Contents

1 Theoretical questions

1.1 Problem 2.1

Determine $\boldsymbol{M_g}$ $\boldsymbol{M_v}$ $\boldsymbol{M_\omega}$

Starting from the force term:

$$\boldsymbol{f^B} = \underbrace{\boldsymbol{M_g a_g}}_{\boldsymbol{f_g^B}} + \underbrace{\boldsymbol{M_v f_r}}_{\boldsymbol{f_{rot}}} \tag{1}$$

$$\boldsymbol{f^B} = \boldsymbol{f_g^B} + \boldsymbol{f_{rot}} \tag{2}$$

Taking as Inertial Reference Frame (IRF) a NED Frame fixed in a generic point, we can easily say that, under hypothesis of flat earth and drone mass m constant, the gravitational force vector will be constant, with the only non null component along IRF z axis (which points downward). Its expression will be:

$$\boldsymbol{f_g^I} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{3}$$

Now, as the equations of the drone dynamics are expressed in the Body Reference Frame (BRF), we have to apply a change of basis of this vector in such a way to express its components in the BRF. This change of basis can be applied through the rotation matrix $\boldsymbol{R}$ transposed. Collecting the mass factor m outside the vector we get to our final equation.

$$\boldsymbol{f_g^B} = \boldsymbol{R(\lambda)^T} m \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{4}$$

where

$$\boldsymbol{M_g} = \boldsymbol{R(\lambda)^T} m \tag{5}$$

In order to exploit the expression of the second matrix we have to look carefully at the drone geometry. Considering all the forces generated by the four rotors, we can understand that these forces are displaced only along the BRF z axis for every flight configuration. Thus our force vector expressed in BRF will have only the z component non null and it will be the sum of the four rotor forces.

$$\boldsymbol{f_{rot}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{6}$$

where

$$\boldsymbol{M_v} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 \end{bmatrix} \tag{7}$$

The minus is due to the opposite force direction with the BRF z axis.

We have now to analyse the torque term:

$$\boldsymbol{n} = \boldsymbol{M_\omega f_r} \tag{8}$$

Again, taking into consideration the drone geometry we can relate the expressions of roll, pitch and yaw to our rotor forces. The variable d represents the arm between the drone centre of gravity and the single rotor force application point. The yaw term is given by the torque reactions between rotors and the drone body. We can relate them to the rotor forces thanks to a given relation which imply a constant value c equal for every rotor $n_i = cf_i$.

$$n_p = f_2 d - f_4 d \tag{9a}$$
$$n_q = f_1 d - f_3 d \tag{9b}$$
$$n_r = n_1 - n_2 + n_3 - n_4 \tag{9c}$$

All the signs are in accordance to the positive BRF axis. Writing the equations 9 in a matrix form we will obtain:

$$\boldsymbol{M_\omega f_r} = \begin{bmatrix} 0 & d & 0 & -d \\ d & 0 & -d & 0 \\ c & -c & c & -c \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{10}$$

1.2 Problem 2.2

If we rotate of 45° our drone around the z axes in the positive direction, we will have the x axis between rotor 1 and 2 while y axis between rotor 4 and 1. In this case every rotor force has a non null arm with respect to both roll and pitch axis and it will be $d \sin 45$. The first two rows of the matrix change while the third remains the same.

$$\boldsymbol{M_{\omega_{new}}} = \begin{bmatrix} -d\sin 45 & d\sin 45 & d\sin 45 & -d\sin 45 \\ d\sin 45 & d\sin 45 & -d\sin 45 & -d\sin 45 \\ c & -c & c & -c \end{bmatrix} \tag{11}$$

1.3 Problem 2.3

Defining a new control input:

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = -u_w \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{12}$$

This gives the equation:

$$-f_1 - f_2 - f_3 - f_4 = -u_w \tag{13}$$

$$\boldsymbol{M_{\omega_{new}} f_r} = \boldsymbol{u_\omega} \quad with \quad \boldsymbol{u_\omega} = \begin{bmatrix} u_p \\ u_q \\ u_r \end{bmatrix} \tag{14}$$

Combining equation 13 with equation 14, and considering that $\boldsymbol{u} = \begin{bmatrix} u_w \\ \boldsymbol{u_\omega} \end{bmatrix}$

The input can be expressed as

$$\boldsymbol{u} = \begin{bmatrix} -1 & -1 & -1 & -1 \\ -d\sin 45 & d\sin 45 & d\sin 45 & -d\sin 45 \\ d\sin 45 & d\sin 45 & -d\sin 45 & -d\sin 45 \\ c & -c & c & -c \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{15}$$

1.3.1 Problem 2.3.1

The usefulness of having the input in this form is that u_w is the input for the force along the BRF z axis while $\boldsymbol{u_\omega}$ is the input for the drone roll, pitch and yaw. In this way we have separated the two contributes.

1.3.2 Problem 2.3.2

By using equation 15 $\boldsymbol{f}$ can be expressed as a function of $\boldsymbol{u}$

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 & -1 \\ -d\sin 45 & d\sin 45 & d\sin 45 & -d\sin 45 \\ d\sin 45 & d\sin 45 & -d\sin 45 & -d\sin 45 \\ c & -c & c & -c \end{bmatrix}^{-1} \boldsymbol{u} \tag{16}$$

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} -1/4 & \frac{-1}{4d\sin 45} & \frac{1}{4d\sin 45} & \frac{1}{4c} \\ -1/4 & \frac{1}{4d\sin 45} & \frac{1}{4d\sin 45} & \frac{-1}{4c} \\ -1/4 & \frac{1}{4d\sin 45} & \frac{-1}{4d\sin 45} & \frac{1}{4c} \\ -1/4 & \frac{-1}{4d\sin 45} & \frac{-1}{4d\sin 45} & \frac{-1}{4c} \end{bmatrix} \begin{bmatrix} u_w \\ u_p \\ u_q \\ u_r \end{bmatrix} \tag{17}$$

1.4 Problem 2.4

The twelve non linear equations are given by the six kinematics relations and six dynamic relations: we have to insert into the dynamic one the action of our input $\boldsymbol{u}$. For the the Drone has a symmetric structure we will consider the inertia matrix as diagonal.

$$\boldsymbol{g} = \begin{pmatrix} (s(\phi)\ s(\psi) + c(\phi)\ c(\psi)\ s(\theta)) - v\ (c(\phi)\ s(\psi) - c(\psi)\ s(\phi)\ s(\theta)) + u\, c(\psi)\ c(\theta) \\ v\ (c(\phi)\ c(\psi) + s(\phi)\ s(\psi)\ s(\theta)) - w\ (c(\psi)\ s(\phi) - c(\phi)\ s(\psi)\ s(\theta)) + u\, c(\theta)\ s(\psi) \\ w\, c(\phi)\ c(\theta) - u\, s(\theta) + v\, c(\theta)\ s(\phi) \\ p + r\, c(\phi)\ tg(\theta) + q\, s(\phi)\ tg(\theta) \\ q\, c(\phi) - r\, s(\phi) \\ \frac{r\, c(\phi)}{c(\theta)} + \frac{q\, s(\phi)}{c(\theta)} \\ r\, v - q\, w - g\, s(\theta) \\ p\, w - r\, u + g\, s(\theta)\ s(\phi) \\ q\, u - p\, v + \frac{u_w + g\, m\, c(\phi)\, c(\theta)}{m} \\ \frac{l + (J_y - J_z) qr}{J_x} \\ \frac{m + (J_z - J_x) pr}{J_y} \\ \frac{n + (J_x - J_y) pq}{J_z} \end{pmatrix} \tag{18}$$

1.5 Problem 2.5

We have to study an equilibrium condition in which the drone is in hover in a fixed position and with a fixed yaw angle. In order to fully determine $\boldsymbol{x_0}$ we have to consider that:

- in hover roll and pitch angle are null, while yaw angle is defined as constant: $\boldsymbol{\lambda_0} = \begin{bmatrix} 0 \\ 0 \\ \psi_0 \end{bmatrix}$
- in hover u, v and w are null: $\boldsymbol{v_0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$
- in hover p and q are null and r will be null as well because of the hypothesis of constant yaw angle: $\boldsymbol{\omega_0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

Thus our equilibrium state vector will be

$$\boldsymbol{x_0} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 0 \\ 0 \\ \psi_0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{19}$$

If we evaluate the drone dynamics equations in equilibrium condition we can figure out that our input has to satisfy some specific conditions in order to grant the equilibrium.

$$\begin{cases} \boldsymbol{0} = \boldsymbol{f^B(u_0)} \\ \boldsymbol{0} = \boldsymbol{n(u_0)} \end{cases} \tag{20}$$

Reminding equation 8 and 14 we can immediately say that $\boldsymbol{u_{\omega_0}} = \boldsymbol{0}$.

The first equation, instead, tells us that $u_{w_0} = mg$

Thus our equilibrium input vector will be

$$\boldsymbol{u_0} = \begin{bmatrix} mg \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{21}$$

1.6 Problem 2.6

By inserting the equilibrium condition $\boldsymbol{x_0}$ in the Jacobian matrix calculated using the Matlab symbolic toolbox, we obtained the following $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices:

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \cos(\psi_0) & -\sin(\psi_0) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sin(\psi_0) & \cos(\psi_0) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{22}$$

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & \frac{1}{J_x} & 0 & 0 \\ 0 & 0 & \frac{1}{\mathrm{J_y}} & 0 \\ 0 & 0 & 0 & \frac{1}{J_z} \end{pmatrix} \tag{23}$$

1.7 Problem 2.7

Laplace transforms of $\delta \boldsymbol{x}$ and $\delta \boldsymbol{u}$:

$$G_\phi(s) = \frac{\Phi(s)}{U_p(s)} = \frac{1}{J_x s^2} \tag{24}$$

$$G_\theta(s) = \frac{\Theta(s)}{U_q(s)} = \frac{1}{J_y s^2} \tag{25}$$

$$G_\psi(s) = \frac{\Psi(s)}{U_r(s)} = \frac{1}{J_z s^2} \tag{26}$$

$$G_x(s) = \frac{X(s)}{U_q(s)} = -\cos\psi_0 \frac{g}{J_y s^4} \tag{27}$$

$$G_y(s) = \frac{Y(s)}{U_p(s)} = \cos\psi_0 \frac{g}{J_x s^4} \tag{28}$$

$$G_z(s) = \frac{Z(s)}{U_w(s)} = \frac{1}{m s^2} \tag{29}$$

Comment on $G_x(s)/G_y(s)$: Used superposition, $U_p = 0/U_q = 0$, the output X(s)/Y(s) if U_q/U_p is acting alone respectively.

1.8 Problem 2.8

X(s) related to $\Theta(s)$:

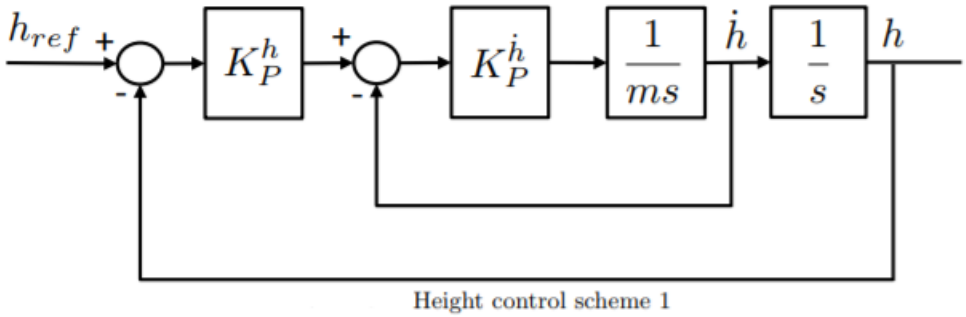
$$X(s) = -\cos\psi_0 \frac{g}{s^2}\Theta - \sin\psi_0 \frac{g}{s^2}\Phi \tag{30}$$

Y(s) related to $\Phi(s)$:

$$Y(s) = \cos\psi_0 \frac{g}{s^2}\Phi - \sin\psi_0 \frac{g}{s^2}\Theta \tag{31}$$

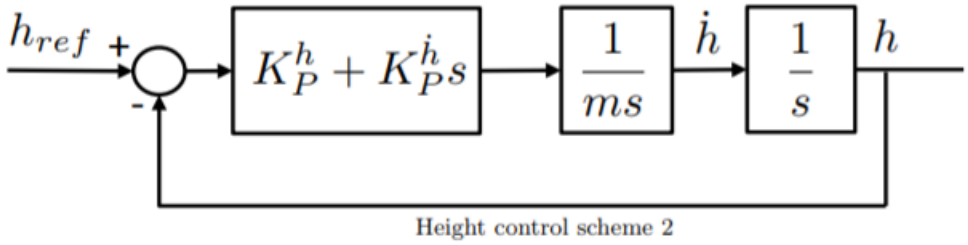
It is possible to control δx independently from $\delta\theta$ because by keeping $\delta\theta$ fixed δx can be controlled with $\delta\phi$. This kind on control is not possible when ψ_0 is exactly equal to $0 + k\pi$.

1.9 Problem 2.9



Height control scheme 1

Figure 1: Scheme1



Height control scheme 2

Figure 2: Scheme2

Analysing the general features of the two control schemes we can say that the first scheme Fig(1) may be more precise as it includes a closed loop on the $\dot{h}$ but it requires a sensor to measure it, so

it is more subject to noise diffusion inside the closed loop. The second scheme Fig(2) includes just one closed loop with a proportional derivative controller: this makes the system less precise but at least it requires just the acquisition of h.

Performing some runs with Simulink changing the proportional and integrative gains we could get the results shown in Fig(3). We can see the different behaviour of the two closed control loop in different situations: we notice that the first scheme is in general quicker in getting to the reference state, except when both gains are very small. In this case, however, the second control scheme has a non negligible overshoot.

1.10 Problem 2.10

By applying the block rules for the transfer functions we get:

$$G_{clh}(s) = \frac{K_P^h K_P^{\dot{h}}}{s(ms + K_P^{\dot{h}}) + K_P^h K_P^{\dot{h}}} \tag{32}$$

1.11 Problem 2.11

Again applying the block rules we get:

$$G_{cl\theta}(s) = S_\theta \frac{(K_P^\theta s + K_I^\theta) K_P^q}{s^2(J_{yy}s + K_P^q) + (K_p^\theta s + K_I^\theta) K_P^q} \tag{33}$$

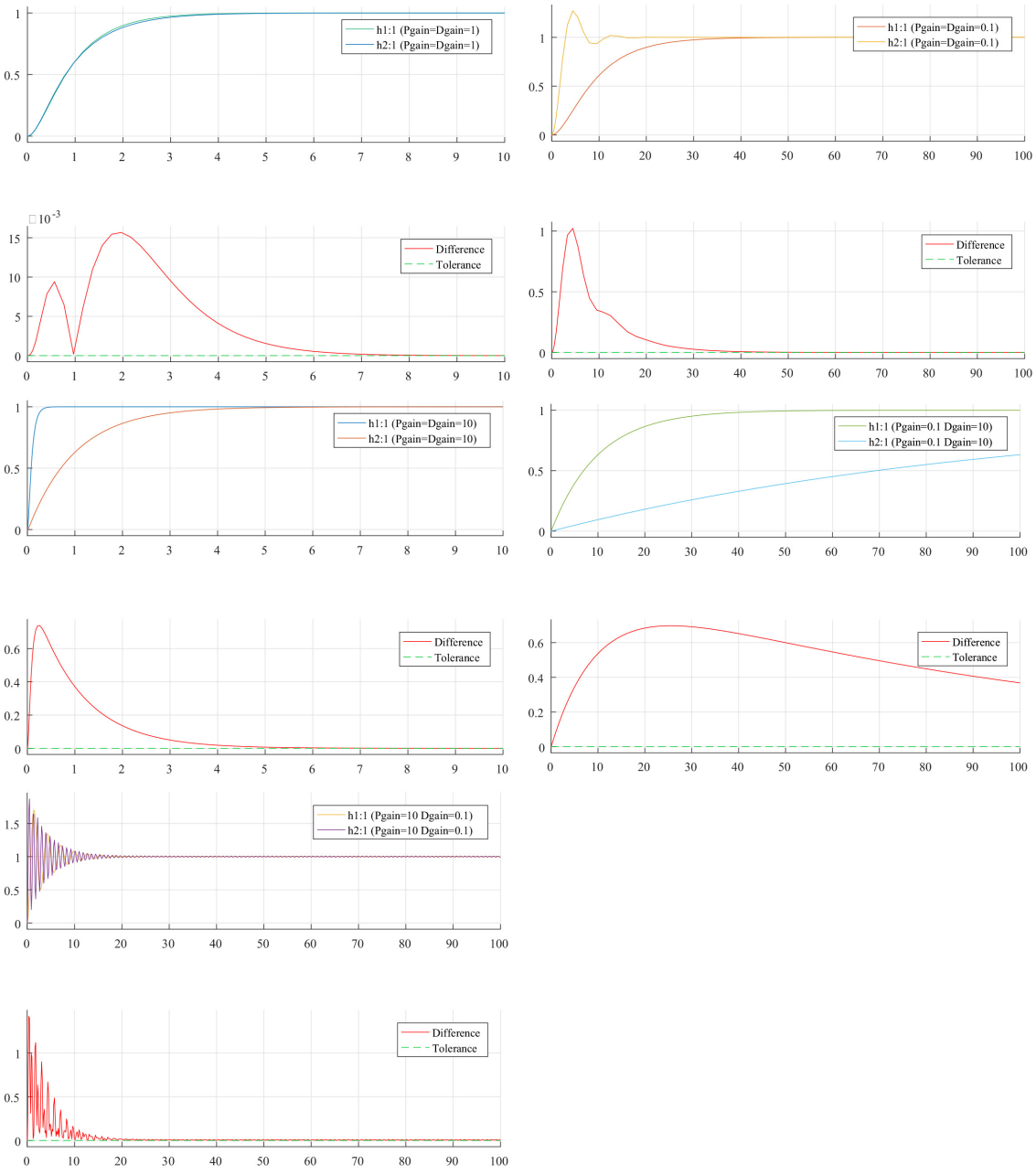


Figure 3: Runs with different gains

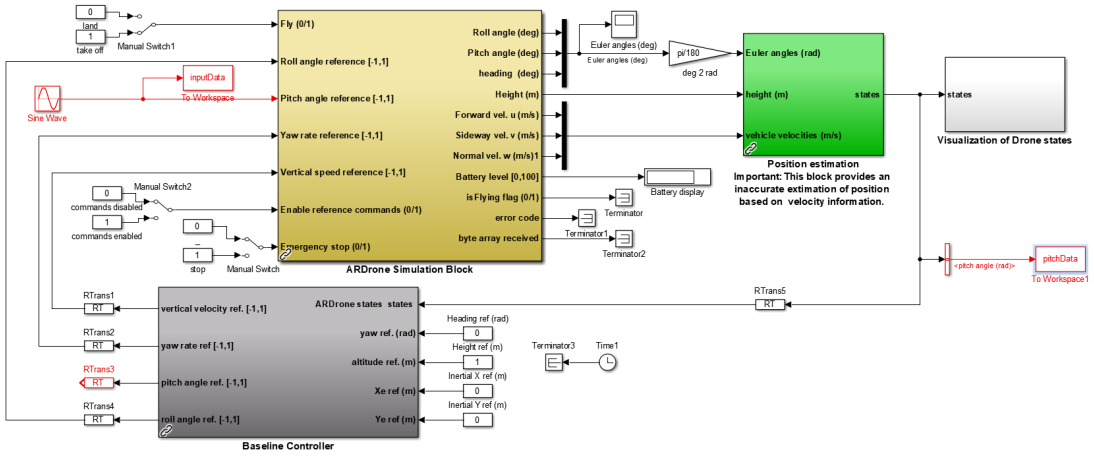


Figure 4: Modified Model

2 Simulations Questions

2.1 Problem 3.1

In order to collect all the required data, the drone Simulink model has been modified in a way shown in Fig(4). In this way different pulsation values can be set for the sinusoidal input, recording both input and output signals as well.
Once collected all the data each run has been analysed using the fast Fourier transform Matlab Function of both input and output. After some manipulations, the amplitude of the output signal has been processed. Each run gives a single point into the Bode diagram, thus the result would be a series of points in Fig(5).

2.2 Problem 3.2

Using the Nlinfit Matlab function, which perform a non linear regression for a given input and output vector, the transfer function of the system has been estimated:

$$G_{est}(s) = \frac{2.527s + 4.912}{s^2 + 3.958s + 12.03} \tag{34}$$

The plot of the Bode diagram of this transfer function together with the simulation points in Fig(6) can prove that the estimation has been performed correctly.

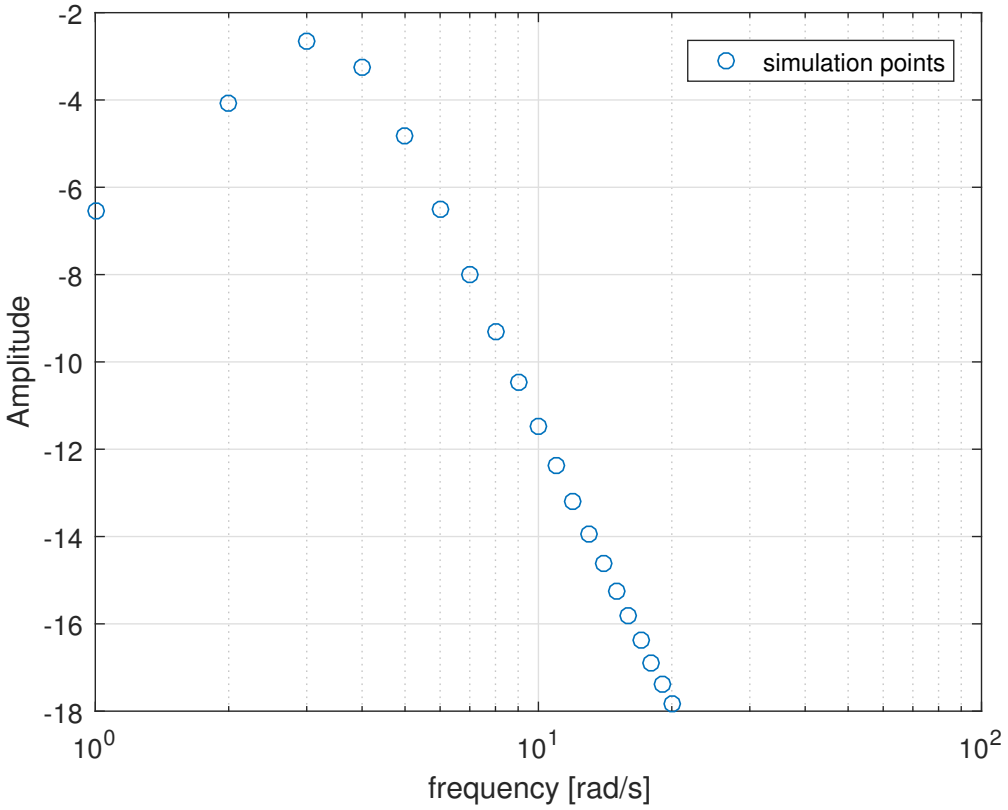


Figure 5: Bode Diagram of Simulation Data

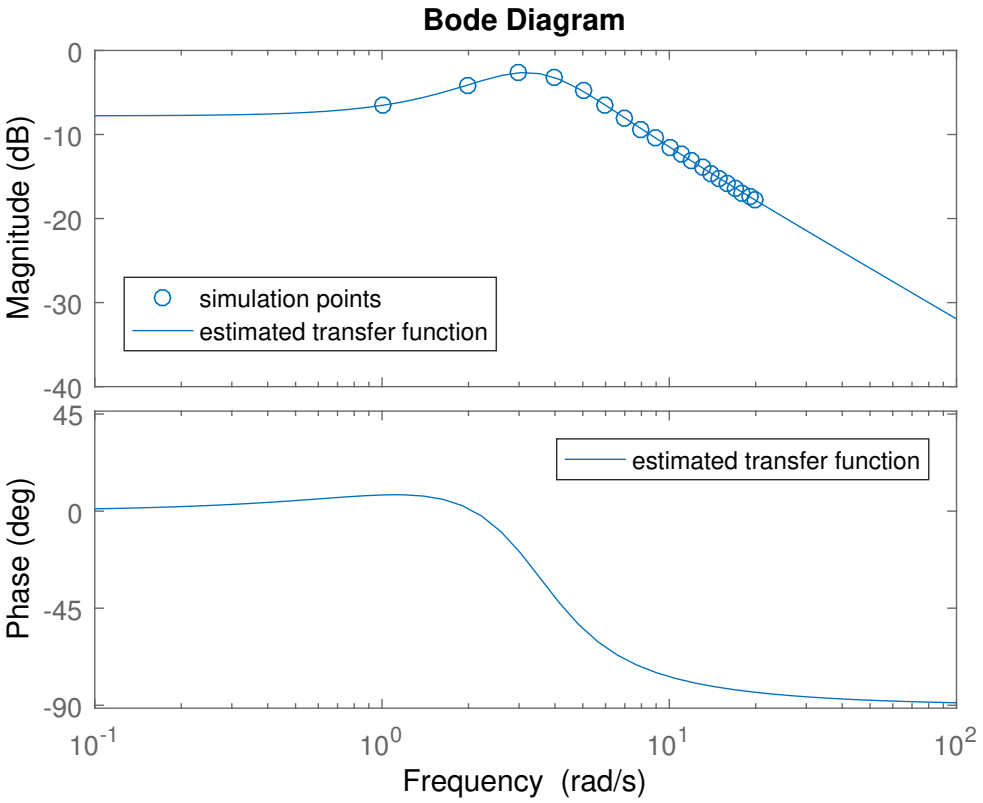


Figure 6: Estimated TF Bode Plot

2.3 Problem 3.3

The main difference between the transfer function (34) and (33) is basically the order of the characteristic polynomial (denominator of the transfer function). In case of the transfer function (33), a third order polynomial is present, which means three poles in the Bode diagram. In the case of the estimated transfer function (34), instead, a second order polynomial is present, thus just two poles. The only way in which this second order approximation could work is when one of the three poles of (33) is at a very high frequency in the Bode diagram.
In order to prove this fact a third order estimation has been calculated for the simulation data, and the result is:

$$G^*_{est}(s) = \frac{1.396 \cdot 10^9 s + 2.713 \cdot 10^9}{s^3 + 5.523 \cdot 10^8 s^2 + 2.186 \cdot 10^9 s + 6.646 \cdot 10^9} \tag{35}$$

Cleaning the second order term of the denominator the transfer function becomes:

$$G^*_{est}(s) = \frac{2.527s + 4.912}{1.811 \cdot 10^{-9} s^3 + s^2 + 3.958s + 12.03} \tag{36}$$

In this case the term s^3 could be neglected as it is multiplied by an extremely small coefficient. Plotting the pole and zero map of this third order estimated transfer function in Fig(7) is easy to understand that one of the three poles is orders of magnitude far from the other two and from the zero.

This considerations bring to the conclusion that this approximation is valid until the coefficient ahead of the s^3 is around 10^{-9} and becomes less precise when this coefficient grows. Analysing the symbolic expression of (33) we can easily say that the s^3 coefficient is $\frac{J_{yy}}{K^q_P}$ so if we want to use the transfer function (34) instead of the complete third order one, we should be sure that this ratio is small enough.

2.4 Problem 3.4

The root locus of the closed loop transfer function (33) can be drawn taking into consideration the behaviour of the roots of the its denominator with a changing value of K^θ_P. This procedure has been applied for both (34) and (35) in order to see if the approximation from the third to the second order implies any alteration in the system stability.
The results shown in Fig(8) and in Fig(9) prove that the system remains stable for every positive value of K^θ_P. Moreover the two graphs does not display any difference, leading us to the conclusion that our approximation is correct.

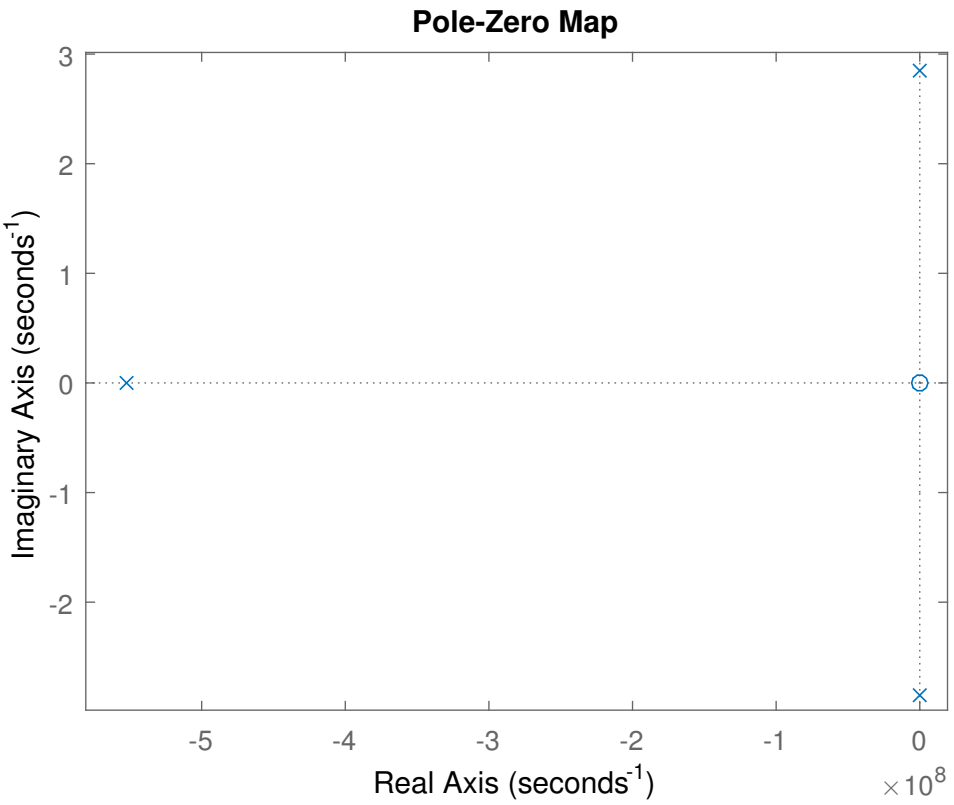


Figure 7: Poles and Zeros Map

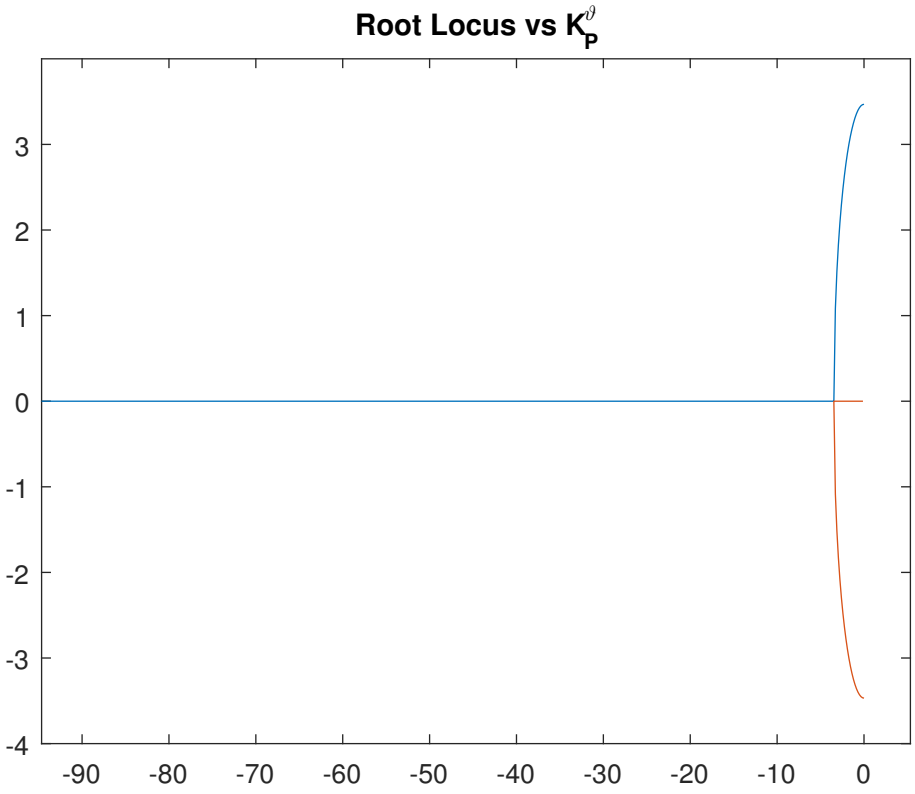


Figure 8: Root Locus of 2nd Order System

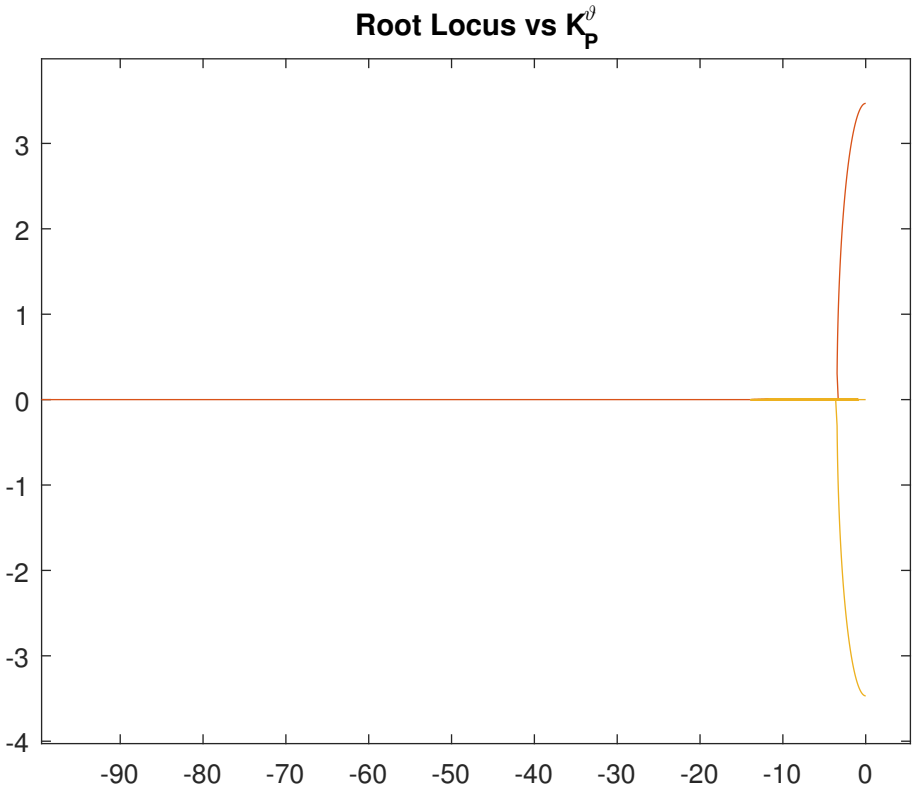


Figure 9: Root Locus of 3rd Order System

3 Experiments

3.1 Experiment 3

After performing the experiments in the arena, the data collected has been processed in a way to show the Bode plot of the real system in Fig(10). We can notice that the behaviour of the experimental points is less regular if compared to the simulation diagram. This is obviously due to all the disturbances present in the real environment, such as the boundary of the working area, the presence of other drones nearby, etc...

3.2 Task 3.6

It is still possible to identify an estimated transfer function from all the points: in this case the very first point at $1[rad/sec]$ frequency has been neglected because it would bring silly results in the approximation.

The approximated transfer function has been calculated with the same procedure of the simulation part:

$$F_{est}(s) = \frac{1.078s + 11.78}{s^2 + 6.462s + 32.18} \tag{37}$$

We can see in Fig(11) the Bode plot of the estimated transfer function compared with the experimental points and how the very first point is far away from the approximation.

In Fig(12) can be seen the difference between the two transfer function, one estimated by simulation while the other with experimental data: we can notice a small difference in the static gain and a less prominent peak in the zone of the two poles for the experimental transfer function . In Fig(13), instead, are plotted the pole and zeros in the imaginary plane.

3.3 Task 3.7

The last experiment for the Pitch control system identification required to acquire data from a variable input:

$$\theta_{ref}(t) = \begin{cases} 0, & 0 \leq t < 15s \\ 0.2, & 15 \leq t < 18 \\ 0, & t \geq 18 \end{cases} \tag{38}$$

The experimental data has been compared to a simulation run with the drone Simulink Model. The result is shown in Fig(14). This graph can be splitted into three main part:

- $0 < t < 15$: theoretically nothing should happen in this part, as the simulation graph confirms. However in the real test environment many disturbances are present. At the very beginning a negative pitch angle is recorded probably due to a sort of asymmetry during the take off.

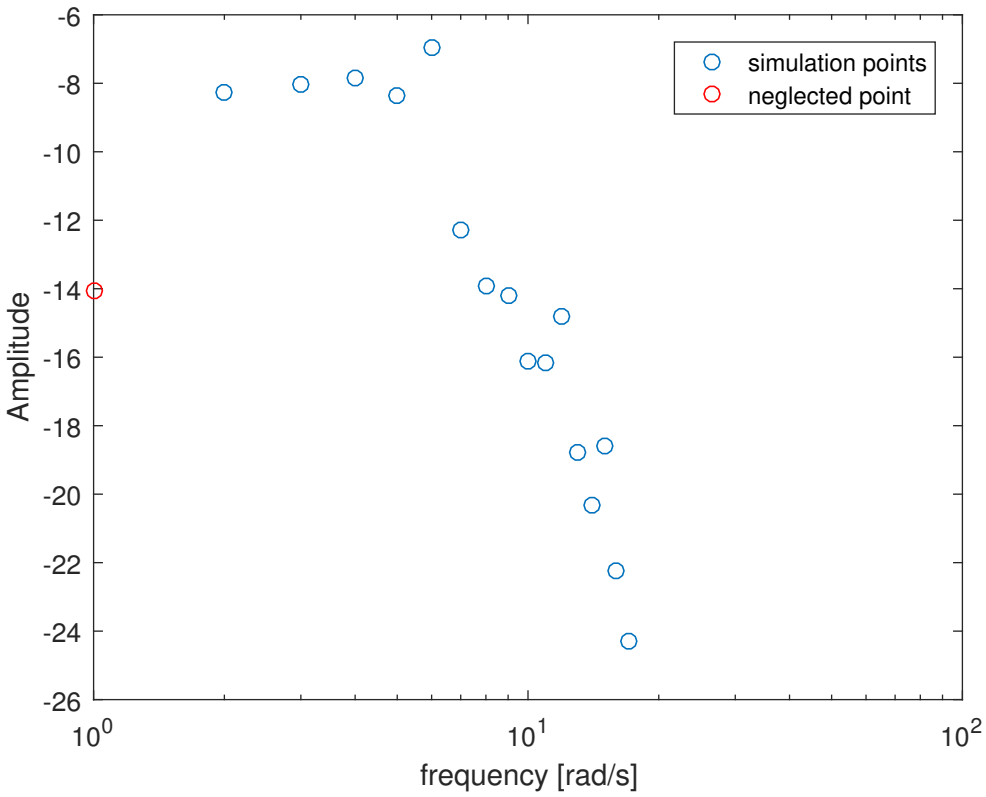


Figure 10: Real Data Bode Plot

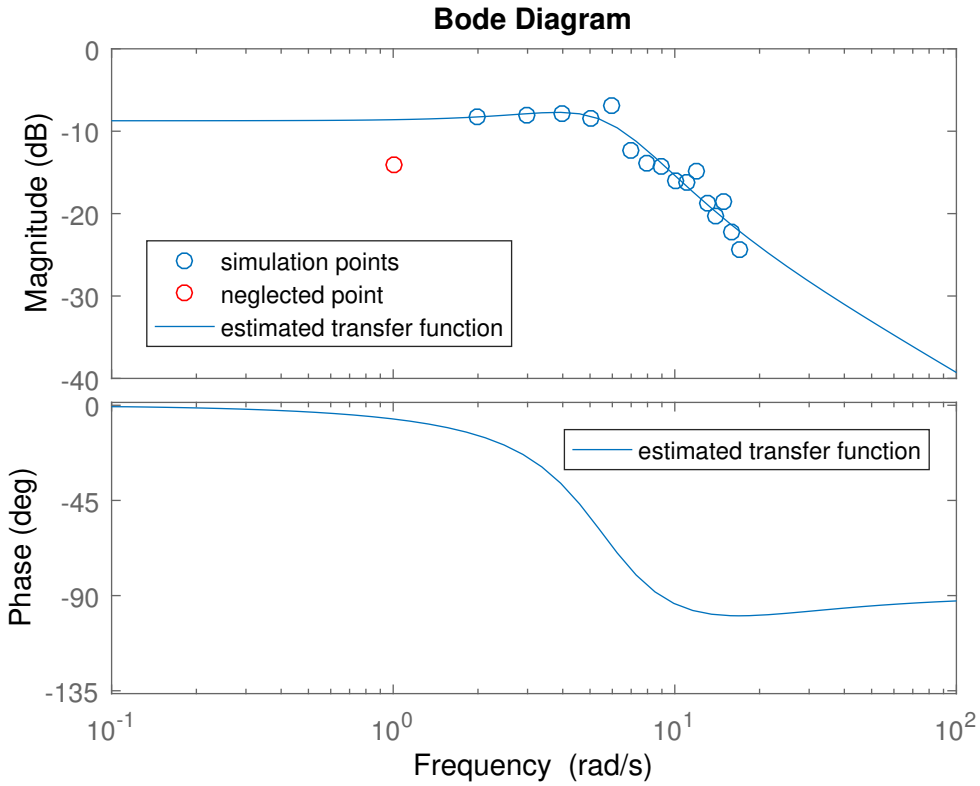


Figure 11: Estimated Transfer Function Bode Plot

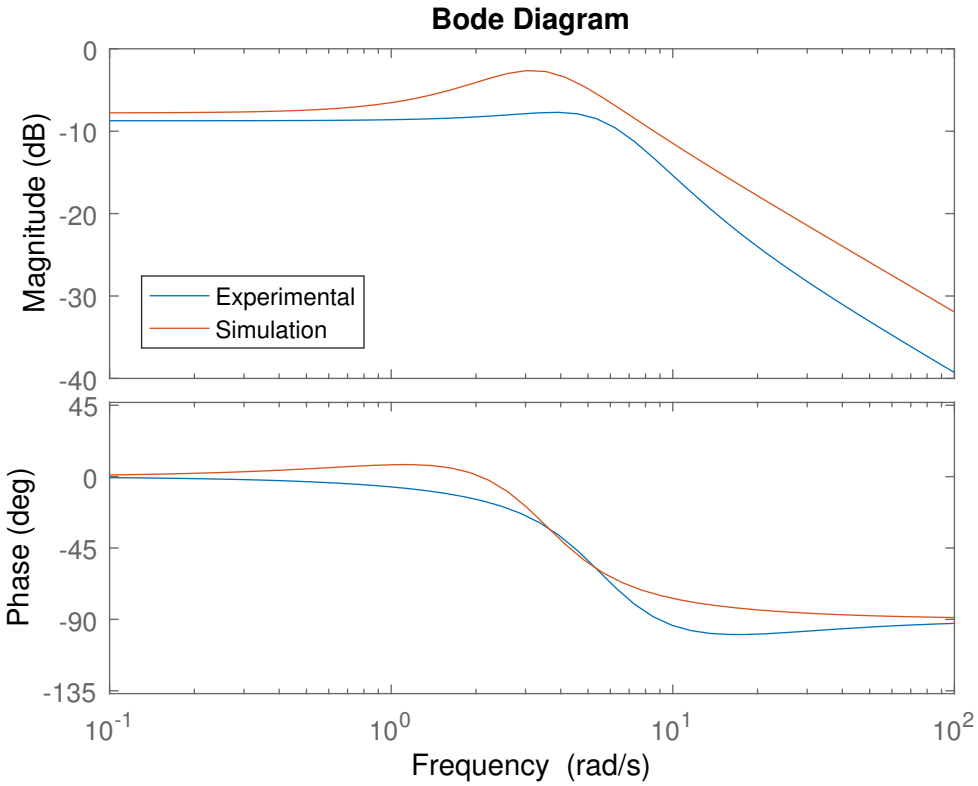


Figure 12: Experimental and Simulation Comparison

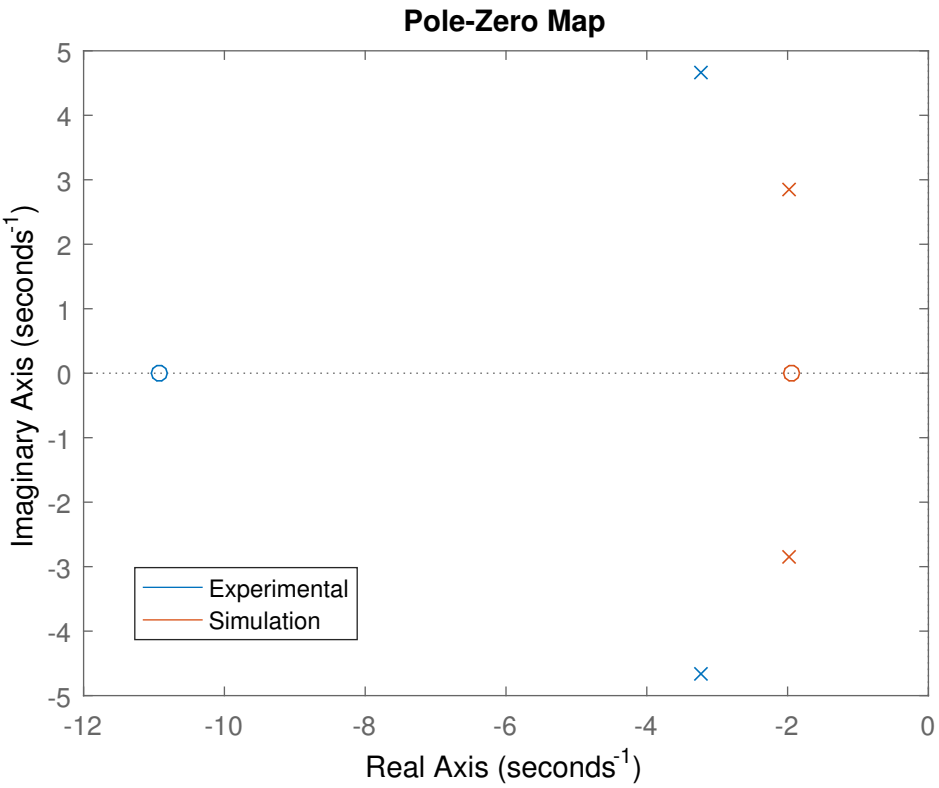


Figure 13: Pole and Zeros Map Comparison

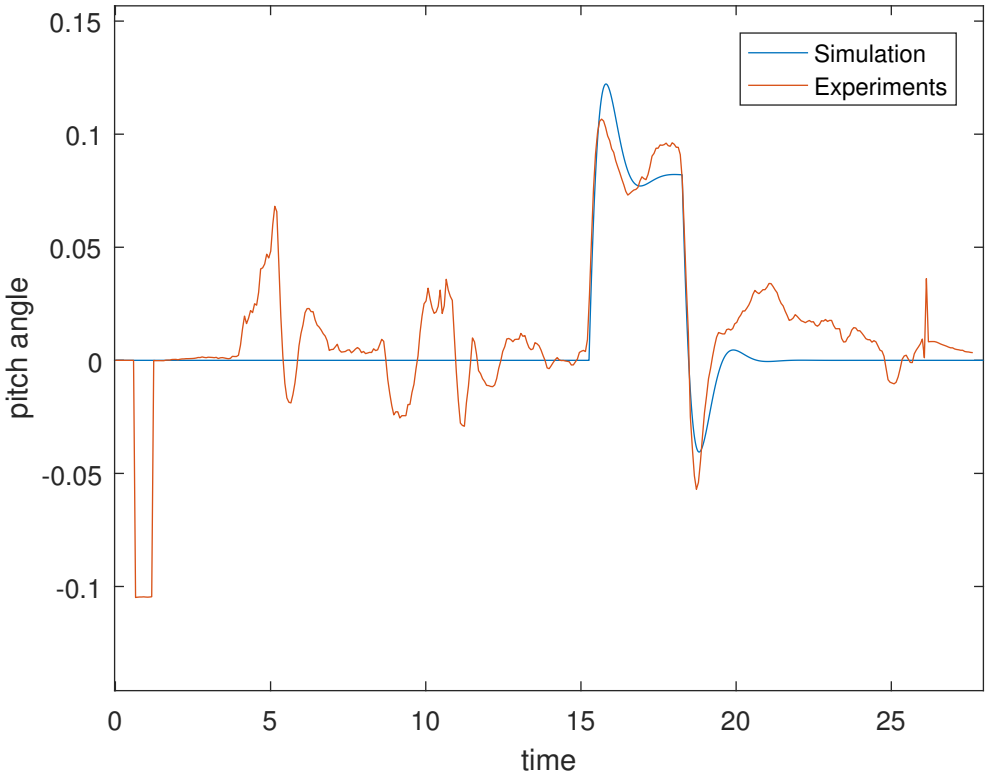


Figure 14: Pitch Response

After this short time (around 2 seconds) the pitch angle is stabilised in a neighborhood of the reference: obviously it is never set at the precise reference value as all the disturbances previously said are present.

- $15 \leq t < 22$: in this region a non null input is given to the system. The steepness of the time response is equal for both experimental and simulation data, while some differences can be found in the peak and in the static gain: the experimental response has a lower overshoot and a higher static gain. This last information can not be completely validated but performing other experiments: the short time in which the input is given does not allow us to say if this higher gain is truly a higher gain or just a temporary effect of some noise or disturbance. Once the input is brought back to zero the inverse process can be observed: in this case the experimental response is the one with the higher overshoot.
- $22 \leq t < +\infty$ again here it can be observed the presence of disturbances that drive our pitch angle away from the set reference.

3.4 Experiment 4

The height feedback has been disconnected from the baseline controller and the Simulink model has been modified outside to get a clear view of the model. The modified Simulink model can be viewed in figure 15. A switch has been added to separate the two systems (see Fig(16)). The first system is for the drone to take off to 1 meter. After 12 seconds the switch changes to the experimental gain and the step starts after 15 seconds.

The data used in this task has to be extracted from "step response starts here" to "ends here" as shown in Fig(16). This is performed in Matlab and then imported as input and output in system identification toolbox. Fig(17) shows all the extracted data for each step response with the corresponding input data.

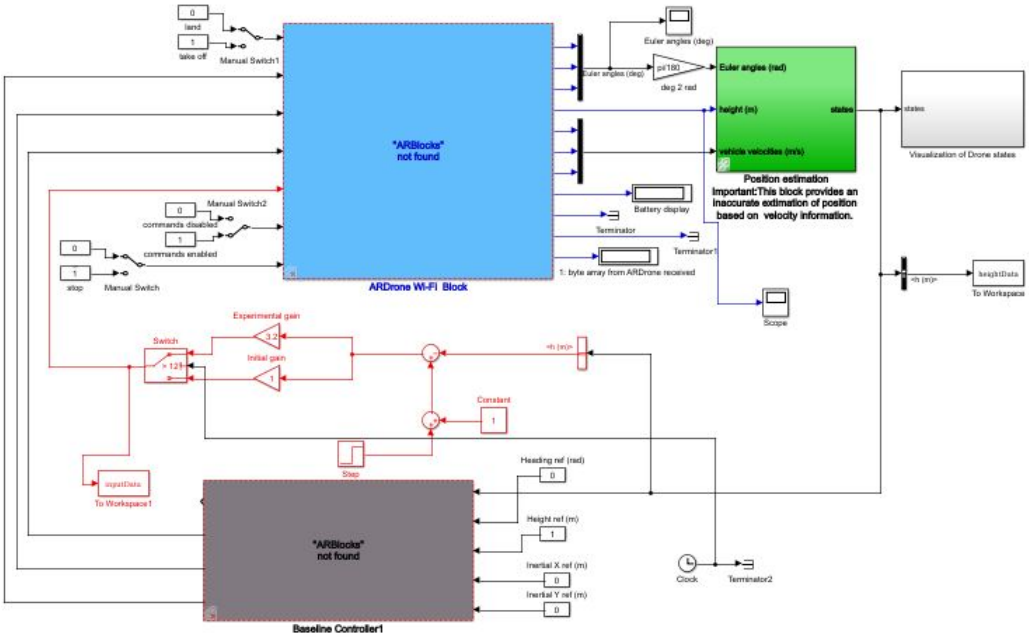


Figure 15: Modified Simulink Model

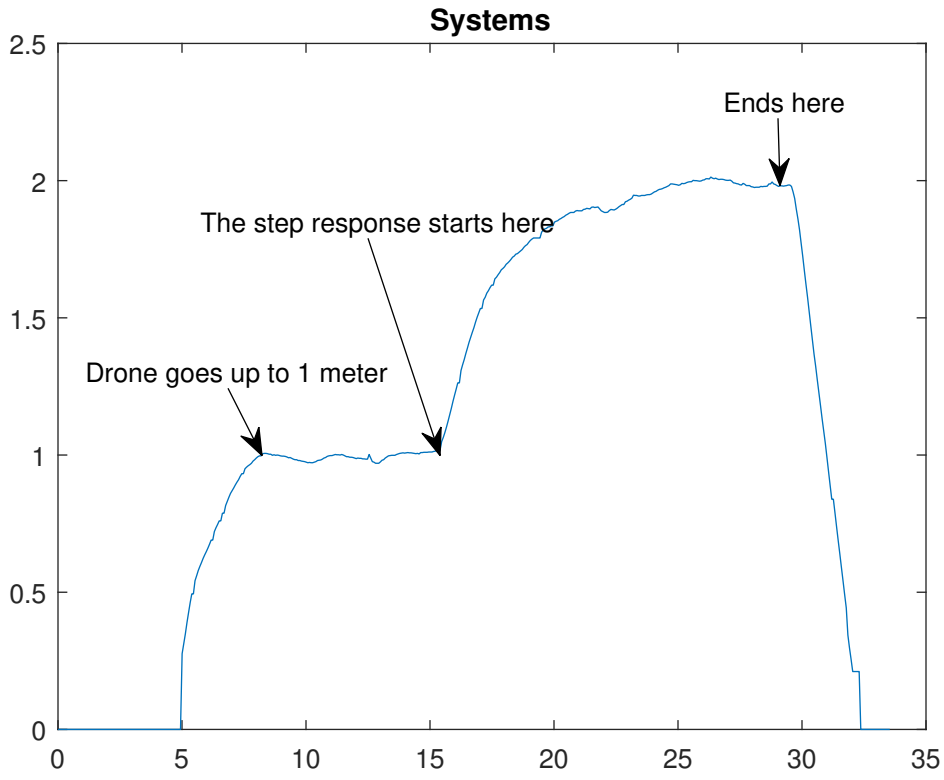


Figure 16: The two systems obtained as the drone takes off and the step starts.

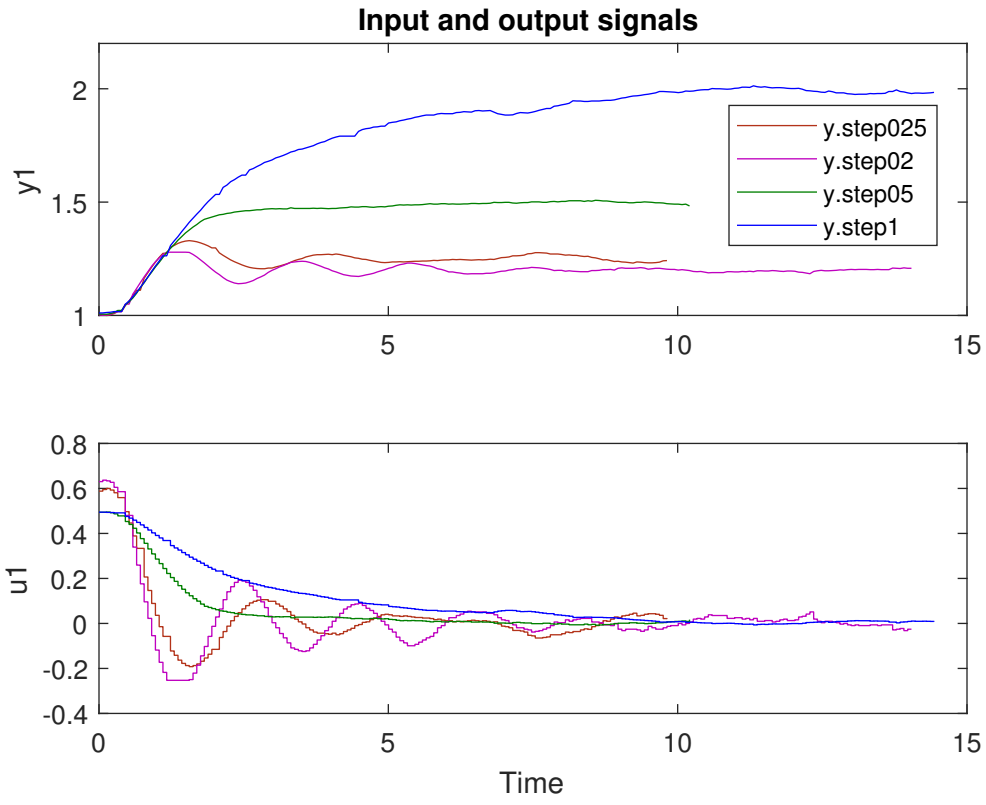


Figure 17: Plots of measured extracted output and input for the step response

2 poles	Best fit Step 1	Best fit Step 0.5	Best fit Step 0.25	Best fit Step 0.2
$Tf1=\frac{-0.4326}{s^2+0.7963s+1.988\cdot10^{-10}}$	22.65	***72.11***	25.95	8.134
$Tf3=\frac{-1.266}{s^2+1.18s+6.753\cdot10^{-11}}$	-50.77	17.93	19.42	-6.176
$Tf5=\frac{2.287}{s^2+2.108s+0.001346}$	***50.82***	61.06	***70.99***	***69.26***
$Tf7=\frac{23.65}{s^2+12.03s+0.0007296}$	-54.51	-14.71	-6.215	-41.71
3 poles	Best fit Step 1	Best fit Step 0.5	Best fit Step 0.25	Best fit Step 0.2
$Tf2=\frac{0.628}{s^3+0.5624s^2+0.915s+9.495\cdot10^{-6}}$	***93.83***	75.55	27.24	10.18
$Tf4=\frac{2.035}{s^3+1.007s^2+2.549s+1.289\cdot10^{-10}}$	83.84	***87.24***	***39.47***	***18.66***
$Tf6=\frac{-6.372}{s^3+0.5624s^2+0.915s+9.495\cdot10^{-6}}$	-165	-28.91	13.63	-5.244
$Tf8=\frac{-7.782}{s^3+3.165s^2+4.222s+0.01468}$	-134.9	-26.73	14.51	-11.32

Figure 18: Table of all the approximated transfer functions. The numbers which are bold and underlined are the overall best fit, if it is only bold then it is the best fit for just the transfer functions with 2 poles or 3 poles.

3.5 Task 4.2

The approximated transfer function tf5 with 2 poles has the best fit for the step 1 response with an accuracy of 50.82%. This is not a satisfying approximation. However, tf1, manage to fit 72.11% to the measured data for step 0.5. Fig(23a) shows how the pink line follows the measured data with a tolerable fit. However, a second order transfer function usually has a tendency to overshoots, it is faster, and is the best approximation for the systems oscillating responses. For the measured responses for step 0.2 and step 0.25 the transfer function tf5 (Fig(18)) manage to approximate with a best fit of 69.26% and 70.99%. The data measured in these two step responses are sort of corrupted, but since the experiments were only executed once for each step. The poles considering Fig(19) are on the real axis and the system is therefore overdamped with a general homogeneous solution described by equation 39. The smallest pole will control the rate at which $y_h(t)$ decays to zero. Tf7 has one negative real pole with large magnitude compared with the others, this will decay rapidly, but the other real pole is close to the imaginary axis and will decay much slower. Combined with a large gain the transfer function is not even close to approximate the data. Tf3 has a negative gain which forces the output in negative direction in the beginning of step 1 and step 0.5. Then the transfer function overshoots and settles lower than the measured data.

$$y_{h_{tf5}}(t) = C_1 e^{-2.11t} + C_2 e^{-1.04 \cdot 10^{-4} t} \tag{39}$$

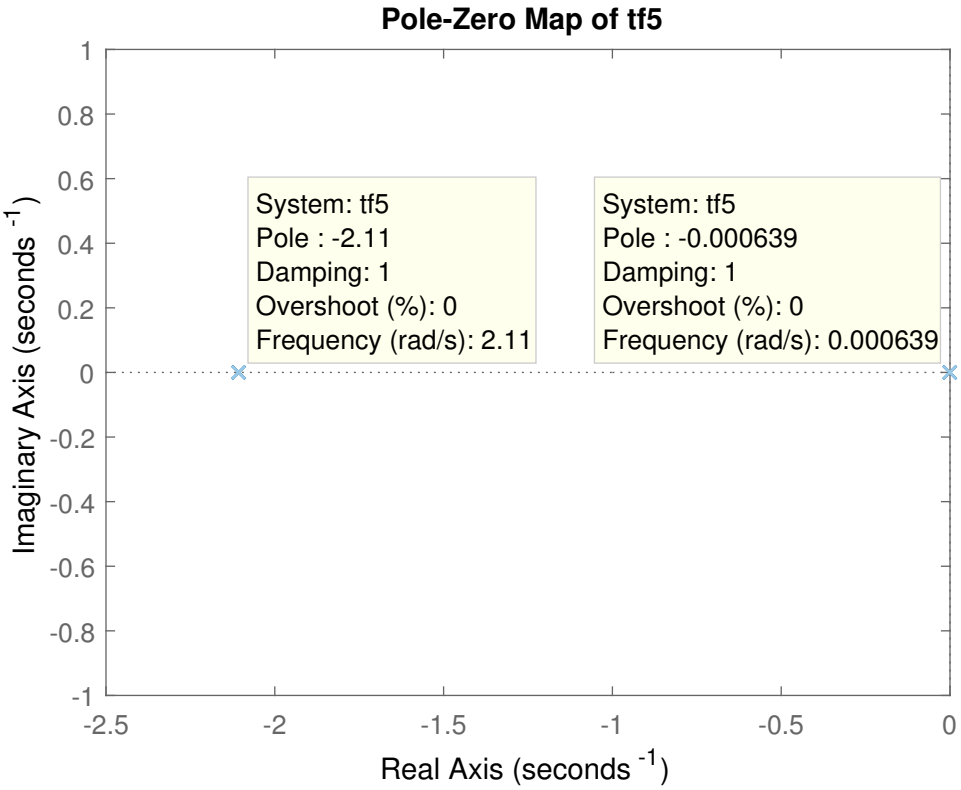


Figure 19: Pole zero map of the transfer function tf5 (2 poles)

3.6 Task 4.3

Looking at the pole-zero plot of tf2 in Fig(20) pair of complex conjugated poles and a real pole close to the imaginary axis are present. This means that the system is asymptotically stable since it approaches zero as t goes to ∞. Equation 40 is the total homogeneous response of the system. $\tau_1 \approx 3.56$ s and $\tau_2 \approx 9.61 \cdot 10^4$ the dominant long term response is thus τ_2. Transfer function tf2 has 3 poles and has a fit of 93.83% for step 1. The reasons for this great approximation can be explained by the fact that a third order transfer function decreases the overshoot and gives a slower response. The same goes for tf4's approximation of step 0.5. Furthermore, tf6 and tf8 do not fit for any of the data. The negative gains will make the output move in negative direction when a positive step is applied to the system. This can be seen in Fig(22b). The imaginary part of the complex conjugated pole pairs of the two transfer functions are quite large as can be seen in Fig(21). This will cause the sinus in the homogeneous solution of the two systems to oscillate with a larger frequency compared with tf2 and tf4.

$$y_{h_{tf2}}(t) = C_1 e^{-1.04 \cdot 10^{-5} t} + A e^{-0.281 t} \sin(0.914 t + \phi) \tag{40}$$

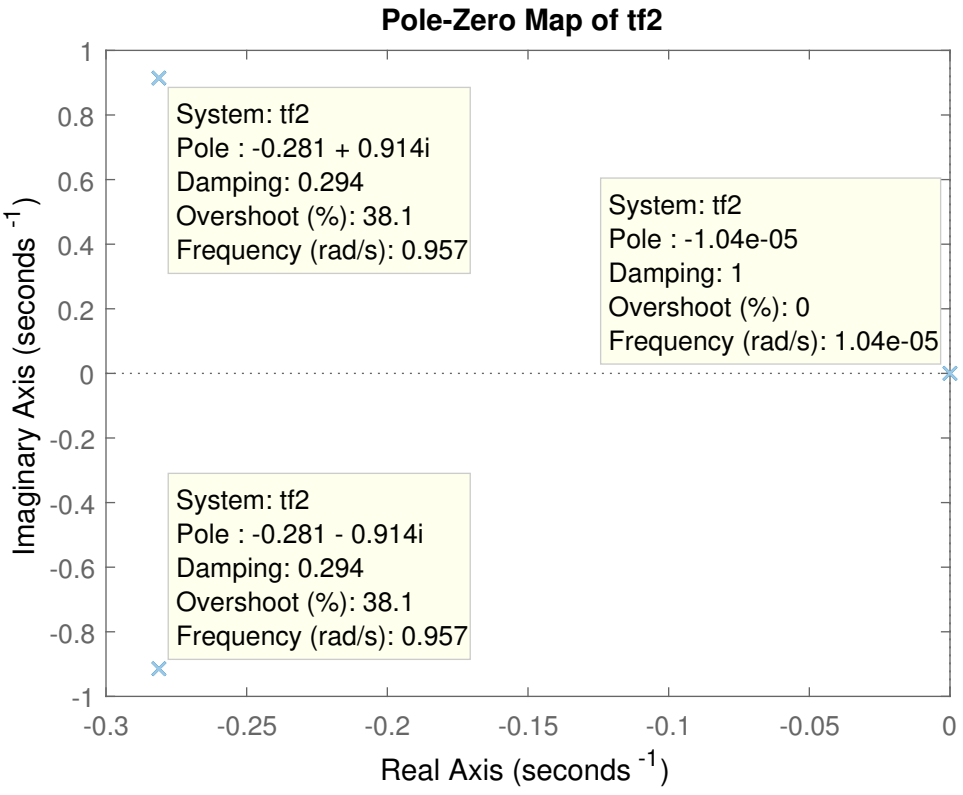


Figure 20: Pole zero map of the transfer function tf2 (3 poles)

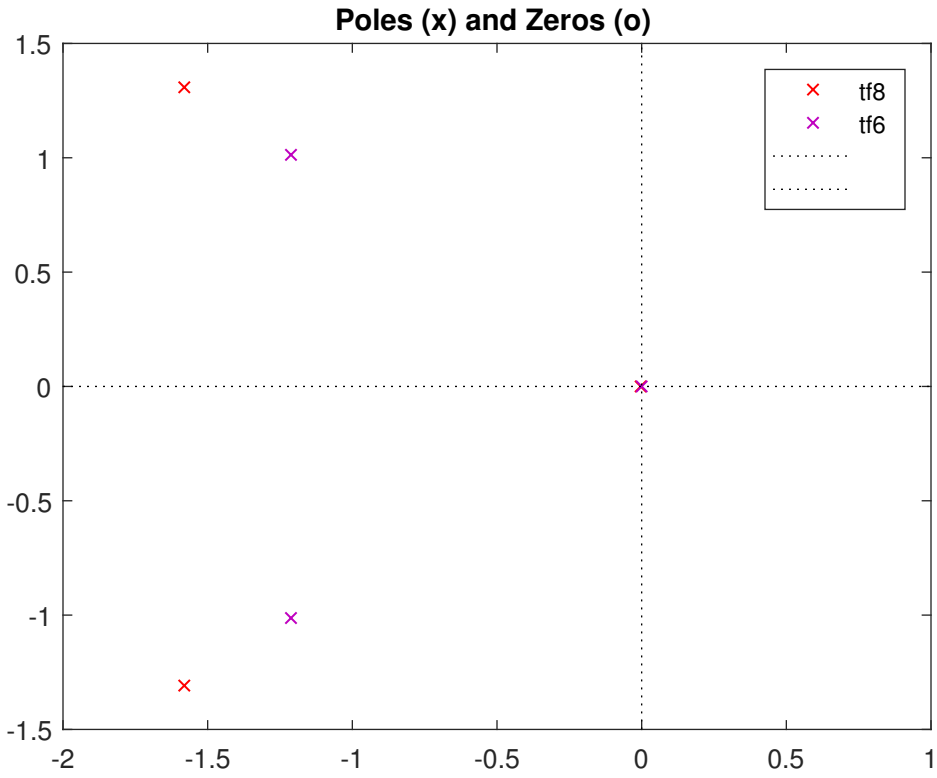
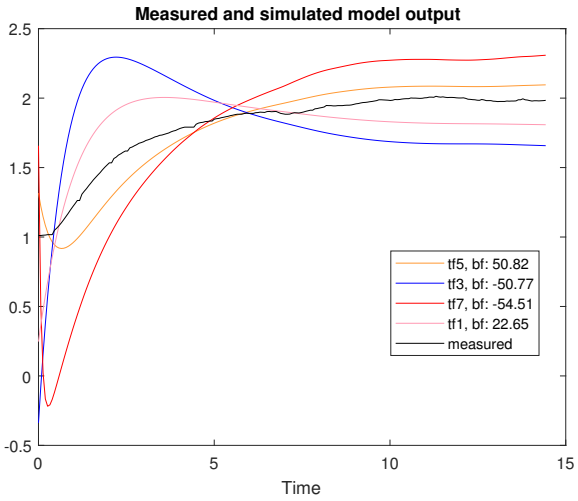
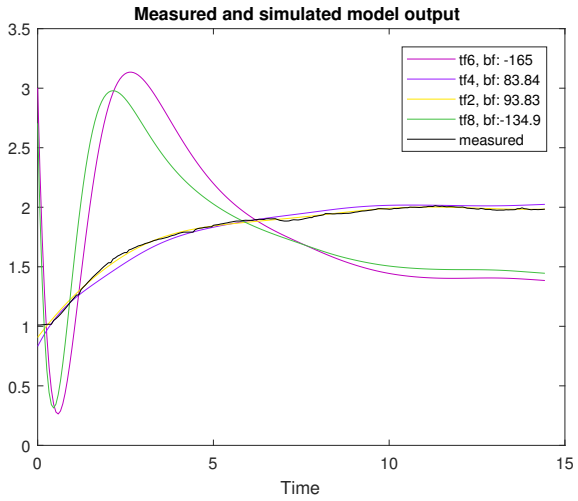


Figure 21: Pole zero map of the transfer function tf6 and tf8 (3 poles)

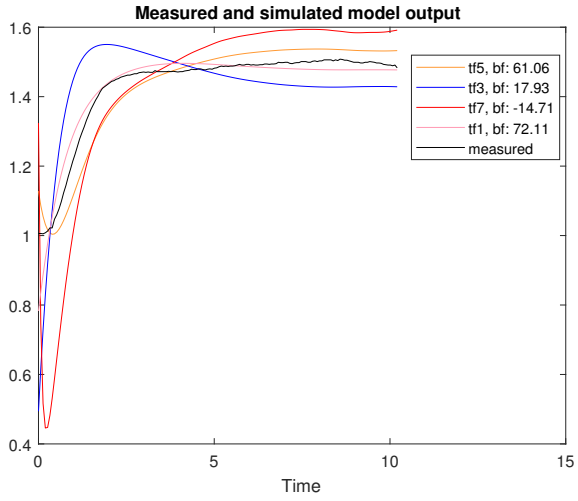


((a)) Transfer functions with 2 poles

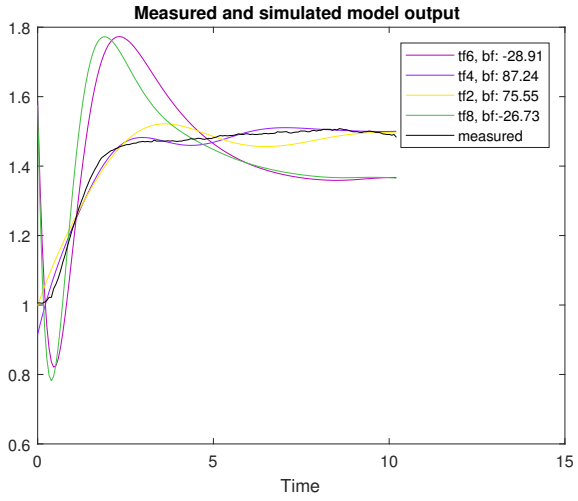


((b)) Transfer functions with 3 poles

Figure 22: Step 1: Approximated transfer functions

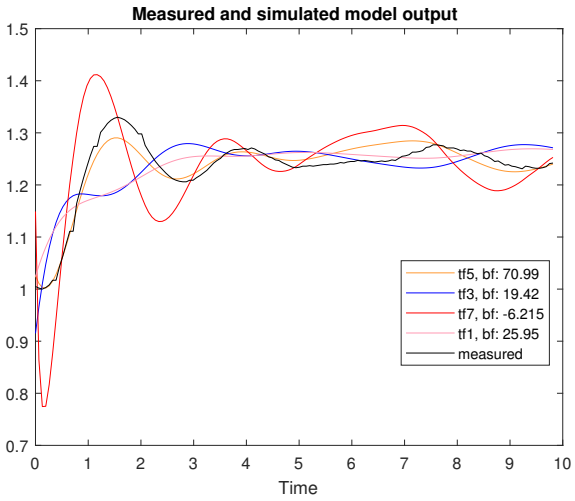


((a)) Transfer functions with 2 poles

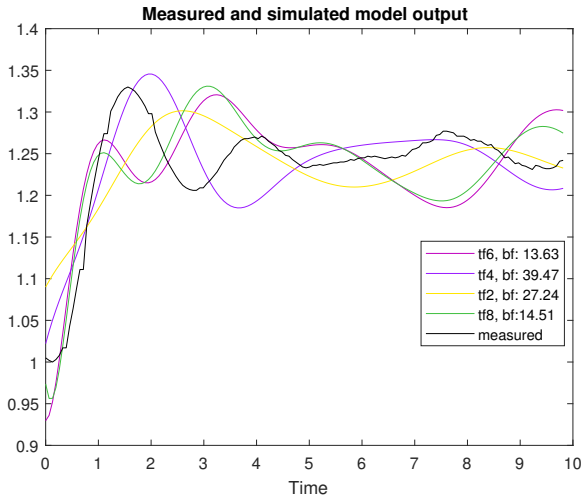


((b)) Transfer functions with 3 poles

Figure 23: Step 0.5: Approximated transfer functions

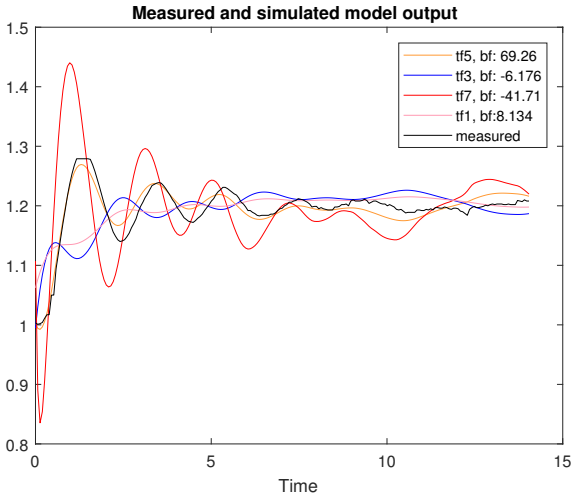


((a)) Transfer functions with 2 poles

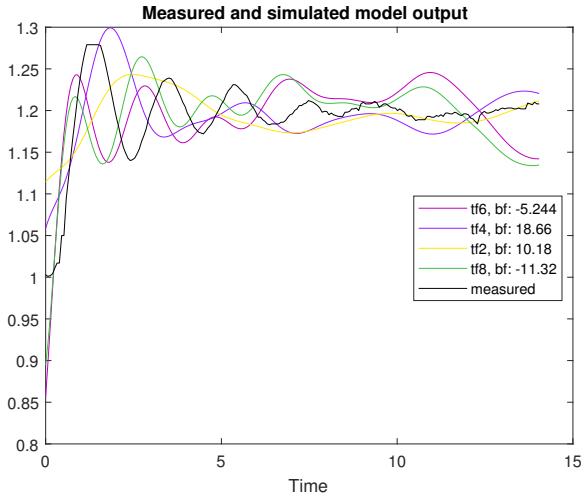


((b)) Transfer functions with 3 poles

Figure 24: Step 0.25: Approximated transfer functions



((a)) Transfer functions with 2 poles



((b)) Transfer functions with 3 poles

Figure 25: Step 0.2: Approximated transfer functions