# ME-425 : Model Predictive Control Mini-project

Group D

Alessandro Dalbesio, 352298
Rhea Saber, 302674

12th January 2024

# Contents

**1 System Linearization** **1**

**2 Deliverable 3.1: Design MPC regulators** **1**
2.1 Recursive Constraints Satisfaction and stabilization to the origin . . . . . 1
2.2 Parameters tuning . . . . . 3
2.3 Terminal invariant sets . . . . . 3
2.4 Open-loop and closed-loop results . . . . . 4

**3 Deliverable 3.2: Design MPC Tracking Controllers** **6**
3.1 Tuning the parameters Q,R,H,P . . . . . 7
3.2 Open-loop and closed-loop results . . . . . 7

**4 Deliverable 4: Simulation with Nonlinear Rocket** **8**

**5 Deliverable 5.1: Offset-Free Tracking** **10**

**6 Deliverable 5.2: Offset-Free tracking with fuel consumption** **12**
6.1 Impact of having a thrust-dependent mass decrease during flight on the controller . 12

**7 Deliverable 6.1: Nonlinear MPC** **13**
7.1 Problem formulation . . . . . 14
7.1.1 System discretization . . . . . 14
7.1.2 Optimization problem definition . . . . . 14
7.2 Parameters tuning . . . . . 14
7.3 Results . . . . . 15
7.4 Comparison with the linear controller . . . . . 16
7.4.1 Comparison for a maximum roll angle of 15° . . . . . 16
7.4.2 Comparison for a maximum roll angle of 50° . . . . . 17
7.4.3 Discussion . . . . . 17

**8 Deliverable 6.2: Nonlinear MPC with delay** **18**
8.1 Effect of the delay on the nominal system . . . . . 18
8.2 Delay compensator design . . . . . 19

# 1 System Linearization

Here we want to obtain a linearized version of the rocket to see if we can, in some way, simplify the control of the rocket.

Firstly we compute an equilibrium point $(x_s, u_s)$ such that $0 = f(x_s, u_s)$.

```
rocket = Rocket(Ts);
[xs,us] = rocket.trim();
```

Then we can linearize the nonlinear model of the system.

```
sys = rocket.linearize(xs, us);
```

By doing this procedure we obtain a LTI system.

By studying the transfer function of the system we can see that we can simplify the control by splitting the system into four sub-systems:

```
[sys_x, sys_y, sys_z, sys_roll] = rocket.decompose(sys, xs, us)
```

Intuitively this can be explained by verifying that at steady-state (and in positions very close to it) the rocket is in vertical position (all states equals to zero) so:

- Changes in the deflection angle of servo 1, $\delta_1$, cause a variation of the speed only in $y$ direction and variation of the speed cause changes in the position $y$. Additionally changes of $\delta_1$ cause changes only in the rotational speed around the x axis $\omega_x$ so of $\alpha$.
- As for $\delta_1$ changes in the deflection angle of servo 2, $\delta_2$, cause a variation of $v_x$,$x$, $\omega_y$ and $\beta$.
- Changes in the average throttle will change the vertical propulsion of the rocket. As a consequence this imput will only have an effect on speed $v_z$ which will affect only $z$.
- Changes in the difference between the two propellers causes the rocket to rotate around the $z$ axis. This means that only $\omega_z$ and $\gamma$ change.

It's important to underline that this approximation is valid only when the rocket is very close to the steady-state.

# 2 Deliverable 3.1: Design MPC regulators

Here the goal is to design, for each of the dimensions x, y, z and roll, a MPC controller with the following properties:

- Recursive satisfaction of the input and angle constraints.
- Stabilization of the system to the origin (i.e. all states equal to zero).
- Settling time no more than seven seconds when starting stationary at 3 meters from the origin (for x,y and z) or stationary at 30° for roll.

## 2.1 Recursive Constraints Satisfaction and stabilization to the origin

To explicitly ensure stability and feasibility we need an infinite horizon which is computationally intractable.

To overcome this limitation we can split the infinite horizon problem into two sub-problems:

- From $i = 0$ to $i = N$, where the constraints might be active, we solve the constrained optimization problem

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i \\ \text{s.t.} \quad & x_{i+1} = A x_i + B u_i, \quad i = 0, 1, \ldots, N-1 \\ & x_0 = x_{\text{init}} \\ & H_x \cdot x_i \leq k_x, \quad i = 0, 1, \ldots, N \\ & H_u \cdot u_i \leq k_u, \quad i = 0, 1, \ldots, N-1 \end{aligned} \tag{1}$$

- For $i > N$, where the constraints are not active, we use the LQR control law starting from state $\boldsymbol{x_N}$

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & \sum_{i=N}^{\infty} x_i^T Q x_i + u_i^T R u_i \\ \text{s.t.} \quad & x_{i+1} = A x_i + B u_i, \quad i = N, N+1, \ldots, \infty \end{aligned} \tag{2}$$

The resulting MPC formulation is

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & \sum_{i=0}^{N-1} \left( x_i^T Q x_i + u_i^T R u_i \right) + x_N^T P x_N \\ \text{s.t.} \quad & x_{i+1} = A x_i + B u_i, \quad i = 0, 1, \ldots, N-1 \\ & x_0 = x_{\text{init}} \\ & H_x \cdot x_i \leq k_x, \quad i = 0, 1, \ldots, N \\ & H_u \cdot u_i \leq k_u, \quad i = 0, 1, \ldots, N-1 \\ & x_N \in X_f \end{aligned} \tag{3}$$

where $x_N^T P x_N$ is the corresponding infinite horizon cost with $P$ that is the solution of the discrete-time algebraic Riccati equation and $X_f$ is the maximum invariant set obtained for a defined $A$,$B$,$Q$,$R$ and for defined constraints. These two components make it possible to have recursive feasibility and stability.

After defining the parameters the terminal invariant set can be easily computed by implementing the following conceptual algorithm:

**Conceptual Algorithm to Compute Invariant Set**

$\Omega_0 \leftarrow \mathbb{X}$
**loop**
  $\Omega_{i+1} \leftarrow \text{pre}(\Omega_i) \cap \Omega_i$
  **if** $\Omega_{i+1} = \Omega_i$ **then**
    **return** $\mathcal{O}_\infty = \Omega_i$
  **end if**
**end loop**

## 2.2 Parameters tuning

Q and R are symmetric positive definite matrices that determines relative importance of the states and control signals in the optimization problem cost.

In the case of the linearized system we can split the matrices $Q$ and $R$ as:

| **Controller** | $Q_{11}$ | $Q_{22}$ | $Q_{33}$ | $Q_{44}$ | **R** |
|---|---|---|---|---|---|
| X | $\omega_y$ | $\beta$ | $v_x$ | x | $\delta_1$ |
| Y | $\omega_x$ | $\alpha$ | $v_y$ | y | $\delta_2$ |
| Z | $v_z$ | z | - | - | $P_{\text{avg}}$ |
| Roll | $\omega_z$ | $\gamma$ | - | - | $P_{\text{diff}}$ |

Table 1: order of the parameters for the matrices Q and R

By knowing that we can easily tune the parameters based on the relative importance of each state or input with respect to the others. In this case we have tuned them to have less than seven seconds as settling time when starting stationary at 3 meters from the origin or stationary at 30° for roll. In this case we observed that by assigning a weight of 10 to the states and a weight of 0.1 to the inputs we were able to obtain the desired performances. In this case the tuning was rather simple thanks to the fact that the four system are completely independent.

We set the horizon length at H=8 to have a good balance between computational complexity and predictive ability.

As discussed before the terminal cost matrix $P$ is the solution of the DARE equation obtained for the defined $A, B, Q$ and $R$. We will refer to it as $Q_f$.

In the table 2 we summarize the chosen parameters for each sub-system

| **Parameters** | **Controller X** | **Controller Y** | **Controller Z** | **Controller Roll** |
|---|---|---|---|---|
| Q | 10 $I_4$ | 10 $I_4$ | 10 $I_2$ | 10 $I_2$ |
| R | 0.1 | 0.1 | 0.1 | 0.1 |
| H | 8 | 8 | 8 | 8 |
| P | Qf | Qf | Qf | Qf |

Table 2: Chosen parameters for the controllers

## 2.3 Terminal invariant sets

The terminal invariant sets obtained with the parameters chosen in Table 2 are displayed below.

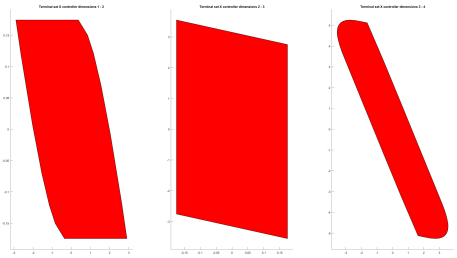


Figure 1: Terminal invariant set X

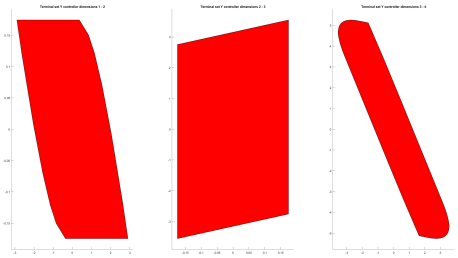


Figure 2: Terminal invariant set Y

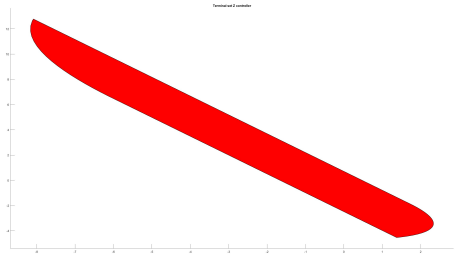


Figure 3: Terminal invariant set Z

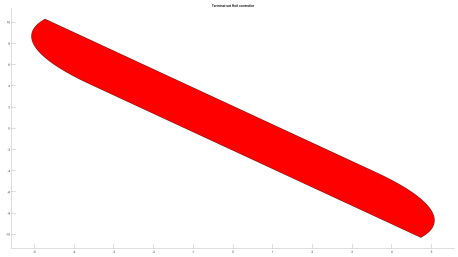
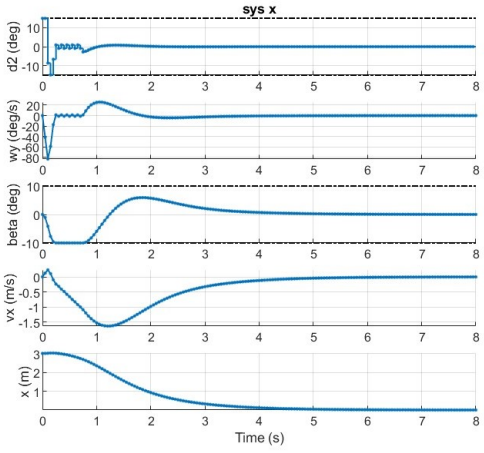


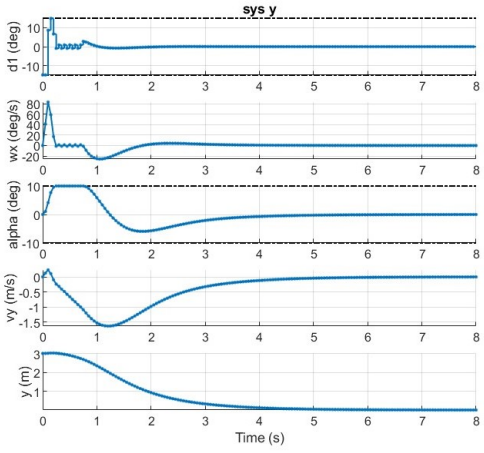
Figure 4: Terminal invariant set Roll

## 2.4 Open-loop and closed-loop results

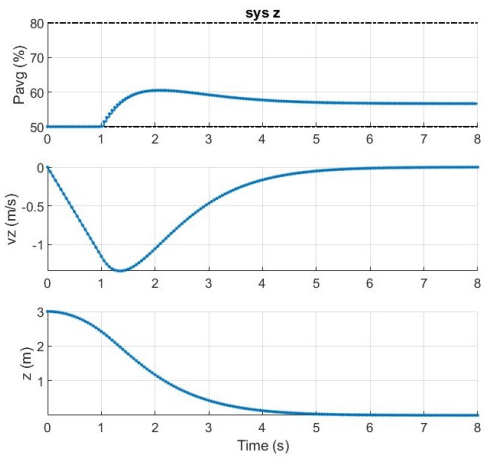
The obtained open-loop results are displayed below.



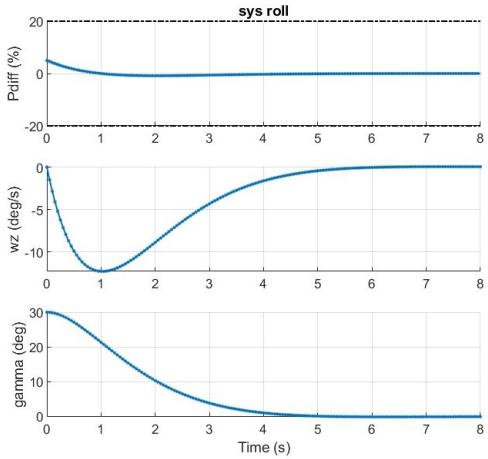

(a) Open-loop evolution of sub-system X




(b) Open-loop evolution of sub-system Y

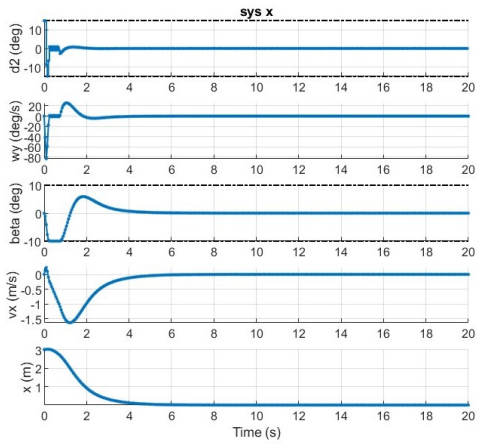



(a) Open-loop evolution of sub-system Z

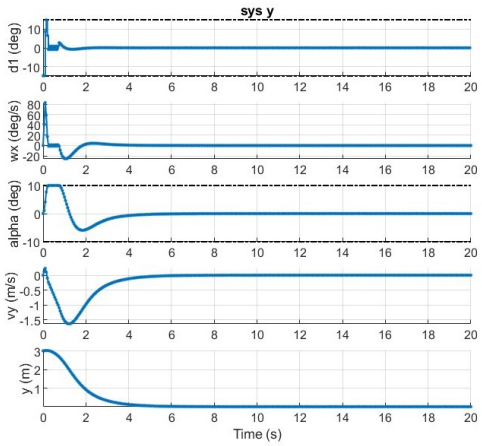



(b) Open-loop evolution of sub-system Roll

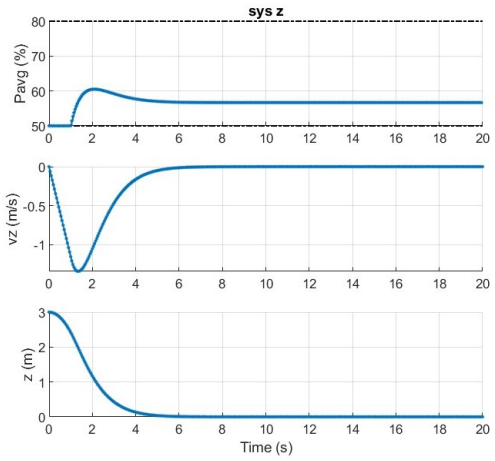
The closed-loop results are displayed below.



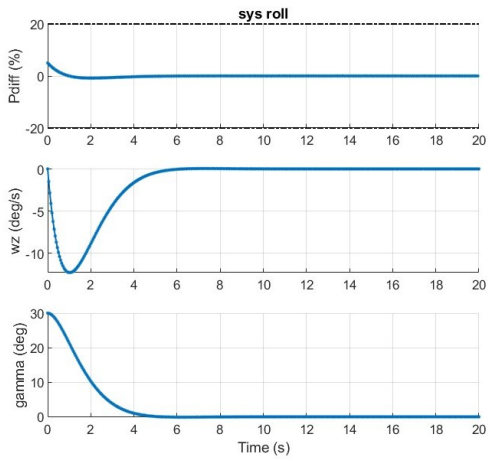

(a) Closed-loop evolution of sub-system X




(b) Closed-loop evolution of sub-system Y




(a) Closed-loop evolution of sub-system Z




(b) Closed-loop evolution of sub-system Roll

The results obtained fully satisfy the requirements.

# 3 Deliverable 3.2: Design MPC Tracking Controllers

Here we want to extend our MPC controllers to make them track a defined reference.
In this case the chosen reference is -4 meters for x,y,z and 35° for roll.

Here we drop the requirements of invariance and therefore we have no terminal set. Since we still have stability requirements we keep the terminal cost.

To be able to follow a defined reference we can write the MPC formulation as:

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & \sum_{i=0}^{N-1} \left((x_i - x_s)^T Q (x_i - x_s) + (u_i - u_s)^T R (u_i - u_s)\right) + V_f(x_i - x_s) \\ \text{s.t.} \quad & x_{i+1} = A x_i + B u_i, \quad i = 0, 1, \ldots, N-1 \\ & x_0 = x_{\text{init}} \\ & H_x \cdot x_i \leq k_x \\ & H_u \cdot u_i \leq k_u \end{aligned} \tag{4}$$

where $u_s$ and $x_s$ that are the solution of the minimization problem:

$$\begin{aligned} \min_{x_s, u_s} \quad & u_s^T R_s u_s \\ \text{s.t.} \quad & \begin{bmatrix} I - A & -B \\ C & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} 0 \\ r \end{bmatrix} \\ & H_x x_s \leq k_x \\ & H_u u_s \leq k_u \end{aligned} \tag{5}$$

Here we assume that the target problem is feasible and we consider $R_s = 1$.

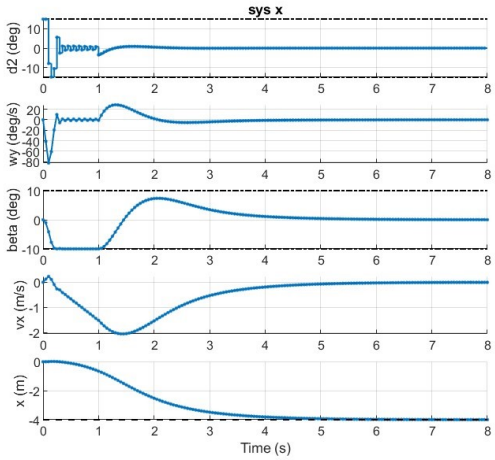
## 3.1 Tuning the parameters Q,R,H,P

Here we tune our parameter with the same method we used in deliverable 3.1.

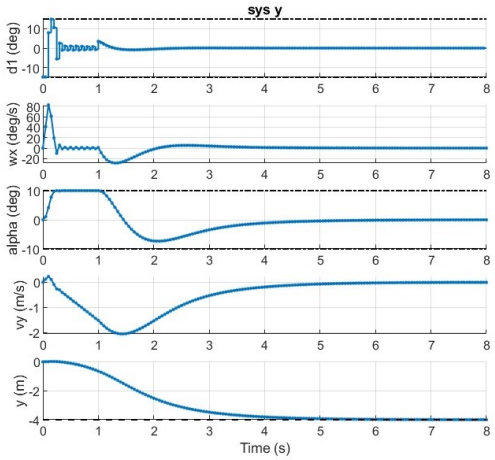
With the parameters defined before, and showed in table 2, we are able to track our reference without oscillations and we reach the reference in less than 5 seconds both for open and closed loop. As terminal cost we have chosen to use $Q_f$. This choice has been made to have stability guarantees.

## 3.2 Open-loop and closed-loop results

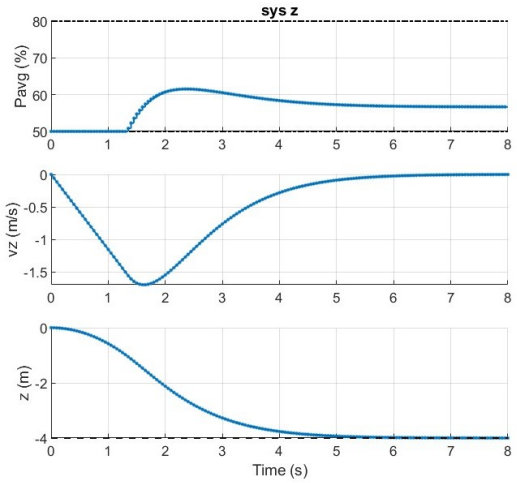
The obtained open-loop results are displayed below.



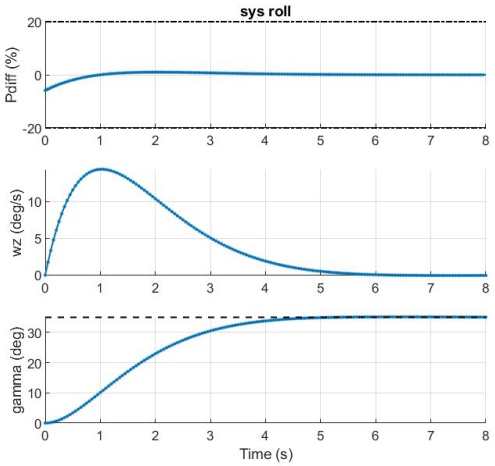

(a) Open-loop evolution of sub-system X




(b) Open-loop evolution of sub-system Y

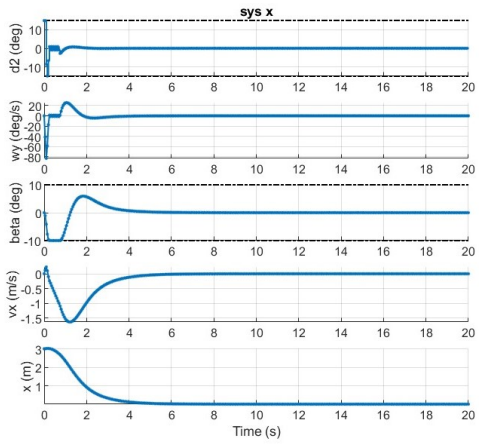



(a) Open-loop evolution of sub-system Z

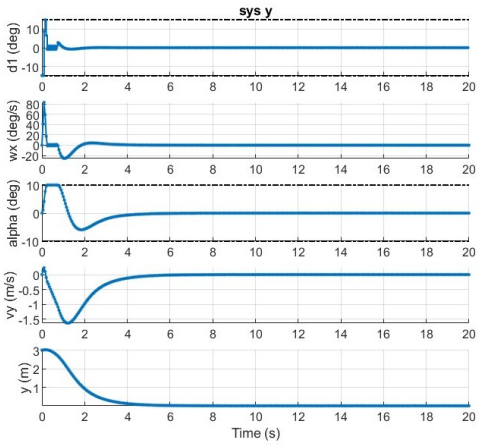



(b) Open-loop evolution of sub-system Roll

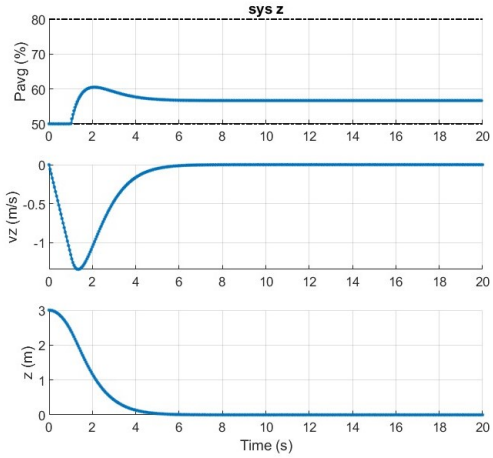
The closed-loop results are displayed below.



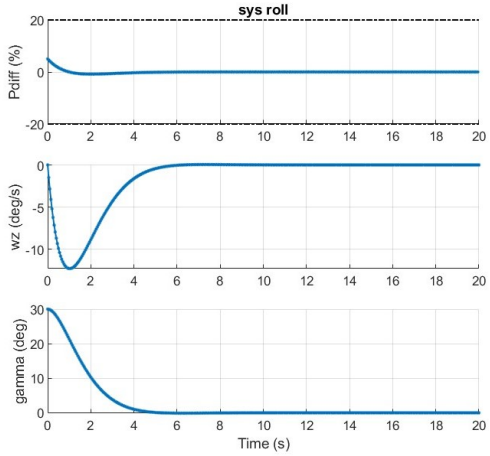

(a) Closed-loop evolution of sub-system X




(b) Closed-loop evolution of sub-system Y




(a) Closed-loop evolution of sub-system Z




(b) Closed-loop evolution of sub-system Roll

As before the results fully satisfy the requirements.

# 4 Deliverable 4: Simulation with Nonlinear Rocket

Here we are going to use the previously defined linear-controllers to control the nonlinear system to make it follows a sequence of references. The MPC formulation used is the same as 4.

Firstly we test if the choice of parameters designed in the previous steps (Table 2) can be used.

The results obtained with the parameters summarized in Table 2 are displayed in Figure 13.

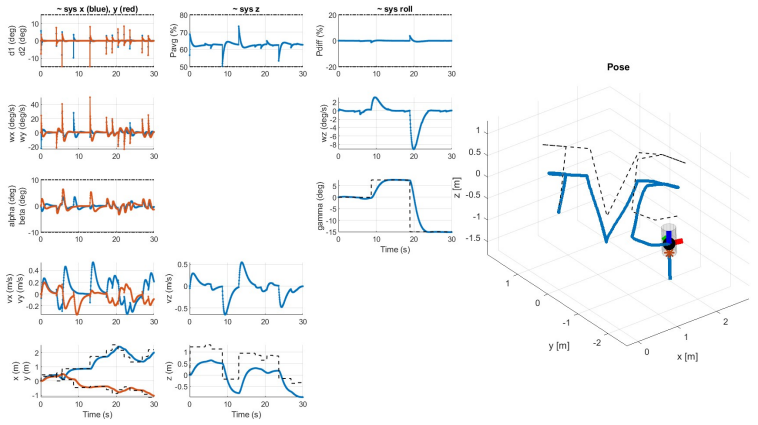



Figure 13: Results obtained with the parameters summarized in *Table 3*

We observe that the results obtained with the parameters previously defined are, as expected, not very good.
This is not unexpected mainly for two reasons:

- The parameters defined in *Table 3* have been chosen to have a settling time that is less than 7 seconds, while here we need a faster system.
- In this case we are controlling the non-linear system instead of the linearized system as before, so the parameters found might not be ideal. Additionally we should consider that in this case we don't have completely independent sub-systems as before.

To obtain better results we have chosen to re-tune the parameters of our controllers. In the re-tuning we have chosen to follow these rules:

- Reduce as much as possible the steady-state error (no steady-state errors).
- Reduce as much as possible any oscillations.
- Go to the reference as fast as possible.

In the tuning we have to take care that while previously we had perfectly independent sub-system while here we observed that this simplification is not completely valid anymore.

The chosen parameters are summarized in *Table 3*

| Parameter | X Controller | Y Controller | Z Controller | Roll Controller |
|---|---|---|---|---|
| Q | diag([400,10,25,275]) | diag([400,10,25,275]) | diag([125,1500]) | diag([500,1500]) |
| R | 100 | 100 | 0.01 | 0.01 |
| P | $Q_f$ | $Q_f$ | $Q_f$ | $Q_f$ |
| H | 8 | 8 | 8 | 8 |

Table 3: Parameters chosen for each controller

The results obtained with the parameters summarized in *Table 3* are displayed below in *Figure 14*.

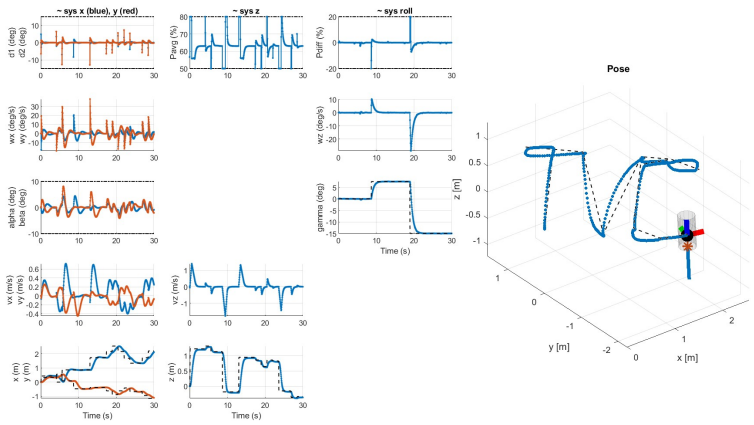



Figure 14: Results obtained with the parameters summarized in *Table 3*

With the chosen parameters we have obtained fast control with a good tracking of the references. We observe, indeed, that $x, y, z, \delta$ reach the reference value with very small oscillations and with a very small steady-state error.

In the tuning of the parameters we have given a high cost to the reference states to reduce as much as possible the steady-state error.

We observed that we were able to control the oscillations of the reference state mainly through the associated speeds ($V_x$ for $x$, $V_y$ for $y$, $V_z$ for $z$ and $\omega_z$ for $\delta$) and for the Z controller and Roll Controller also with the inputs.

We observed that when we had very large $\omega_x$ and $\omega_y$ when changing the reference we also had large oscillations on the states $\omega_z$ and $\gamma$.
As a consequence we have chosen to increase the weights associated to be able to limit this behavior. Since we know that the inputs directly affect $\omega_x$ and $\omega_y$ we have chosen to increase the inputs associated to the X controller and the Y controller to be able to have a smoother control.
The inputs of the other controllers have been kept low to make control reactive.
We observed that this choice hasn't affected in a significant way the speed of the control as we are able to reach the reference fast enough.

Since we were able to obtain good results with the chosen parameters and we observed that the states are not close to saturation at any time, we haven't implemented soft-constraints.

# 5 Deliverable 5.1: Offset-Free Tracking

In this section we assume that the mass of the rocket is significantly different from the model and we extend the z-controller to compensate it.

To do this we firstly augment the model of the z-subsystem by incorporating the disturbance model

$d_k$. The new model equations are given by:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + B_d d_k \\ d_{k+1} &= d_k \\ y &= Cx_k \end{aligned} \tag{6}$$

we then design the linear state and disturbance estimator based on the augmented model whose expression is given by:

$$\begin{bmatrix} \hat{x}_{k+1} \\ \hat{d}_{k+1} \end{bmatrix} = \begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ \hat{d}_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k + \begin{bmatrix} L_x \\ L_d \end{bmatrix} (C\hat{x}_k - y_k) \tag{7}$$

where $\hat{x}_k$ and $\hat{d}_k$ are the estimated values for the states and for the disturbance.

If we now compute the error dynamic we get:

$$\begin{bmatrix} x_{k+1} - \hat{x}_{k+1} \\ d_{k+1} - \hat{d}_{k+1} \end{bmatrix} = (\begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} + \begin{bmatrix} L_x \\ L_d \end{bmatrix} [C \quad 0]) + \begin{bmatrix} x_k - \hat{x}_k \\ d_k - \hat{d}_k \end{bmatrix} \tag{8}$$

From this equations we can see that we can control the convergence of the error by modifying the poles of the matrix $\bar{A} + L\bar{C}$ by defining the matrix $L$.

In the tuning have to consider two trade-offs: if we place the poles of $\bar{A} + L\bar{C}$ close to the origin the estimation error will go to zero faster but this will also make the estimation less smooth and with more overshoots while with eigenvalues with a norm close to 1 we will have a slower estimation process, but decrease the overshoot of the estimate d.

We chose the triplets of poles E = [0.3,0.35,0.4] because we saw they were giving both smooth estimation and fast convergence.
We have used the same MPC formulation as 4 with the only difference that $x_0 = \hat{x}$ where $\hat{x}$ is the estimated state. The parameters used summarized in table 3.

The disturbance estimation is plotted in figure 15 together with the system evolution.

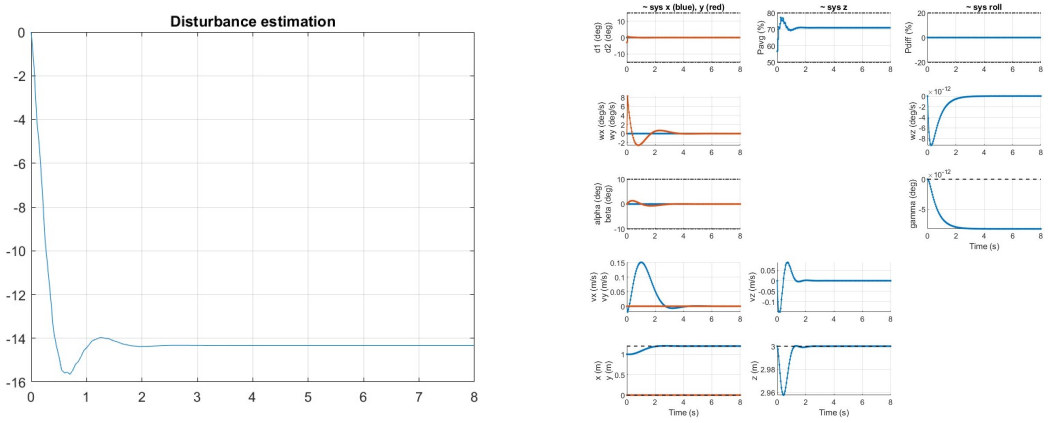
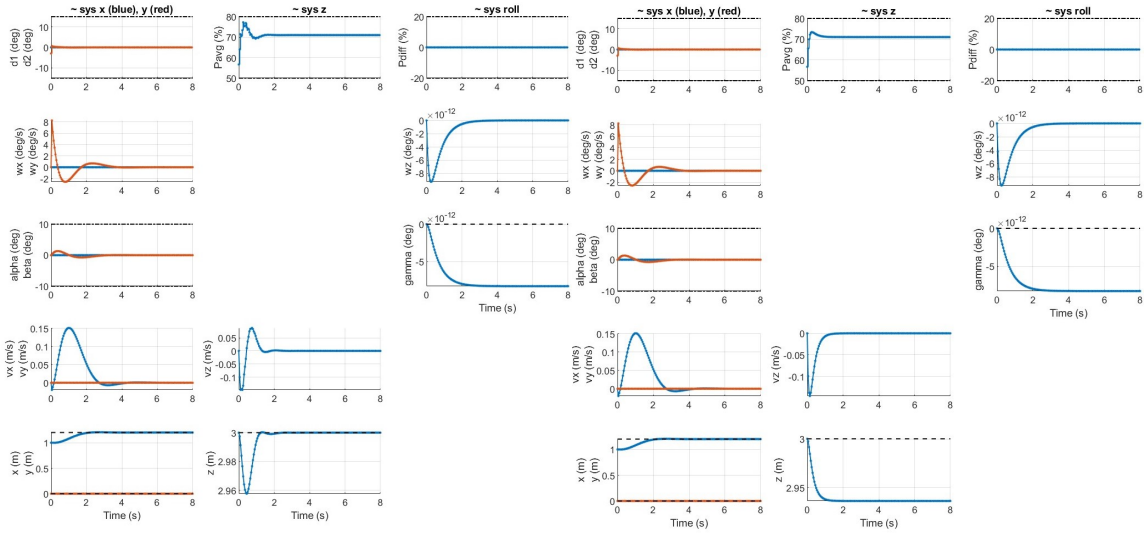



Figure 15: Disturbance estimation and system evolution

Then we want to address the benefits of using an offset-free tracking. For this reason we compare the results obtained with and without the estimator. The results are shown in 16




(a) m=2.13 kg with offset-free-tracking

(b) m=2.13 kg without offset-free tracking

Figure 16: Comparison between offset-free tracking enabled and disabled

Here we can see that after some time the controller with offset-free tracking is able to estimate correctly the additional mass. We can see this because we don't have any offset in the z state. Instead without the estimator the error is not compensated and we can see a constant offset in the z state.

# 6 Deliverable 5.2: Offset-Free tracking with fuel consumption

In addition to the different total mass from Deliverable 5.1 we now assume that half of the initial rocket mass is actually fuel, and it will decrease depending on the power consumption of the motor. Once all the fuel/energy is consumed, the motor will not produce any thrust anymore and the remaining rocket mass will be half of the initial mass.
To simulate the system with changing mass, we added the thrust-dependent mass decrease rate of -0.27 during the flight.
Here we will use the same formulation as Deliverable 5.1 but we will drop the inputs constraints in the computation of the reference states and input.

## 6.1 Impact of having a thrust-dependent mass decrease during flight on the controller

In figure 17 we shows the impact of having a thrust-dependant mass decrease:

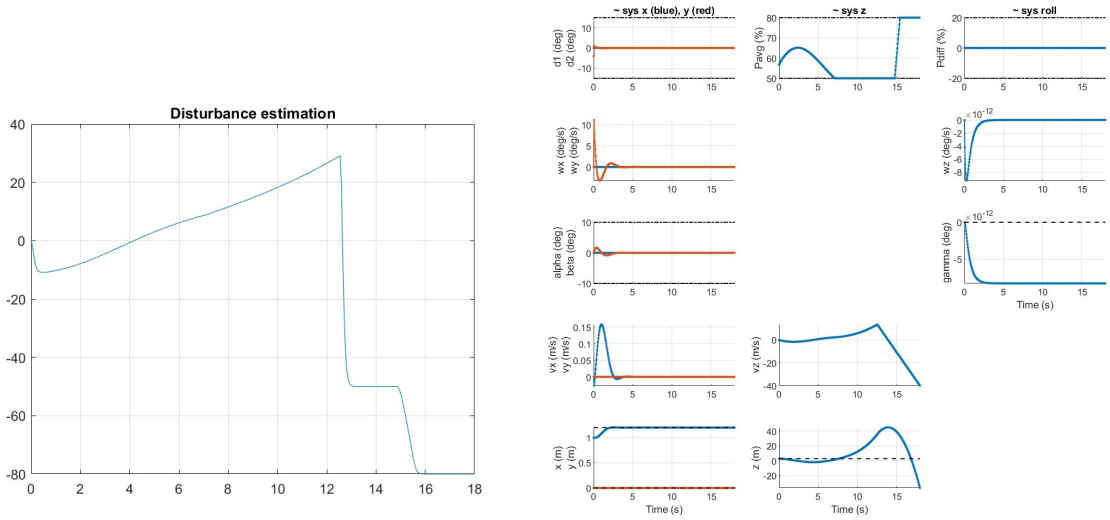



Figure 17: State and system estimation with a thrust-dependant mass decrease

In the first couple of seconds of the simulation, despite the estimator, there is a little tracking offset in height. In this case the main problem is that we are trying to estimate a constant disturbance while the mass rate decrease means that we are dealing with a variable disturbance. This directly affect the estimation of the states and of the disturbance, which will be imprecise, and this make the performances of the estimator worst.
To modify this one solution might be to model the disturbance and include this model in the estimator. This might significantly improve our results.

We observed that, after a first stabilization period, when the mass decrease also the $P_{avg}$ will decrease. This makes sense because the less the mass of the rocket the less power is needed to lift it up.
We see that when the mass is below a certain value $P_{avg}$ tries to go to below 50% but since we have constrained the input it will stays at this value. As a consequence the rocket will increase it's acceleration in $z$ direction and we see that the rocket increase its speed in z direction and height.
After some time the rocket fuel has finished and the rocket starts to fall. The system at this point try to use as much $P_{avg}$ as possible to compensate the free fall but since we have run out of fuel the system is not able to apply this input.

# 7 Deliverable 6.1: Nonlinear MPC

Here we want to develop a nonlinear MPC controller for the rocket system.

## 7.1 Problem formulation

### 7.1.1 System discretization

The rocket is a continuous time system and to be able to control it with MPC we firstly need to discretize it.
To discretize our system we choose to use *Runge-Kutta 4* method. This method, using a second-order Taylor series expansion around a desired state for a given input, provides a good precision and that's the reason why we have chosen it.
Given the sampling time $h$, the discretized state $x_{k+1}$, the current state $x_k$, the current input $u_k$ and the system dynamic $\dot{x} = f(x, u)$ the discretized state is:

$$x_{k+1} = x_k + h \cdot \left( \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} \right) \tag{9}$$

with

$$\begin{aligned} k_1 &= f(x_k, u_k) \\ k_2 &= f(x_k + \frac{h}{2} \cdot k_1, u_k) \\ k_3 &= f(x_k + \frac{h}{2} \cdot k_2, u_k) \\ k_4 &= f(x_k + h \cdot k_2, u_k) \end{aligned}$$

### 7.1.2 Optimization problem definition

Once we have discretized our system we need to define the optimization problem.
Since we want to track a given reference and we have constraints in polyhedral form we can write our optimization problem as:

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & \sum_{i=0}^{N-1} \left( (x_i - x_{ref})^T Q (x_i - x_{ref}) + (u_i - u_{ref})^T R (u_i - u_{ref}) \right) + x_N^T Q_f x_N \\ \text{s.t.} \quad & x_{i+1} = f(x_i, u_i), \quad i = 0, 1, \ldots, N-1 \\ & x_0 = x_{\text{init}} \\ & H_x \cdot x_i \leq k_x \\ & H_u \cdot u_i \leq k_u \end{aligned} \tag{10}$$

where

- $x_{ref}$ and $u_{ref}$ are the reference points to track.
- $Q_f$ is the terminal set computed on the linearized system at the equilibrium point.

## 7.2 Parameters tuning

In the parameters tuning we encountered in lots of trade-off.

As already discussed when tuning the linear controllers to control the nonlinear model we observed

that by having an high $\omega_x$ and $\omega_y$ we had problems with $\gamma$ when the rocket was changing the reference. For this reason we had to put high values on the weights associated to $\omega_x$ and $\omega_y$ to avoid having lots of oscillations on $\gamma$.
To additionally avoid too rapid variations for $\omega_x$ and $\omega_y$ we have chosen to use high values for the associated inputs.
This choice hasn't affected a lot the performances of $x$ and $y$ since we are able to track the reference values very quickly.
In table 4 we have summarized the parameters used.
We have used the same parameters for $\gamma_{max} = 15°$ and $\gamma_{max} = 50°$

| x | y | z | $\omega_x$ | $\omega_y$ | $\omega_z$ | $v_x$ | $v_y$ | $v_z$ | $\alpha$ | $\beta$ | $\gamma$ | $\delta_1$ | $\delta_2$ | $P_{avg}$ | $P_{diff}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 375 | 375 | 1500 | 400 | 400 | 250 | 25 | 25 | 150 | 25 | 25 | 1750 | 0.5 | 0.5 | 0.01 | 1.00 |

Table 4: Parameters used with nonlinear controller

## 7.3 Results

The results obtained with $\delta_{max} = 15°$ are plot in figure 18

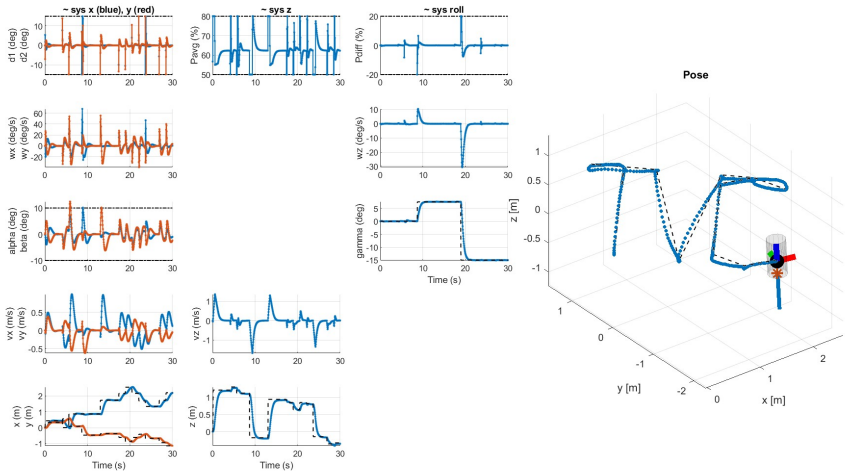



Figure 18: System with $\delta_{max} = 15°$

The results obtained with $\delta_{max} = 50°$ are plot in figure 19

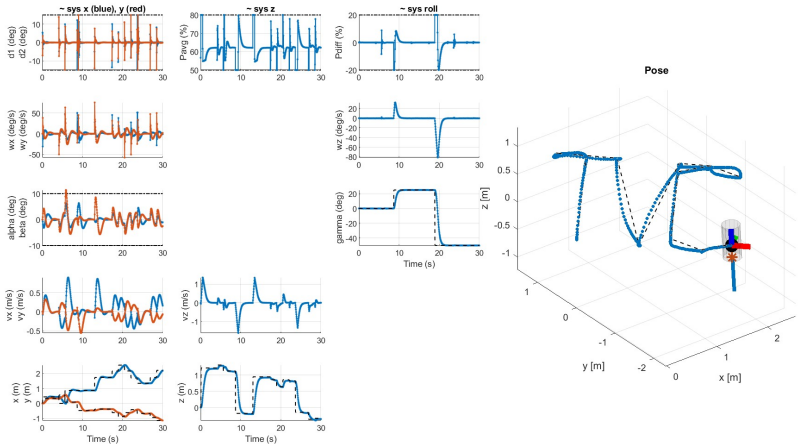
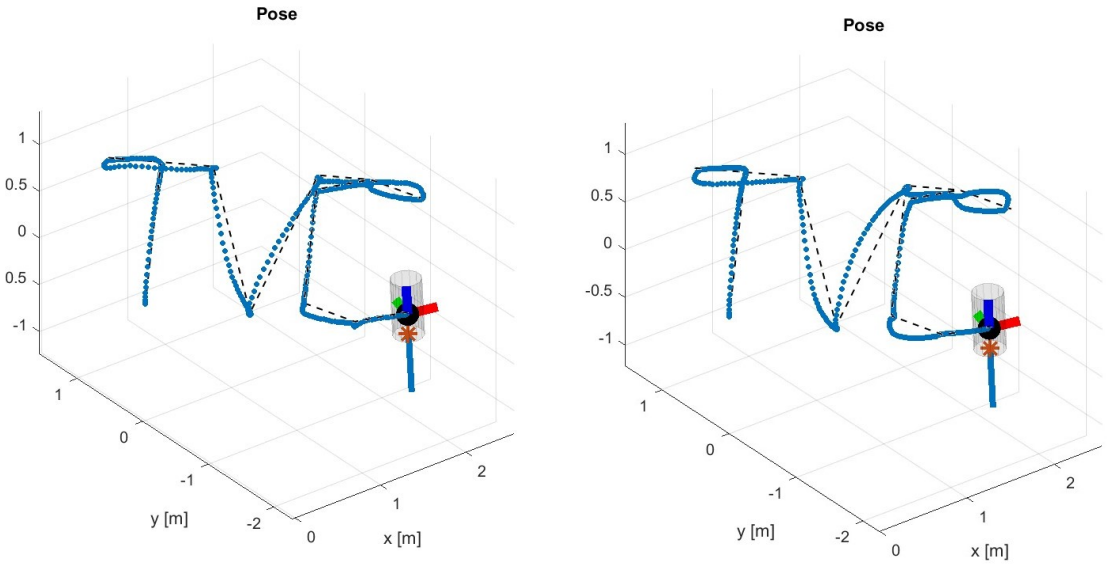



Figure 19: System with $\delta_{max} = 50°$

## 7.4 Comparison with the linear controller

### 7.4.1 Comparison for a maximum roll angle of 15°

We can see, from the plot below, that for a maximum roll angle of 15° the performances between the nonlinear and the linear controller are very similar.



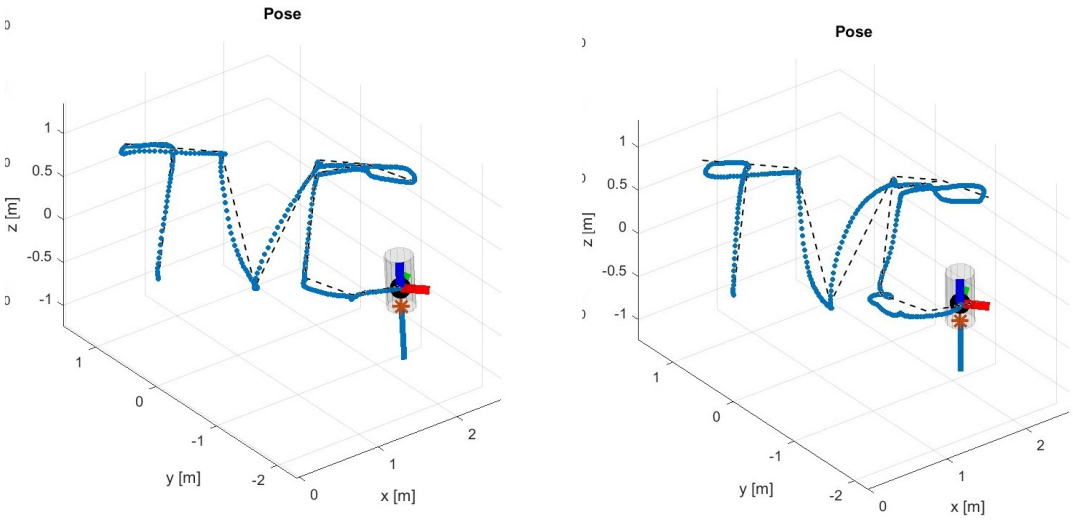

(a) Performances obtained with nonlinear controller (b) Performances obtained with linear controller

### 7.4.2 Comparison for a maximum roll angle of 50°

In the case of a maximum roll angle of 50° we observed that with the linear controllers we had lots of issues. In particular we verified that we had to lower the performances, and to re-tune the controllers, to be able to have a feasible problem.
With the new weights we were able to delay the oscillations to the end and in this way we were able to have a feasible problem. This doesn't solve the problem because, if we increase the simulation time, we can clearly see that the rocket keeps oscillating.

This behaviour is not unexpected. The linear controllers has been designed on a linearized version of the system, which validity is only close to the linearization point. The more we go away from this linearization point the more the approximation is less imprecise.
In this case we are very far away from the linearization point and this has tragic effects on the system behavior.

With the nonlinear controllers, since we don't have any linearization assumption, the results are very good.




(a) Performances obtained with nonlinear controller (b) Performances obtained with linear controller

### 7.4.3 Discussion

In the case of linear controller vs nonlinear controller we can say that:

- In the case of the linear controllers we were able to tune the parameters more easily. Each subsystem had its own controller and the weights were only relative to that subsystem. In the case of the non-linear controller we weren't able to do so and we had to tune the parameters of all the states together. In this case since the system was quite simple the process was not too difficult but if more state are added then the tuning process with the nonlinear controller might becomes really challenging.

- In the case of linear controllers we had lots of assumptions and limitation. As shown in the previous results the performances obtained with the linear controller, when we go away from the linearization point, becomes terrible and the system becomes unstable.

# 8 Deliverable 6.2: Nonlinear MPC with delay

Here we assume that the time needed to solve the MPC problem is significantly high and as a consequence we will have some delay in the control of our system.
Here we will use the same formulation as Deliverable 6.1 with just a small modification in the matrices $Q$ and $R$ to have faster tracking performances.

## 8.1 Effect of the delay on the nominal system

Firstly we start to see how the increase of the delay affect the close-loop behavior of our system. In figure 22 we have plot the nominal system to use it as a reference.

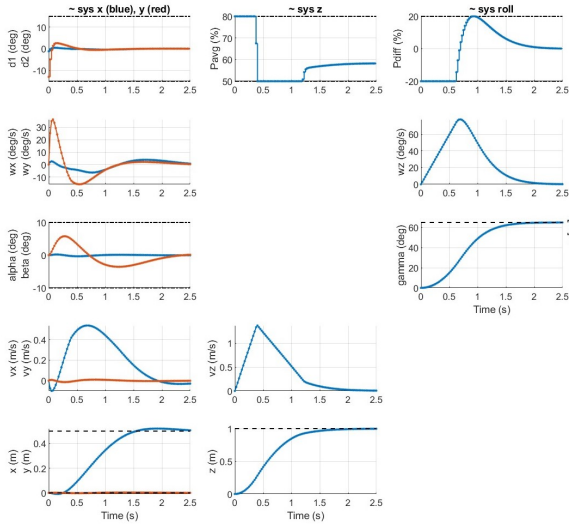


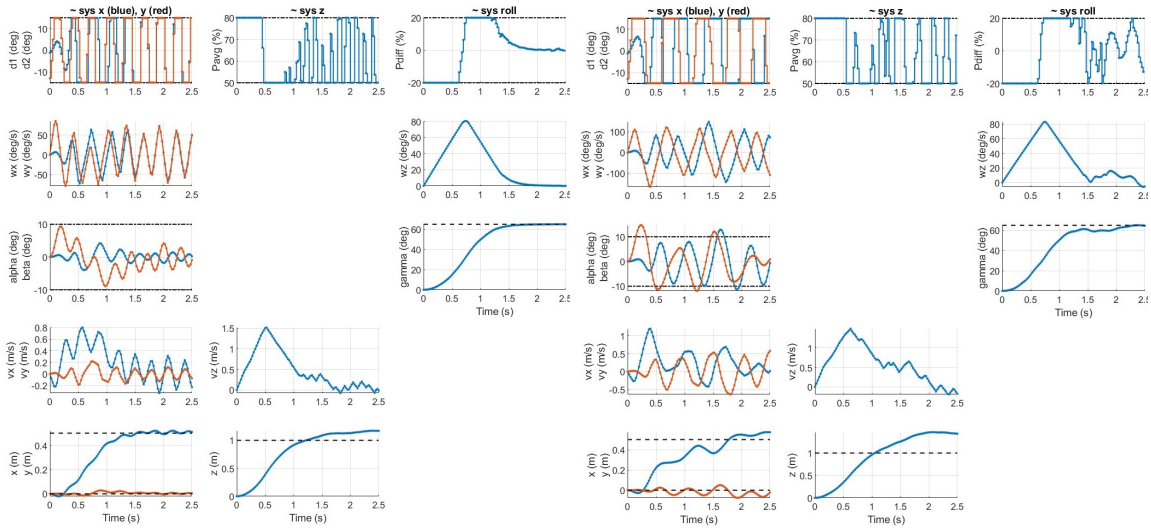

Figure 22: System without delay

We observed that with a delay of 25 ms the inputs of the system starts to oscillate in a significant way even if the performances in terms of tracking are not too affected. The rocket is indeed still able to track the given reference. Similar, but worst, results are obtained with a delay of 50 ms. In figure 23a we plot the obtained results with a delay of 50 ms.

With a delay of 75 ms the system is oscillating in an aggressive way. In figure 23b we plot the obtained results.

From the results obtained we can say that with a computational delay that of even 25 ms the performances start to degrade but the system is still able to track the reference (even if with some oscillations).

With a delay that is bigger than 75 ms the system oscillate aggressively and the system is completely unstable.
This is very interesting as it shows how even the smallest delay can quickly degraded the performances of a system.




(a) Delay of 50 ms

(b) Delay of 75 ms

## 8.2 Delay compensator design

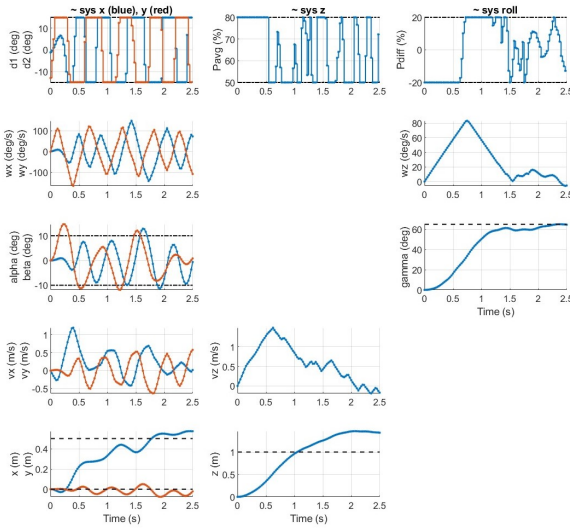
The idea behind the delay compensator is very simple: if we have an estimate of the computational delay $\delta$ instead of computing at time $k$ the optimal input based on the state $x(k)$ which will be applied at time $k+\delta$ we can estimate the state $x(k+\delta)$ and compute the estimated optimal input $u(k+\delta)$. Then, this estimated optimal input will be applied at time $k+\delta$.

To estimate the next state at time $x_{k+1}$ given the state at time $x_k$, the sampling time $T_s$ and the system dynamic $f(x, y)$ we can use the Euler integration:

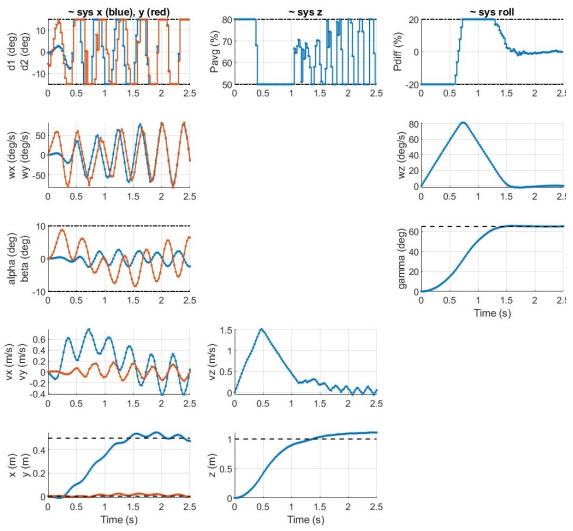
$$x_{k+1} = x_k + T_s \cdot f(x_k, u_k)$$

Starting from $x_k$ we can estimate the states $[x_k, ..., x_{k+\delta}]$ given that we know the inputs we will apply $[u_k, ..., u_{k+\delta-1}]$.

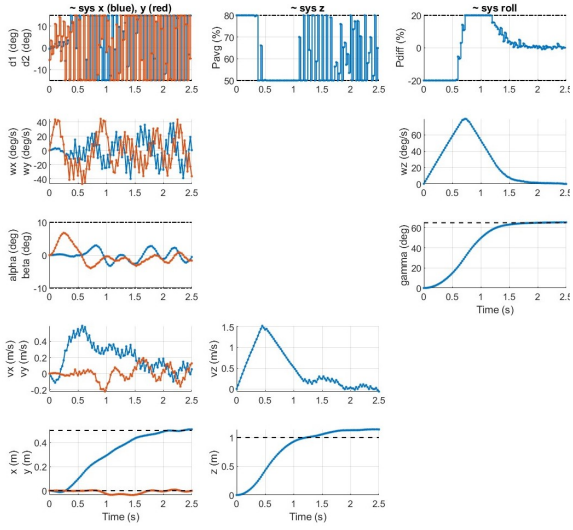
Below, in figure 24a, 24b, 25a and 25b, we plot the results obtained with the implemented compensator for a delay of 75 ms and different values of estimated delay.
From the results we can see how important it is not only to compensate the delay but also to estimate correctly the delay of the system.



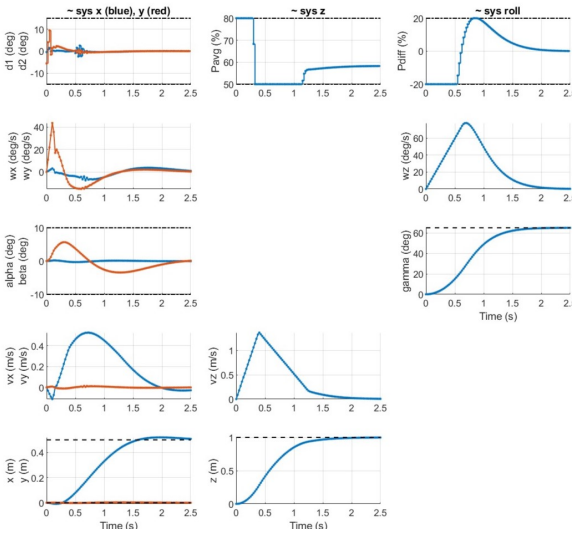

(a) Delay 75 ms - Estimated delay of 0 ms




(b) Delay 75 ms - Estimated delay of 25 ms




(a) Delay 75 ms - Estimated delay of 50 ms




(b) Delay 75 ms - Estimated delay of 75 ms