63 bd De Brandebourg
94200 Ivry-sur-Seine

Au511 – Practical work

Control of aircraft

Aymeric LEGRAND
Mathys VINATIER

Teacher: M. Julien LUBRANO

TABLE OF CONTENTS

I – Control aircraft

IN FLIGHT OPERATING POINT

Studying an aircraft requires knowing some parameters without which nothing can be done. Thus, the altitude and the speed are the “must-know” numbers to determine the values of all the environment variables. Let us look at the table below:

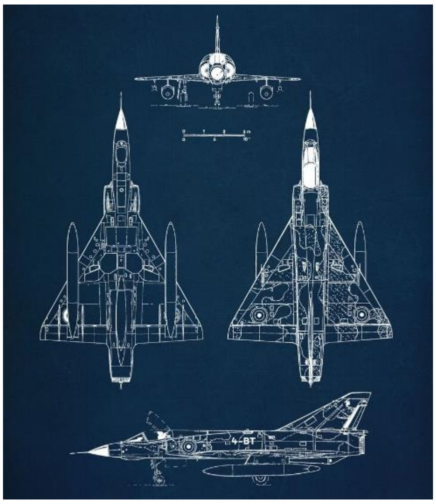
Alt (ft) \ Mach	0.76	0.91	1.18	1.29	1.49	1.68	1.88
575	11	12	13	14	15	16	17
3015	21	22	23	24	25	26	27
6115	31	32	33	34	35	36	37
10085	41	42	43	44	45	46	47
13105	51	52	53	54	55	56	57
16335	61	62	63	64	65	66	67
19365	71	72	73	74	75	76	77
22265	81	82	83	84	85	86	87

We decided to take the operating point 42, meaning:

- Altitude = **10085ft**
- Mach = **0.91**

AIRCRAFT CHARACTERISTICS

The plane : Dassault Mirage III



The **Dassault Mirage III** is a French supersonic fighter aircraft developed in the late 1950s by Dassault Aviation. Known for its distinctive delta-wing design, it was one of the first operational aircraft to achieve Mach 2. Originally conceived as an interceptor, the Mirage III proved highly versatile, adapting to roles such as ground attack and reconnaissance.

Exported in several continents (Asia, Southa America, Africa), it served in the air forces of over 20 countries and played a significant role in key conflicts, including the Six-Day War (1967). Its design, performance, and adaptability made it a landmark in aviation history, with some upgraded variants remaining in use today.

Technical sheet

Here are the technical characteristics of our fighter:

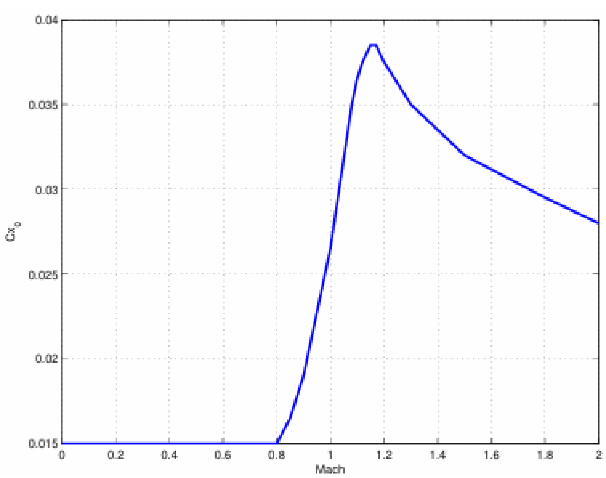
Total length	$l_t = \frac{3}{2} l_{ref}$
Mass	$m = 8400kg$
Aircraft centering (center of gravity position as % of total length)	$c = 52\%$
Reference surface (wings)	$S = 34m^2$
Radius of gyration	$r_g = 2.65m$
Reference length	$l_{ref} = 5.24m$

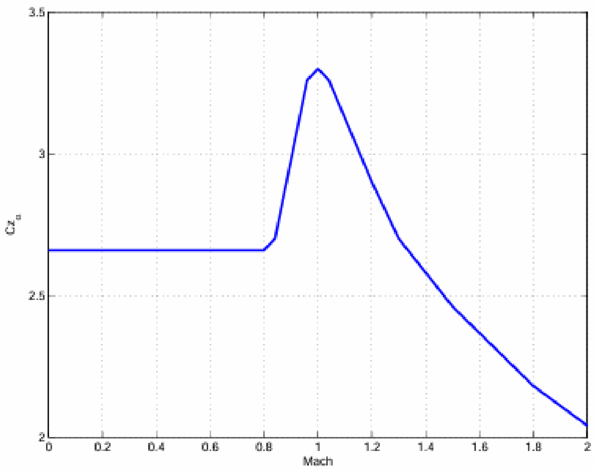
For the calculus of air density and speed of sound as a function of altitude, we will use the US 76 standard atmosphere model. More of that, we will consider the following hypothesis:

- Symmetrical flight, in the vertical plane (null sideslip and roll)
- Thrust axis merged with aircraft longitudinal axis
- Inertia principal axis = aircraft transverse axis (diagonal inertia matrix)
- Fin control loop: its dynamics will be neglected for the controller synthesis
- The altitude sensor is modeled by a 1st order transfer function with a time constant = 075 s

Aircraft aerodynamic model

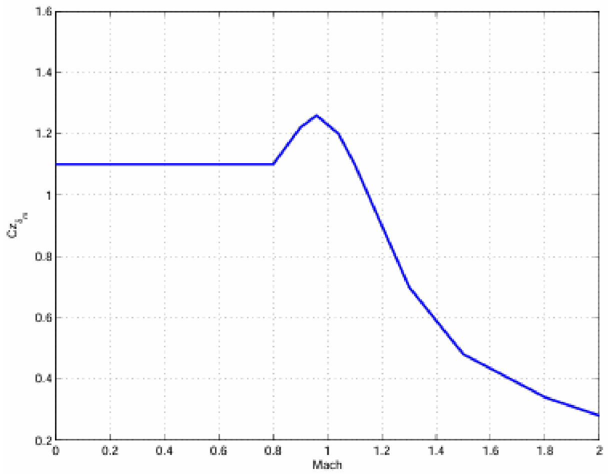
The aircraft aerodynamic coefficients for the longitudinal motion (drag, gradient of drag and lift, aerodynamic center for body and fins, polar coefficient and damping coefficient) are αgiven below as functions of Mach number.

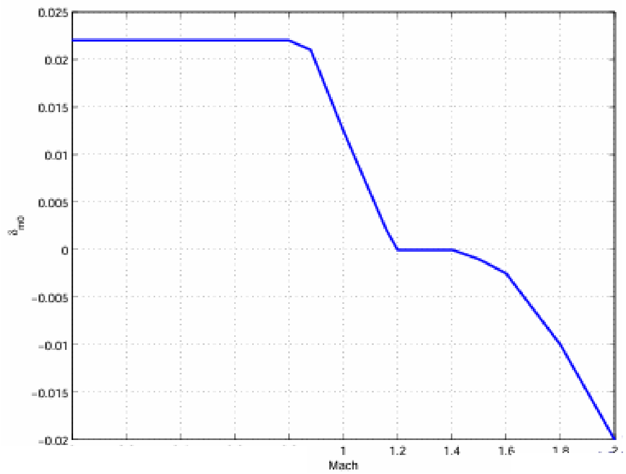




<u>Left</u> : drag coefficient for null incidence C_{x_0}

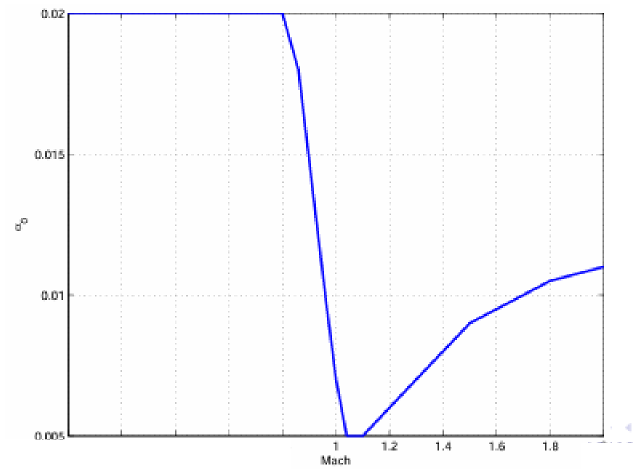
<u>Right</u> : Lift gradient coefficient $WRT\ \alpha\ C_{z_\alpha}\ (rad^{-1})$

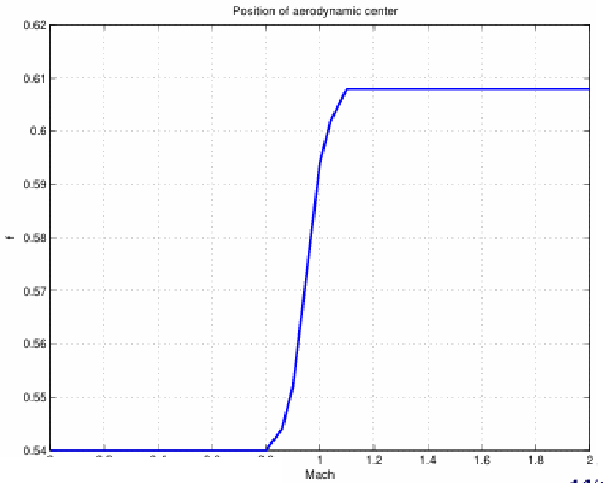




Left : lift gradient coefficient WRT δ_m $C_{z_{\delta_m}}$ (rad^{-1})

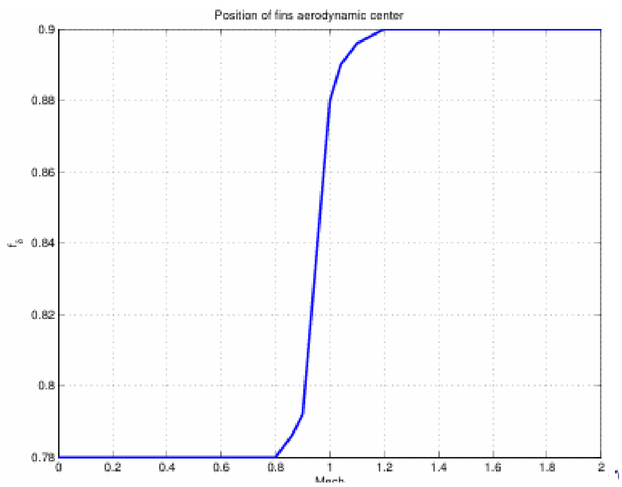
Right : Equilibrium fin deflection for null lift δ_{m_0} *(rad)*

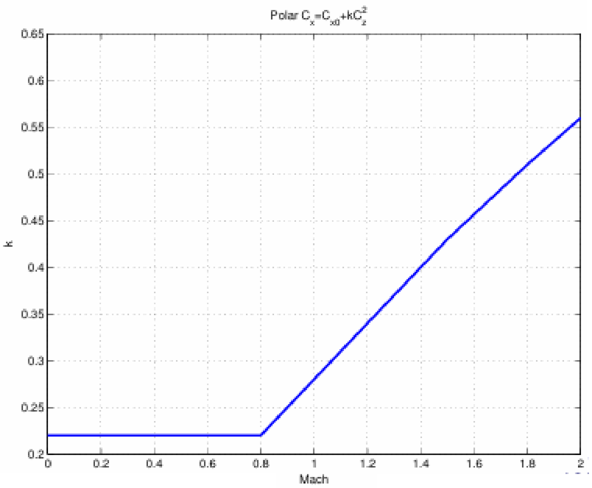




Left : incidence for null lift and null fin deflection α_0 *(rad)*

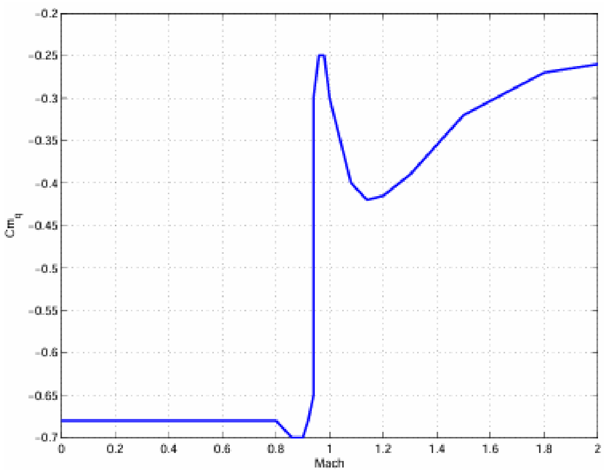
Right : Aerodynamic center of body and wings f





Left : Aerodynamic center of fins (pitch axis) f_δ

Right : polar coefficient k



Damping coefficient C_{m_q} *(s/rad)*

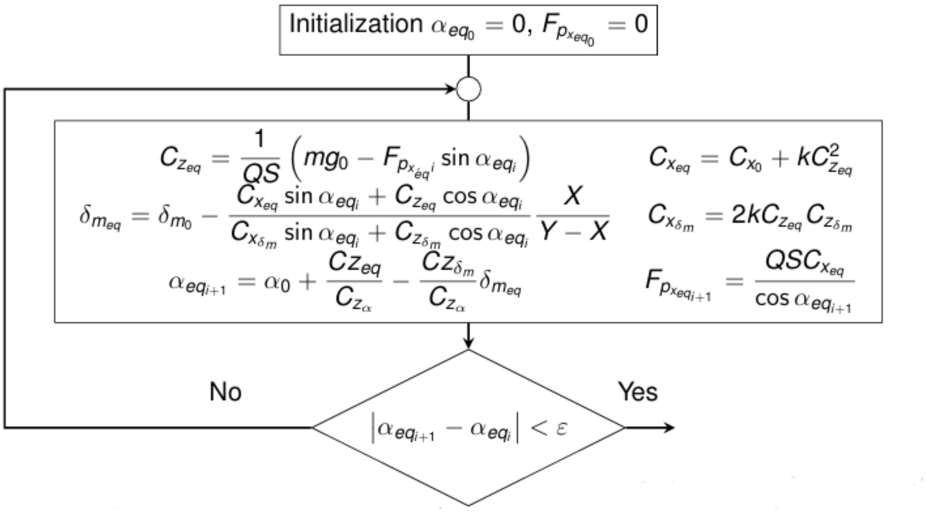
For M = 0.91, we can find the values of the missing parameters by reading them on the different graphics. Thus, we got:

Drag coefficient	C_{x_0}	0.022
Lift gradient coefficient $WRT\ \alpha$	C_{z_α}	3.2
Lift gradient coefficient $WRT\ \delta_m$	$C_{z_{\delta m}}$	1.25
Equilibrium fin deflection for null lift	δ_{m_0}	0.02
Incidence for null lift and null fin deflection	α_0	0.01
Aerodynamic center of body and wings	f	0.56
Aerodynamic center of fins	f_δ	0.81
Polar coefficient	k	0.26
Damping coefficient	C_{m_q}	-0.68

STUDY OF THE UNCONTROLLED AIRCRAFT

Equilibrium point computing algorithm

First, we will study the behavior of the aircraft without control. Then, we will develop open and close loops to control the plane. To do so, we will implement the following formulas on Python:



Then, we can compute the control matrices A and B, given the following state space model:

$$\dot{X} = AX + BU$$

$$\begin{pmatrix} \dot{V} \\ \dot{\gamma} \\ \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \\ \dot{z} \end{pmatrix} = \begin{pmatrix} -X_V & -X_\gamma & -X_\alpha & 0 & 0 & 0 \\ Z_V & 0 & Z_\alpha & 0 & 0 & 0 \\ -Z_V & 0 & -Z_\alpha & 1 & 0 & 0 \\ 0 & 0 & m_\alpha & m_q & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & V_{\acute{e}q} & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} V \\ \gamma \\ \alpha \\ q \\ \theta \\ z \end{pmatrix} + \begin{pmatrix} 0 \\ Z_{\delta_m} \\ -Z_{\delta_m} \\ m_{\delta_m} \\ 0 \\ 0 \end{pmatrix} (\delta_m)$$

According to the formulas of the *"Aircraft longitudinal dynamics"* course, we have:

$$X_V = \frac{2QSC_{x_{eq}}}{mV_{eq}} \qquad m_V = 0 \qquad Z_V = \frac{2QSC_{z_{eq}}}{mV_{eq}} \approx \frac{2g_0}{V_{eq}}$$

$$X_\alpha = \frac{F_{eq}}{mV_{eq}} \sin\alpha_{eq} + \frac{QSC_{x_\alpha}}{mV_{eq}} \qquad m_\alpha = \frac{QS\ell_{ref} C_{m_\alpha}}{I_{YY}} \qquad Z_\alpha = \frac{F_{eq}}{mV_{eq}} \cos\alpha_{eq} + \frac{QSC_{z_\alpha}}{mV_{eq}}$$

$$X_\gamma = \frac{g_0 \cos\gamma_{eq}}{V_{eq}} \qquad m_q = \frac{QS\ell_{ref}^2 C_{m_q}}{V_{eq} I_{YY}} \qquad Z_\gamma = \frac{g_0 \sin\gamma_{eq}}{V_{eq}}$$

$$X_{\delta_m} = \frac{QSC_{x_{\delta_m}}}{mV_{eq}} \qquad Z_{\delta_m} = \frac{QSC_{z_{\delta_m}}}{mV_{eq}}$$

$$X_\tau = -\frac{F_\tau \cos\alpha_{eq}}{mV_{eq}} \qquad m_{\delta_m} = \frac{QS\ell_{ref} C_{m_{\delta_m}}}{I_{YY}} \qquad Z_\tau = \frac{F_\tau \sin\alpha_{eq}}{mV_{eq}}$$

Thus, thanks to Python, we got:

$$A = \begin{pmatrix} -0.0251 & -0.0328 & -0.0545 & 0 & 0 & 0 \\ 0.0653 & 0 & 1.7656 & 0 & 0 & 0 \\ -0.0653 & 0 & -1.7656 & 1 & 0 & 0 \\ 0 & 0 & -23.4526 & -1.4566 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 298.7143 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 \\ 0.6848 \\ -0.6848 \\ -66.4184 \\ 0 \\ 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad D = [0]$$

From that, we can determine the poles, damping ratios and proper pulsations of short period and phugoid modes:

	Pole	Damping ratio (ζ)	Proper pulsation (w_n)
	0	1	0
Short period mode	[-1.611+4.84j; -1.611-4.84j]	0.3158	5.101 rad/s
Phugoid mode	[-0.012+0.042j; [-0.012-0.042j]	0.2711	0.044 rad/s

According to the values of the proper pulsation, we can determine the flight mode of each pole. Indeed, we can see a big gap between the first pair of complexes and the second one (x100). Therefore, each pair of complexes is associated with one mode; the fastest one with the short period mode and the slowest with the phugoid mode.

Study of short period mode

Two variables are associated with the short period mode : α and q. Therefore, we got two transfer functions, one for each:

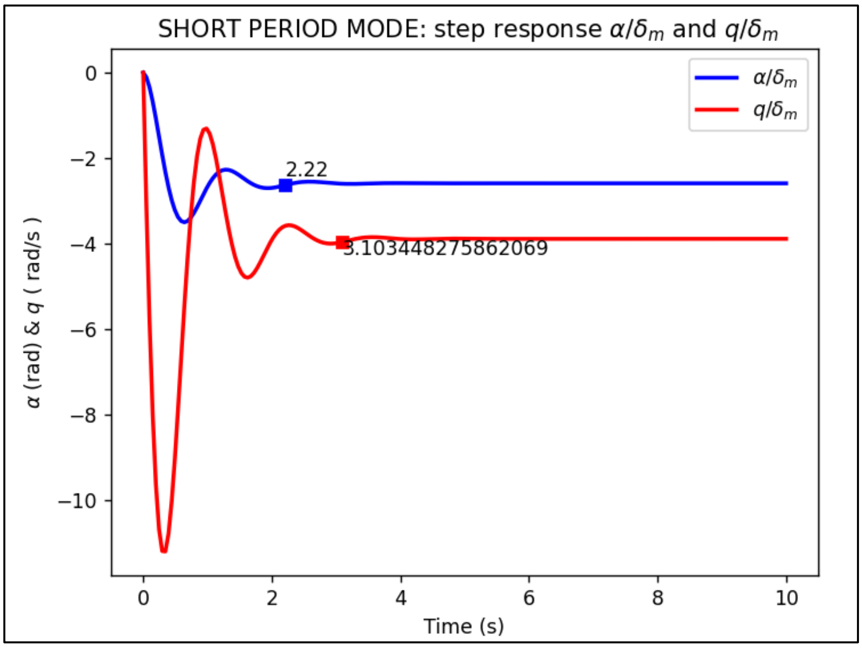
For $\frac{q}{\delta_m}$:

$$\boldsymbol{TF = \frac{-66.42\,s - 101.2}{s^2 + 3.222\,s + 26.02}}$$

For $\frac{\alpha}{\delta_m}$:

$$\boldsymbol{TF = \frac{-0.6848\,s - 67.42}{s^2 + 3.222\,s + 26.02}}$$

Giving the following curves:



By reading the graphics, we can determine the settling time of the two variables, meaning the time after which the curve is stable:

- α: 2.22 s
- q: 3.10 s

The settling time is pretty low, which is very satisfying.

Study of phugoid mode

Two variables are associated with the short period mode : α and q. Therefore, we got two transfer functions, one for each:

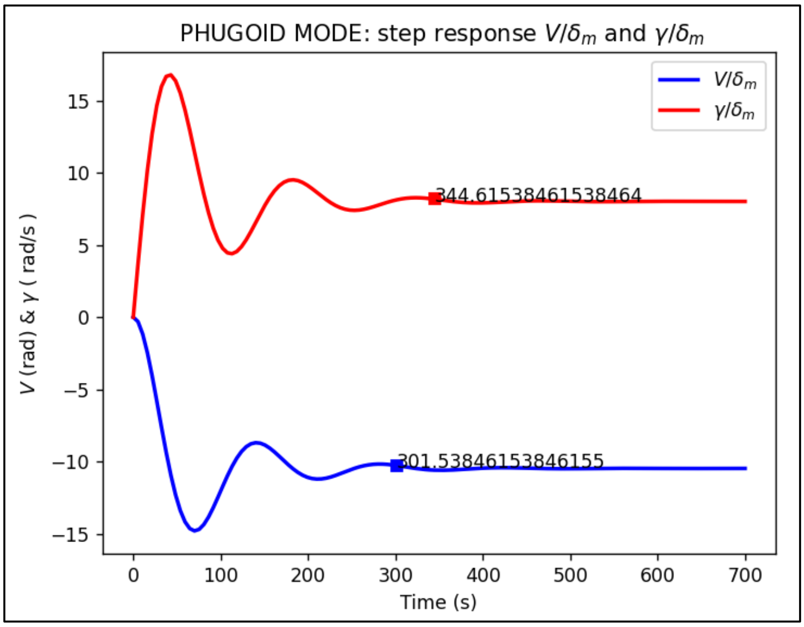
For $\frac{v}{\delta_m}$:

$$\boldsymbol{TF = \frac{-6.939 * 10^{-18}\, s + 0.02249}{s^2 + 0.02512\, s + 0.002146}}$$

For $\frac{\gamma}{\delta_m}$:

$$\boldsymbol{TF = \frac{0.6848\, s\ +\ 0.0172}{s^2 + 0.02512\, s + 0.002146}}$$

Giving the following curves:



By reading the graphics, we can determine the settling time of the two variables, meaning the time after which the curve is stable:

- V: 344.62 s
- γ: 301.54 s

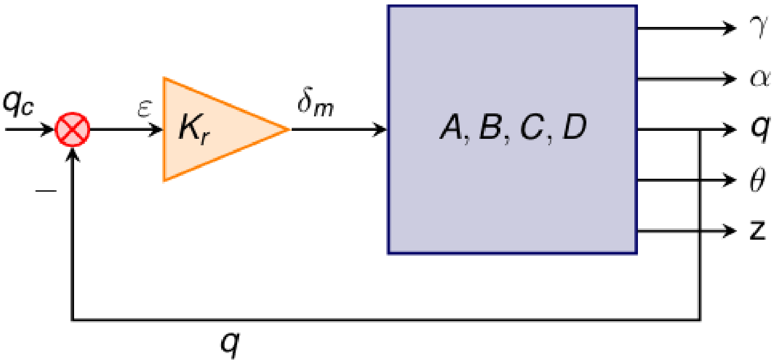
Contrary to the short period mode, the settling time is very high. To improve that, we will add a closed loop. To do so, we will consider that the speed is controlled with a perfect auto-throttle (instantaneous response). Therefore, we can modify the state vector X such as:

$$X = (\gamma \ \ \alpha \ \ q \ \ \theta \ \ z)^T$$

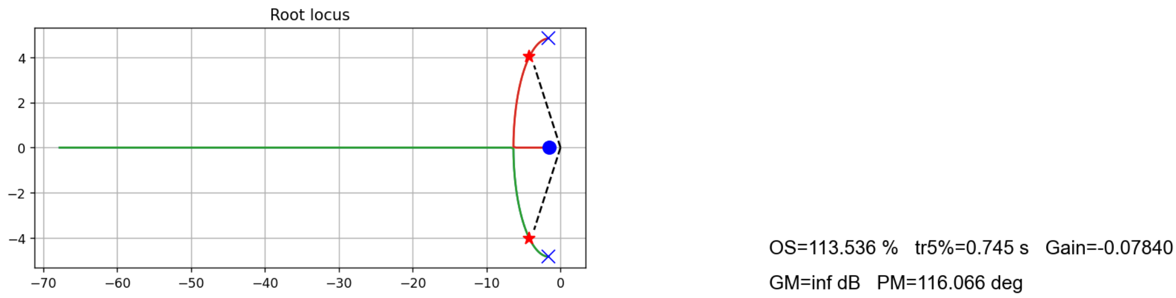
II – Controller synthesis

q FEEDBACK LOOP

We are beginning to build an autopilot by adding a gyrometric feedback loop (with q as the measured variable).



Then, we need to find the value of the gain K_r. To do so, we can use the $sisopy31$ python file and more specifically the root locus. The first step is to put the damping ratio at 0.7. It means overlaying the red stars and the black slashed lines, and then, reading the gain.



Thus, we got: $\boldsymbol{K_r = -0.07840}$

From this value, we can calculate the new state space system:

$$\begin{cases} A_k = AK_rBC_q \\ B_k = K_rB \\ C_k = C_{out} = C_q \\ D_k = K_rD \end{cases}$$

With:

$$A_k = \begin{pmatrix} 0 & 1.7656 & 0.0537 & 0 & 0 \\ 0 & -1.7656 & 0.9463 & 0 & 0 \\ 0 & -23.4526 & -6.6638 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad B_k = \begin{pmatrix} -0.0537 \\ 0.0537 \\ 5.2072 \\ 0 \\ 0 \end{pmatrix}$$

$$C_k = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \qquad D_k = [0]$$

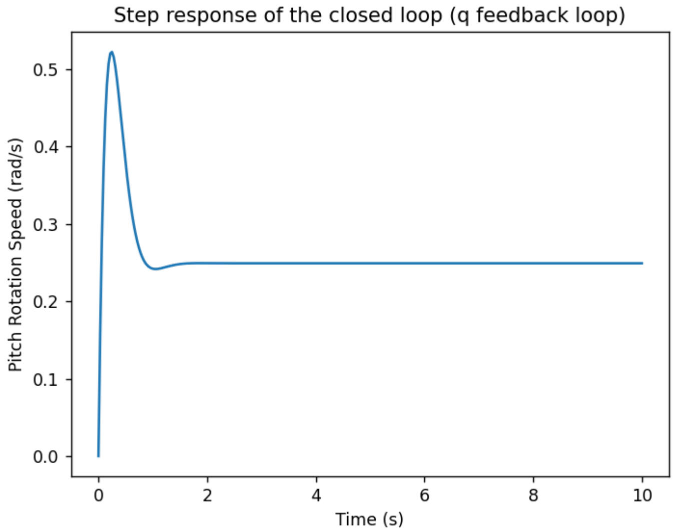
We also got the following transfer function:

$$\boldsymbol{TF} = \frac{\mathbf{5.207\ s + 7.935}}{\boldsymbol{s^2} + \mathbf{8.429\ s + 33.96}}$$

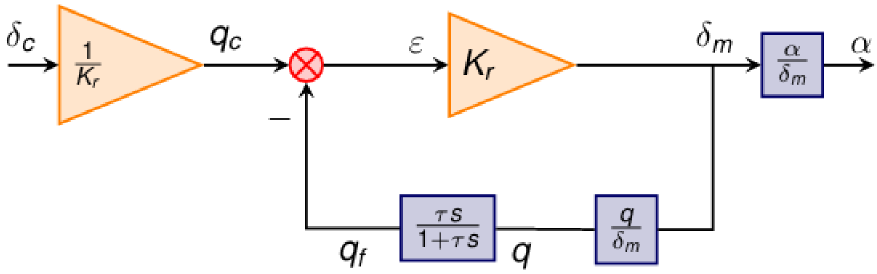
the following parameters:

Pole	Damping ratio (ζ)	Proper pulsation (w_n)
0	1	0
[-4.215+4.04j; -4.215-4.04j]	0.723	5.827 rad/s

And the step response of the closed loop:



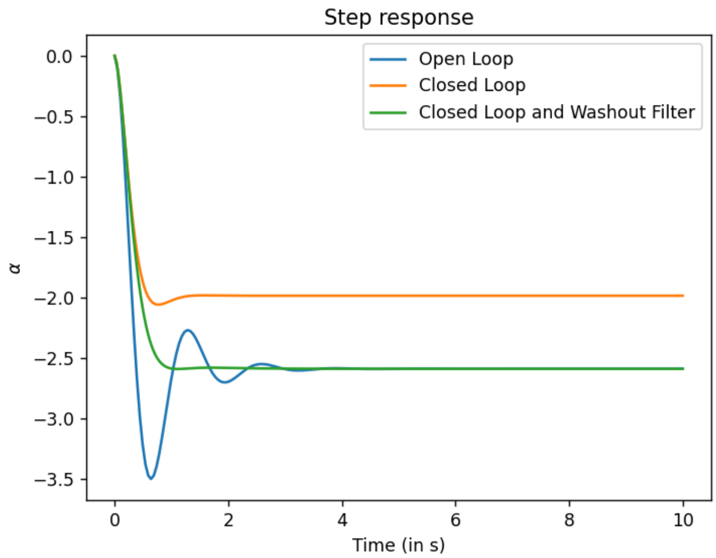
Now, the settling time is equal to 1.8 seconds, which is much better. However, to deal with the big overshot at the very beginning, we will use a certain type of filter called “washout filter”. The new system will look like:



A washout filter requires a value of the time constant τ. According to the subject, we will choose:

$$\tau = 0.7\text{s}$$

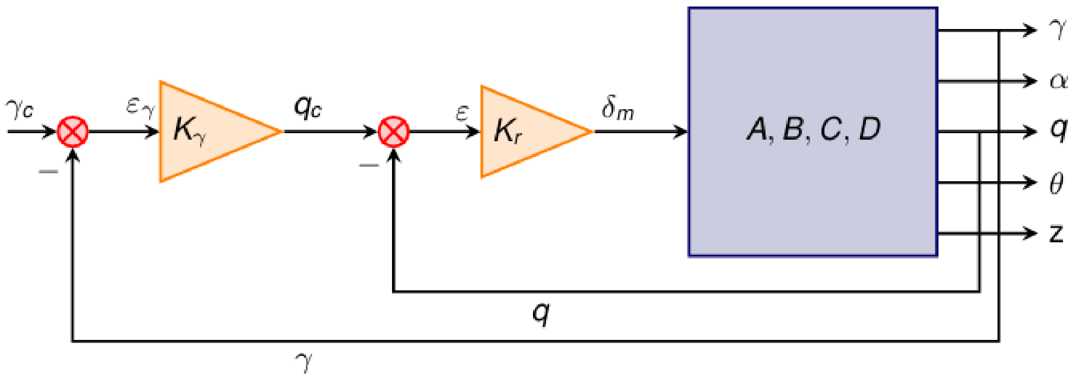
This would allow to have the same steady state gain for α with or without the q feedback. By plotting the three curves on a same graphics (Open loop, closed loop and washout filter), we obtain:



We can clearly see the effect of the filter. We get the same result as the open loop (same stable value) but we avoid the overshot we had with the closed loop (the curve of the washout filter is almost a line while the closed loop curve has a little bump at the beginning). Therefore, the filter is efficient.

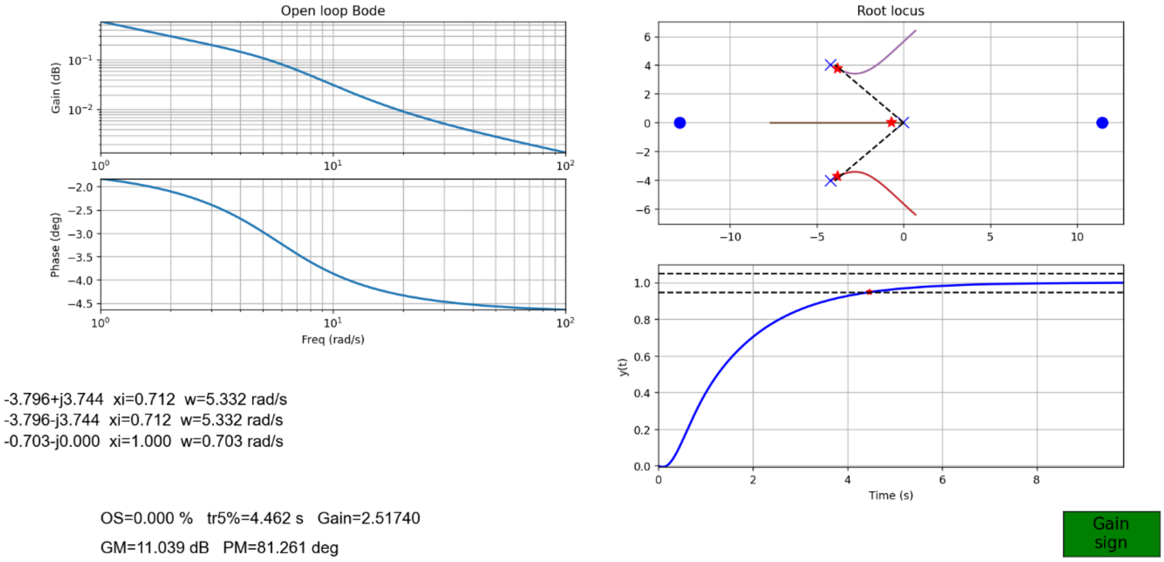
γ FEEDBACK LOOP

We consider that the auto-throttle perfectly ensures that the speed is constant, so that $\dot{v} = \frac{dv}{dt} = 0$. A flight path angle feedback loop is added to the preceding controlled system (with the q feedback loop, keeping the preceding K_r tuning).



At first, we will choose an estimation of the gain K_γ, based on the following parameters:

- Gain margin: $GM \geq 8dB$
- Phase margin: $PM \geq 35°$
- Settling time: $\leq 5\%$



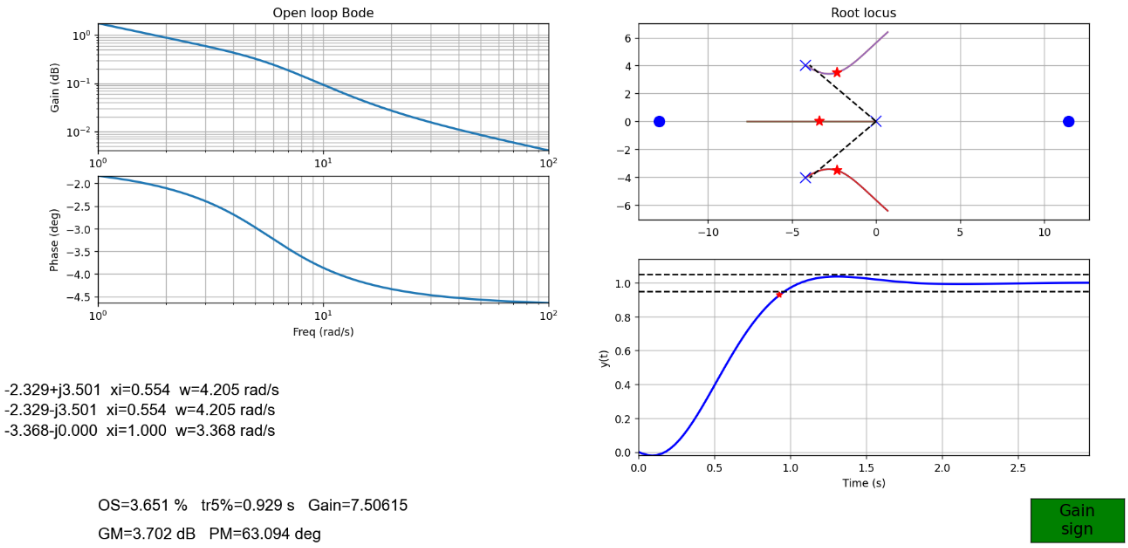
On this graph:

- Gain margin: $GM = 11.039dB \geq 8dB$
- Phase margin: $PM = 81.261° \geq 35°$
- Settling time: $\leq 5\%$

All the criteria are fulfilled, therefore, the first estimation of K_γ would be: $K_\gamma = 2.517$.

This is a very low estimation; we will define a second tuning to get a better value. Here are the criteria:

- Overshoot: $D_1 \leq 5\%$
- Settling time: within 5% with an optimized step response



On this graph:

- Overshoot: $D_1 \leq 5\%$
- Settling time: within 5% with an optimized step response

All the criteria are fulfilled, therefore, the second estimation of K_γ would be: $K_\gamma = 7.506$.

Closed loop state space with γ as the output:

$$A_\gamma = \begin{pmatrix} 0.9649 & 1.7656 & 0.0537 & 0 & 0 \\ -0.9649 & -1.7656 & 0.9463 & 0 & 0 \\ -93.5882 & -23.4526 & -6.6638 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 289.7143 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad B_\gamma = \begin{pmatrix} -0.9649 \\ 0.9649 \\ 93.5882 \\ 0 \\ 0 \end{pmatrix}$$

$$C_\gamma = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad D_\gamma = [0]$$

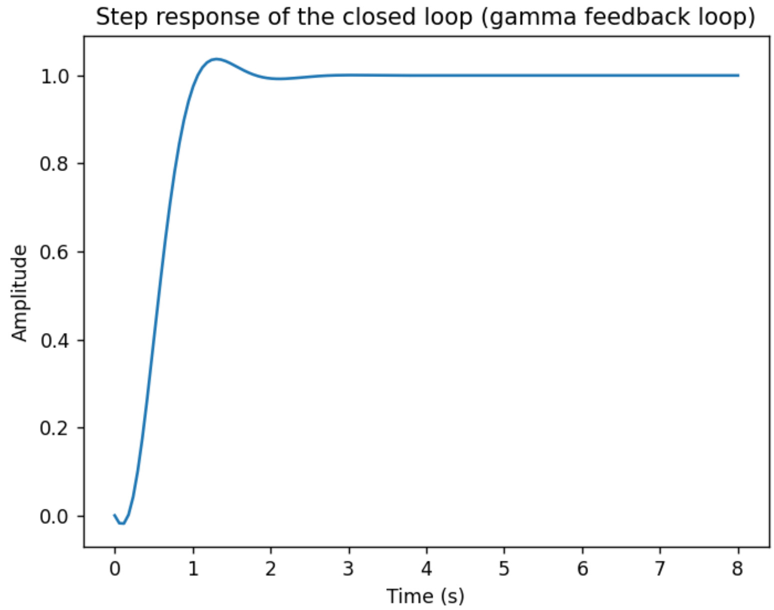
We also got the following transfer function:

$$\boldsymbol{TF = \frac{-0.9649\, s^2 - 1.405\, s + 142.6}{s^3 + 7.464\, s^2 + 32.55s + 142.6}}$$

the following parameters:

Pole	Damping ratio (ζ)	Proper pulsation (w_n)
0	1	0
[-0.732+4.82j; -0.732-4.82j]	0.150	4.875 rad/s
-6	1	6 rad/s

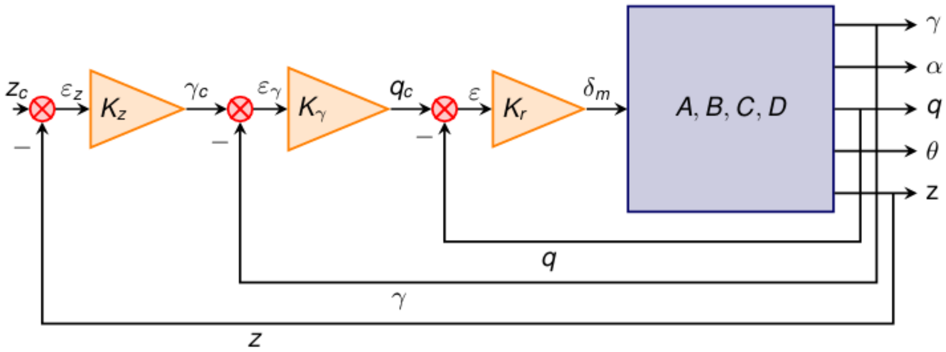
And the step response of the closed loop:



The settling time is under 2 seconds, which is ok. Moreover, the overshot is very small. So, the results are very satisfying.

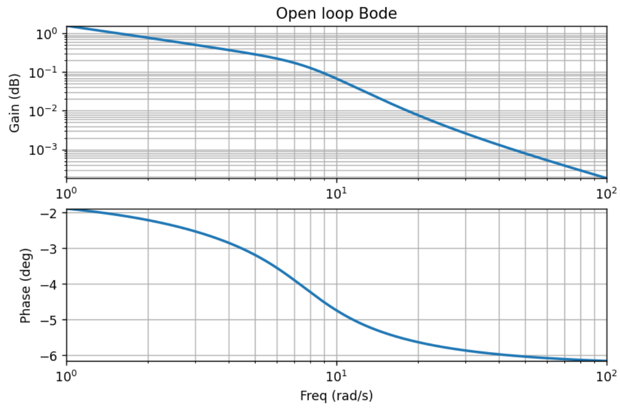
z FEEDBACK LOOP

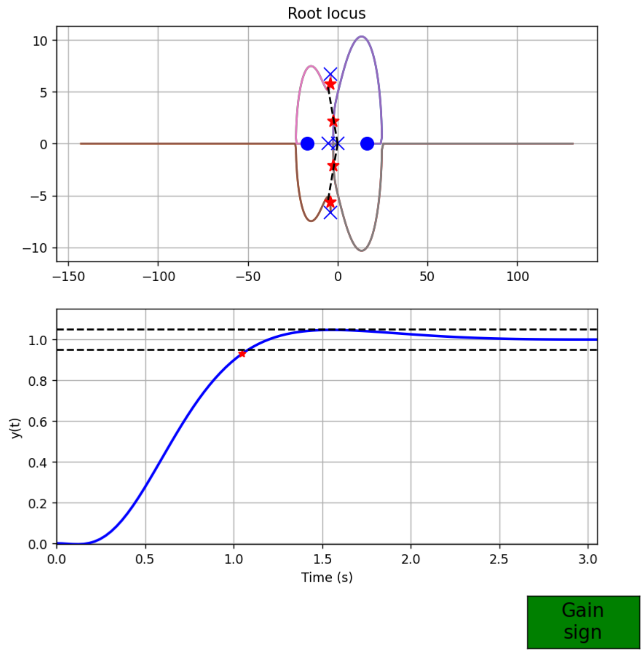
We add another control loop, using the measurement of the altitude z to the previous controlled system (aircraft + q feedback loop + γ feedback loop, while keeping the K_r and K_γ tuning).



At first, we will choose an estimation of the gain K_z, based on the following parameters:

- Overshoot: $D_1 \leq 5\%$
- Settling time: within 5% with an optimized step response
- Pseudo-periodic modes must be correctly damped: $\xi \geq 0.5$





-4.210+j5.689 xi=0.595 w=7.077 rad/s
-4.210-j5.689 xi=0.595 w=7.077 rad/s
-2.263+j2.163 xi=0.723 w=3.130 rad/s
-2.263-j2.163 xi=0.723 w=3.130 rad/s

OS=4.715 % tr5%=1.048 s Gain=0.00372

GM=3.368 dB PM=61.218 deg

On this graph:

- Overshoot: $D_1 \leq 5\%$
- Settling time: within 5% with an optimized step response
- Pseudo-periodic modes must be correctly damped: $\xi \geq 0.5$

All the criteria are fulfilled, therefore, the estimation of K_z would be: $K_z = 0.00372$.

Closed loop state space with z as the output:

$$A_\gamma = \begin{pmatrix} 0.403 & 1.7656 & 0.0537 & 0 & 0.0017 \\ -0.403 & -1.7656 & 0.9463 & 0 & -0.0017 \\ -39.0853 & -23.4526 & -6.6638 & 0 & -0.1606 \\ 0 & 0 & 1 & 0 & 0 \\ 289.7143 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad B_\gamma = \begin{pmatrix} -0.0017 \\ 0.0017 \\ 0.1606 \\ 0 \\ 0 \end{pmatrix}$$

$$C_\gamma = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \qquad D_\gamma = [0]$$

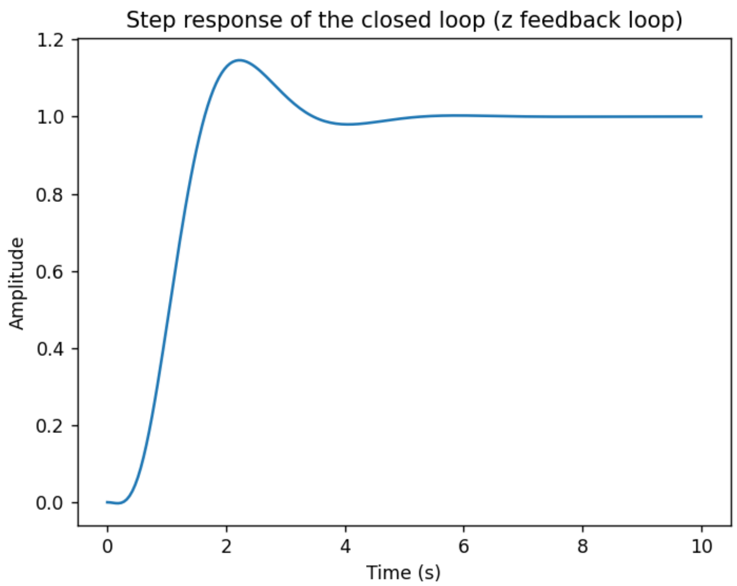
We also got the following transfer function:

$$\boldsymbol{TF = \frac{-1.776 * 10^{-14}\, s^3 - 0.4947\, s^2 - 0.7206\, s^2 + 73.12}{s^4 + 8.026\, s^3 + 32.88\, s^2 + 58.84\, s + 73.12}}$$

the following parameters:

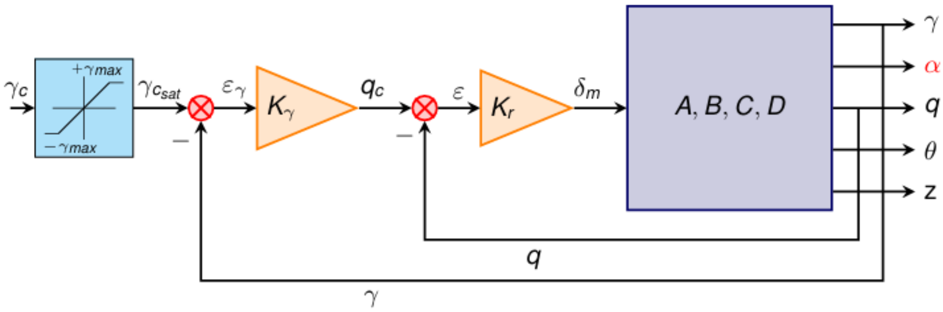
Pole	Damping ratio (ζ)	Proper pulsation (w_n)
0	1	0
[-0.960+1.852j; -0.960-1.852j]	0.46	2.086 rad/s
[-3.053+2.735j; -3.053-2.735j]	0.745	4.099 rad/s

And the step response of the closed loop:



SATURATION

A saturation is added at the input of the γ feedback loop. In this question, we are going to determine the value of $\gamma_{c_{sat}}$, but we will not implement the nonlinear simulation of the saturated autopilot.



Given we couldn't install slycot, we didn't do this part

NEW CENTER OG GRAVITY (COG)

We define a new position of the center of gravity of the aircraft by modifying the value of c.

$$c = f * 1.1$$

It modifies the values of the lengths X and Y, and consequently changes the state space representation of the aircraft.

By computing the new state space, we got the new matrices as follows:

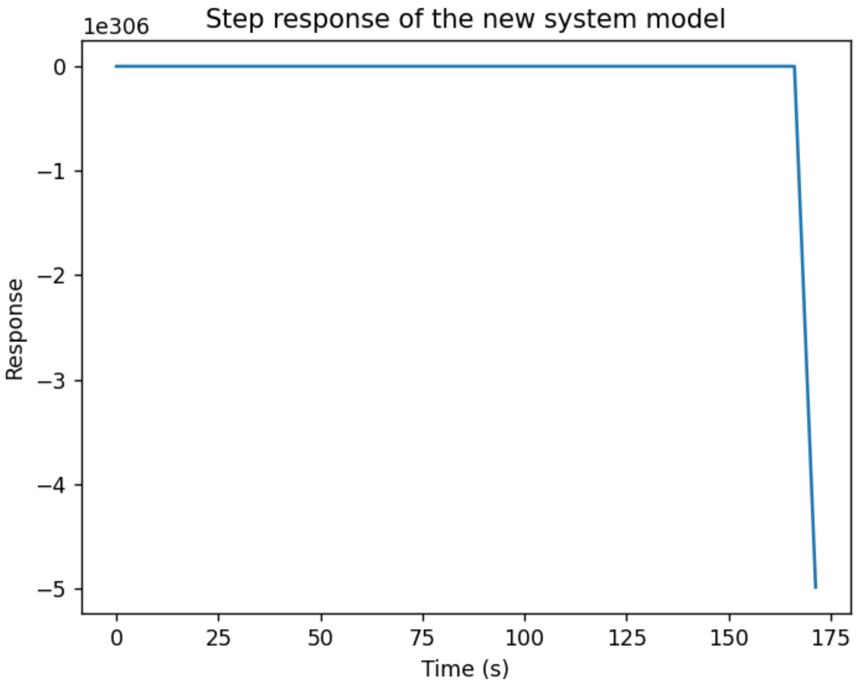
$$A = \begin{pmatrix} -0.0251 & -0.0328 & -0.0546 & 0 & 0 & 0 \\ 0.0655 & 0 & 1.7656 & 0 & 0 & 0 \\ -0.0655 & 0 & -1.7656 & 1 & 0 & 0 \\ 0 & 0 & -32.8288 & -1.4566 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 298.7143 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 \\ 0.6848 \\ -0.6848 \\ -44.4251 \\ 0 \\ 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad D = [0]$$

From that, we can determine the poles, damping ratios and proper pulsations of short period and phugoid modes:

	Pole	Damping ratio (ζ)	Proper pulsation (w_n)
	0	1	0
Short period mode	-7.343	1	7.343 rad/s
	4.121	1	4.121 rad/s
Phugoid mode	[-0.013+0.047j; [-0.013-0.047j]	0.2633	0.04831 rad/s

And the following step response:



This curve is obviously incorrect, but we couldn't find the problem.

Conclusion

To conclude, we managed to model the systems of an automatic pilot in different configurations and for several flight modes. Moreover, we managed to make it reusable, by changing the center of gravity and being able to compute all the work again with the new values.

We discovered and managed also to use new python libraries, especially the control library. Therefore, we were able to use matlab in Python, making it easier to work on one software instead of two.

However, we couldn't achieve to do all the work due to a problem to install slycot.