MPC mini-project

Guibbert, Arnaud
arnaud.guibbert@epfl.ch
321085

Papadatos, Henry
henry.papadatos@epfl.ch
284446

Waelbroeck, Aitana
aitana.waelbroeck@epfl.ch
282764

January 29, 2022

Contents

1 Introduction

The goal of this project is to implement different types of controllers to fly a thrust vectoring rocket. The first approach that will be presented, is to linearize the model and apply a linear MPC controller. The second approach, is to apply a non-linear MPC controller.

2 Deliverable 2.1

The steady state of the rocket corresponds to the situation where the average throttle of the rocket P_{avg} alone compensates gravity, maintaining the position of the rocket straight and at the origin (relative to the world frame). Linearizing the model around this trim point, means that small angles approximation can be done since the linear model will only be accurate for small values of α and β. The linearization broke the model into four independent, linear and continuous time sub-systems of the form ($\dot{x} = Ax + Bu$) :

1. *sys_x*, with $x = [\omega_y, \beta, v_x, x]$ and $u = \delta_2$
   The deflection angle of the second servo $u = \delta_2$, only makes the rocket turn around the y_b axis, generating an angular velocity ω_y and affecting the Euler Angle β. The deflection δ_2 leads to a displacement along the x_w and z_w axis, but around the steady state position, due to the small angles approximation, the displacement along the z_w axis is negligible.

2. *sys_y*, with $x = [\omega_x, \alpha, v_y, y]$ and $u = \delta_1$
   The physical explanation for *sys_x*, is symmetrically valid here. The deflection angle of the first servo $u = \delta_1$, only makes the rocket turn around the x_b axis, generating an angular velocity ω_x and affecting the Euler Angle α. The deflection δ_1 leads to a displacement along the y_w and z_w axis, but around the steady state position, due to the small angles approximation, the displacement along z_w is negligible.

3. *sys_z*, with $x = [v_z, z]$ and $u = P_{avg}$.
   Intuitively, when considering small angle approximation (straight rocket) the z position only depends on its previous z position and the v_z speed at which the rockets flies in the world frame. Of course, P_{avg} will directly have an impact on the position z of the rocket, specially around the equilibrium point, where the rocket stays almost perfectly straight at all time. Around the steady state, the effect of the deflections δ_1 and δ_2 on z are negligible, since α and β are really small.

4. *sys_roll*, with $x = [\omega_z, \gamma]$ and $u = P_{diff}$
   The roll γ is directly linked to the angular speed ω_z generated by the throttle difference between the two motors. This is only true with the small angle approximation, taking into account that the rocket is almost straight at all times.

3 Deliverable 3.1

Recursive constraint satisfaction

Ensuring recursive constraint satisfaction and stability can be achieved if the following conditions are satisfied:

1. **The stage cost is a positive definite function**
   To fulfill this first point, we choose the cost matrices R, Q such that they are positive definite (achieved by using diagonal matrices with strictly positive coefficients).

2. **The terminal set $\mathcal{X}_f$ is invariant under the local control law $\kappa_f(x)$ (terminal controller) and all state and input constraints are satisfied in $\mathcal{X}_f$.**
   To satisfy these conditions, the terminal controller is defined as the unconstrained LQR controller ($u^* = Kx$) and the terminal set is defined as the maximum control invariant set for the unconstrained LQR controller[1] $\mathcal{X}_f$ such that $\mathcal{X}_f \in \mathbf{X}$ and $K\mathcal{X}_f \in \mathbf{U}$.

3. **The terminal cost $V_f(x)$ is a continuous Lyapunov function in the terminal set $\mathcal{X}_f$ that satisfies:**

$$V_f(x_+) - V_f(x) < -I(x, \kappa_f(x)) \;\; \forall x \in \mathcal{X}_f$$

   To fulfill this last condition, the terminal cost is defined as the optimal LQR cost, i.e. $V_f(x) = x^T P x$ where P is solution to the discrete time algebraic Riccati equation.

Procedure to tune the parameters

To ensure the good functioning and performances of the controller, some parameters need to be tuned. For each subsystem, one needs to wisely define the cost matrices R, Q, the horizon length N as well as the terminal cost. Here these parameters have been chosen according to the specifications: the system should be able to:

- Stabilize at the origin in less than 8 seconds when starting at 5 five meters from it (along x, y or z)
- Stabilize at the origin in less than 8 seconds when starting at 45° for roll.

The terminal cost is actually already chosen such that it ensures the stability of the controller.
The choice of the cost matrices is specific to each subsystem and is the result of a physical approach as well as an empirical testing. The cost coefficients are detailed for each subsystem in the following subsections.
The choice of the horizon length N is not an easy task. The horizon should be sufficiently high such that the system is able to reach the terminal set $\mathcal{X}_f$, but not too high, as the computational cost dramatically increases with N. Basically, for each subsystem, we started with a reasonable high horizon length (around 1.5 seconds, $N = 30$), then we adjusted the cost matrices and finally tried to reduce the horizon length as long as the terminal set remained reachable. Even if at the end we should select only one horizon length value for the merged controller, it was interesting to assess how each subsystem was behaving depending on the horizon length.

3.1 `Sys_x` controller

The `Sys_x` system consists of the following state: w_y, β, v_x, x and the control input is δ_2. The rocket should be able to stabilize at the origin in less than 8 seconds when starting at five meters from it along x. To do so, the rocket will need to rotate around the y axis such that it tilts the rocket towards the origin and produces a thrust along the x axis. However, one does not want to have a high angular velocity about y as the rocket risks to flip over and then free fall. Besides, the system should satisfy the constraints on β (± 5). To avoid this behavior, one should limit the angular velocity w_y. This can be achieved by adding a high penalty on it. Regarding the input δ_2, we decided to lower its cost to let the system use it (experience showed that it was not heavily used while obtaining good performances). Thus, the cost matrices used for this system are described below:

$$Q_1 = \begin{pmatrix} 80 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad R_1 = 0.1$$

Following the procedure described below, the horizon length was set to $1.1s$ [$N = 22$]. Figure (1) shows the terminal set, and figure (2) presents the final results obtained with the tuning parameters described above.

[1] maximum invariant set for the system $x^+ = (A + BK)x$

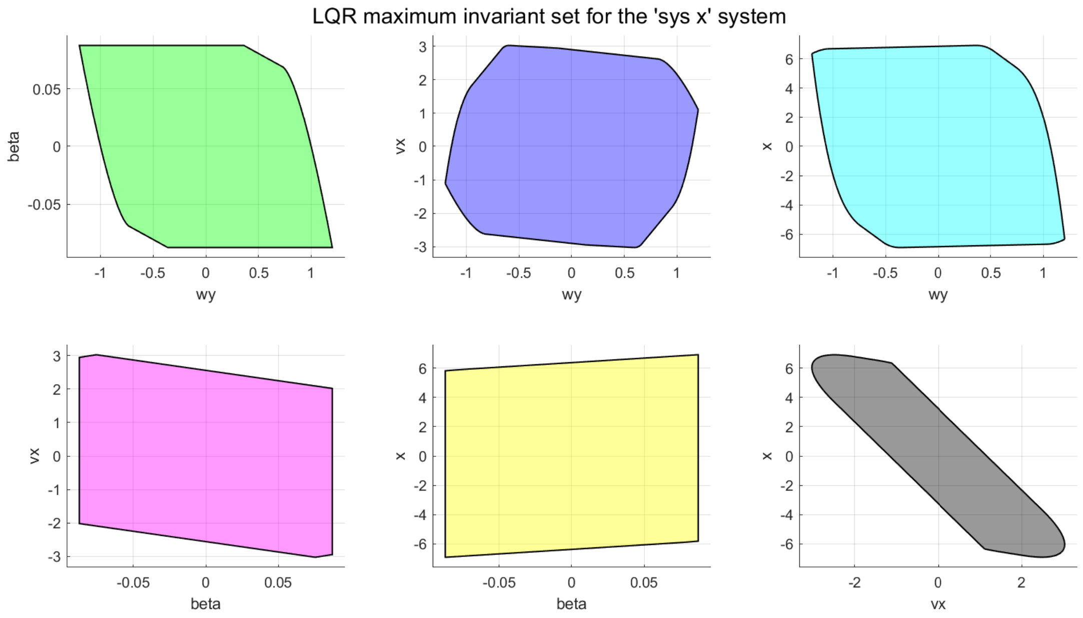
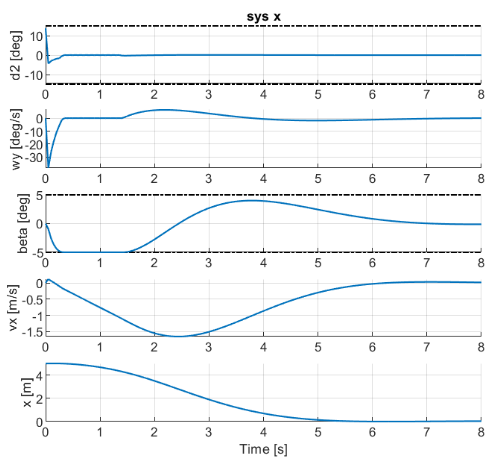


Figure 1: Terminal set $\mathcal{X}_f$ for the `Sys_x` system. The terminal set is the maximum control invariant set for the unconstrained LQR controller (i.e. maximum invariant set for the system $x^+ = (A + BK)x$).



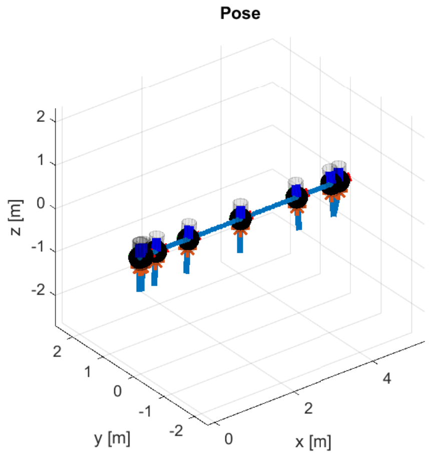


Figure 2: Controller's performances when starting stationary at five meters from the origin along x.

3.2 `Sys_y` controller

The `Sys_y` system consists of the following state: w_x, α, v_y, y and the control input is δ_x. Due to the symmetry of the rocket, this system is actually very similar to `Sys_x`: instead of having a thrust along x, we have a thrust along y and the rocket is able to generate it by tilting around x. Again, we want to avoid that the rocket flips over, that's why we set a high penalty coefficient on w_x. The resulting cost matrices are similar to the ones set for `Sys_x`:

$$Q_2 = \begin{pmatrix} 80 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad R_2 = 0.1$$

The horizon was also set to $1.1s$ $[N = 22]$. Figure (3) shows the terminal set, and figure (13) presents the final results obtained with the tuning parameters described above.

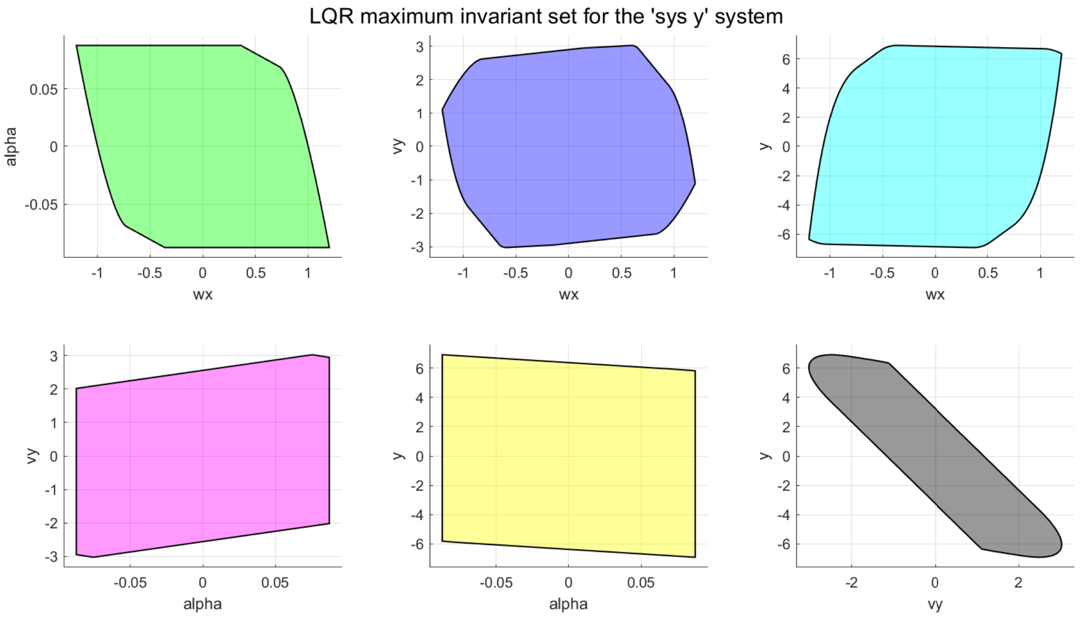


Figure 3: Terminal set $\mathcal{X}_f$ for the `Sys_y` system. The terminal set is the maximum control invariant set for the unconstrained LQR controller (i.e. maximum invariant set for the system $x^+ = (A + BK)x$).

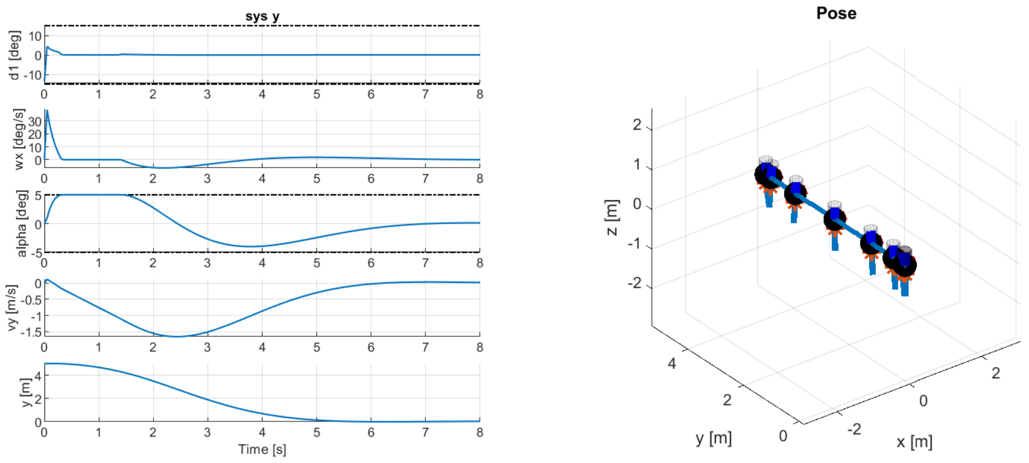


Figure 4: Controller's performances when starting stationary at five meters from the origin along y.

3.3 Sys_z controller

First, regarding the code, one could notice that a value $us = 56.667$ was hard-coded and that the constraints are shifted by the amount us. This comes from the fact that at the equilibrium point, a positive thrust P_{avg} is needed to compensate the gravity. Here we are minimizing over $\Delta u = u - u_s$, thus the constraints are shifted. If in the other subsystems, there was no similar hard-coded value, it is simply because at the equilibrium point all the other state variables are equal to 0 (thus implicitly hard-coded), P_{avg} is the only non-zero variable.

For `Sys_z`, the main issue was about the capability of the rocket to reach the terminal set given the horizon length. Indeed, the acceleration of the rocket towards negative z is both limited by the gravity and by the fact that we constrained P_{avg} to be higher than 50%. That's why the horizon length was set to $1.5s$ here. Regarding the parameters, we lowered the cost associated with P_{avg} such that the rocket could heavily use it and then avoid an eventual overshoot regarding z. The resulting cost matrices are given below:

$$Q_3 = I \quad R_3 = 0.05$$

Figure (5) shows the terminal set, and figure (6) presents the final results obtained with the tuning parameters described above.

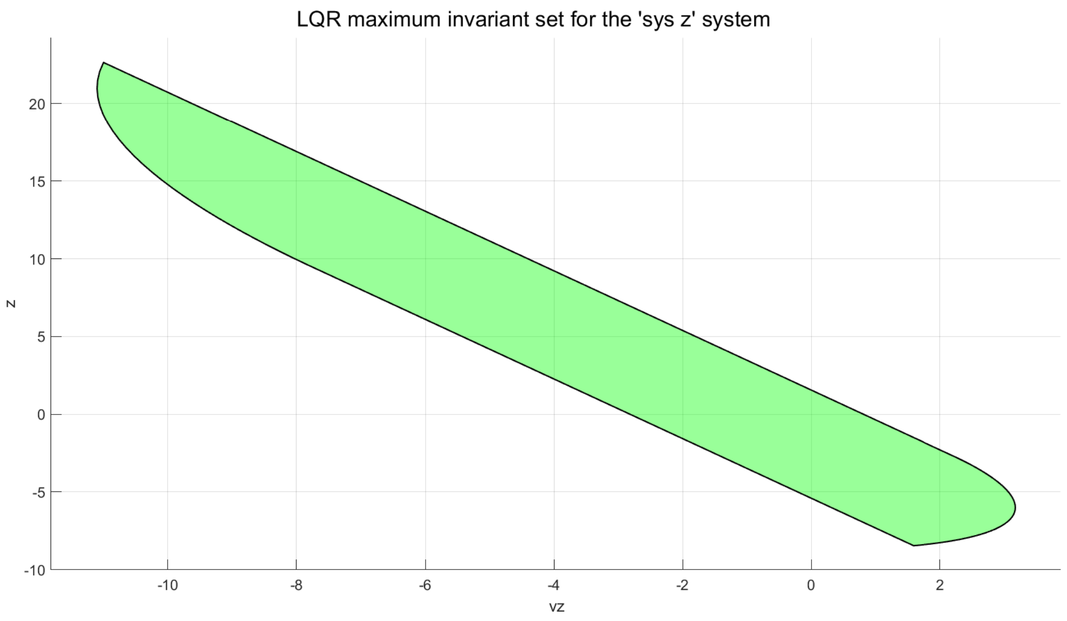


Figure 5: Terminal set $\mathcal{X}_f$ for the `Sys_z` system. The terminal set is the maximum control invariant set for the unconstrained LQR controller (i.e. maximum invariant set for the system $x^+ = (A + BK)x$).

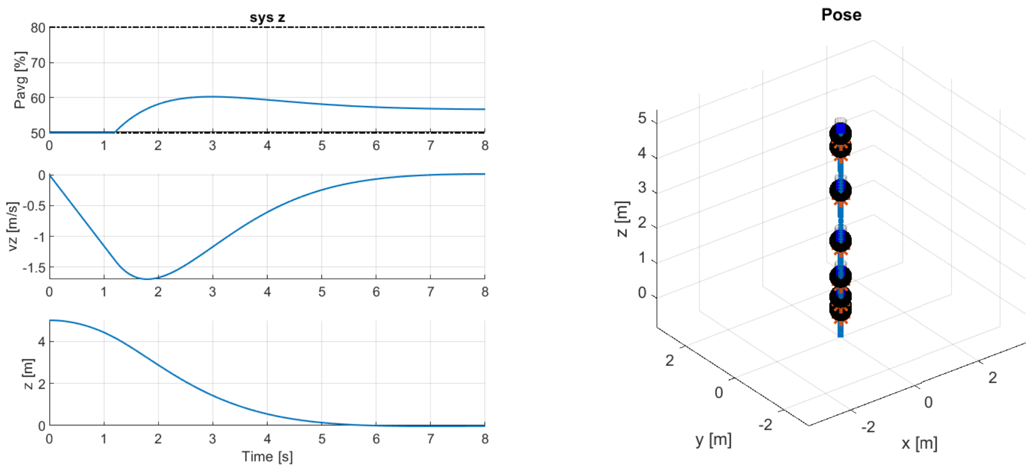


Figure 6: Controller's performances when starting stationary at five meters from the origin along z.

3.4 `Sys_roll` controller

For `Sys_roll`, as we did previously, we lowered the cost associated with P_{diff} to both let the controller use it and emphasize and scale the cost associated with γ and w_z (expressed in radians). The resulting cost matrices are given below:

$$Q_4 = I \quad R_4 = 0.01$$

Regarding the horizon, $H = 0.5s$ $[N = 10]$ was sufficient to achieve good performances (w.r.t. the specifications). The figure (7) shows the terminal set, and the figure (8) presents the final results obtained with the tuning parameters described above.

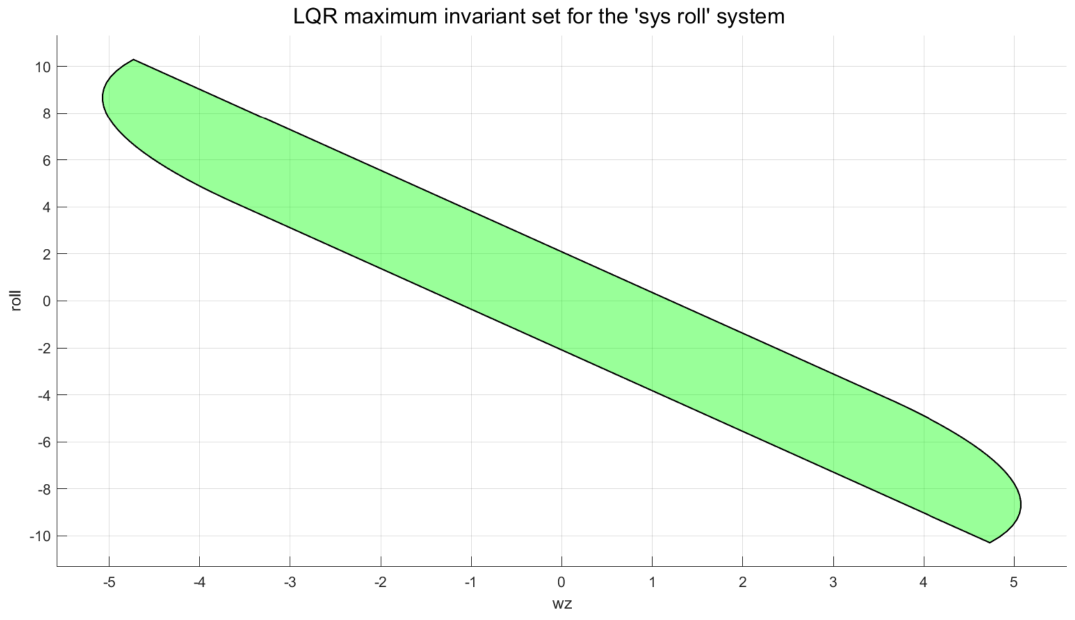


Figure 7: Terminal set $\mathcal{X}_f$ for the `Sys_roll` system. The terminal set is the maximum control invariant set for the unconstrained LQR controller (i.e. maximum invariant set for the system $x^+ = (A + BK)x$).

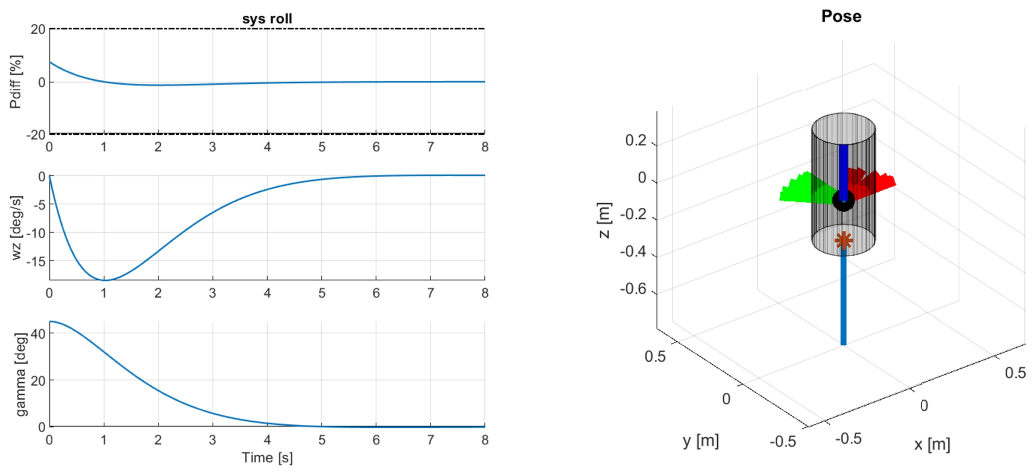


Figure 8: Controller's performances when starting stationary at 45° for roll.

4 Deliverable 3.2

Design procedure

The terminal set reduces the feasible set. A solution is to drop it, but it comes at a cost: the feasible set will not be invariant anymore. We can maintain the stability by taking a sufficiently large horizon length and by keeping the terminal cost.

Therefore, we decided to drop the terminal set and to check the stability by sampling. Our controller is indeed stable when tracking the reference of part 4 (5).

Steady state target

Here we want to track a reference r. In order to track a non-zero reference, we need to know towards which value x_s the state has to converge and towards which value u_s the input has to converge.
If x_s and u_s satisfy:

$$x_s = Ax_s + Bu_s$$
$$r = Cx_s + Du_s$$

We will have that $y_k \to r$ when $k \to \infty$.
We also want x_s and u_s to satisfy constraints, and we would like to minimize u_s.

Therefore, to find x_s and u_s we defined an optimizer as follows:

$$\min\ u_s^T R_s u_s$$

subject to:

$$\begin{pmatrix} I-A & -B \\ C & D \end{pmatrix} \begin{pmatrix} x_s \\ u_s \end{pmatrix} = \begin{pmatrix} 0 \\ r \end{pmatrix}$$
$$H_x x_s \leq k_x$$
$$H_x u_s \leq k_u$$

For our use case, the target problem was always feasible.

The optimization problem that needs to be solved at each iteration in order to find the optimal inputs is the same as in Deliverable 3.1 (3). Except that the cost function to minimize is:

$$(x - x_s)^T Q(x - x_s) + (u - u_s)^T R(u - u_s)$$

By minimizing this cost, the system will converge to r.

Soft constraints

For `Sys_x` and `Sys_y`, we imposed the state constraints:

$$\alpha \leq 5^\circ$$
$$\beta \leq 5^\circ$$

Indeed, our linearization, only works for small angles. If the angle of the rocket gets too big, our approximate model will not correspond to the reality anymore. However, this choice of 5° is quite arbitrary, and clearly the error of the model won't change much if the rocket has an angle of 5.1° instead of 5°. Therefore, as we thought that this state constraint could be temporary violated if necessary, we implemented a soft constraint. To do so, we need to introduce the slack constraint ϵ. The new state constraint can be written as follows:

$$H_x x_s \leq k_x + \epsilon$$

As we want to penalize the amount of constraint violation, we need to include ϵ in the cost function. We opted for a quadratic penalty as it is the simplest and it worked well for our use case. Therefore, we added this term in the cost function:

$$\epsilon^T S \epsilon$$

We opted for a big S so that the constraint is violated a big longer, but it is not violated by a lot (so our model doesn't diverge too much from reality).

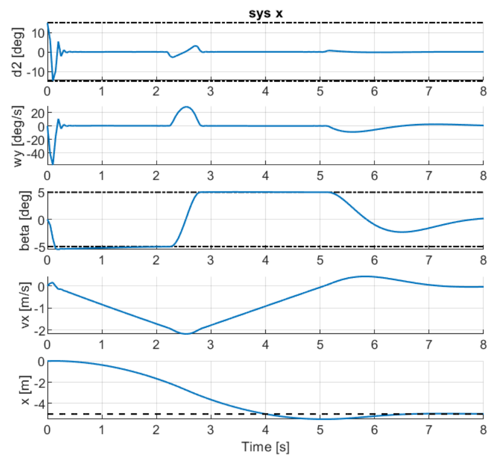
Procedure to tune the parameters

The tuning procedure is globally the same as in deliverable 3.1 (3). There is a difference with the horizon length though. As the tuning will be focused on having a nice behavior for task 4, we did not bother to find the best horizon length for each sub-system. Therefore, we choose a horizon length of $1.5s$ for all the sub-systems.

Note that the controllers were tuned to track the reference of task 4. This is why the tuning led to different values compared to task 3.1. We made sure to use our inputs sufficiently without having an unstable behavior. We also took care not to violate the soft constraint too much when tracking the letters "MPC". The procedure to tune S is specified above and we finally chose $S = 10^6$.

4.1 Sys_x controller

$$Q_2 = \begin{pmatrix} 100 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 80 \end{pmatrix} \qquad R_2 = 1$$



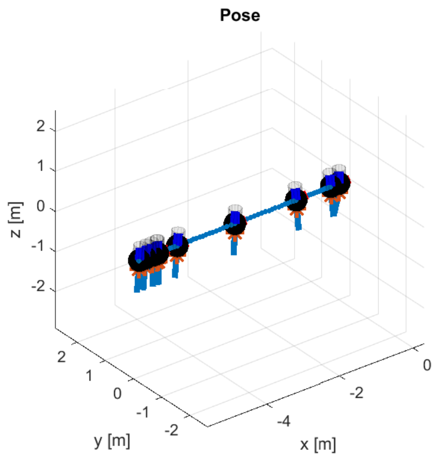


Figure 9: Controller's performances when starting stationary at origin and tracking $x = -5m$.

4.2 Sys_y controller

$$Q_1 = \begin{pmatrix} 100 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 80 \end{pmatrix} \qquad R_1 = 1$$

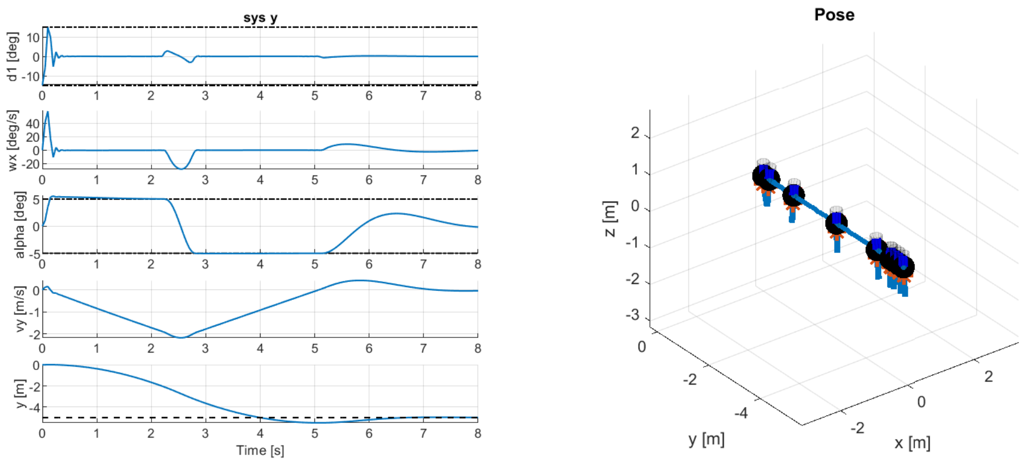


Figure 10: Controller's performances when starting stationary at origin and tracking $y = -5m$.

4.3 Sys_z controller

$$Q_3 = \begin{pmatrix} 1 & 0 \\ 0 & 170 \end{pmatrix} \qquad R_3 = 1$$

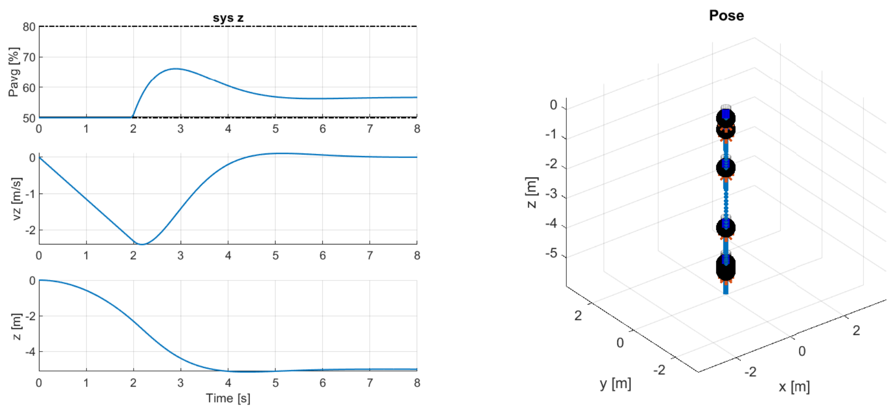
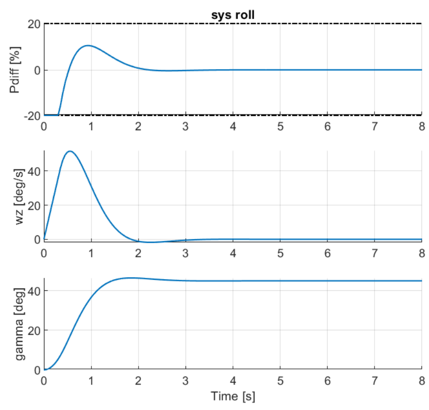


Figure 11: Controller's performances when starting stationary at origin and tracking $z = -5m$.

4.4 Sys_roll controller

$$Q_4 = \begin{pmatrix} 100 & 0 \\ 0 & 5000 \end{pmatrix} \qquad R_4 = 1$$



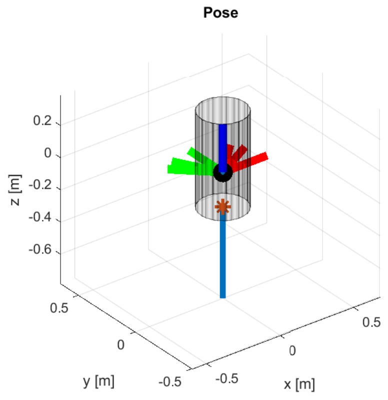


Figure 12: Controller's performances when starting stationary at an angle of 0 and tracking $gamma = 45°$.

5 Deliverable 4.1

This deliverable consists of merging the 4 sub-controllers developed in the previous task. As those sub-controllers were specifically tuned to track the letters "MPC", there is not much to add.

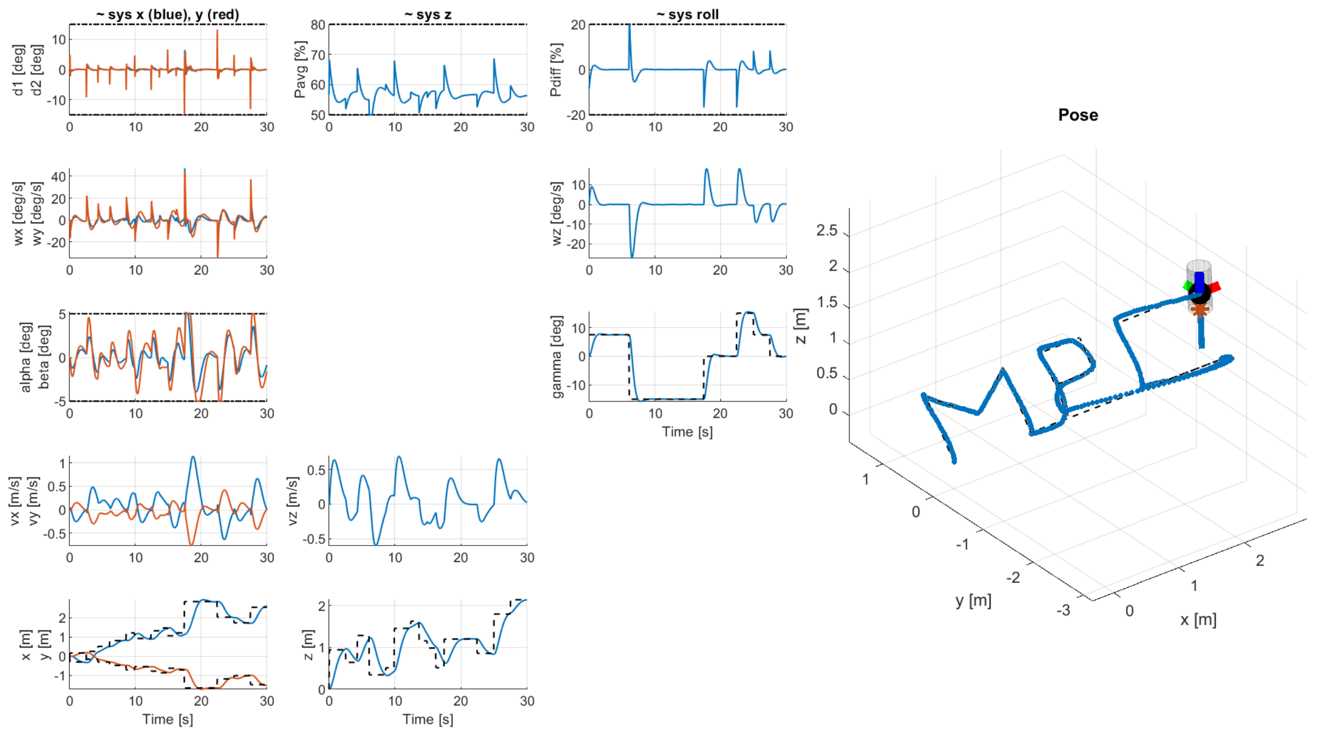


Figure 13: Controller's performances when starting tracking *MPC*

.

Note that we could have made `Sys_z` a bit more aggressive. But then the drawing would have appeared less nice because diagonal lines would not have been straight anymore (z would have been corrected too fast compared to x and y).

6 Deliverable 5.1

In this section, the goal is to design an offset-free tracking controller for the z–axis, to compensate a change in the rocket mass. In order to reject a constant disturbance on the mass of the rocket we now consider the dynamic of the system in the z – direction as:

$$x^+ = Ax + Bu + Bd \tag{1}$$

$$y = Cx$$

Where the disturbance d only affects the state system, and not the observable y. We build an estimator in order to estimate the disturbance $\hat{d}$ and the state of the system $\hat{x}$:

$$\begin{pmatrix} \hat{x}_{k+1} \\ \hat{d}_{k+1} \end{pmatrix} = \underbrace{\begin{pmatrix} A & B \\ 0 & I \end{pmatrix}}_{\bar{A}} \begin{pmatrix} \hat{x}_k \\ \hat{d}_k \end{pmatrix} + \underbrace{\begin{pmatrix} B \\ 0 \end{pmatrix}}_{\bar{B}} u_k + \underbrace{\begin{pmatrix} L_x \\ L_d \end{pmatrix}}_{L} (\underbrace{\begin{pmatrix} C & 0 \end{pmatrix}}_{\bar{C}} \begin{pmatrix} \hat{x}_k \\ \hat{d}_k \end{pmatrix} - y_k)$$

This can be re-written in a more compact form:

$$\begin{pmatrix} \hat{x}_{k+1} \\ \hat{d}_{k+1} \end{pmatrix} = \bar{A} \begin{pmatrix} \hat{x}_k \\ \hat{d}_k \end{pmatrix} + \bar{B}u_k + L(\bar{C} \begin{pmatrix} \hat{x}_k \\ \hat{d}_k \end{pmatrix} - y_k)$$

L is chosen such that the error dynamics model:

$$\begin{pmatrix} x_{k+1} - \hat{x}_{k+1} \\ d_{k+1} - \hat{d}_{k+1} \end{pmatrix} = (\bar{A} + L\bar{C}) \begin{pmatrix} x_k - \hat{x}_k \\ d_k - \hat{d}_k \end{pmatrix}$$

Poles selection

The poles of $\bar{A} + L\bar{C}$ need to be carefully defined such that we rapidly and smoothly converge to the true values. The poles must be placed into the unit circle such that the disturbance estimation as well as the state estimation converge to the true values. One could choose the poles as small as possible to ensure a faster convergence but it could lead to an overshoot of the estimation as well as oscillatory phenomenons. To choose the poles and avoid such behaviors, we tested different configurations depicted on figure 14. It shows the evolution of the estimated disturbance for a new mass of 1.783 kg and constant reference/target set to 1m (starting from the origin). As one can notice, choosing poles close to 0 leads to a high overshoot and introduces oscillatory phenomenons. On the contrary, if the poles are close to 1, the estimation does not overshoot but the convergence time is higher. The configuration **(0.4,0.5,0.6)** was a good trade-off between overshoot and settle time/convergence time.

Results

Both the controller and the steady state computation need to take into account the new dynamic system (1), satisfying now $x_s = Ax_s + Bu_s + B\hat{d}$ for the steady state.

When introducing a constant disturbance on the rocket mass, by increasing the mass (1.783 Kg instead of 1.7 Kg), the controller from Deliverable 4 does not perform well. As we can see from Figure 15, the tracking of the rocket has a negative offset on the z direction due to the mass increase. After adding an observer, the rocket is able to correctly track the path (Figure 16) without any offset. The performance is as good as the one in Deliverable 4, and no further tuning is necessary. In addition you can also visualize, on figure 17, the evolution of the estimation of the disturbance during the simulation.

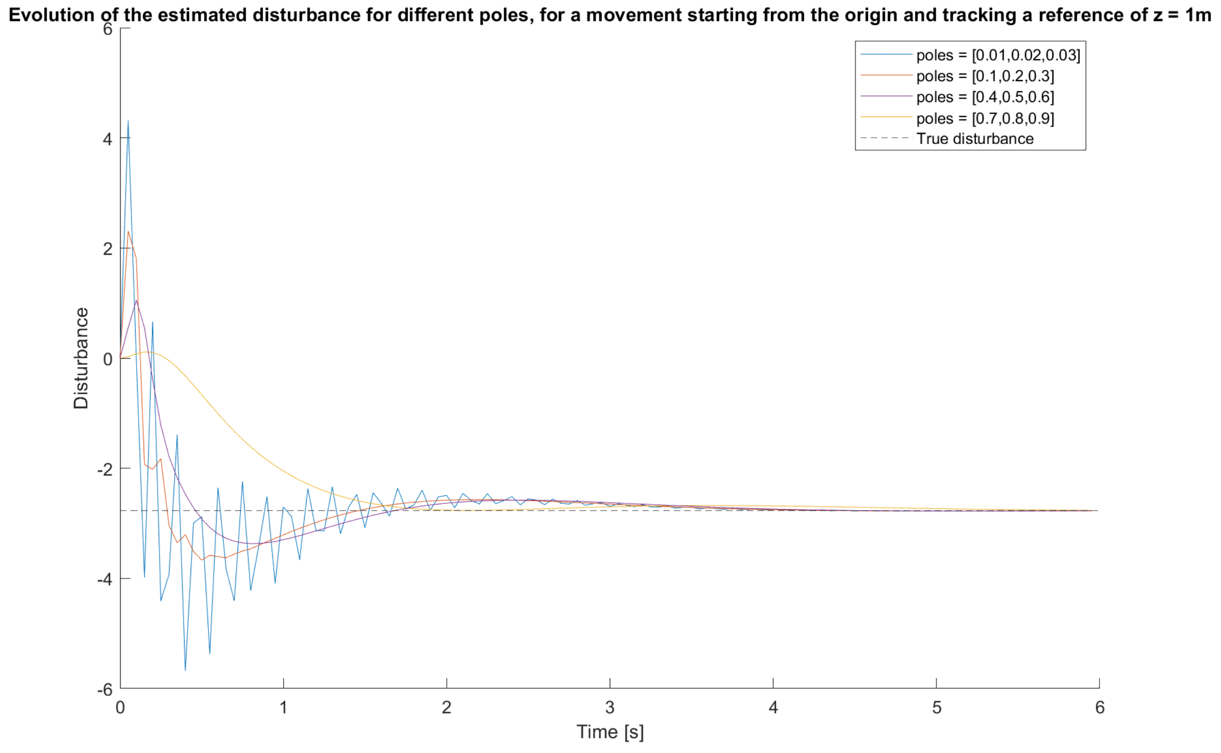


Figure 14: Evolution of the disturbance estimation for different pole values, mass-rocket = 1.783 kg, and z reference = 1 m (starting at origin).

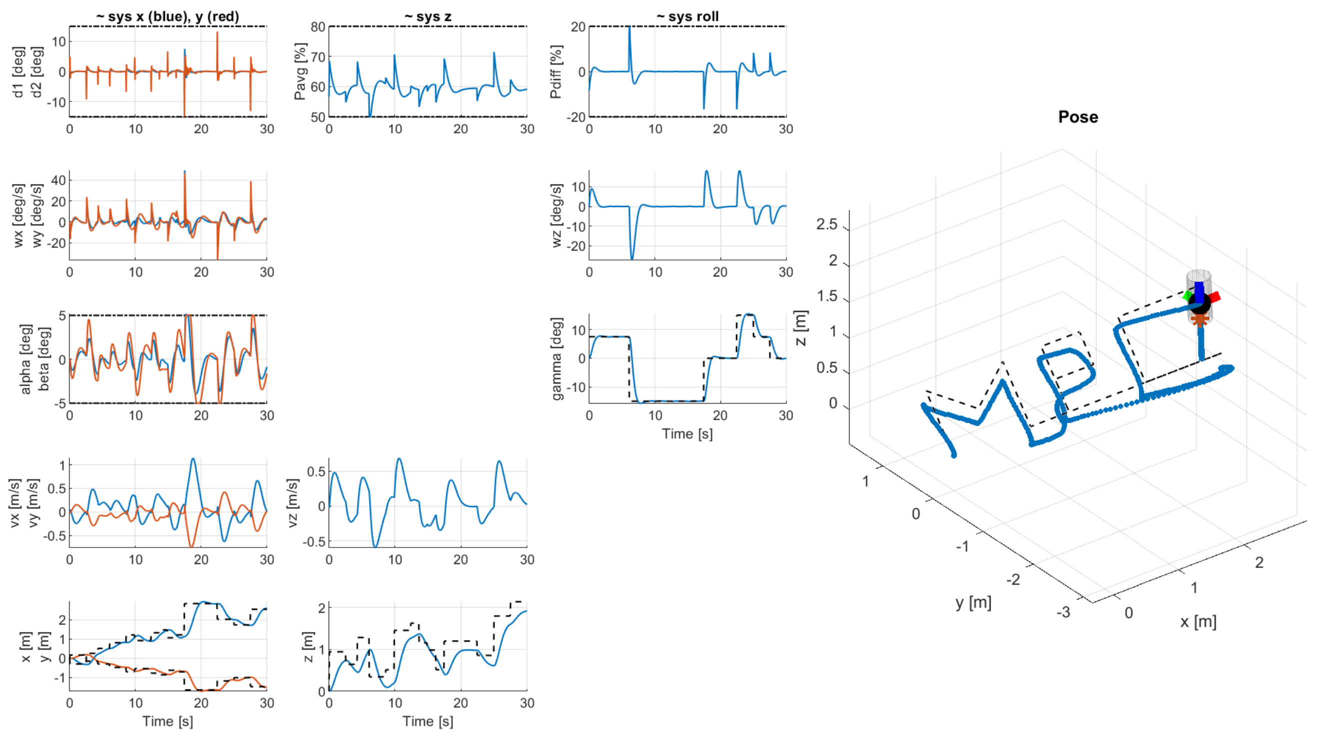


Figure 15: Impact of changing the mass on the controller from Part 4: mass-rocket = 1.783 Kg

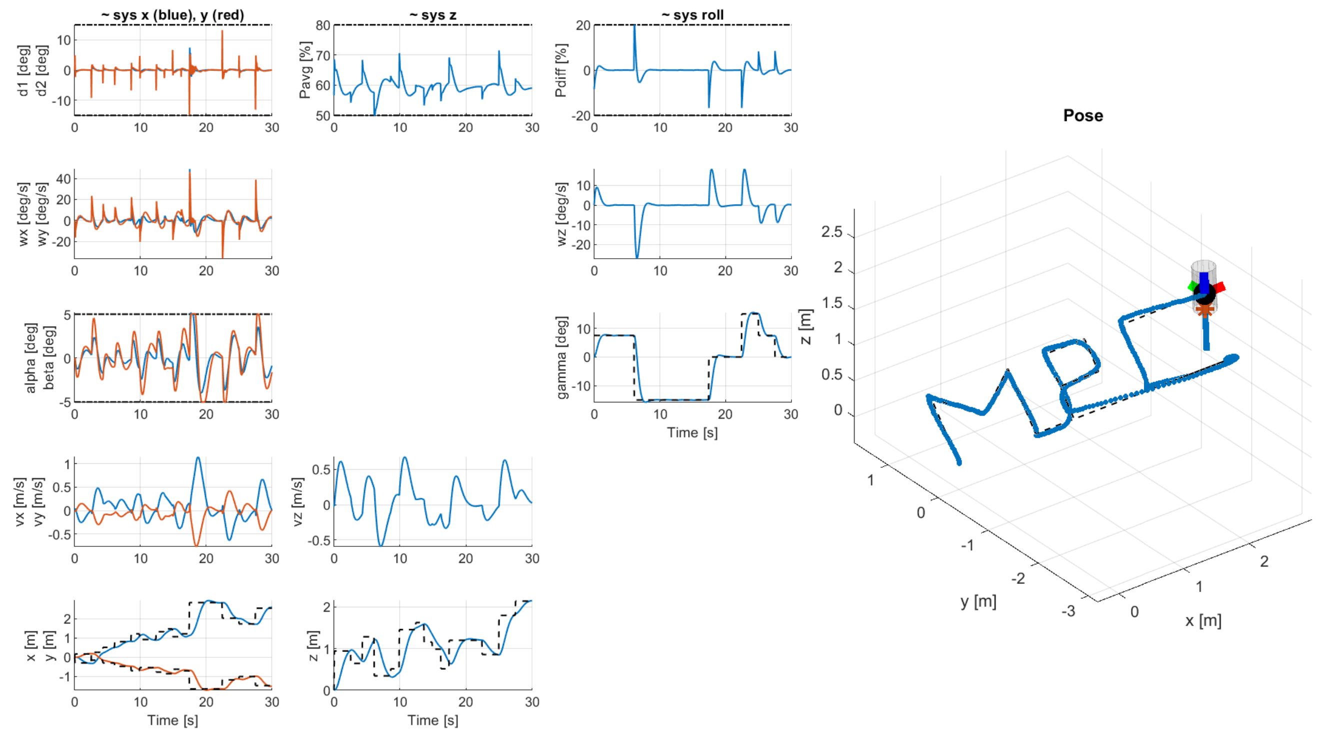


Figure 16: Offset-free tracking of the updated z–controller: mass-rocket = 1.783 Kg

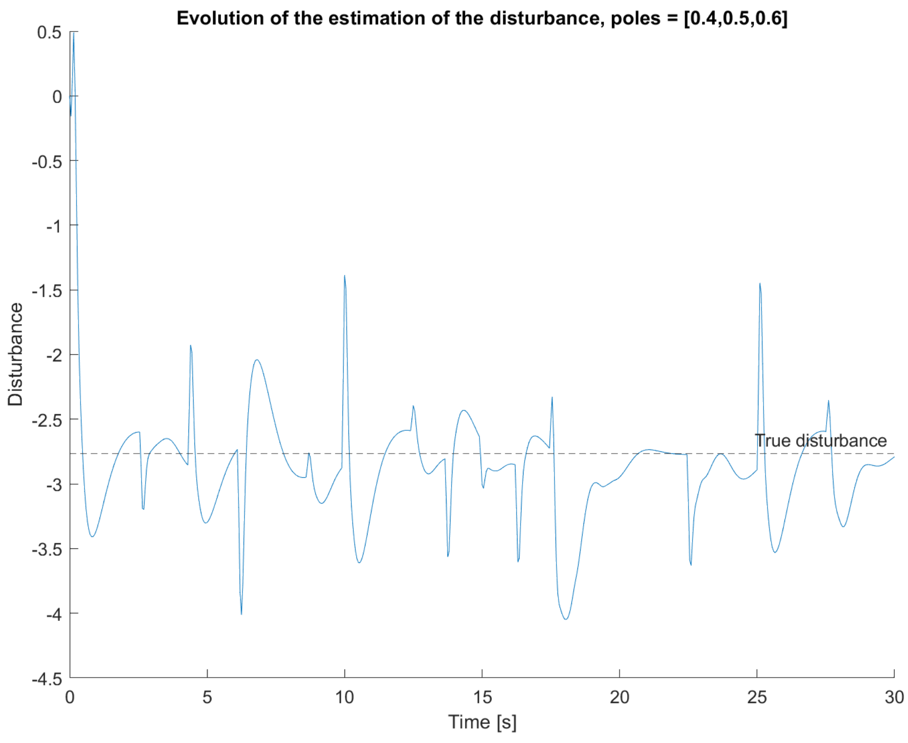


Figure 17: Evolution of disturbance estimation for mass-rocket = 1.783 Kg and when tracking the letters "MPC".

7 Deliverable 6.1

Design procedure and fine tuning

Regarding the design procedure, we added a terminal cost computed based on the linearized model around the equilibrium point considered previously. This terminal cost allows us to keep the horizon length quite small while keeping the optimization problem feasible. The equations used for the terminal cost are detailed below. First we linearize the function f around an equilibrium point using the first order taylor's approximation.

$$\dot{x} = f(x,u) = \underbrace{f(x_s,u_s)}_{0} + J_{f/x}(x_s,u_s)(x-x_s) + J_{f/u}(x_s,u_s)(u-u_s)$$

Where $J_{f/x}$ and $J_{f/u}$ are the jacobian of f w.r.t. x and u respectively. Assuming that the sampling rate h is small emough then:

$$x_{k+1} \approx x_k + h\dot{x}_k \approx x_k + hJ_{f/x}(x_k - x_s) + hJ_{f/u}(u_k - u_s)$$

Subtracting x_s from both side of the equation leads to:

$$x_{k+1} - x_s \approx \underbrace{(I + hJ_{f/x})}_{A}(x_k - x_s) + \underbrace{hJ_{f/u}}_{B}(u_k - u_s)$$

Then to predict the next state based on the differential equation $\dot{x} = f(x,u)$, we used the RK4 method. Regarding the fine tuning procedure, we started with the configuration we had with the linearized system in 5 and then we slightly modified them to get better performances. The final fine-tuning parameters for a maximum roll angle of 15 degrees are listed below:

$$Q = \text{diag}(600, 1100, 100, 1, 1, 1000, 1, 1, 1, 1000, 1000, 1000) \quad R = \text{diag}(1, 1, 5, 1)$$

The performances of the non-linear controller are shown in 18 and 20 with a maximum roll angle of respectively 15° and 50°. With a maximum roll angle of 15°, we can notice that the performances are slightly better than the ones provided by the linear controller but the difference is not huge. On the contrary, with a maximum roll angle of 50°, the performances of the linear controller shown in 21 drop significantly while the non-linear controller still perform a good job. Bad performances of the linear controller with a roll of 50° can be explained by the fact that, with such a high roll angle, the linearized model is no longer a good approximation of the dynamic of the rocket as we are far from the point where the linearization was performed.

In addition, we also tried to introduce other non-linear functions in the cost function to improve performances for x, y, z and γ tracking when the error is low. To do so, we added a L1 error (absolute value error) to emphasize small errors[2], i.e. below 1. The new equation of the stage cost function is given below:

$$x^T Q x + u^T R u + \rho\big(\mid x - x_{target} \mid + \mid y - y_{target} \mid + \mid z - z_{target} \mid + \mid \gamma - \gamma_{target} \mid\big)$$

With $\rho = 0.01$. The results with this cost function are shown in 19.

Pros and cons

Using a non-linear controller exhibits some advantages and drawbacks compared to linear controller:

- One of the main drawback of the nonlinear version is the computational time that is far higher than the one in the linear version. Then this high computational time limits the horizon length we can choose.
- In the non-linear version, determining a terminal set is not as easy as it in the linear version. Hence we cannot ensure that the feasible set will be invariant, and we needed to determine it empirically.
- At equal horizon length, the non-linear version gives better results than the linear one. As expected and shown in 21 and 20, non-linear controller gives far better performances when we are far from the point where the linearization was performed.
- Contrary to the linear controller, no constraints were put on α and β. Then the controller was allowed to go beyond 5 degrees, which leads to better performances
- Finally the non-linear version allowed us to use any other kind of non-linear functions in the cost function, that gives more freedom in the fine-tuning procedure.

[2]Indeed when the error is below 1, the quadratic cost reduces the error while absolute cost keeps it unchanged.

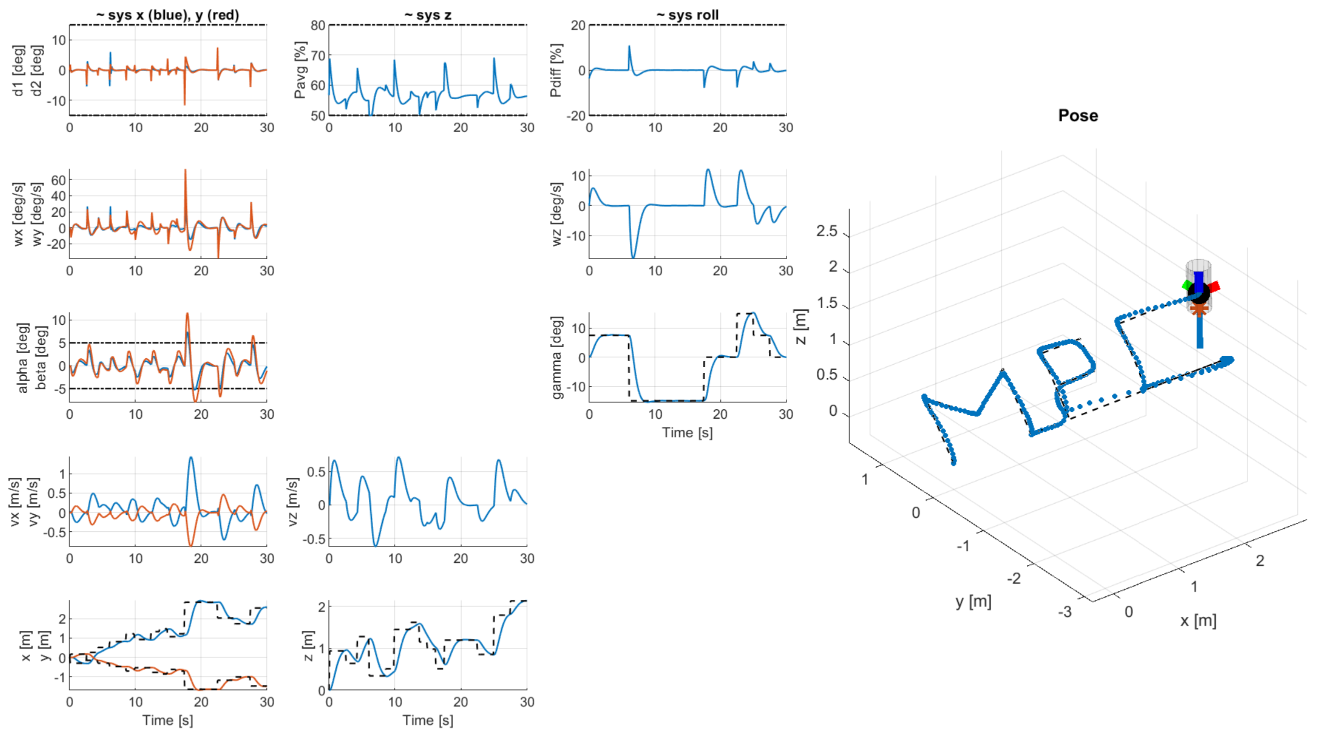


Figure 18: Performances of the non-linear controller with a maximum roll angle set to 15°.

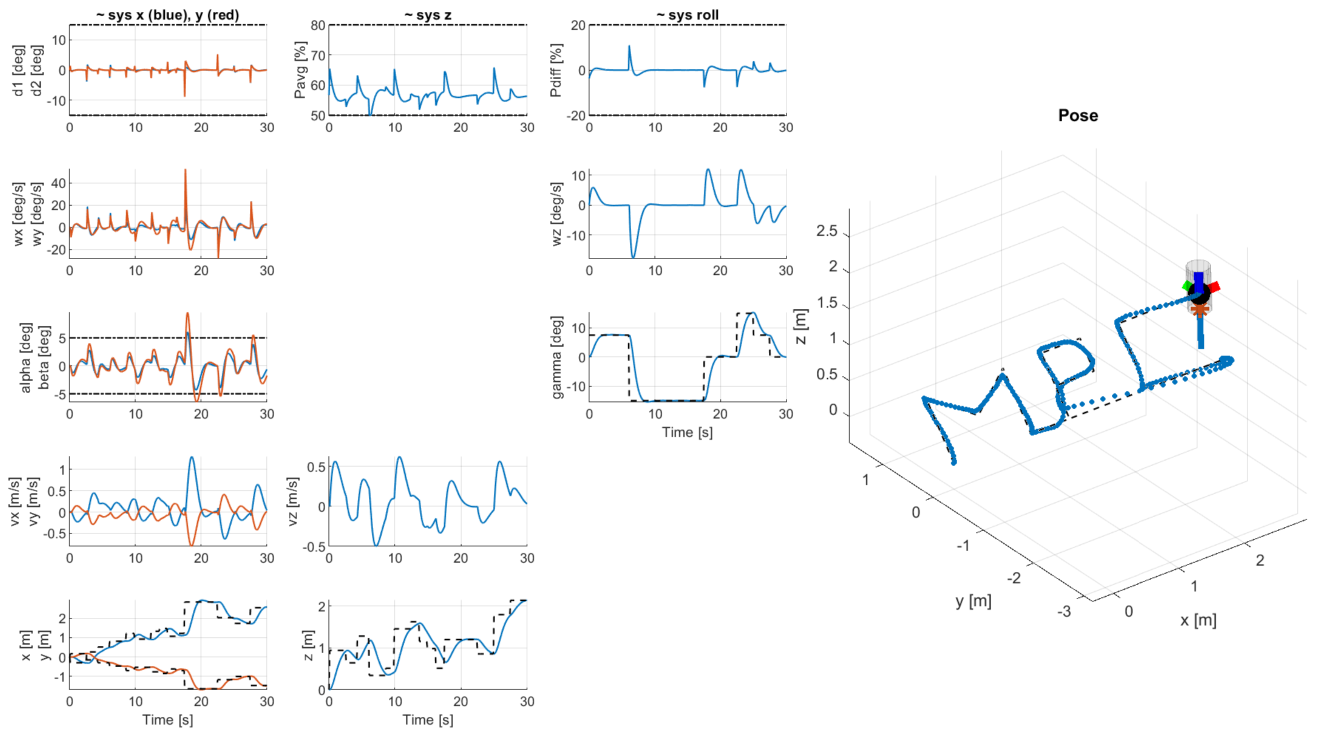


Figure 19: Performances of the non-linear controller with a maximum roll angle set to 15° and with absolute error cost function added to the initial cost function.

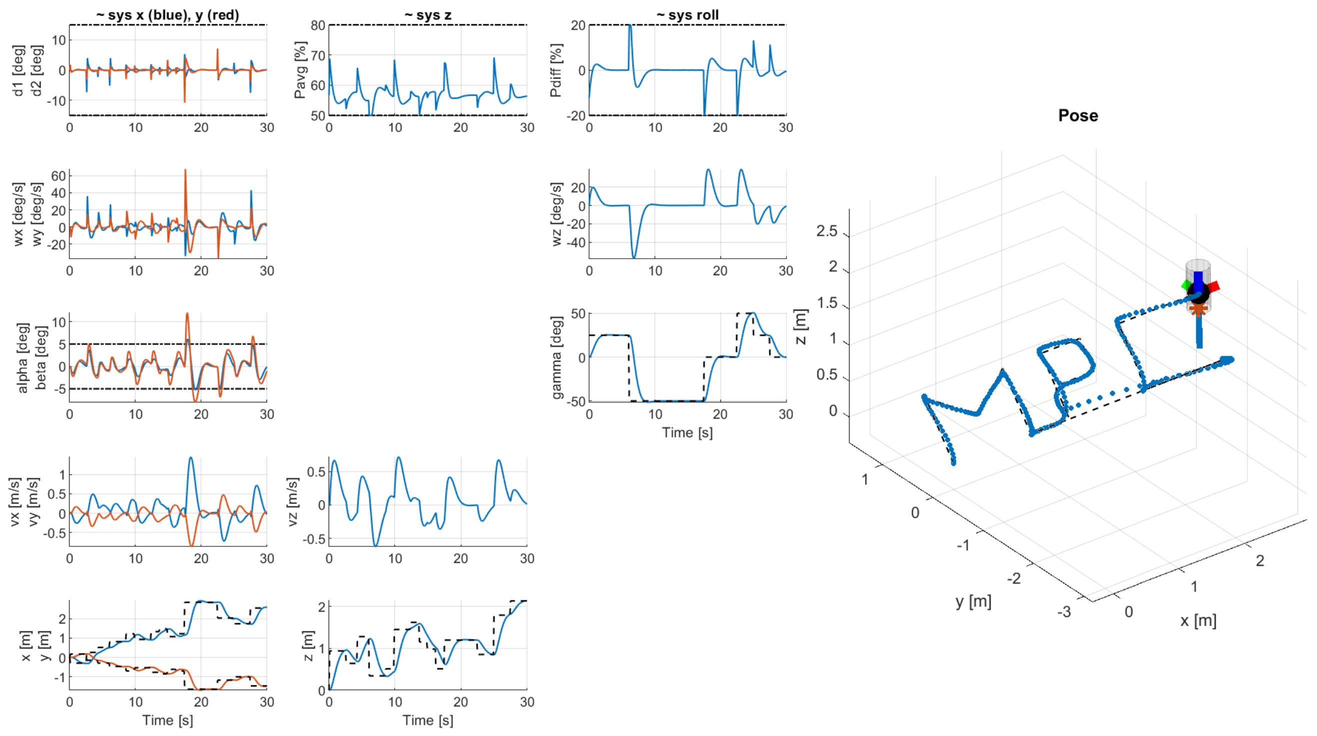


Figure 20: Performances of the non-linear controller with a maximum roll angle set to 50°.

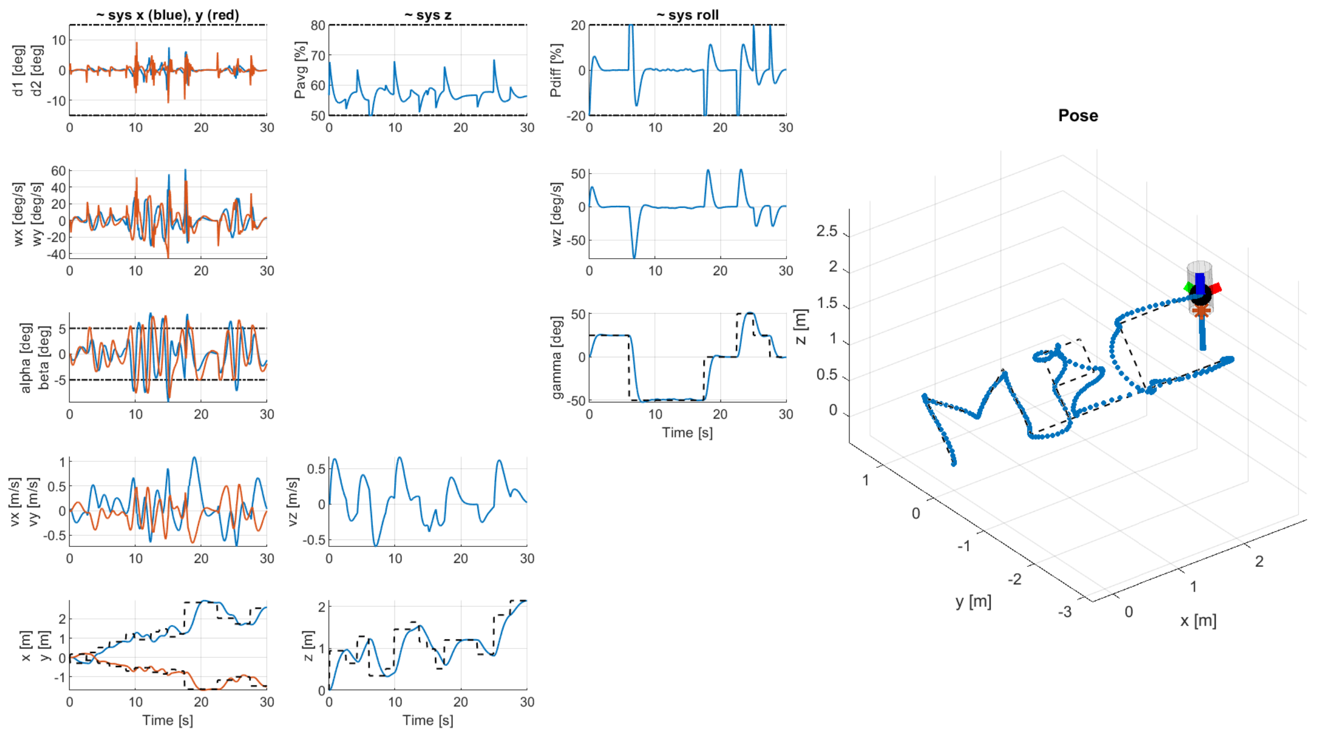


Figure 21: Performances of the linear controller (from deliverable 4.1) with a maximum roll angle set to 50°. As one can notice that the performance is far worse than the nonlinear controller's one.

8 Conclusion

It was interesting to be able to compare a linear and non-linear MPC approach for this model. We had the chance to experience the pros and cons of both methods, already mentioned above 7.
Thanks to this project, we now have a better grasp on the concepts and limitations of MPC.