Report MPC - Rocket control

Agatha Duranceau (271165), Francesco Nonis (300540), Vincent Philippoz (300559)

December 2021

Figure 1: Illustration of a rocket. Source: *Shutterstock*

1 Introduction

The aim of this project is to control and simulate the motion of a rocket following a desired path using MPC. In the first part, the system will be linearized around an equilibrium position, which will allow us to break it into four independent subsystems. In the second part, we will consider the whole non-linear system, and compare the performance of the two methods for following a non-linear path.

2 Deliverables

2.1 Deliverable 2.1

The input δ_1 (deflection angle of the servomotor that inclines the propeller around axis x) has an impact on the orientation angle β in the y-z plane, and thus impacts the y coordinate and speed v_y, as well as the angular speed ω_x. Similarly, δ_2 impacts x, v_x, α the orientation in the x-z plane, and the angular speed ω_y. The average power P_{avg} only has an impact on the z position and speed v_z, and finally, P_{diff} defines how the rocket will turn on itself with the angular velocity ω_z. Therefore, it is possible to break the linearized system into four independent sub-systems corresponding to the four inputs δ_1, δ_2, P_{avg} and P_{diff}.

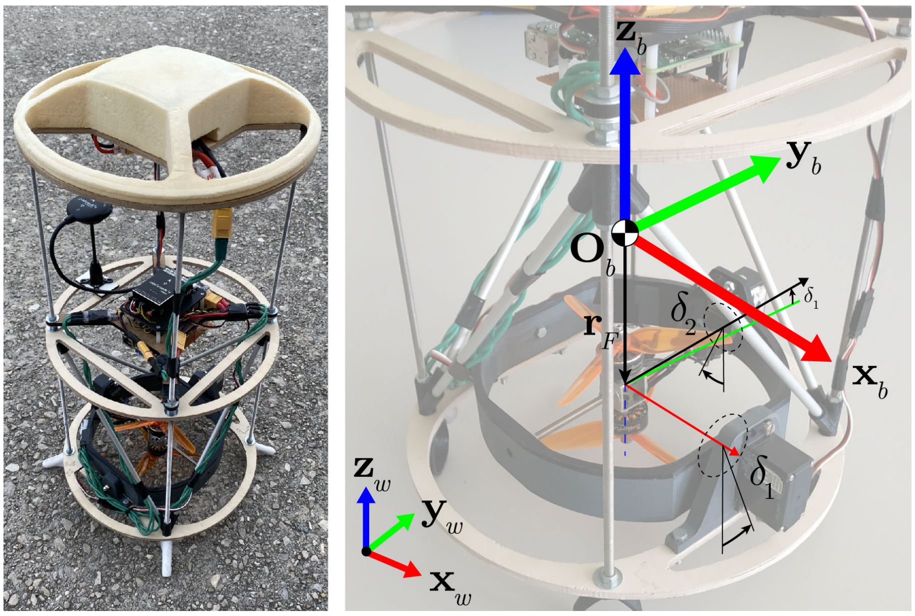


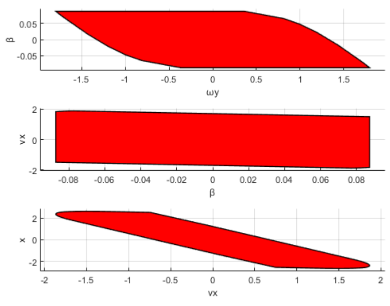
Figure 2: Rocket and visualization of its coordinates, taken from the project brief

2.2 Deliverable 3.1

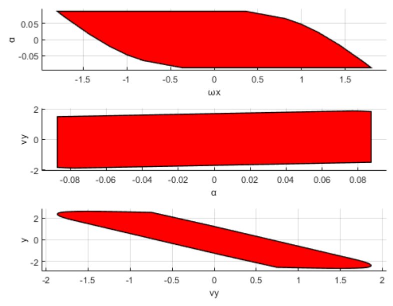
We want to make sure that the system $X^+ = A \cdot X + B \cdot U$ will converge to the origin whilst remaining within desired state and input constraints at all time. The optimal controller that minimizes the cost of the trajectory while respecting these conditions is returned as a YALMIP optimizer object.

In order to design such a controller, one must define a cost function to minimize, as well as constraints to respect for all next steps taken as inputs of the optimizer function. The constraints are defined by the physical limits of the system as given in the project brief, and the cost function is tuned with the parameters Q and R according to what one wants to penalize for.

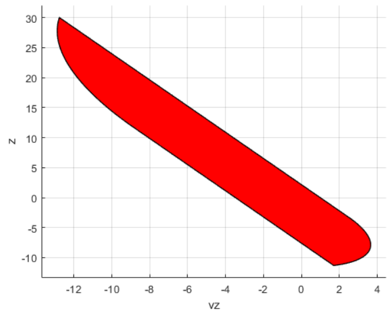
However, to make sure that a feasible solution exists, one must find a region of attraction for which it is guaranteed that the system will converge to the origin, i.e. the terminal set. It is computed as the maximum control invariant set for which there will always exist an input ensuring that the next step will remain inside the invariant set and where an LQR controller can be applied to reach the origin. The cost within this set, i.e. the terminal cost, is thus defined by LQR controller parameter Q_f.The terminal sets for the four controllers can be seen in figure 3.



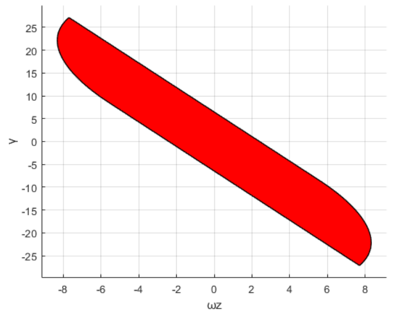
(a) Projections of the terminal set for state x



(b) Projections of the terminal set for state y



(c) Terminal set for state z



(d) Terminal set for state γ (roll)

Figure 3: Terminal sets for the four controllers

We want to force the system to enter this terminal set within the horizon N (defined by H). Thus, the horizon length, H, must be large enough to make the problem feasible. In our case for instance, 1s was too short but 5s gave a satisfactory result for the four controllers.

Before reaching the terminal region, the stage cost is given by $X' \cdot Q \cdot X + U' \cdot R \cdot U$. The matrix Q penalizes for how far the rocket states are from the origin. Thus, by increasing its value, the system will converge faster. The identity matrix seemed sufficient for x and y. However, for the controllers on the z position and on the rolling-around of the rocket, it took too much time, so we multiplied Q by 10 in order to reach a satisfactory convergence (under 8 seconds).

R penalises the input parameters. Increasing it is useful when one wants to minimize energy consumption for instance, which is not the case here. Thus we kept $R = 1$ for the four controllers.

The resulting trajectories simulated for the four independent systems can be seen in figures 4 to 7.

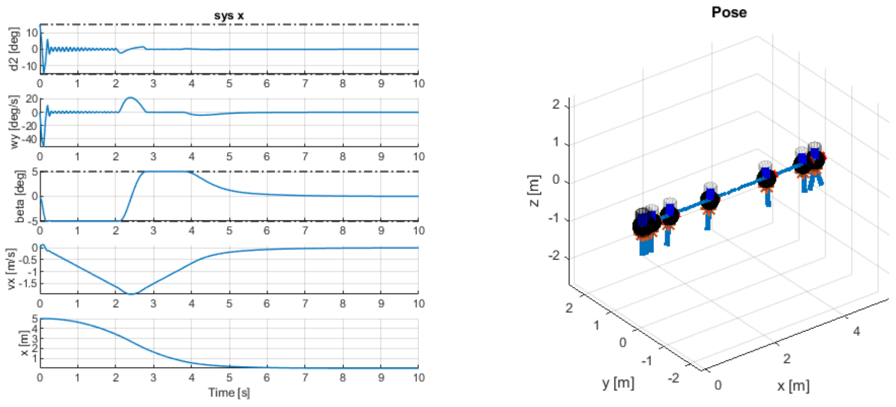


Figure 4: Trajectory along axis x

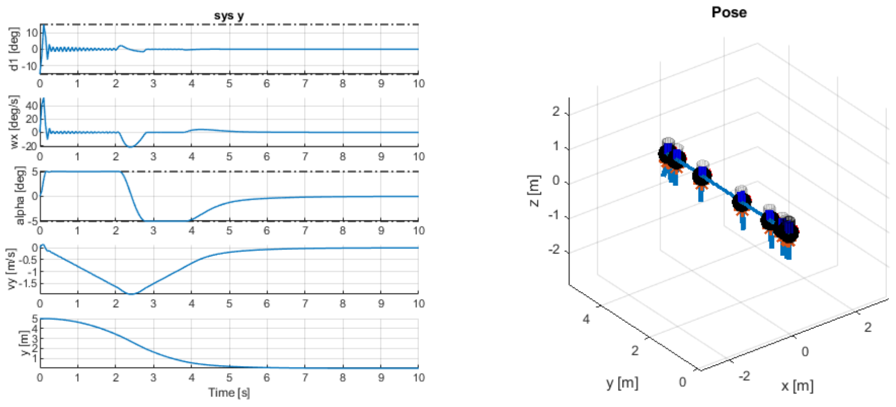


Figure 5: Trajectory along axis y

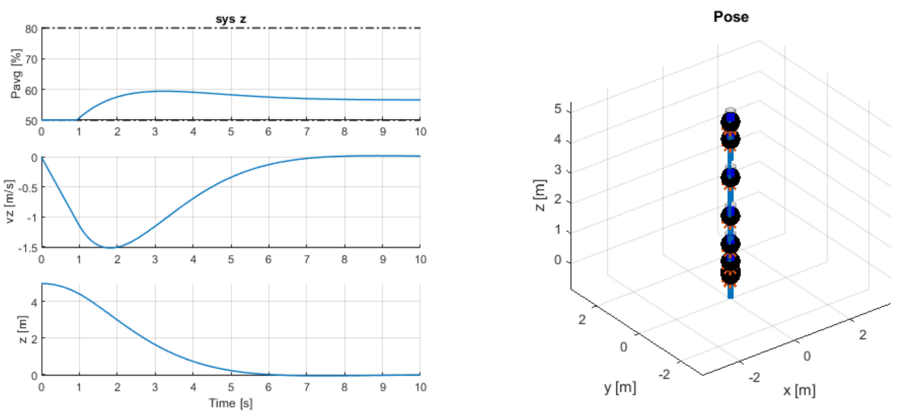


Figure 6: Trajectory along axis z

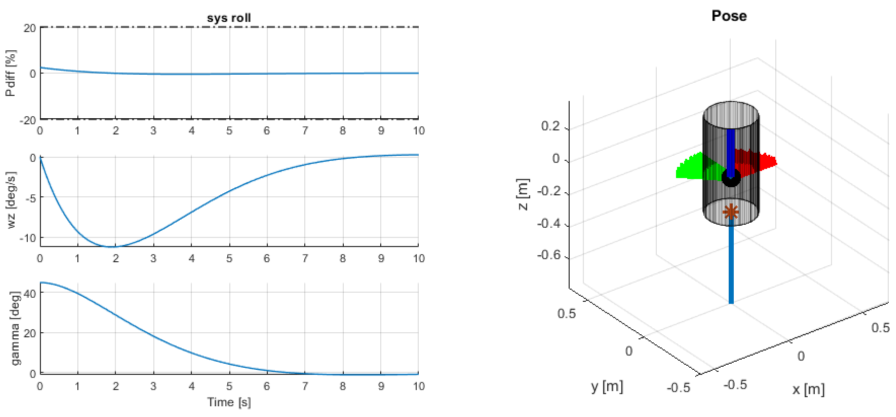


Figure 7: Rotation γ (roll)

2.3 Deliverable 3.2

From now on, we want the rocket to converge to a chosen non-zero reference, in this case a position. To do so, we must compute a feasible steady state target x_s and u_s, corresponding to the desired target reference (i.e. $C \cdot x_s = ref$). By definition, a steady state is such that the system converges to it, or in other words, that the next position will be the same as the previous one (i.e. $x_s = A \cdot x_s + B \cdot u_s$). To control the system, we simply need to use the same controllers as in the previous part but replacing X by its shifted value $X - x_{ref}$ and U by $U - u_{ref}$.

Almost all the controllers performed as well as in the previous part, except controller_roll for which we had to increase the value of Q so that the system converges to the desired angle fast enough, and without any overshoot.

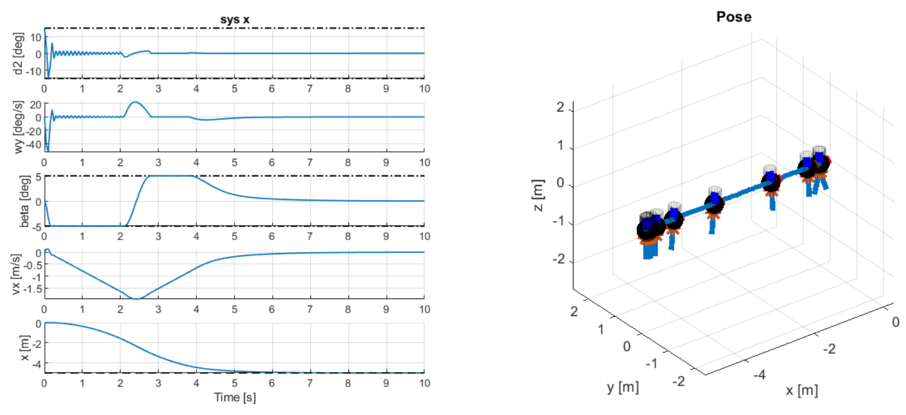


Figure 8: Trajectory along axis x

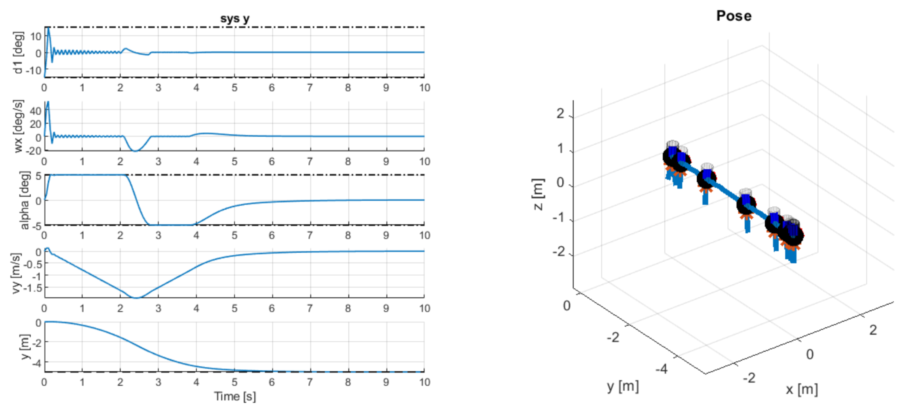


Figure 9: Trajectory along axis y

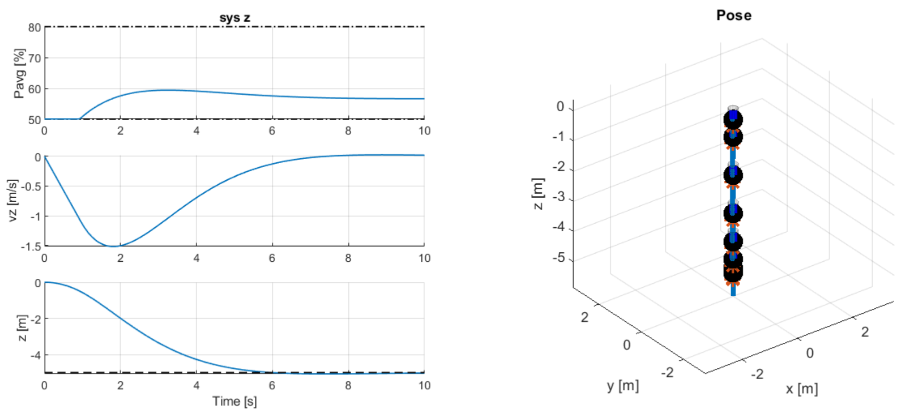


Figure 10: Trajectory along axis z

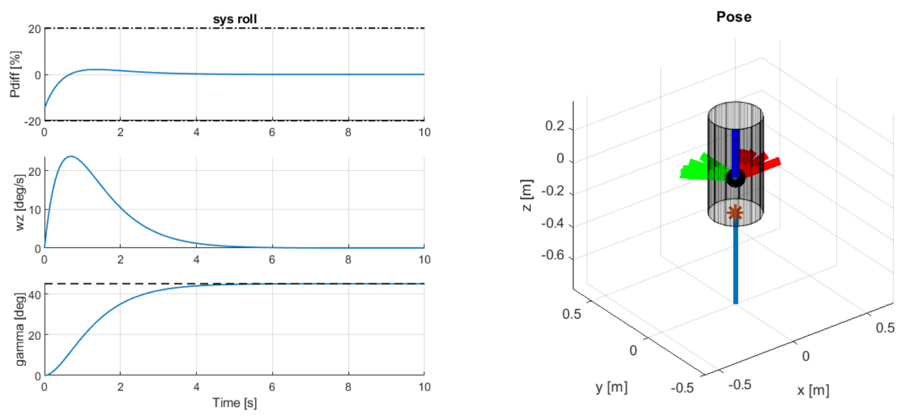


Figure 11: Rotation γ (roll)

2.4 Deliverable 4.1

In this section, the four controllers have been combined so that the rocket can follow a complete non-linear trajectory. The initial performance was quite poor (see figure 12), so we have increased the values of the Q matrices for the four controllers, especially for the variables z and $roll$, until the trajectory was close enough to the desired path and when the performance did not seem to increase significantly any more (see figure 13).

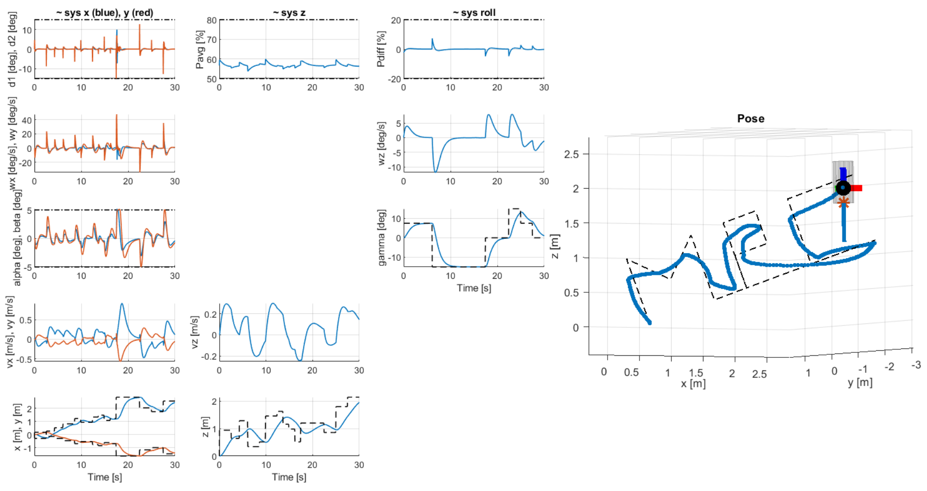


Figure 12: Rocket trajectory with initial parameters

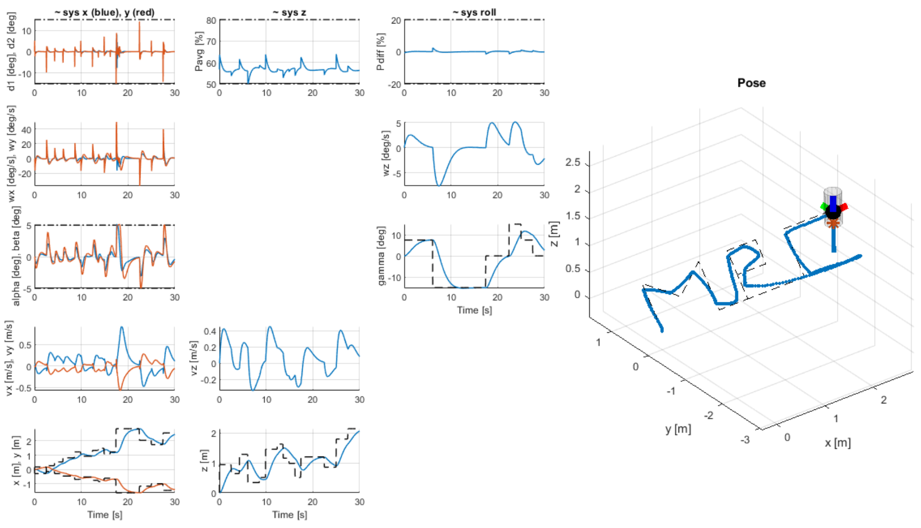


Figure 13: Rocket trajectory after tuning of parameters

Let us note that increasing the parameter T_f (time in which the system must reach the end of the full reference path), also allows for a better trajectory of the rocket.

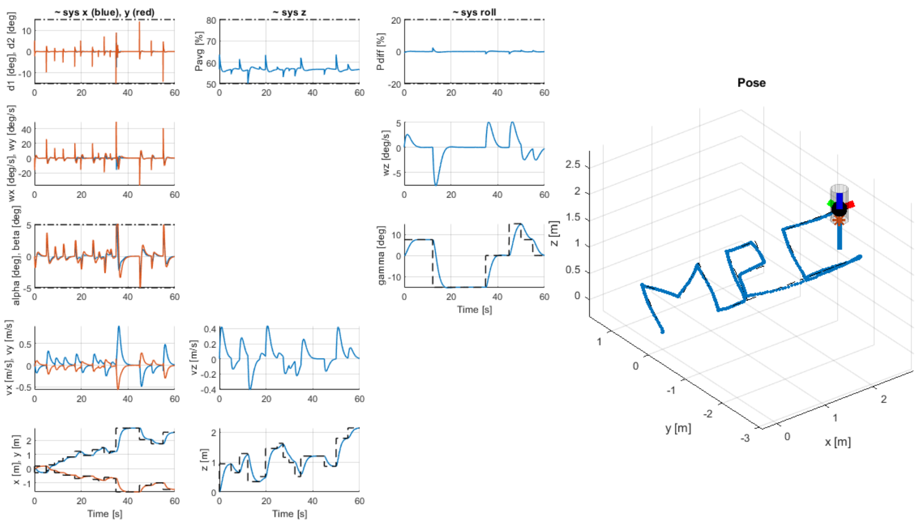


Figure 14: Rocket trajectory with longer time to reach end of path (60s)

2.5 Deliverable 5.1

In reality, the mass of a rocket varies as fuel is consumed during flight. Thus the equilibrium value of the input P_{diff} that stabilizes the z position will also vary. Keeping the previously designed controller with a different mass would therefore make the tuning of the parameters, the physical model used until now and the computation of the target states (for tracking) obsolete and lead to an incorrect trajectory, as it can be seen in figure 15.

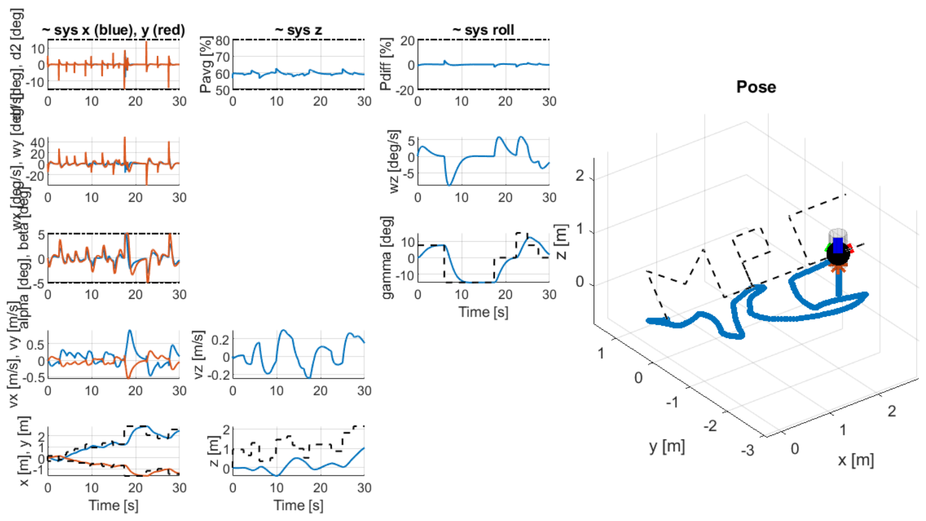


Figure 15: Rocket trajectory after change in mass without any modification to the controller

In order to compensate for this change of mass and avoid tuning the parameters every time, one can build an estimator to estimate the impact of the mass variation modelled as an input disturbance d on the subsystem z.

To do so, one must define an augmented model $\overline{x_{next}} = \bar{A}\cdot\bar{x}+\bar{B}\cdot u+L\cdot(\bar{C}\cdot\bar{x}-y)$, with $\bar{A}$, $\bar{B}$ and $\bar{C}$ the augmented matrices that depend on the system dynamics A, B and C. $\bar{x}$ includes the state x as well as the disturbance d. L is the observer matrix that is used to observe the difference between the last predicted output from the augmented model and the measured output of the system (i.e. error we want to minimize) and it can be designed using pole placement, ensuring that the error dynamics converge to zero (i.e. poles < 1).

This allows us to get an estimation of the disturbance d_{est} that will be added to the inputs of the system for the computation of the reference target. The state constraints will therefore become $x_s = A \cdot x_s + B \cdot u_s + B \cdot d_{est}$ instead of $x_s = A \cdot x_s + B \cdot u_s$. The final result can be seen in figure 16.

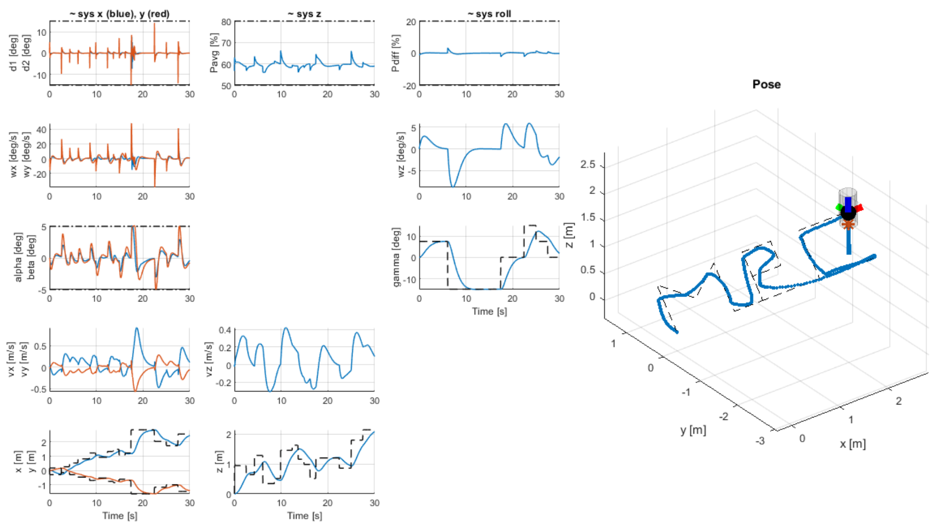


Figure 16: Rocket trajectory after change in mass and implementation of tracking for constant disturbances (Offset-free tracking)

2.6 Deliverable 6.1

The complete rocket system contains twelve state variables X = (w_x,w_y,w_z,α, β, γ, v_x, v_y, v_z, x, y, z). It is is submitted to the continuous dynamics function $f(X, U)$, with state constraints $F \cdot X \leq f$, input constraints $M \cdot U \leq m$ and initial state X^0.

There are four inputs U = (δ_1,δ_2,P_{avg},P_{diff}) to control the motion of the system and these have a direct impact on the variables (x, y, z, γ,). Thus the reference to track will include four parameters (ref_x, ref_y, ref_z, ref_γ) and we assume that the targets for the other variables will be zero since we want do not want the rocket to move once it has reached it's target position (speeds are zero) and we want the rocket to be vertical ($\alpha = 0$, $\beta = 0$). Therefore, the target state X_{target} will be equal to (0,0,0,0, 0, ref_γ, 0, 0, 0, ref_x, ref_y, ref_z) .

For every state variable X_i, and at every iteration, the cost to minimize will be $(X_i - X_{target_i})' Q_i (X_i - X_{target_i})$, where Q_i is tuned to penalize more for being far from the desired state for variable i. In our case we have penalized mostly for being far from the desired targets along x, y and z because it is what is most important for following a 3D trajectory.

Moreover, in order to keep the computation time under control, we want the horizon to be short enough. To make this possible, we use a terminal cost Q_f, which is computed as the cost of an LQR controller for the linearized system as seen in the previous parts of the project.

The next state is given by the Runge-Kutta 4 function which approximates the integral of the system's continuous dynamics function $f(X, U)$ around the current state and input for a chosen sampling period. This next state is submitted to the same constraints as the previous ones.

Given all these conditions, one can compute the best input sequence that will bring the system to the desired output using CASADI for optimization.

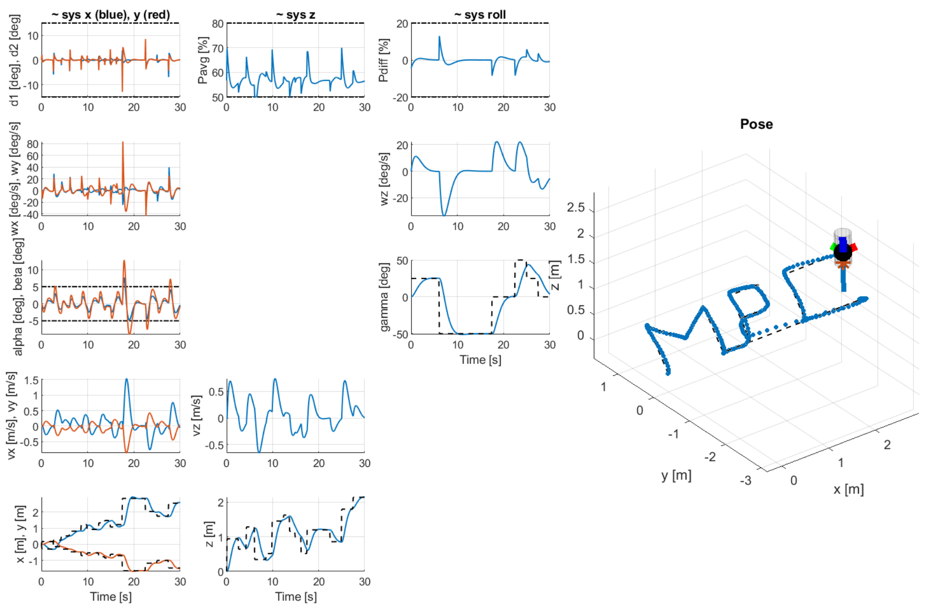


Figure 17: Rocket trajectory with Nonlinear MPC controller

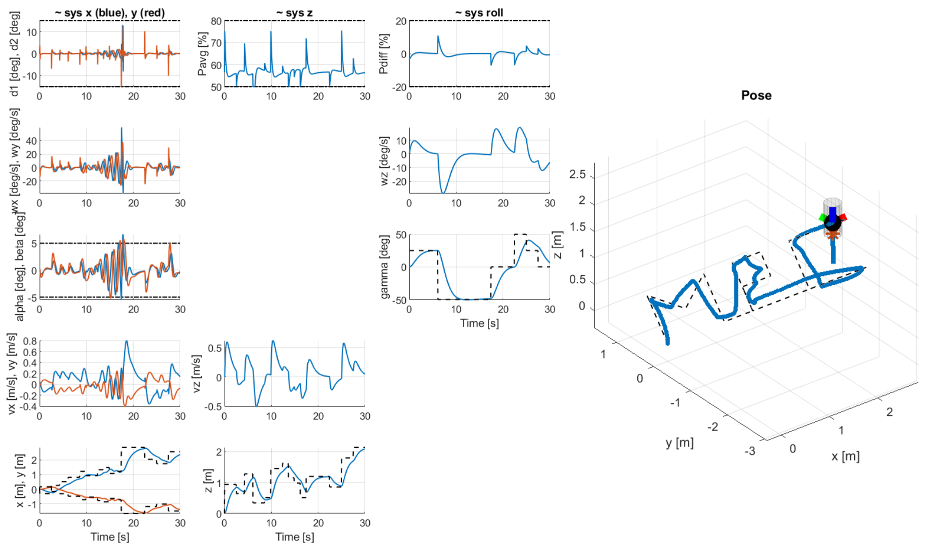


Figure 18: Rocket trajectory with linear MPC controller for a gamma trajectory up to $\pm 50°$

As we can see in figure 18, the linear controller struggles to follow the trajectory when it has to track some higher change in γ_{ref}.

Non-linear MPC improves greatly the trajectory of the rocket when tuned correctly compared with the linearized system. However, computation cost has increased. Whilst the linear MPC controller could track the trajectory reasonably well when the trajectory was "simple" (a maximum change of $|\gamma_{ref}| = 15°$), its performance drops significantly when the trajectory becomes more complex. Therefore, choosing between a linearized or a nonlinear MPC controller is a trade-off between the computation cost of the controller and the complexity of the trajectory to follow.