.500016 N
f_ey = 0.500016 N
f_ez = -2.452553 N
tau_ex = 0.000003 Nm
tau_ey = 0.008139 Nm
tau_ez = 0.199680 Nm

The real mass of UAV is 1.249995 kg
```

5 Exercise 5

In this exercise we need to fill the inner loop and the closed loop of the given geometric controller with the formulas that have been presented during the course.

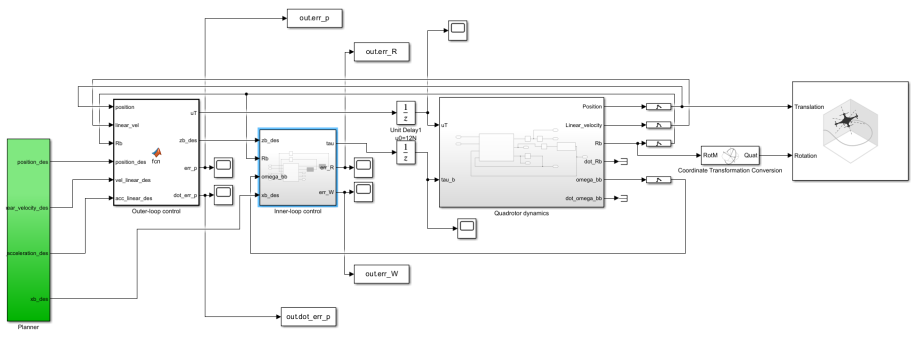
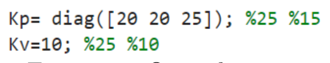


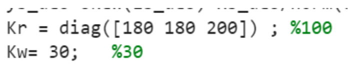
Figure 24: Simulink scheme of the controller

For the gains' choice a trial and error method has been applied. Following the principle that the inner loop should be faster than the outer loop, I've given an higher magnitude to the gains relative to the angular part.



```
Kp= diag([20 20 25]); %25 %15
Kv=10; %25 %10
```

Figure 25: Outer loop gains



```
Kr = diag([180 180 200]) ; %100
Kw= 30;   %30
```

Figure 26: Inner loop gains

Here I'll show the results obtained of the errors with those gains for the linear part:

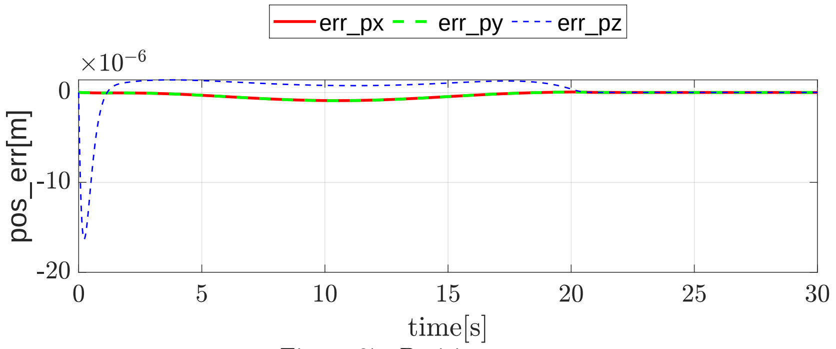


Figure 27: Position errors

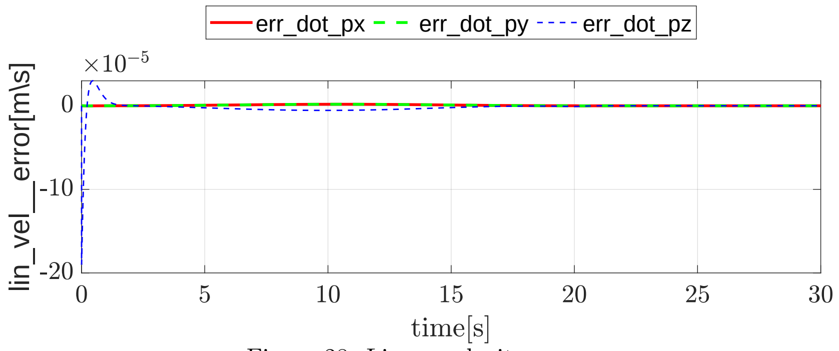


Figure 28: Linear velocity errors

And here I'll show the results of the errors for the angular part:

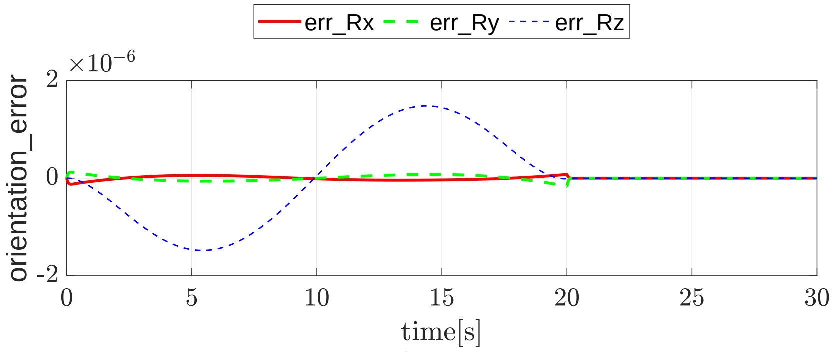


Figure 29: Attitude errors

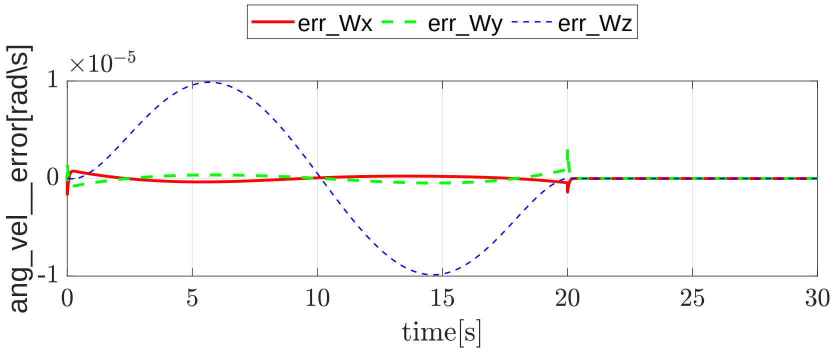


Figure 30: Angular velocity errors

We choose the total thrust, u_T , and the desired reduced attitude, $R_{b,d}e_3$,for the third body-fixed axis such that they stabilize the zero equilibrium of the tracking error for the translational dynamics. But, we can also say that this controller is designed such that the position tracking error converges to zero when there is no attitude error, and it is limited for non-zero attitude tracking errors to achieve asymptotic stability of the complete closed-loop dynamics.
As last thing I'll show the evolution of the control inputs:

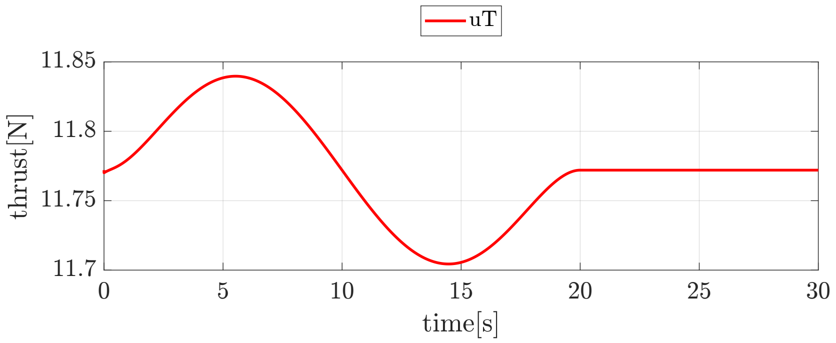


Figure 31: Total thrust

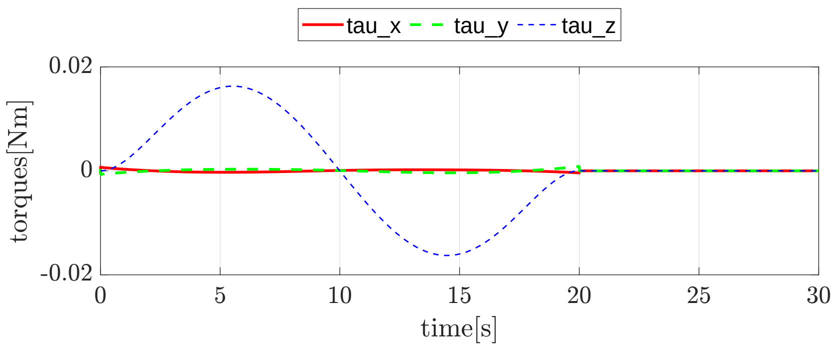


Figure 32: torques

As we can see at 20 seconds the drone reaches the goal and so the torques goes to 0 while the thrust remains at a constant value, this behavior can be interpreted as an hovering situation.

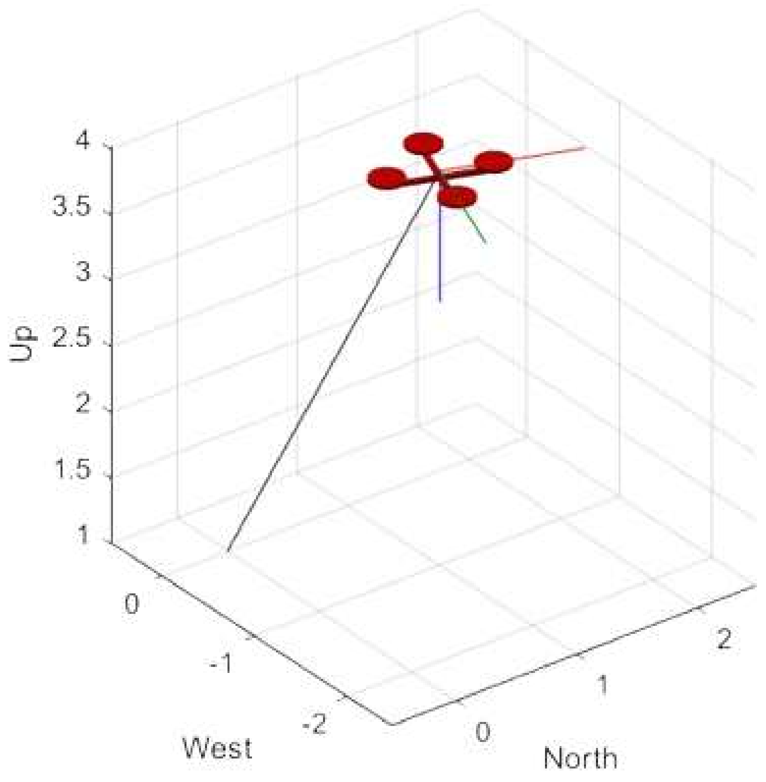


Figure 33: Final configuration