GROUP 78
Bonvalot Isione 283395
Gieruc Théo 282360
In-Albon Malika 283437

Mini-Project
Course : Model Predictive Control
Fall 2021

2022-01-31

Part 1 : Introduction

This project is part of the Course *Model Predictive Control* at EPFL. The objective is to develop an MPC controller to fly a rocket prototype.

The system will first be linearized in order to use linear MPC. We will design an MPC regulator, we will then add the tracking and an estimator to cancel a possible offset caused by a change in the mass of the rocket. This model will be tested in a non-linear simulation. This will show the weakness of applying a linear MPC on a non-linear system.

Finally, a non-linear MPC will be designed using CASADI and its performances will be compared to the linear MPC.

Part 2 : Linearization

Deliverable 2.1 : Explain from an intuitive physical / mechanical perspective, why this separation into independent subsystems occurs

As explained in the project description given, the system is described by 12 states where $\mathbf{x} = [\vec{\omega}^T \vec{\phi}^T \vec{v}^T \vec{p}^T]$ is the state vector. ω are the angular velocities around the body axes, ϕ are the Euler angles, **v** is the velocity and **p** the position.
The dynamic equation for the rocket is $\dot{x} = f(x, u)$.

The detailed equations in the project description show a link between the states:

- Thrust vector angle δ_2 to position x
- Thrust vector angle δ_1 to position y
- Average throttle P_{avg} to height z
- Differential throttle P_{diff} to roll angle γ

This allows a separation into four independent subsystems with some specific inputs controlling each subsystem. For example, the sys_x subsystem will only be affected by the angular velocity on y, the β angle, the velocity on x and the position x. So there is no need to include the other states in the controller for this particular system. The explanation is similar for the three other subsystems.

Another explanation of the separation into independent subsystems is done by using the physical influence of each state on the subsystems. This explanation only works with very small α and β angles. This is why, in the nonlinear case, it is not possible to separate our model into multiple subsystems.

- In the X subsystem, the position and movement of the rocket along the X direction can be described by the rotation around the Y axis (β) and its angular velocity (ω_y), as by its position (x) and speed (v_x) in the plan. The input δ_2 has a direct effect on β and ω_y.
- Because of its symmetry, the rocket moves in the Y direction in a similar way as in the X direction. The input δ_1 influences the state α, ω_x, y, v_y which describes the movement of the rocket along the Y direction.
- For the Z direction and small α and β angles, the input P_{avg} influences only the speed v_z and the position z.
- The roll of the rocket is described by its rotation around the Z axis and its rotational speed (γ and ω_z). The differential throttle P_{diff} creates a torque on the rocket around the Z axis, making it roll.

Part 3 : Design MPC Controllers for Each Sub-System

Deliverable 3.1 : Design MPC Regulators

The optimization problem to solve for each of the independent subsystem is:

$$J^*(x) = \min \sum_{i=0}^{N-1} l(x_i, u_i) + x_N^T Q_f x_N$$

$$\begin{aligned} s.t.\ \ x_{i+1} &= Ax_i + Bu_i \\ Fx_i &\leq f \\ Mu_i &\leq m \\ x_N &\in X \end{aligned}$$

Constraints

The design process has been the same for the four controllers. First the constraints are set. Due to the linearization, both the X and Y controllers have constraints on their state, on β and α respectively. These constraints are not applied on the first step but only on the following steps.

- **For the X controller**: $Mu \leq m$ with $M = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ and $m = \begin{bmatrix} 0.26 \\ 0.26 \end{bmatrix}$ in order to keep δ_2 between $\pm$0.26 rad.
  $Fx \leq f$ with $F = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}$ and $f = \begin{bmatrix} 0.0873 \\ 0.0873 \end{bmatrix}$ in order to keep β between $\pm$0.873 rad.
- **For the Y controller**: $Mu \leq m$ with $M = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ and $m = \begin{bmatrix} 0.26 \\ 0.26 \end{bmatrix}$ in order to keep δ_1 between $\pm$0.26 rad.
  $Fx \leq f$ with $F = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}$ and $f = \begin{bmatrix} 0.0873 \\ 0.0873 \end{bmatrix}$ in order to keep α between $\pm$0.873 rad.
- **For the Z controller**: $Mu \leq m$ with $M = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ and $m = \begin{bmatrix} 23.33 \\ 6.66 \end{bmatrix}$ in order to keep P_{avg} between 50 and 80%. There are no constrain on v_z and z.
- **For the Roll controller**: $Mu \leq m$ with $M = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ and $m = \begin{bmatrix} 20 \\ 20 \end{bmatrix}$ in order to keep P_{diff} between $\pm$ 20%. There are no constraint on the ω_z and γ.

Cost Matrices

The cost matrices were found by trial and error, until having a controller stable but fast. The LQR controller for unconstrained system is also computed, as well as the maximal invariant set in order to have X_f, F_f, Q_f and f_f. The Horizon length has been set to 4 seconds, which gives enough time to the controller to reach the Terminal set.

The quadratic cost function to optimize in a LQR controller is

$$l(x, u) = x^T Q x + u^T R u$$

The two cost matrices, Q and R, need to be tuned. It is a trade-off between the size of the terminal set and the tracking performance. A small terminal set leads more easily to an infeasible set. In this section, a large feasible set is not required.
Generally, the weights on the tracking (x for the X controller, y for the Y controller, ...) will be set high, in order to have good tracking. In the case of the X and Y controllers, the weight on ω_y and ω_x respectively will also be set high, in order to avoid violating the conditions on β and α.

- **For the X controller**: Q = $\begin{array}{c} \\ \omega_y \\ \beta \\ v_x \\ x \end{array} \begin{array}{c} \begin{array}{cccc} \omega_y & \beta & v_x & x \end{array} \\ \begin{pmatrix} 10 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 10 \end{pmatrix} \end{array}$ and R = 1.

To avoid angle constraint violations, the weight on ω_y has been set high. In order to have good regulation on x, its weight has also been increased.
As the input δ_2 does not have any requirement, R is equal to 1.

The Terminal set is given at Figure 1. It is useful as it indicates there in which the rocket should be after the Horizon time.

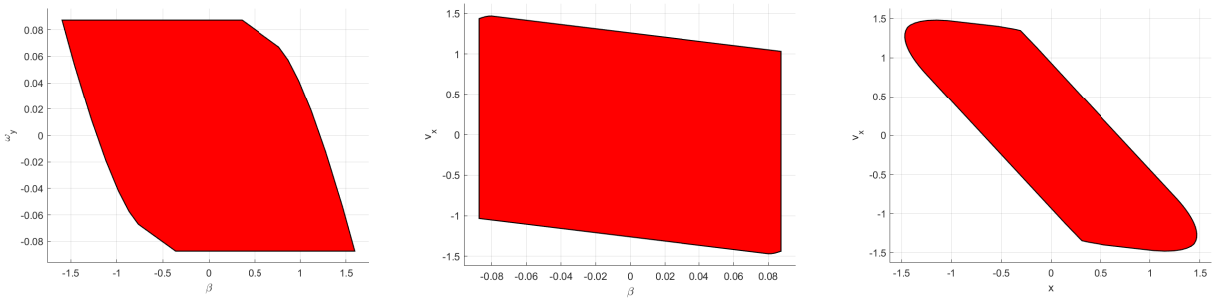

Figure 1: The three projections of the terminal set of the X controller

- **For the Y controller**: $Q = \begin{array}{c} \\ \omega_x \\ \alpha \\ v_y \\ y \end{array} \begin{array}{c} \begin{array}{cccc} \omega_x & \alpha & v_y & y \end{array} \\ \begin{pmatrix} 10 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 10 \end{pmatrix} \end{array}$ and $R = 1$.

  This controller is the same than the X controller. The terminal set is also identical.

- **For the Z controller**: $Q = \begin{array}{c} \\ v_z \\ z \end{array} \begin{array}{c} \begin{array}{cc} v_z & z \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0 & 40 \end{pmatrix} \end{array}$ and $R = 1$.

  We want to have a fast regulation in z, so its weight in the Q matrix is high.

  The Terminal set is given in Figure 2, projected on v_z and z. Because of the asymmetry of the constraints on P_{avg} and the gravity, the terminal is also asymmetrical.

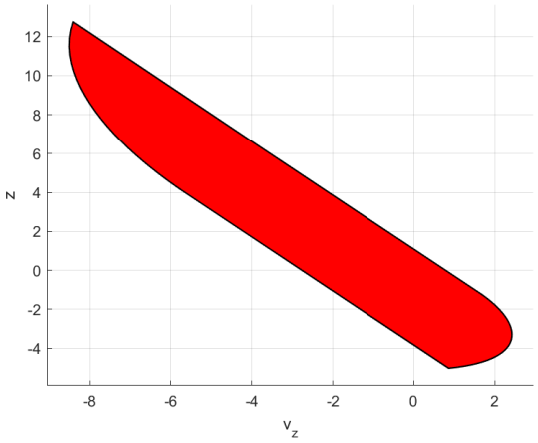

Figure 2: Terminal set of the Z Controller

- **For the Roll controller**: $Q = \begin{array}{c} \\ w_z \\ \gamma \end{array} \begin{array}{c} \begin{array}{cc} w_z & \gamma \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0 & 20 \end{pmatrix} \end{array}$ and $R = 0.01$.

  It is a quite aggressive regulator that allows fast regulation. The small R parameter allows the controller to use the full range of P_{diff}.

  The Terminal set is given at Figure 3, projected on ω_z and γ.

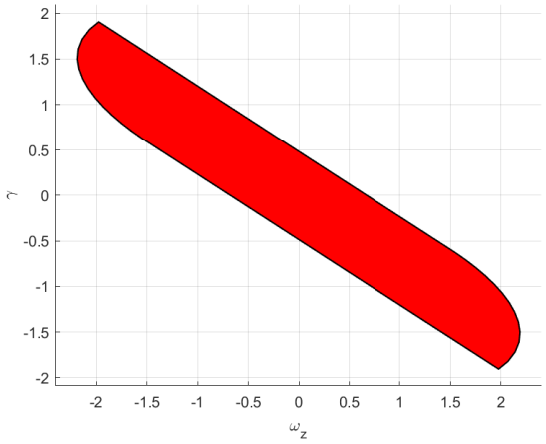

Figure 3: Terminal set of the Roll Controller

Finally, the problem constraints and the objective are set.

$$con = \left(\sum_{i=1}^{N-1} X_{i+1} == AX_i + BU_i\right) + \left(\sum_{i=1}^{N-1} MU_i \leq m\right) + \left(\sum_{i=2}^{N-1} FX_i \leq f\right) + (F_f X_N \leq f_f)$$

$$obj = \sum_{i=1}^{N-1} U_i^T R U_i + \sum_{i=2}^{N-1} X_i^T Q X_i + X_N^T Q_f X_N$$

These constraints and the objective are finally given to the Gurobi optimizer.

Simulation

- **For the X controller:** The rocket starts at position [5, 0, 0] with no speed. It has to go to the origin and stabilize.

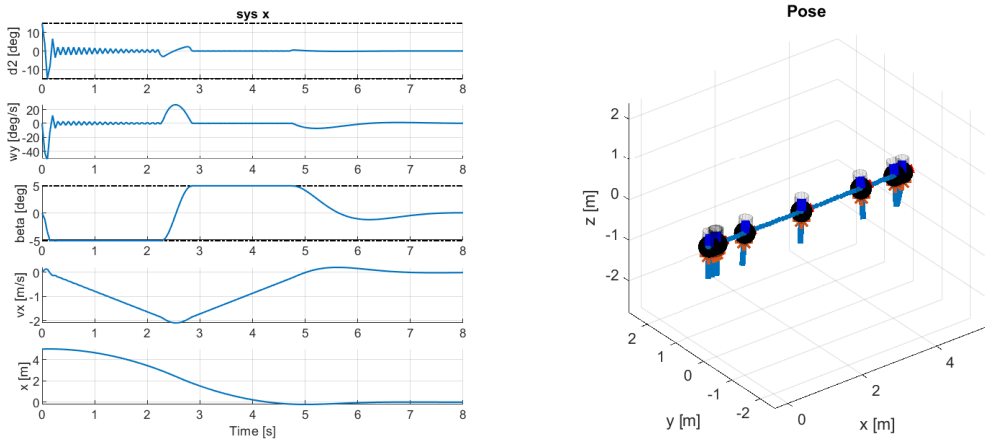

Figure 4: Test of the regulation of the X controller

As shown at Figure 4, no constraint has been violated. The controller needs about five seconds to arrive to the origin point and stabilize.

- **For the Y controller:** The rocket starts at position [0, 5, 0] with no speed. It has to go to the origin and stabilize.

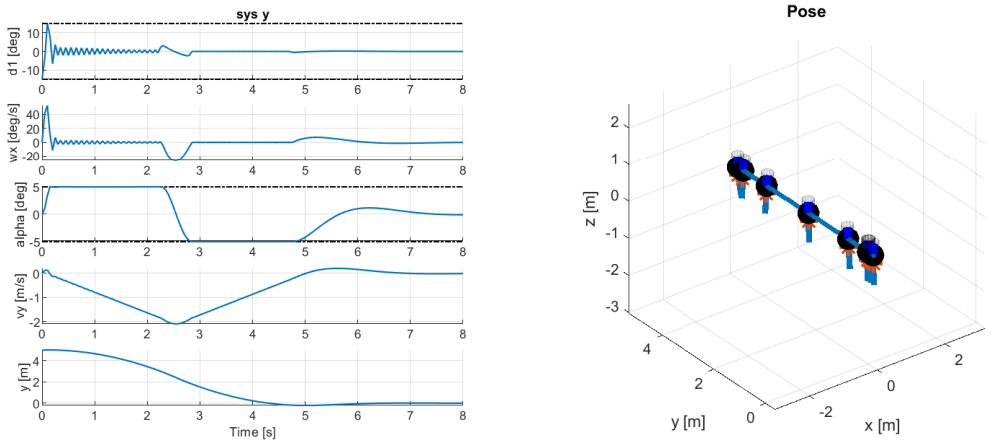

Figure 5: Test of the regulation of the Y controller

It has the same behaviour than the X controller, as they both have the same model and cost matrices.

- **For the Z controller:** The rocket starts at position [0, 0, 5] with no speed. It has to go to the origin and stabilize.

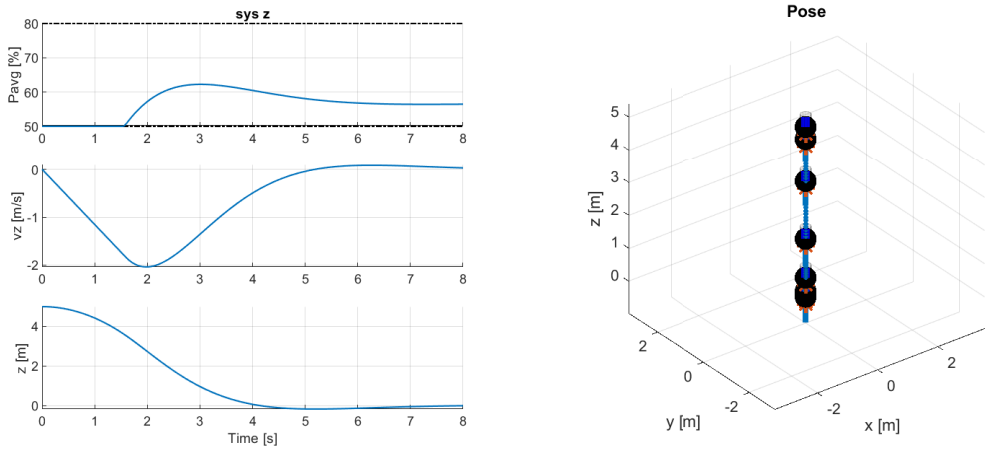

Figure 6: Test of the regulation of the Z controller

As shown at Figure 6, no constraint has been violated. The controller needs a little less than five seconds to arrive to the origin point and stabilize. The rocket uses its maximal power to descend in the first 1.5 seconds, before slowing down in order to stabilize with little to no overshoot.

- **For the Roll controller:** The rocket starts at position [0, 0, 0] with a 90°angle in the Z axis. It has to stay at the origin point and turn on itself until $\gamma = 0$.

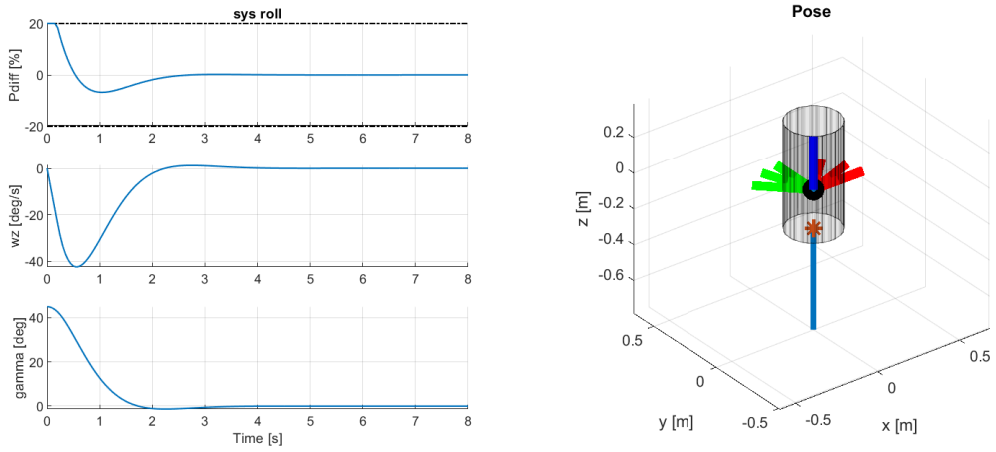

Figure 7: Test of the regulation of the Roll controller

As shown at Figure 7, no constraint has been violated. The controller is very fast but has a little overshoot.

Deliverable 3.2 : Design MPC Tracking Controllers

In this part, we have to design a MPC tracking controller. The optimization problem and the constraints remain the same as the previous section, except that we define deviation variables with the delta formulation:

$$\Delta x = x - x_s$$
$$\Delta u = u - u_s$$

and we replace them in the previous optimization problem of part 3.1.
The constraints and objective are the following for each controller :

$$con = [Mu_s \leq m,\ x_s == Ax_s + Bu_s,\ ref == Cx_s + D]$$
$$obj = u_s^2$$

Figures 8, 9, 10 and 11 show the tracking simulation. These figures show identical properties as the ones in the regulation section.

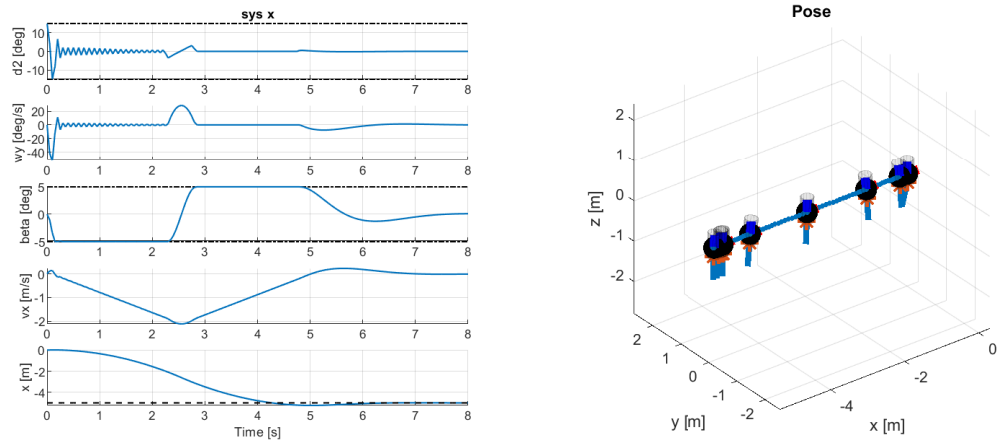

Figure 8: Test of the tracking of the X controller

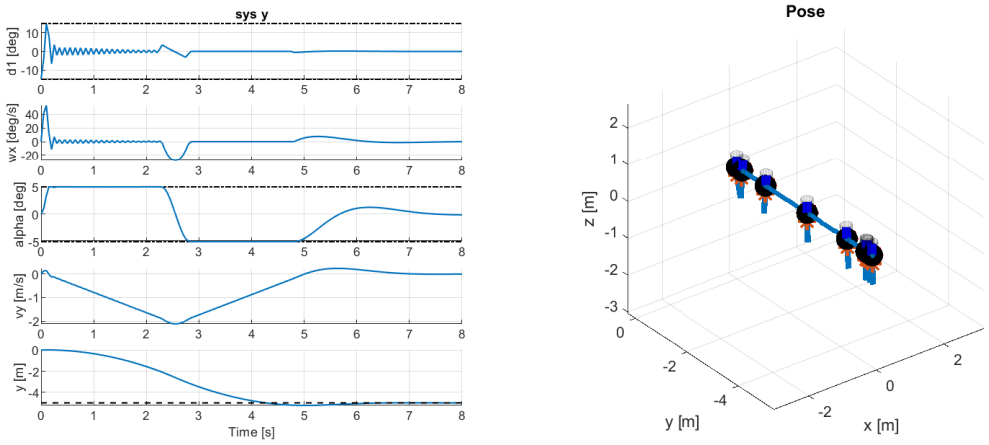

Figure 9: Test of the tracking of the Y controller

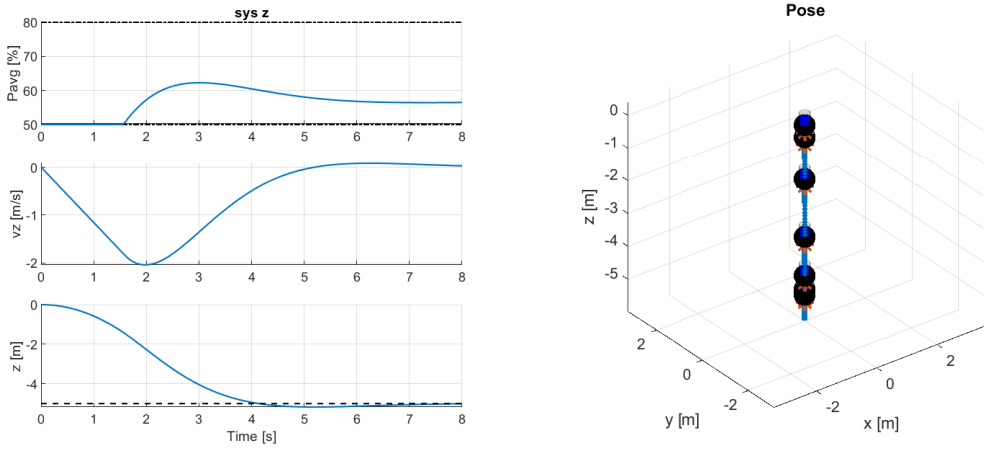

Figure 10: Test of the tracking of the Z controller

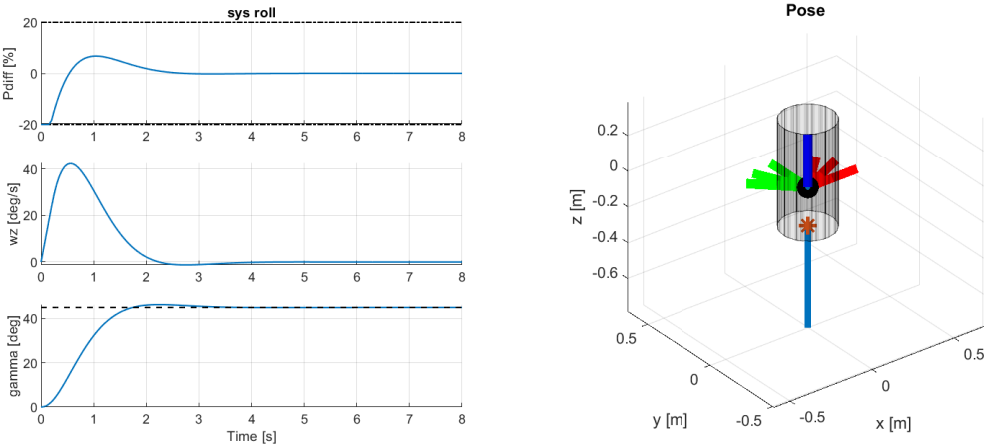

Figure 11: Test of the tracking of the Roll controller

Part 4 : Simulation with Nonlinear Rocket

Deliverable 4.1

Simulate the full nonlinear system with our four controllers from the origin as initial state

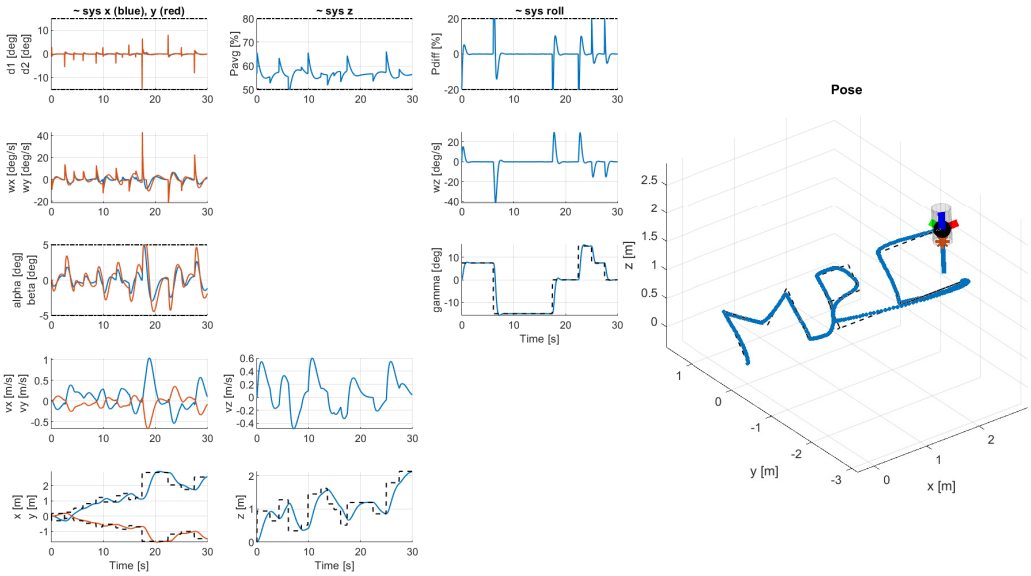

Figure 12: Plot of the linear controllers tracking the MPC reference path and reference roll angle.

As shown in Figure 12, the tracking is not perfect. This is due to the fact that the controllers are linear and the simulation is nonlinear. That causes a mismatch between the predictive controllers and the real simulated system. To improve the performance, we tuned the parameters by trying different combinations, which lead us to have a better understanding of each parameter influence. The number of parameters and the way they interfered with each other made the task difficult.

For systems X and Y, we further increased the weights on the angular velocities to avoid angle constraint violations. Angles α and β are constraint to keep the approximation due to the linearization of the system valid. To keep good tracking performance, we keep the high weight on the position. Here is how we tune the parameter for MPC control of system X, the system Y is tuned in a similar way.

$$\text{X controller: Q} = \begin{array}{c} \\ \omega_y \\ \beta \\ v_x \\ x \end{array} \begin{array}{c} \begin{array}{cccc} \omega_y & \beta & v_x & x \end{array} \\ \begin{pmatrix} 35 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 10 \end{pmatrix} \end{array} \text{ and R} = 1 \ (\delta_2)$$

For the system Z, we lowered the weight of the position in comparison to the normal speed so that its tracking performance is as *bad* as the controllers for the X and Y directions. Having the same tracking performance in all direction helps making straighter lines in the drawing. However, the R parameters of the average throttle is smaller in order to make the system more dynamic, allowing it to use it with less constrains, making the changes of direction faster, with less overshoot.

$$\text{Z controller: Q} = \begin{array}{c} \\ v_z \\ z \end{array} \begin{array}{c} \begin{array}{cc} v_z & z \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0 & 10 \end{pmatrix} \end{array} \text{ and R} = 0.1 \ (P_{avg})$$

For the roll controller, we further increased the weight of the gamma in order to have better tracking performances. The weight of the differential throttle is also decreased, allowing the controller to use the full range of the differential throttle.

$$\text{Roll controller: Q} = \begin{array}{c} \\ w_z \\ \gamma \end{array} \begin{array}{c} \begin{array}{cc} w_z & \gamma \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0 & 400 \end{pmatrix} \end{array} \text{ and R} = 0.01 \ (P_{diff})$$

Part 5 : Offset-Free Tracking

Deliverable 5.1

Explanation of the design procedure and choice of tuning parameters

In this section, we assume that the mass of the rocket changes. The dynamics of the system Z is now

$$\mathbf{x}^+ = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{B}d \tag{1}$$

where d is a constant, unknown disturbance simulating the mass variation of the rocket. Therefore we have to adapt the z controller by adding an estimator.

The augmented model allows us to calculate the state and disturbance estimator L.

$$\begin{bmatrix}\hat{x}_{k+1}\\ \hat{d}_{k+1}\end{bmatrix} = \begin{bmatrix}A & B_d\\ 0 & I\end{bmatrix}\begin{bmatrix}\hat{x}_k\\ \hat{d}_k\end{bmatrix} + \begin{bmatrix}B\\ 0\end{bmatrix}u_k + \begin{bmatrix}L_x\\ L_d\end{bmatrix}(C\hat{x}_k + C_d\hat{d}_k - y_k) \tag{2}$$

where $\hat{x}, \hat{d}$ are estimates of the state and disturbance. The matrices $B_d = B$ and $C_d = 0$.

The error dynamics are the following:

$$\begin{bmatrix}x_{k+1} - \hat{x}_{k+1}\\ d_{k+1} - \hat{d}_{k+1}\end{bmatrix} = \left(\begin{bmatrix}A & B_d\\ 0 & I\end{bmatrix} + \begin{bmatrix}L_x\\ L_d\end{bmatrix}\begin{bmatrix}C & C_d\end{bmatrix}\right)\begin{bmatrix}x_k - \hat{x_k}\\ d_k - \hat{d}_k\end{bmatrix} \tag{3}$$

with $\bar{A} = \begin{bmatrix}A & B\\ 0 & I\end{bmatrix}$ and $\bar{C} = \begin{bmatrix}C & 0\end{bmatrix}$ because $B_d = B$ and $C_d = 0$.
It is now possible to calculate L with the help of the function `place()` of MatLab using $\bar{A}$, $\bar{C}$ and the poles [0.1, 0.2, 0.3]. We chose the poles by trial and error until we got satisfactory results, paying attention to keep them in the unit circle to ensure stability.

Plot showing the impact of changing the mass on our original controller from Part 4, and plot showing that our controller now achieves offset-free tracking.

The Figure 13 shows the system, after a change of its mass, without an estimator. This creates an offset that the system is not able to correct. With an estimator, the system is able to correct this offset. This is shown in Figure 14.

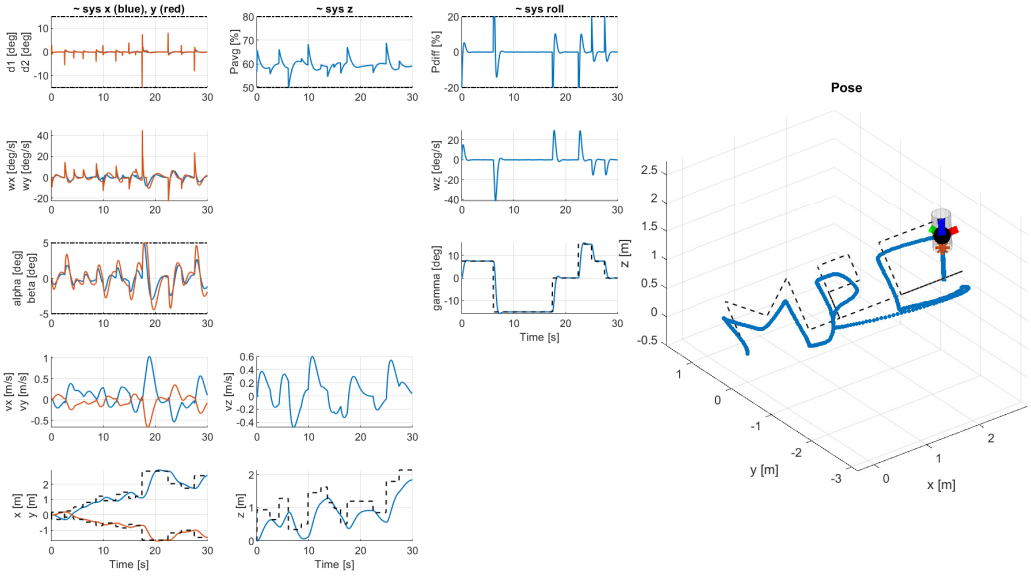

Figure 13: Plot including the disturbance d with linear controllers of part 4

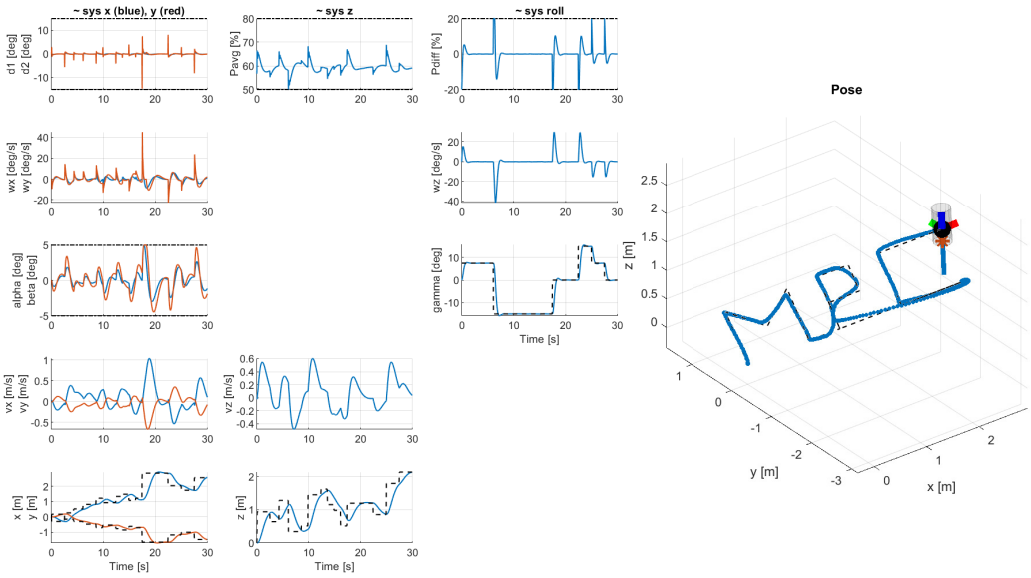

Figure 14: Plot including the disturbance with updated controller to reject the disturbance

Part 6 : Nonlinear MPC

Deliverable 6.1: Develop a nonlinear MPC controller for the rocket using CASADI

Explanation of our design procedure and choice of tuning parameters.

In this part, the goal is to develop a nonlinear MPC controller able to track the given reference. The simulation is done for the default maximum roll angle $|\gamma_{ref}| = 15°$ and for a maximum roll reference of $|\gamma_{ref}| = 50°$. As the computation time could be slower, the sample period has been increased to 0.1 seconds. The horizon is set to $H = 4$ to avoid the need of a terminal cost and still have good tracking performance while limiting overshoot.

The nonlinear optimization problem to solve is:

$$J^*(x) = \min \sum_{i=0}^{N-1} (x_i - x_{ref})^T Q (x_i - x_{ref}) + u_i^T R u_i + (x_N - x_{ref})^T Q (x_N - x_{ref})$$

$$\begin{aligned} s.t.\ \ x_{i+1} &= f_{discrete}(x_i, u_i) \\ Mu_i &\leq m \\ x_0 &= X \end{aligned}$$

The Runge-Kutta 4 was used to discretize the system with a sampling period $T_s = 0.1s$.

$$f_{discrete} = x + \frac{T_s}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$\begin{aligned} k_1 &= f(x, u) \cdot y \\ k_2 &= f(x + \frac{T_s}{2} \cdot k_1, u) \\ k_3 &= f(x + \frac{T_s}{2} \cdot k_2, u) \\ k_4 &= f(x + T_s \cdot k_3, u) \end{aligned}$$

One should also implement the constraints for the nonlinear case.
The constraints on the inputs are:

- δ_1 and δ_2 up to ± 0.26 rad
- P_{avg} between 50% and 80%
- P_{diff} up to $\pm 20\%$

This is done in the code by:

```
opti.subject_to([-0.26; -0.26; 50; -20] <= U_sym <= [0.26; 0.26; 80; 20])
```

There is only one constraint on the states, which is:

- β up to $\pm 85°$

This is done in the code by:

```
opti.subject_to(-deg2rad(85) <= X_sym(5,:) <= deg2rad(85))
```

In the nonlinear case, there is no need to keep the constraints on the small angles like in the linear case ($-5° \leq \alpha, \beta \leq 5°$) as the NMPC controller should cover the whole space. The only constraint is due to the Euler angle attitude representation used in the model, which has a singularity at $\beta = 90°$.
The different matrices of the system are M, Q, R. In this part, we do not decompose the rocket into four sub-systems as we did in the linear dynamic part.

$$M = \begin{array}{c} \\ \omega_x \\ \omega_y \\ \omega_z \\ \alpha \\ \beta \\ \gamma \\ v_x \\ v_y \\ v_z \\ x \\ y \\ z \end{array} \begin{array}{c} \begin{array}{cccc} x & y & z & \gamma \end{array} \\ \begin{pmatrix} 0 & \dots & \dots & 0 \\ \vdots & & & \vdots \\ & & & \\ & \ddots & & \vdots \\ & & 0 & 0 \\ & \dots & 0 & 1 \\ & & & 0 \\ \vdots & & & \vdots \\ 0 & & & \\ 1 & 0 & \dots & \\ 0 & 1 & & \vdots \\ \vdots & 0 & 1 & 0 \end{pmatrix} \end{array} \quad Q = \begin{array}{c} \\ \omega_x \\ \omega_y \\ \omega_z \\ \alpha \\ \beta \\ \gamma \\ v_x \\ v_y \\ v_z \\ x \\ y \\ z \end{array} \begin{array}{c} \begin{array}{ccccccccccccc} \omega_x & \omega_y & \omega_z & \alpha & \beta & \gamma & v_x & v_y & v_z & x & y & z \end{array} \\ \begin{pmatrix} 30 & 0 & \dots & & & & & & & & \dots & 0 \\ 0 & 30 & & & & & & & & & & \vdots \\ \vdots & & 1 & & & & & & & & & \\ & & & 1 & & & & & & & & \\ & & & & 1 & & & & & & & \\ & & & & & 500 & & & & & & \\ & & & & & & 20 & & & & & \\ & & & & & & & 20 & & & & \\ & & & & & & & & 20 & & & \\ & & & & & & & & & 5000 & & \vdots \\ & & & & & & & & & & 5000 & 0 \\ 0 & \dots & & & & & & & & \dots & 0 & 5000 \end{pmatrix} \end{array}$$

In the matrix Q, we gave a very large penalty on the X, Y and Z states to ensure a good tracking of the reference. We assigned a large value on γ state as we wanted our rocket to turn around itself in a precise way to avoid bad reference tracking on the tricky points, as for instance on the "M" vertices of the drawing. The angular velocities on X and Y also have a larger weight than the other states because they have an impact on how the rocket can change its direction during the tracking.

$$R = \begin{array}{c} \\ \delta_1 \\ \delta_2 \\ P_{avg} \\ P_{diff} \end{array} \begin{array}{c} \begin{array}{cccc} \delta_1 & \delta_2 & P_{avg} & P_{diff} \end{array} \\ \begin{pmatrix} 0.0001 & 0 & 0 & 0 \\ 0 & 0.0001 & 0 & 0 \\ 0 & 0 & 1.5 & 0 \\ 0 & 0 & 0 & 0.0001 \end{pmatrix} \end{array}$$

The R matrix is the coefficient for the input cost matrix and it influences the aggressivity of our controller. For δ_1 δ_2 and P_{diff}, the coefficient has been chosen small because these input can be easily applied. However, the coefficient on P_{avg} is much larger, meaning that is it more difficult to be applied to our rocket (increase cost function). This leads to slower movements in the Z direction, similar to the movements in the X and Y directions, in order to have straighter lines in the drawing of the *MPC* path as shown in the Figure 17.

Discuss the pros and cons of our nonlinear controller vs the linear ones we developed earlier.

A nonlinear controller has many advantages as it can describe almost any model and any objectives. Here we can see that our nonlinear model works very well on our rocket model and for the objective of tracking the "MPC" drawing. Moreover, the nonlinear controller has a much larger set of states where the controller is efficient. This means that our nonlinear controller has a large feasible set and is able to work well from almost any initial position. To illustrate this, we tried to see how our nonlinear controller would behave with two very far initial positions. The result is depicted in Figures 15 and 16. One can remark that even if the initial position of the rocket is very distant from the origin, the nonlinear controller needs quite a large time to bring back the rocket to a good position but then, it is able to track the reference with success. This shows the ability of the nonlinear controller to compute a control law from a random initial position.
However, the nonlinear controller is very computationally intense, meaning that we can not have a too small sampling period for the discretization of our model, and that we can not have a too large horizon. We chose to divide our previous horizon by two to have reasonable computation time.

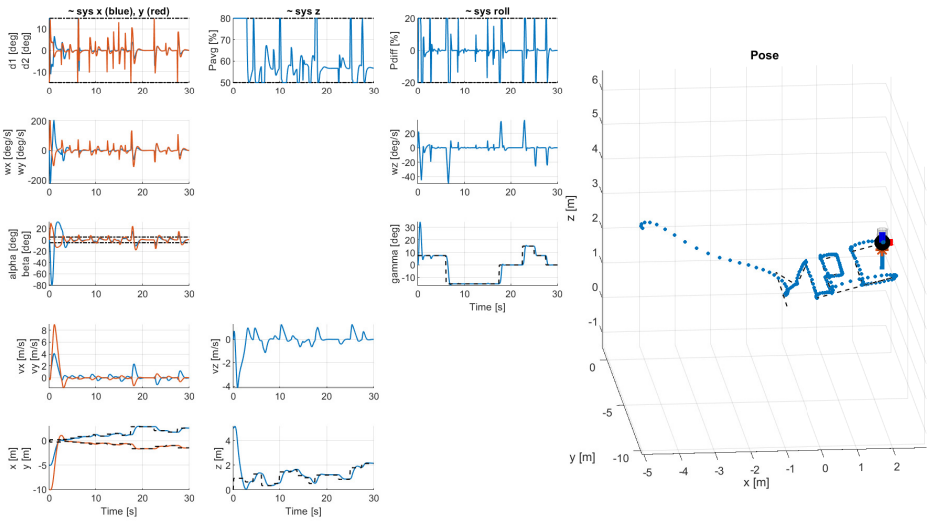

Figure 15: Plot of the NMPC tracking the MPC reference path from initial position (x, y, z) = (-5, -10, 5).

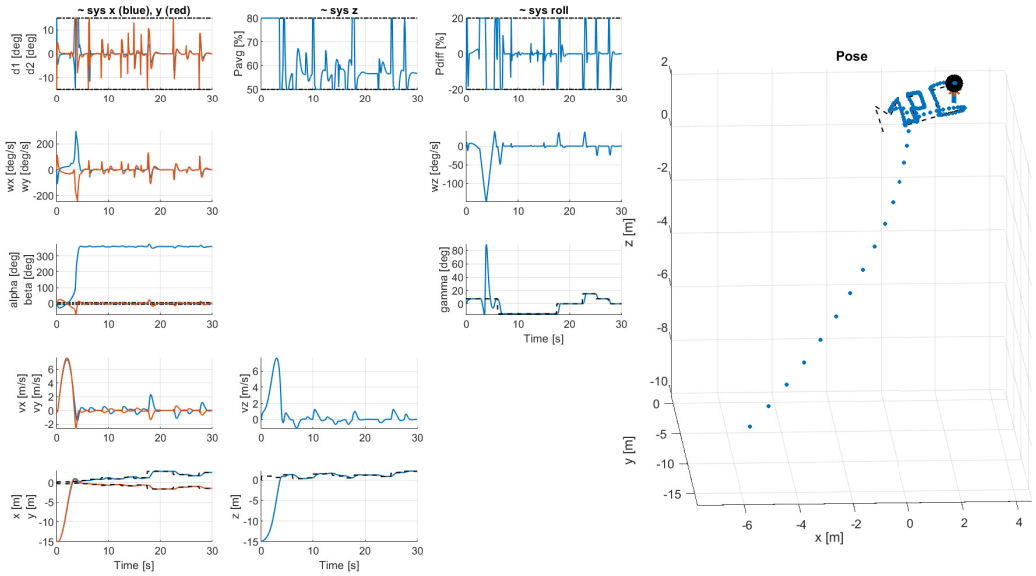

Figure 16: Plot of the NMPC tracking the MPC reference path from initial position (x, y, z) = (-15, -15, -15).

Concerning the constraints, the main difference between the linear and the nonlinear controller remains in the angle linearization constraints. In the linearization part, we needed small angles on α and β to ensure a good linearization of the system, whereas in the nonlinear case, these constraints are not useful anymore. This difference can be observed in Figure 12 where α and β are up to $\pm 5°$ whereas in Figure 17, these angles are in average up to $\pm 10°$ and can be up to $\pm 20°$. This allows for a faster settling time and a better tracking.

Plots showing the performance of our controller.

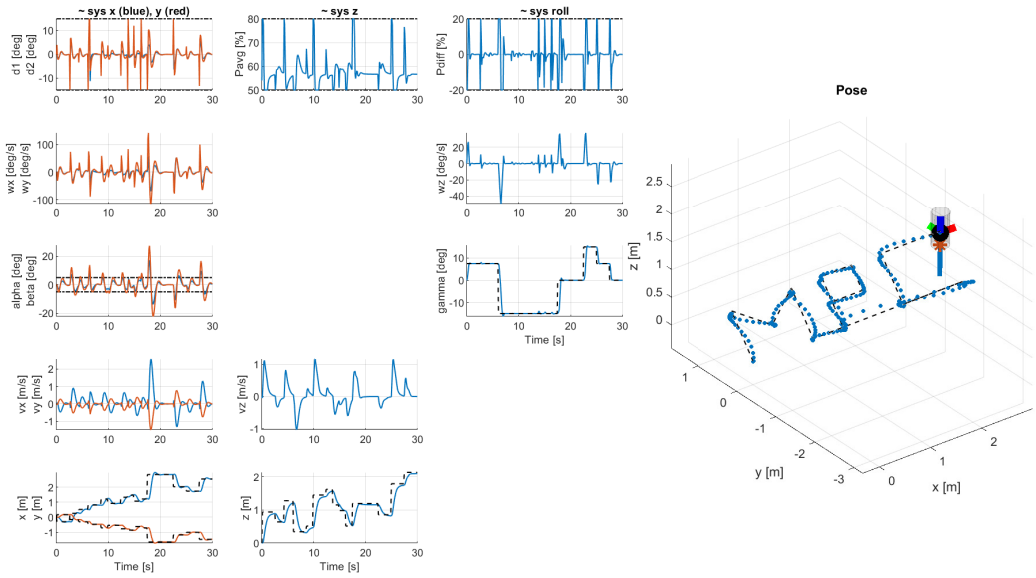

Figure 17: Plot of the NMPC tracking the MPC reference path and default reference roll angle (15°).

As depicted in Figure 17 the trajectory tracking is much better than in the linear controller and it is almost perfect due to the fact that we can implement a more aggressive behavior on some specific states (as detailed previously with the matrix Q) without violating any constraints. This can be seen on the *gamma* subplot in Figure 17.

For a reference angle of 50° with the nonlinear controller

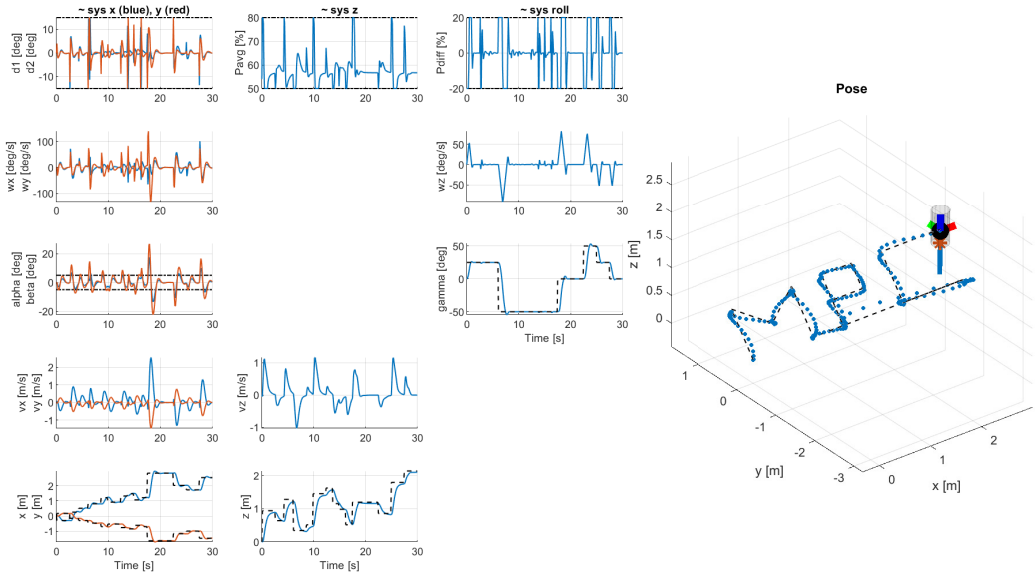

Figure 18: Plot of the NMPC tracking the MPC reference path and reference roll angle (50°).

The Figure 18 illustrates the behavior of the nonlinear controller for a reference tracking of $50°$ on the roll angle. As we can see, the general behavior of the tracking does not differ a lot from the one with a roll reference of $15°$. However, if we look in more details on the plot of the roll and of γ, we observe that for the maximum roll reference of $50°$, the tracking of γ is not as good as for the default roll reference. Furthermore,

the P_{diff} is much more saturated meaning that the throttle difference between the motors is getting maximum to try ensuring a good reference tracking for this larger angle. This can be seen in Figure 19.

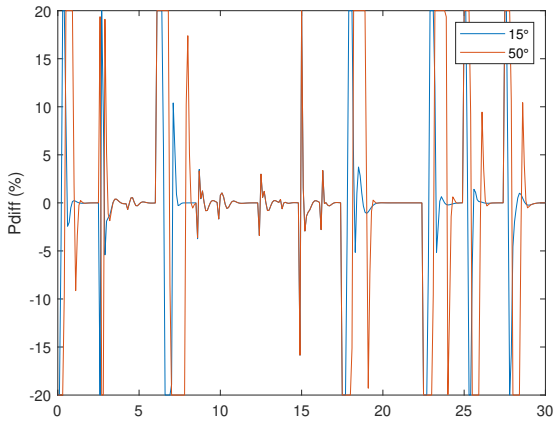

Figure 19: Plot of the P_{diff} comparison with NMPC tracking reference roll angle of 15° and 50°.

For a reference angle of 50° with the linear controller

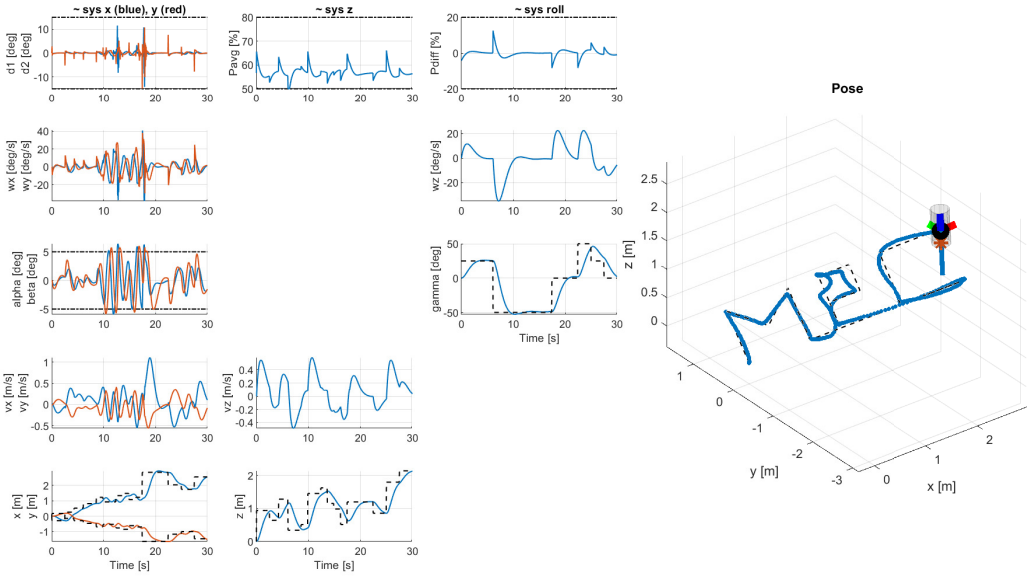

Figure 20: Plot of the linear controller for tracking the MPC reference path and reference roll angle (50°).

The Figure 20 shows the behavior of the linear controller for a reference tracking of $50°$ on the roll angle. It is obvious that the performance of the linear controller is not good due to the extreme reference request. It helps us to see the impact of the roll, especially on the ”P” drawing reference where the rocket should follow a hard trajectory on the γ angle. We observe that a linear model is not able to follow the trajectory as well as the nonlinear did. We can also notice the unstable behavior on the α and β angles in this case. The linearized model does not take into account the effect that the rotation of the rocket around its z axis has on *system X* and *system Y*, whereas the nonlinear model allows to take it into account, resulting in a better tracking.