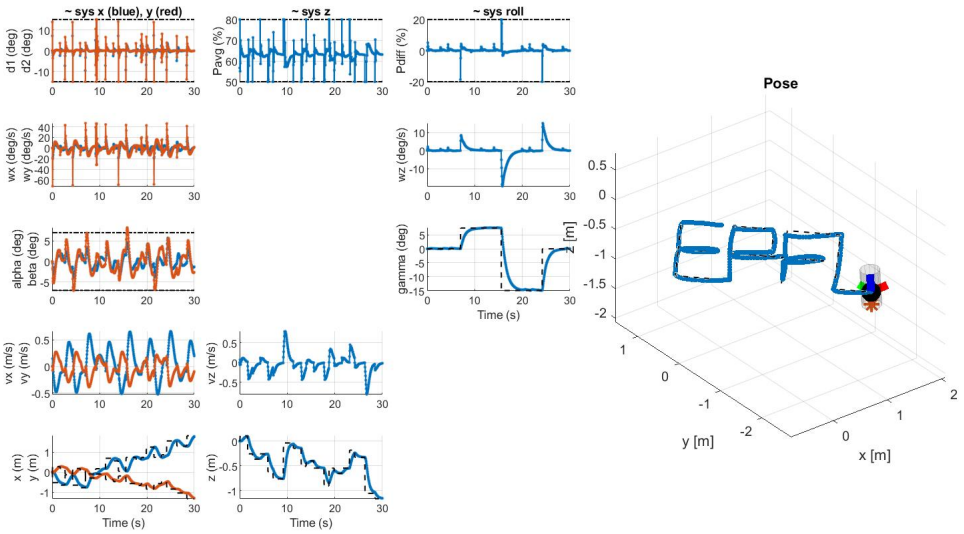
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Fall 2022

Model Predictive Control - Miniproject Report



Group AD:
Bassam EL RAWAS (SCIPER 310635)
Vincent GHEROLD (SCIPER 315014)
Benjamin Noah LUGON-MOULIN (SCIPER 312854)

Table of Contents

1 Introduction

This project aims to develop an MPC controller to fly a prototype of a rocket. We here study a small-scale prototype where the rocket engines are replaced by two counter-rotating drone propellers. The propeller pair is mounted on a gimbal and can be tilted by two servos.

Our report is divided into deliverables, where each deliverable marks a step in the progress of this project. In order to implement the MPC controller, a nonlinear model of the system dynamics of the rocket is already provided, whose states and inputs will be detailed in the next section.

1.1 System Definition

Two reference frames are considered : the first one of the body frame (subscript b) with origin O_b placed at the center of the rocket, and the second one is the fixed and inertial world frame (subscript w).

Our system is defined by the following state vector x and input vector u:

$$x = \begin{bmatrix} \omega \\ \varphi \\ v \\ p \end{bmatrix}, \; u = \begin{bmatrix} \delta_1 \\ \delta_2 \\ P_{avg} \\ P_{diff} \end{bmatrix} \tag{1}$$

where :

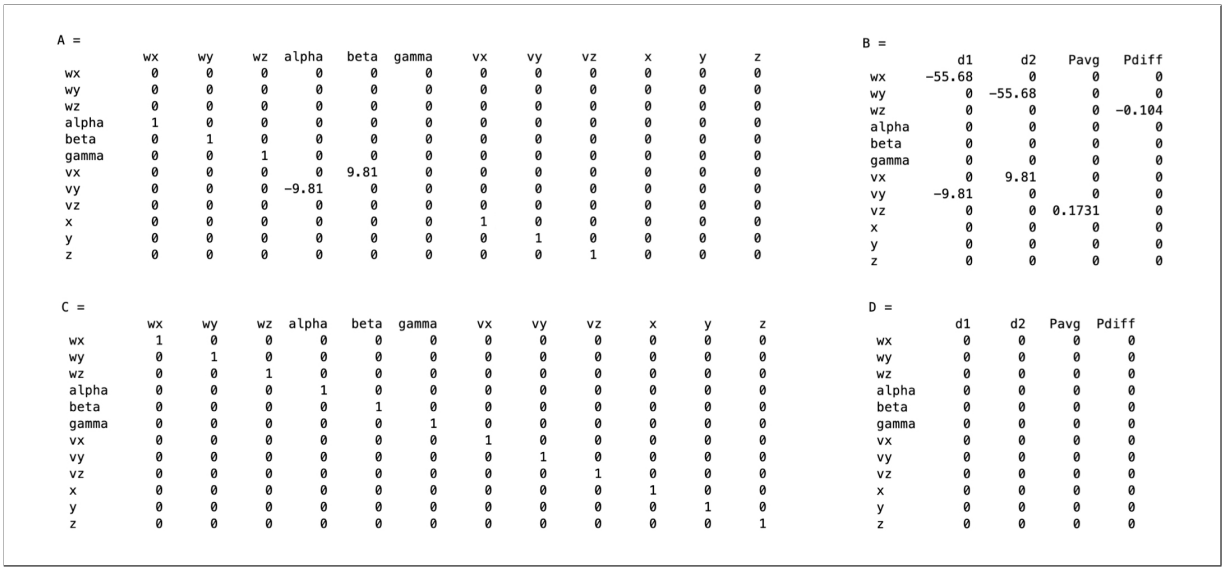
- $\omega = [\omega_x \omega_y \omega_z]^T$ $[rad/s]$ are the angular velocities about the body axes
- $\varphi = [\alpha \beta \gamma]^T$ $[rad]$ are Euler angles that represent the attitude of the body frame with respect to the world frame
- $v = [v_x v_y v_z]^T$ $[m/s]$ and $p = [xyz]^T$ $[m]$ are the velocity and position vectors expressed in the world frame
- δ_1 and δ_2 $[rad]$ are the deflection angles of servo 1 (about axis x_b) and servo 2 (about rotated axis y_b)
- $P_{avg} = (P_1 + P_2)/2$ and $P_{diff} = P_2 - P_1$ $[\%]$ are the average throttle and throttle difference between the motors

From the equations for forces and moments, linear and angular dynamics and attitude kinematics, the dynamic equation of the rocket is modeled as $\dot{x} = f(x, u)$.

2 Linearization

2.1 Deliverable 2.1

Here we control a trimmed and linearized version of the rocket. We start by computing the trim point (x_s, u_s), in other words the state and input equilibrium pair (x_s, u_s) for which the system $\dot{x} = f(x_s, u_s) = 0$. Next we linearize the nonlinear model about this trim point. From this linearization we get the following state-space system:



A =

	wx	wy	wz	alpha	beta	gamma	vx	vy	vz	x	y	z
wx	0	0	0	0	0	0	0	0	0	0	0	0
wy	0	0	0	0	0	0	0	0	0	0	0	0
wz	0	0	0	0	0	0	0	0	0	0	0	0
alpha	1	0	0	0	0	0	0	0	0	0	0	0
beta	0	1	0	0	0	0	0	0	0	0	0	0
gamma	0	0	1	0	0	0	0	0	0	0	0	0
vx	0	0	0	0	9.81	0	0	0	0	0	0	0
vy	0	0	0	-9.81	0	0	0	0	0	0	0	0
vz	0	0	0	0	0	0	0	0	0	0	0	0
x	0	0	0	0	0	0	1	0	0	0	0	0
y	0	0	0	0	0	0	0	1	0	0	0	0
z	0	0	0	0	0	0	0	0	1	0	0	0

B =

	d1	d2	Pavg	Pdiff
wx	-55.68	0	0	0
wy	0	-55.68	0	0
wz	0	0	0	-0.104
alpha	0	0	0	0
beta	0	0	0	0
gamma	0	0	0	0
vx	0	9.81	0	0
vy	-9.81	0	0	0
vz	0	0	0.1731	0
x	0	0	0	0
y	0	0	0	0
z	0	0	0	0

C =

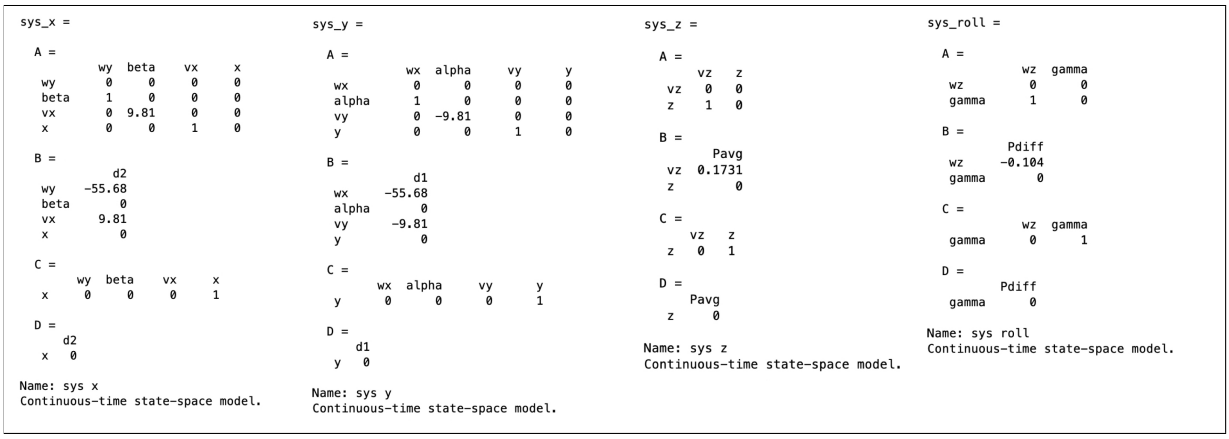
	wx	wy	wz	alpha	beta	gamma	vx	vy	vz	x	y	z
wx	1	0	0	0	0	0	0	0	0	0	0	0
wy	0	1	0	0	0	0	0	0	0	0	0	0
wz	0	0	1	0	0	0	0	0	0	0	0	0
alpha	0	0	0	1	0	0	0	0	0	0	0	0
beta	0	0	0	0	1	0	0	0	0	0	0	0
gamma	0	0	0	0	0	1	0	0	0	0	0	0
vx	0	0	0	0	0	0	1	0	0	0	0	0
vy	0	0	0	0	0	0	0	1	0	0	0	0
vz	0	0	0	0	0	0	0	0	1	0	0	0
x	0	0	0	0	0	0	0	0	0	1	0	0
y	0	0	0	0	0	0	0	0	0	0	1	0
z	0	0	0	0	0	0	0	0	0	0	0	1

D =

	d1	d2	Pavg	Pdiff
wx	0	0	0	0
wy	0	0	0	0
wz	0	0	0	0
alpha	0	0	0	0
beta	0	0	0	0
gamma	0	0	0	0
vx	0	0	0	0
vy	0	0	0	0
vz	0	0	0	0
x	0	0	0	0
y	0	0	0	0
z	0	0	0	0

Figure 1: Linearized state-space system matrices around the trim point

This state-space system can be decomposed into four independent sub-systems:



sys_x =

A =

	wy	beta	vx	x
wy	0	0	0	0
beta	1	0	0	0
vx	0	9.81	0	0
x	0	0	1	0

B =

	d2
wy	-55.68
beta	0
vx	9.81
x	0

C =

	wy	beta	vx	x
x	0	0	0	1

D =

	d2
x	0

Name: sys x
Continuous-time state-space model.

sys_y =

A =

	wx	alpha	vy	y
wx	0	0	0	0
alpha	1	0	0	0
vy	0	-9.81	0	0
y	0	0	1	0

B =

	d1
wx	-55.68
alpha	0
vy	-9.81
y	0

C =

	wx	alpha	vy	y
y	0	0	0	1

D =

	d1
y	0

Name: sys y
Continuous-time state-space model.

sys_z =

A =

	vz	z
vz	0	0
z	1	0

B =

	Pavg
vz	0.1731
z	0

C =

	vz	z
z	0	1

D =

	Pavg
z	0

Name: sys z
Continuous-time state-space model.

sys_roll =

A =

	wz	gamma
wz	0	0
gamma	1	0

B =

	Pdiff
wz	-0.104
gamma	0

C =

	wz	gamma
gamma	0	1

D =

	Pdiff
gamma	0

Name: sys roll
Continuous-time state-space model.

Figure 2: Separation into four non-interacting systems

This decomposition makes sense from a physical perspective : around the trim point, as the rocket is in an equilibrium state (i.e the rocket points straight upwards and maintains a steady hover), height z should only be influenced by the total thrust and hence throttle P_{avg} delivered by the two propellers. Roll angle γ should also only be dependant on the difference in speed and therefore power between the two propellers P_{diff}. Additionally, for small movements around the trim point, position x and y should only be controlled by thrust vector angles δ_2 and δ_1 respectively (a deflection angle of δ_1 induces a rotation about direction x_b, resulting in a horizontal shift in the y_b direction). Note that this is only valid for this trim point, as for example as soon as the rocket tilts vertically by an angle α or β, the positions x, y, z will then become dependant on all the input variables.

3 MPC Controller Design for each Sub-System

3.1 Deliverable 3.1

The design procedure for each of the four subsystems is identical. In order to ensure recursive feasibility, we introduce a terminal set X_f and impose that x_N, the state at step N of the trajectory, lie in this set. This set is computed from the LQR control law, and is positive invariant, hence ensuring recursive feasibility. The MPC problem can then be written as:

$$\begin{aligned} \min_{u} \quad & \sum_{i=1}^{N-1} x_i^T Q x_i + u_i^T R u_i + x_N^T Q_f x_N \\ \text{s.t.} \quad & x_{i+1} = A x_i + B u_i \\ & M x_i \leq m \\ & F u_i \leq f \\ & x_N \in X_f \end{aligned} \tag{2}$$

Where Q_f is computed from the LQR law, M, m, F, f are the constraint matrices on the states and the inputs respectively.

Constraints for each system

There are physical constraints on all the input variables, and extra constraints on the states α and β which come from the linearization of the system. Finally, the trim point (x_s, u_s) has to be taken into account when writing the constraints. In our case, $x_s = 0$ and $u_s = (0, 0, 56.667, 0)^T$.

- **X system:** For $u = \delta_2$, we have $|\delta_2| \leq 0.26$ rad $\iff \begin{pmatrix} 1 \\ -1 \end{pmatrix} u \leq \begin{pmatrix} 0.26 \\ 0.26 \end{pmatrix} \iff Fu \leq f$.

  For $x = (\omega_y, \beta, v_x, x)^T$, we have $|\beta| \leq 0.1222$ rad $\iff \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix} x \leq \begin{pmatrix} 0.1222 \\ 0.1222 \end{pmatrix}$ $\iff Mx \leq m$.

- **Y system:** For $u = \delta_1$, we have $|\delta_1| \leq 0.26$ rad $\iff \begin{pmatrix} 1 \\ -1 \end{pmatrix} u \leq \begin{pmatrix} 0.26 \\ 0.26 \end{pmatrix} \iff Fu \leq f$.

  For $x = (\omega_x, \alpha, v_y, y)^T$, we have $|\alpha| \leq 0.1222$ rad $\iff \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix} x \leq \begin{pmatrix} 0.1222 \\ 0.1222 \end{pmatrix}$ $\iff Mx \leq m$.

- **Z system:** For $u = P_{\text{avg}}$, we have $50\% \leq P_{\text{avg}} \leq 80\% \iff \begin{pmatrix} 1 \\ -1 \end{pmatrix} u \leq \begin{pmatrix} 23.333 \\ 6.667 \end{pmatrix} \iff Fu \leq f$ where we subtracted the trim coordinate for P_{avg} from 50% and 80%. There are no constraints on the states $\text{x} = (v_z, z)^T$.

- **Roll system:** For $u = P_{\text{diff}}$, we have $-20\% \leq P_{\text{diff}} \leq 20\% \iff \begin{pmatrix} 1 \\ -1 \end{pmatrix} u \leq \begin{pmatrix} 20 \\ 20 \end{pmatrix} \iff Fu \leq f$. There are no constraints on the states $\text{x} = (\omega_z, \gamma)^T$.

Cost matrices for each controller

The cost matrices Q and R were found using trial and error, in order to ensure stability and a settling time no bigger than 8 seconds. Horizon length should be chosen big enough to ensure stability, but not too high in order to keep the required computation time reasonable. We chose a horizon length of $H = 4$

seconds, as it represents a good trade-off between guarantee of feasibility and computation time. This choice of horizon length was used for the entire project, up until the tuning of the NMPC controller.

The choice of these cost matrices also has an effect on the size of the terminal invariant sets, but we did not take that into account when determining them as no constraints were imposed on this aspect.

- **X controller:** We chose $Q = \begin{pmatrix} 10 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 10 \end{pmatrix}$ and $R = 1$ for the states ω_y, β, v_x, x and input δ_2. Indeed, we want to quickly stabilize ω_y and x in order to have better tracking and to not violate constraints on the Euler angle β. Having v_x reach 0 a little faster helps the system converge to 0 faster. Since there are no particular constraints on δ_2, we chose $R = 1$.

  The projections of the terminal invariant set looks like the following:

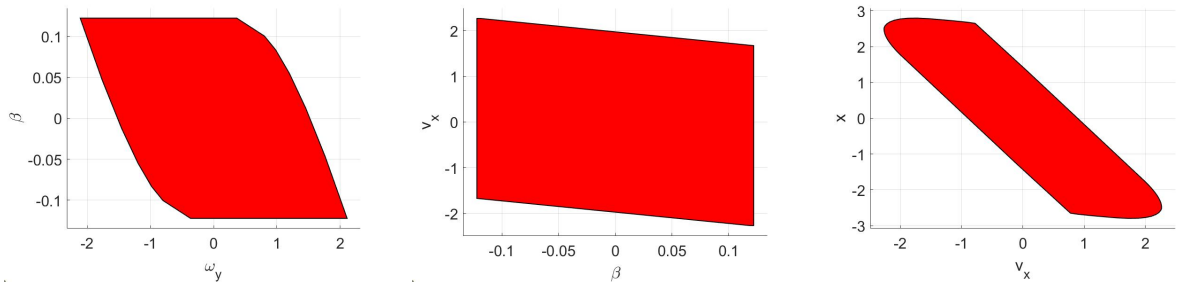


Figure 3: Projections of the terminal set of the X controller

- **Y controller:** The effects of this controller are identical to that of the previous one, with the exception being the direction of movement. The matrices are all the same, and the projections of the terminal invariant set are shown below:

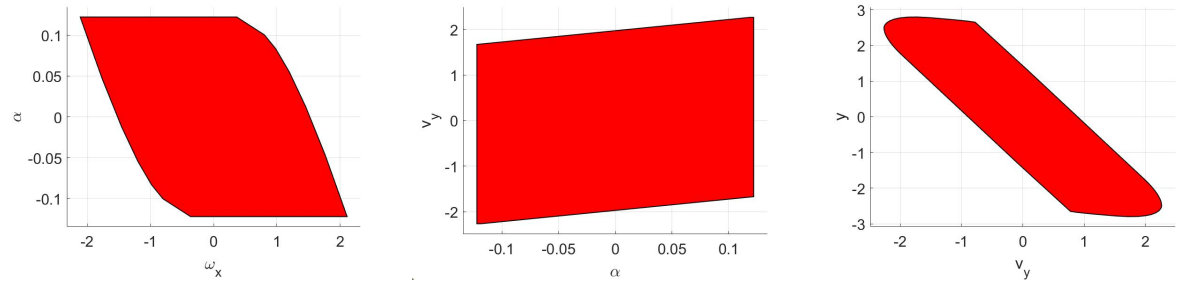


Figure 4: Projections of the terminal set of the Y controller

  As expected, they are (almost) identical to that of the X controller.

- **Z controller:** We chose $Q = \begin{pmatrix} 10 & 0 \\ 0 & 40 \end{pmatrix}$ and $R = 1$ for the states v_z, z and input P_{avg}. Indeed, we want a fast regulation in z, and having the speed converge to 0 faster helps. We also did not want to impose any constraint on the way P_{avg} should behave, hence $R = 1$.

The terminal invariant set is shown in the figure below:

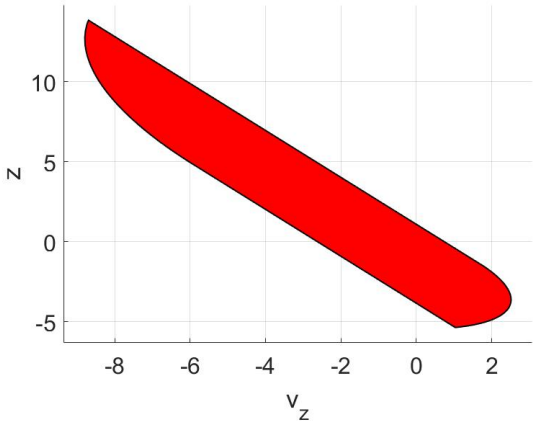


Figure 5: Terminal set of the Z controller

- **Roll controller:** We chose $Q = \begin{pmatrix} 10 & 0 \\ 0 & 20 \end{pmatrix}$ and $R = 0.5$ for the states ω_z, γ and input P_{diff}. We want both states to converge to 0 quickly, and the somewhat small value of R allows this minimization to happen fast.

  The terminal invariant set is shown in the figure below:

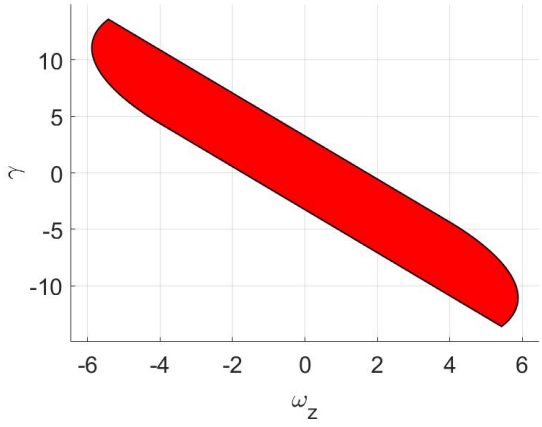


Figure 6: Terminal set of the roll controller

Simulation of each controller

We simulated each controller in open and closed loop, where the starting state was a stationary rocket 4 meters from the origin for x, y, z, and stationary at 45° for roll.

- **X controller:**

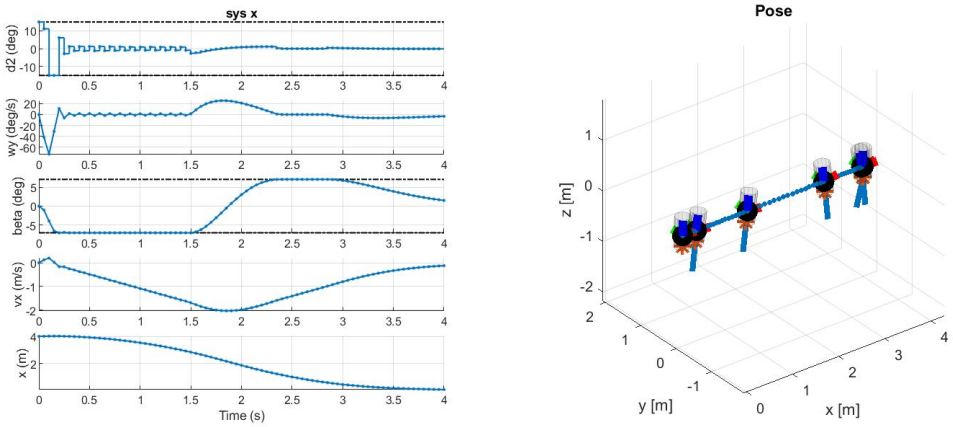


Figure 7: X controller open loop simulation

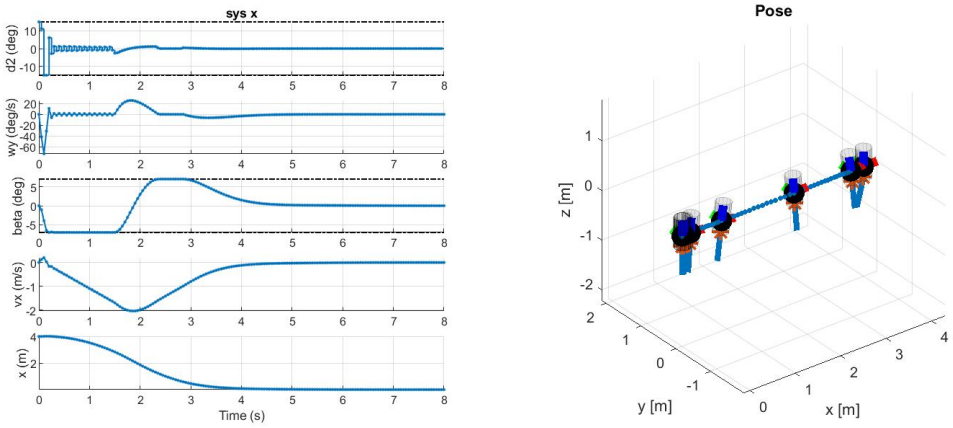


Figure 8: X controller closed loop simulation

- **Y controller:**

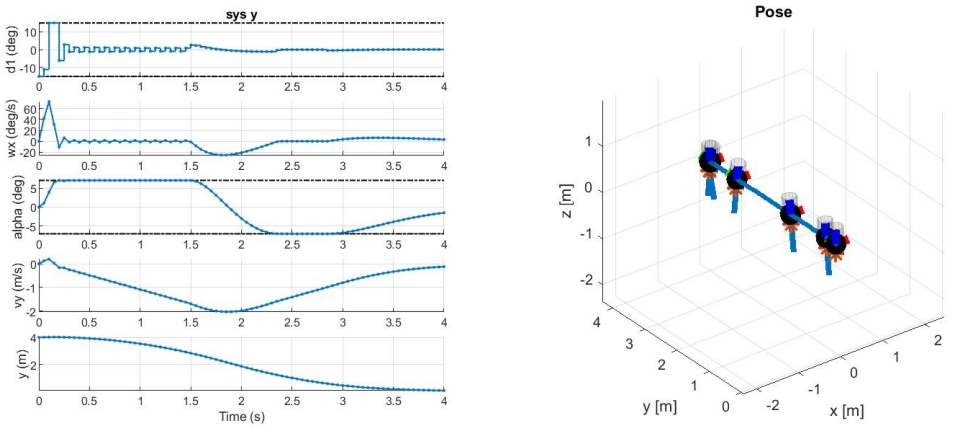


Figure 9: Y controller open loop simulation

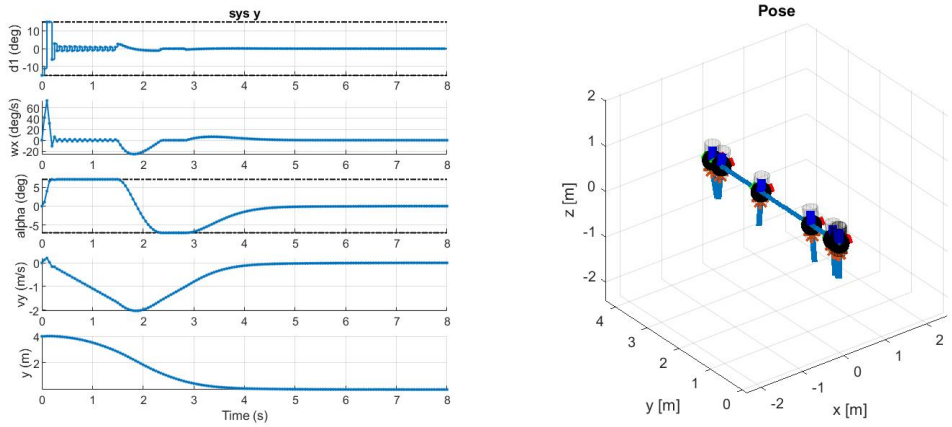


Figure 10: Y controller closed loop simulation

- **Z controller:**

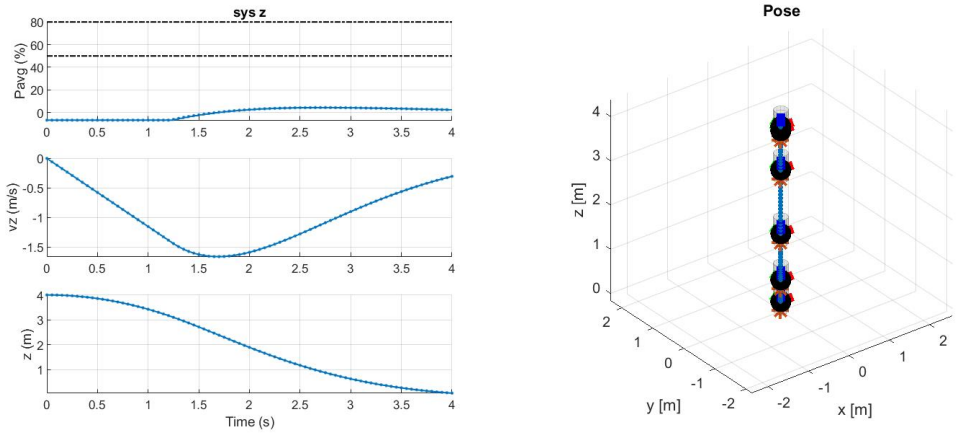


Figure 11: Z controller open loop simulation

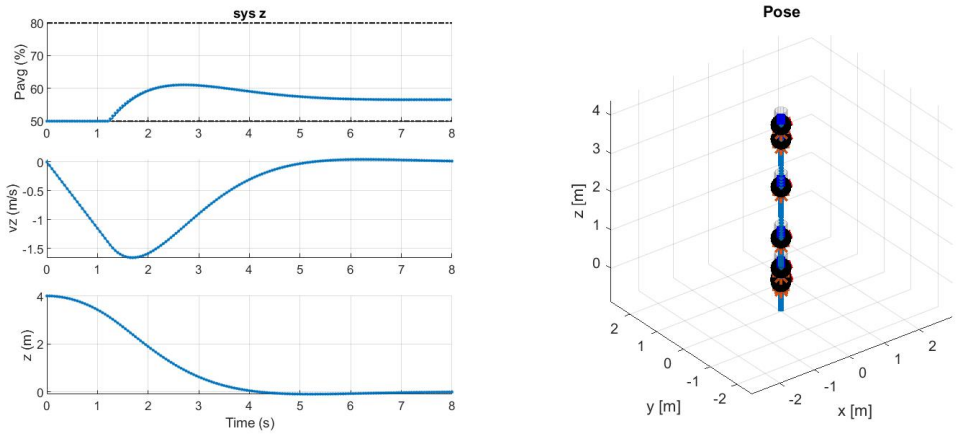


Figure 12: Z controller closed loop simulation

- **Roll controller:**

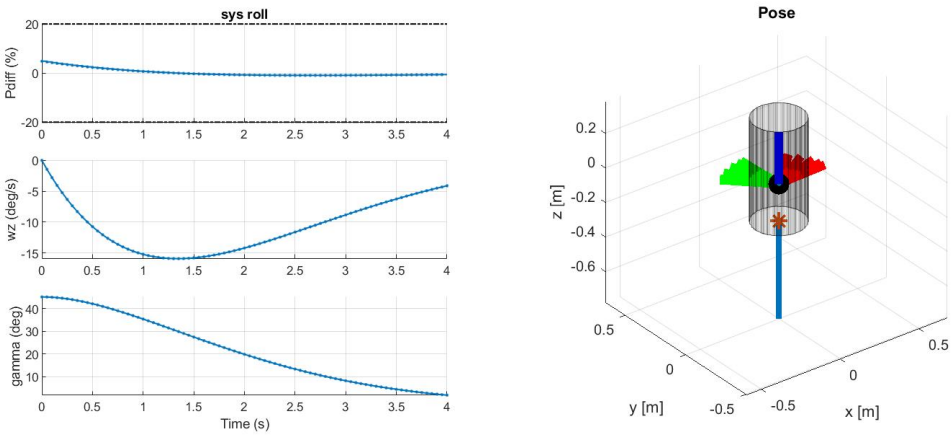


Figure 13: Roll controller open loop simulation

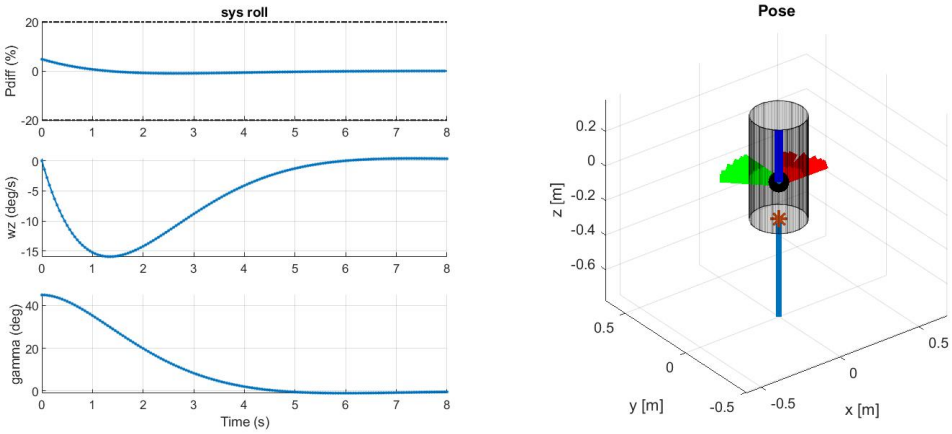


Figure 14: Roll controller closed loop simulation

3.2 Deliverable 3.2

In this section, we implement controllers for tracking. Given a reference r, we first need to compute the steady-state points (x_s, u_s) and use these as a reference for the controllers. Indeed, we need to find the corresponding steady states and inputs (x_s, u_s) that correspond to the output y being equal to r. This is done by solving the following optimization problem:

$$\begin{aligned} \min_{u_s} \quad & u_s^2 \\ \text{s.t.} \quad & x_s = Ax_s + Bu_s \\ & r = Cx_s \\ & Mx_s \leq m \\ & Fu_s \leq f \end{aligned} \tag{3}$$

Once these points are calculated, the optimization problem used to get the sequence of states and inputs is the same as equation (2) but with the origin shifted. Indeed, we define

$$\begin{aligned} \Delta u &= u - u_s \\ \Delta x &= x - x_s \end{aligned} \tag{4}$$

and we calculate

$$\min_{\Delta u} \quad \sum_{i=1}^{N-1} \Delta x_i^T Q \Delta x_i + \Delta u_i^T R \Delta u_i + x_N^T Q_f x_N \tag{5}$$

under the same constraints as equation (2) but with the terminal constraints dropped, and Q_f given by the LQR control law.

Here are the plots of each subsystem, starting at 4 meters from the origin with a roll of 0° and tracking a reference to -4 meters for x, y, z and to 35° for roll:

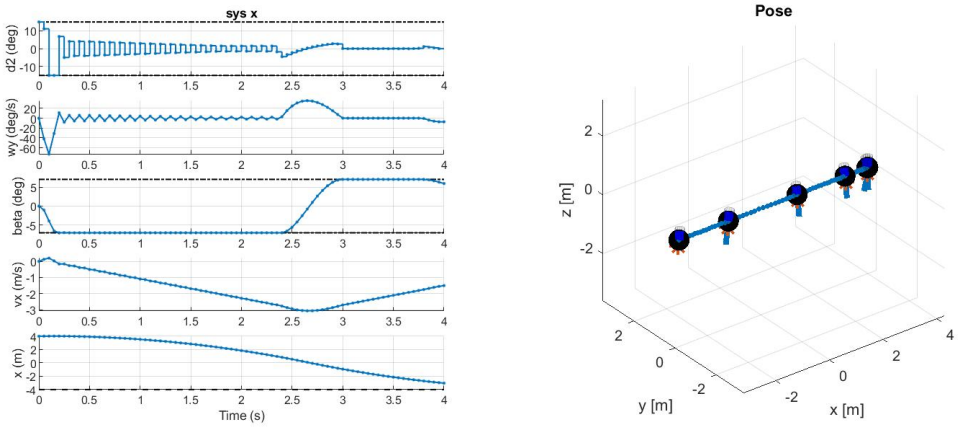


Figure 15: X controller tracking open loop simulation

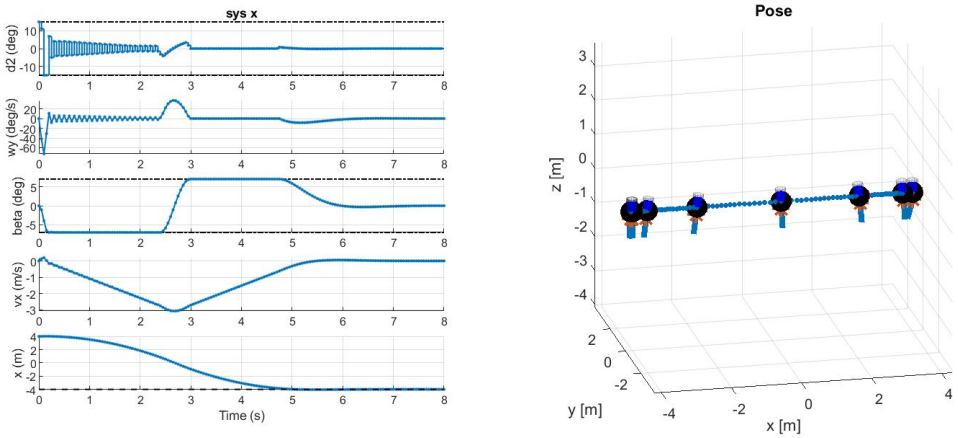


Figure 16: X controller tracking closed loop simulation

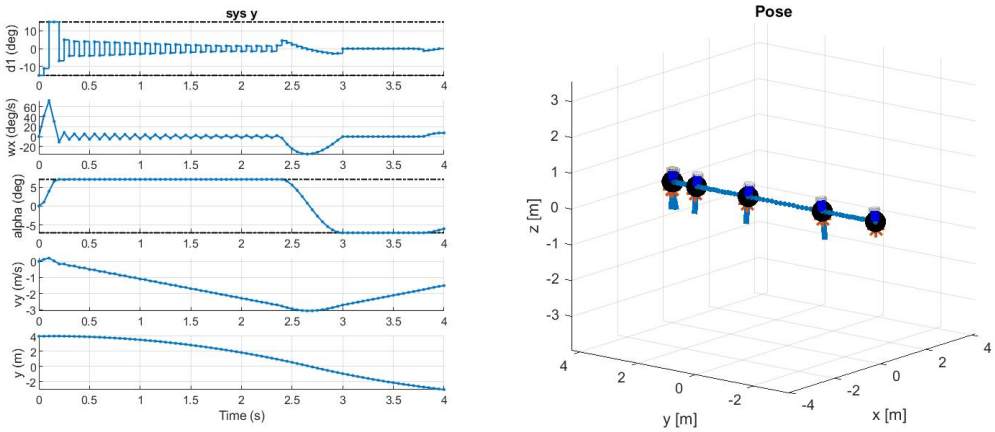


Figure 17: Y controller tracking open loop simulation

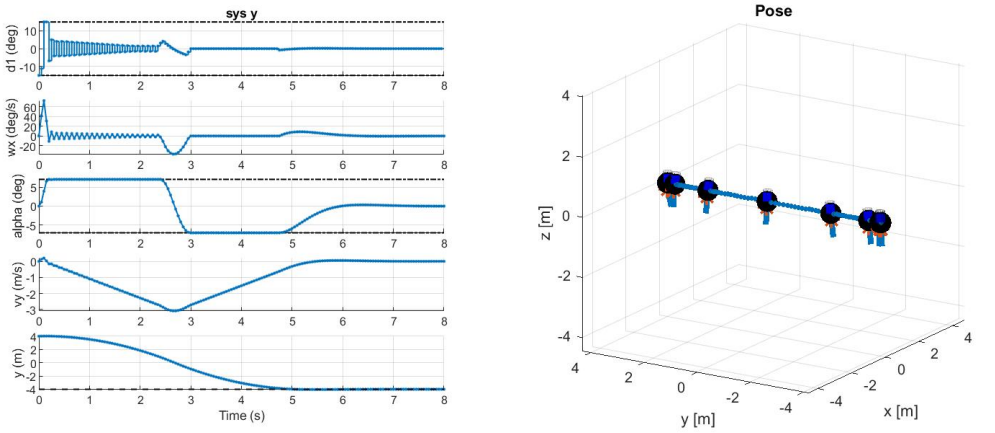


Figure 18: Y controller tracking closed loop simulation

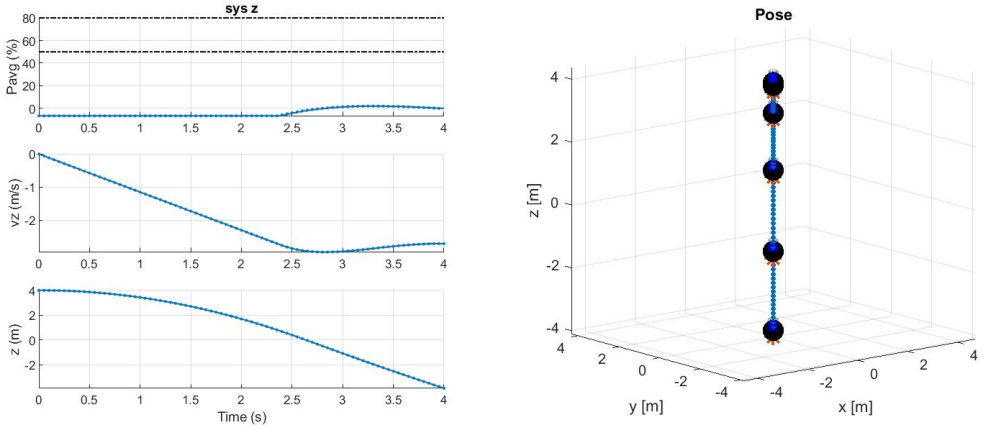


Figure 19: Z controller tracking open loop simulation

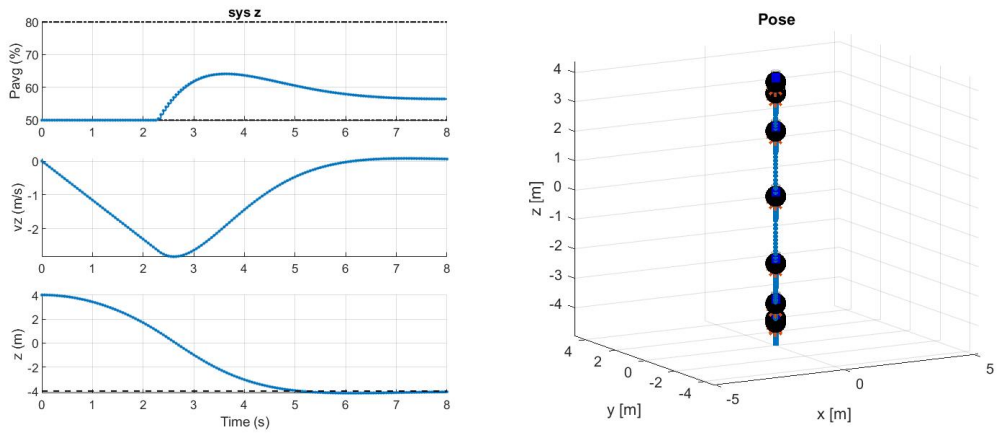


Figure 20: Z controller tracking closed loop simulation

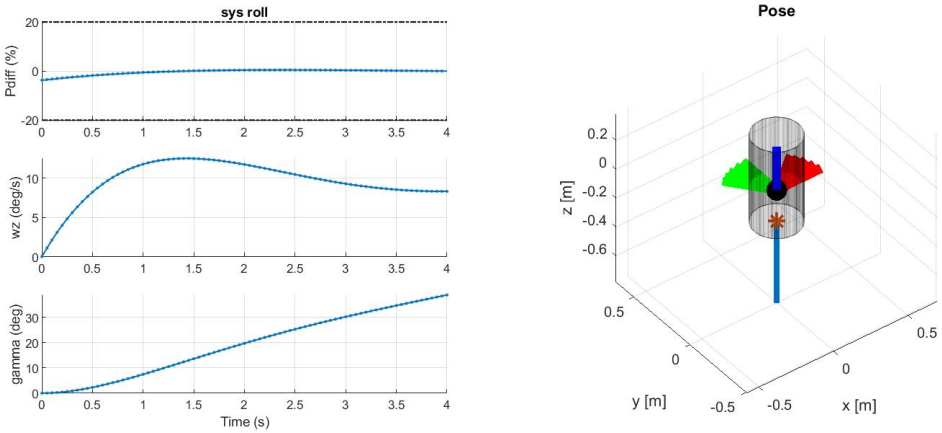


Figure 21: Roll controller tracking open loop simulation

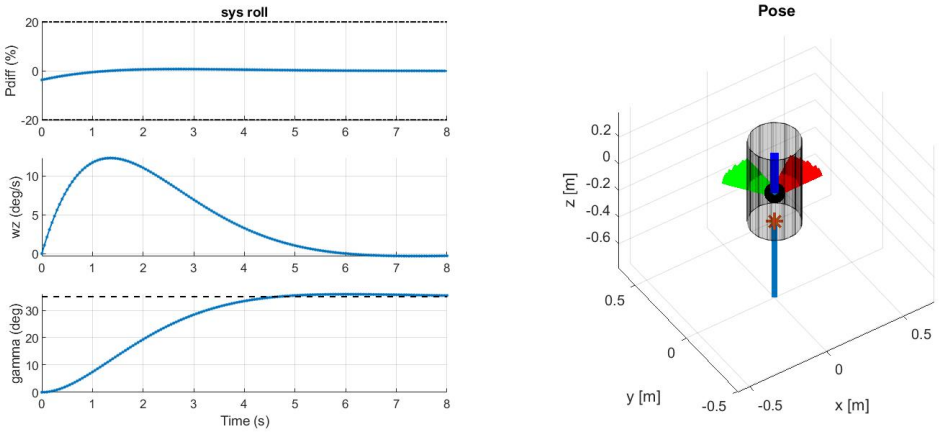


Figure 22: Roll controller tracking closed loop simulation

4 Simulation with Nonlinear Rocket

4.1 Deliverable 4.1

From merging together the previously tuned controllers, we get the following performance for the nonlinear rocket :

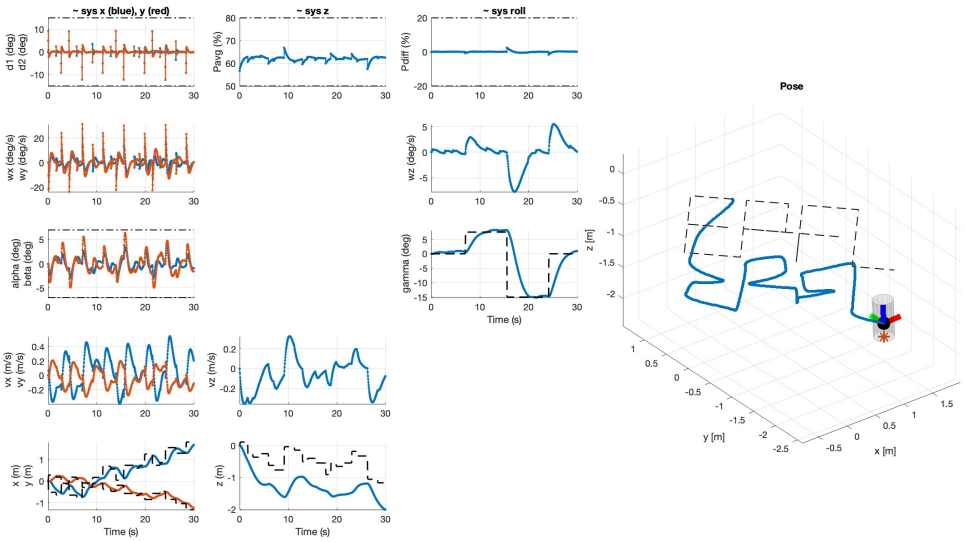


Figure 23: Merged Linear MPC controllers in first nonlinear simulation

From this simulation we first notice that the z controller does a poor job at tracking the reference trajectory. To remediate this, we give more relative importance to the tracking in position z and velocity v_z by setting a lower cost on the inputs : we set $R = 0.001I$, with I being the identity matrix. This already results in much better performance:

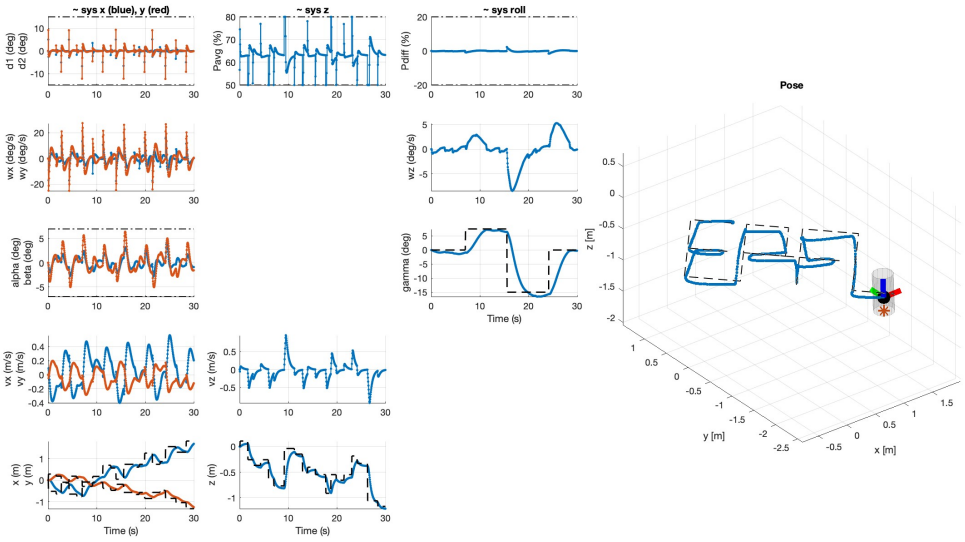


Figure 24: Improved performance when setting $R = 0.001I$

In attempts to further improve tracking for controllers x, y and roll, we decreased the R matrix in a similar fashion to $R = 0.01I$. We also increased the weights in matrices Q corresponding to the positions x, y, z and γ. This however led to infeasibilities. To ensure our problem stayed feasible under model mismatch, we formulated soft state constraints on α and β. We obtain slightly better performances:

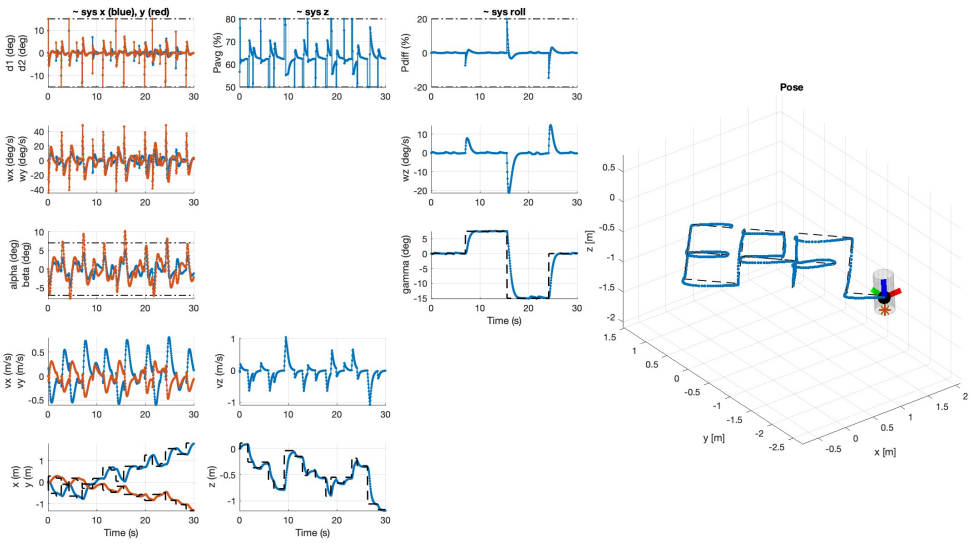


Figure 25: Simulation results with the following tuning parameters: for x and y controllers we have $Q = \text{diag}(10, 1, 5, 40)$, $R = 0.01I$, $S = 10I$, for z controller $Q = \text{diag}(100, 700)$, $R = 0.001I$, for γ controller $Q = \text{diag}(10, 40)$, $R = 0.01$

We can still improve the tracking of the z reference, as well as decrease the constraint overshoots on α and β induced by the slack. After some more tuning, we get the following plot:

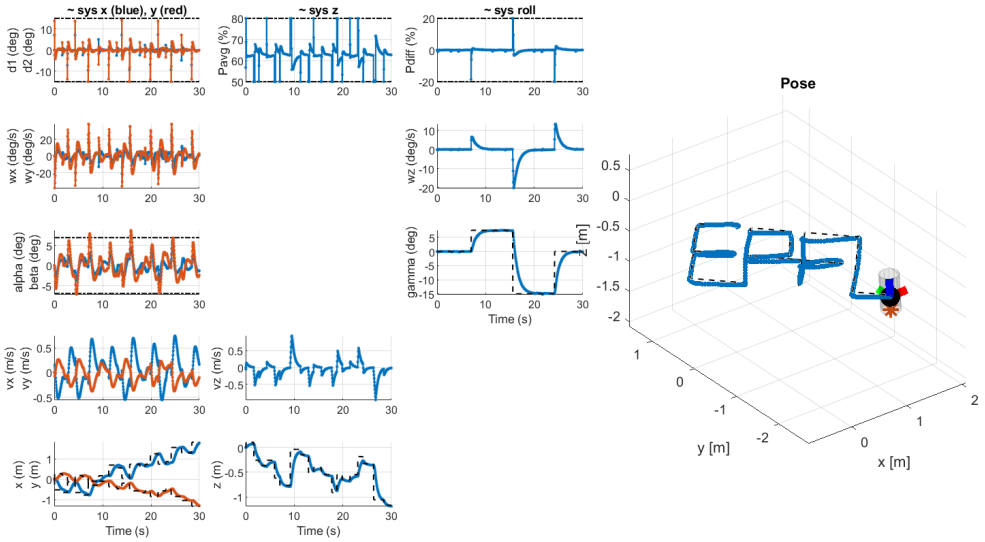


Figure 26: Final closed loop trajectory. For the x and y controllers we have $Q = \text{diag}(20, 10, 5, 45)$, $R = I$, $S = 100I$. For z we have $Q = \text{diag}(250, 900)$ and $R = 0.0009I$. For roll we have $Q = \text{diag}(60, 70)$ and $R = 0.001$.

The noticeable downside is the overshoot on the constraints on α and β. These constraints are not physical, they are imposed only on the linearized system to ensure that the states do not deviate far from the trim point. To avoid these overshoots, we tried reducing the permissible slack as well as removing constraints on the initial state x_0, but our best tracking performance is achieved with the parameters in Figure (26) which do include this overshoot.

5 Offset-Free Tracking

5.1 Deliverable 5.1

This section has the aim to extend the z-controller to compensate for mass changes of the rocket. The dynamics of the system in the z-direction becomes :

$$x^+ = Ax + Bu - Bd \tag{6}$$

where d is an unknown disturbance, which needs to be rejected by the controller.

The approach for offset-free control is to use the output measurements and model to estimate the states and the disturbance. In order to do this, we design a state and disturbance estimator based on the following augmented model :

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

The variables $\hat{x}$ and $\hat{d}$ designate the estimated states and disturbances.. This leads to the following error dynamics:

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

Where the second equality comes from our model which imposes $B_d = B$ and $C_d = 0$. We get the augmented matrices are $\bar{A} = \begin{bmatrix}A & B\\ 0 & I\end{bmatrix}$, $\bar{B} = \begin{bmatrix}B\\ 0\end{bmatrix}$ and $\bar{C} = \begin{bmatrix}C & 0\end{bmatrix}$, and we need to choose $L = \begin{bmatrix}L_x & L_d\end{bmatrix}^T$ to ensure that the error dynamics are stable and converge to zero.

For the estimator poles, we chose 0.5, 0.6, 0.7. Indeed, these poles are first and foremost in the unit circle. They are not too close to 1 and hence not very slow, and based on our simulations, poles that were too close to 0 lead to infeasibilities. When small poles did give us results, they were very aggressive and the reference tracking was not better than what we were able to achieve with 0.5, 0.6, 0.7.

Without the offset-free controller, and by setting `rocket.mass=2.1`, we get the following trajectory:

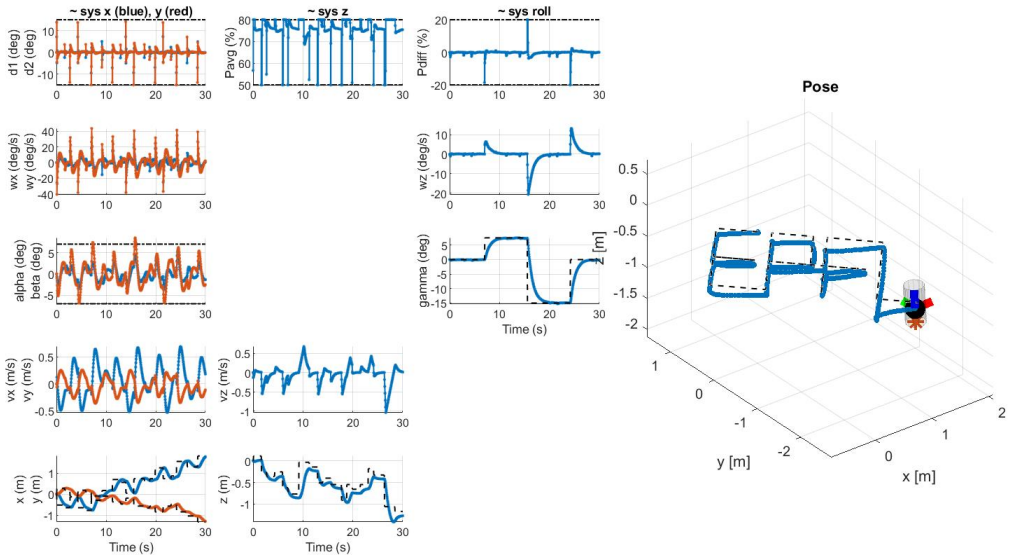


Figure 27: Mass mismatch effect on controllers from Part 4, with `rocket.mass=2.1`

The z controller can be seen struggling to follow its reference, and an increase of the `rocket.mass` variable will make this effect even more apparent.

Implementation of the offset-free controller yields the following much better result:

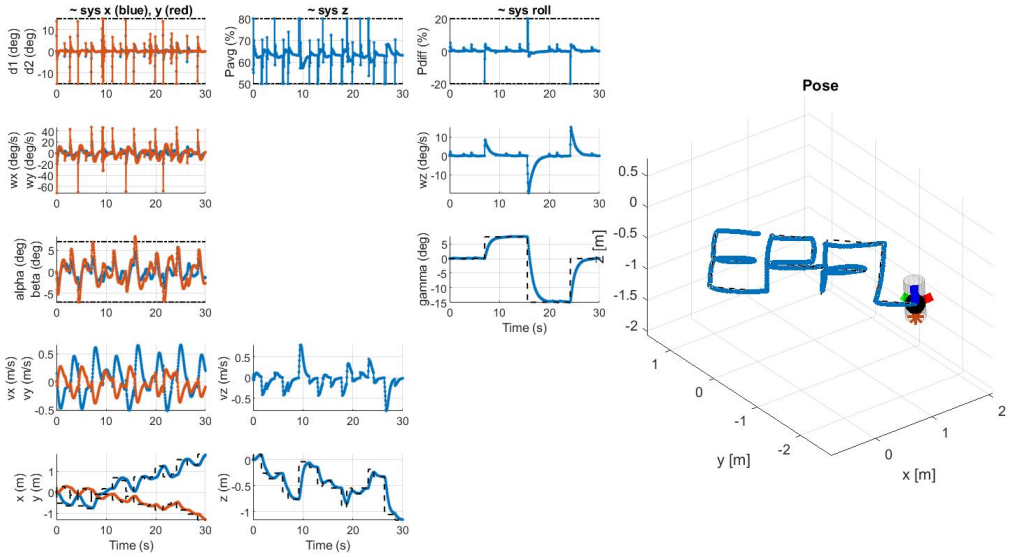


Figure 28: Offset-free tracking of EPFL reference

We can see that the constant disturbance was successfully rejected.

6 Nonlinear MPC

6.1 Deliverable 6.1

The goal of this section is to create a nonlinear controller that covers the full state space of the rocket and provides four input commands (thus we no longer decompose the rocket into four subsystems). Because we no longer linearize our systems, the previous constraints on α and β are dropped. We only impose $|\beta| \leq 80°$ to avoid hitting a singularity in our nonlinear model.
Hence, the constraints on our systems are the following:

$$\begin{aligned} &|\beta| \leq 80° \\ &|\delta_1| \leq 15° \\ &|\delta_2| \leq 15° \\ &50\% \leq P_{\text{avg}} \leq 80\% \\ &|P_{\text{diff}}| \leq 20\% \end{aligned} \tag{9}$$

Because using a nonlinear controller makes computation time longer, the horizon time H was reduced from 4 seconds to 3 seconds. In order to maintain good control while avoiding a large horizon length, we included a terminal cost, which is computed by linearizing and discretizing our model. We use the 4th order Runge-Kutta (RK4) algorithm for the discretization.

The NMPC problem can then be formulated as follows:

$$\begin{aligned} \min_{u} \quad & \sum_{i=1}^{N-1} x_i^T Q x_i + u_i^T R u_i + x_N^T Q_f x_N \\ \text{s.t.} \quad & x_{i+1} = f_{\text{discrete}}(x_i, u_i) \\ & Mx_i \leq m \\ & Fu_i \leq f \end{aligned} \tag{10}$$

Where Q_f is computed from the LQR control law, f_{discrete} is the non linear model, M, m, F, f are the constraint matrices on the states and the inputs respectively.

The control gains for Q and R are as follows:

$$\begin{aligned} Q &= \text{diag}(20, 20, 10, 1, 1, 190, 45, 45, 45, 400, 400, 400) \\ R &= \text{diag}(15, 15, 0.0001, 0.0001) \end{aligned} \tag{11}$$

Indeed, there are high weights for the states x, y, z to ensure good tracking. There is also a high weight for γ, to make sure that the rocket is able to handle sharp turns when tracing out the letters. The weights for P_{avg} and P_{diff} are very low to ensure that the controller has enough margin to track the x, y and z reference.

We obtain the following results with the designed NMPC controller:

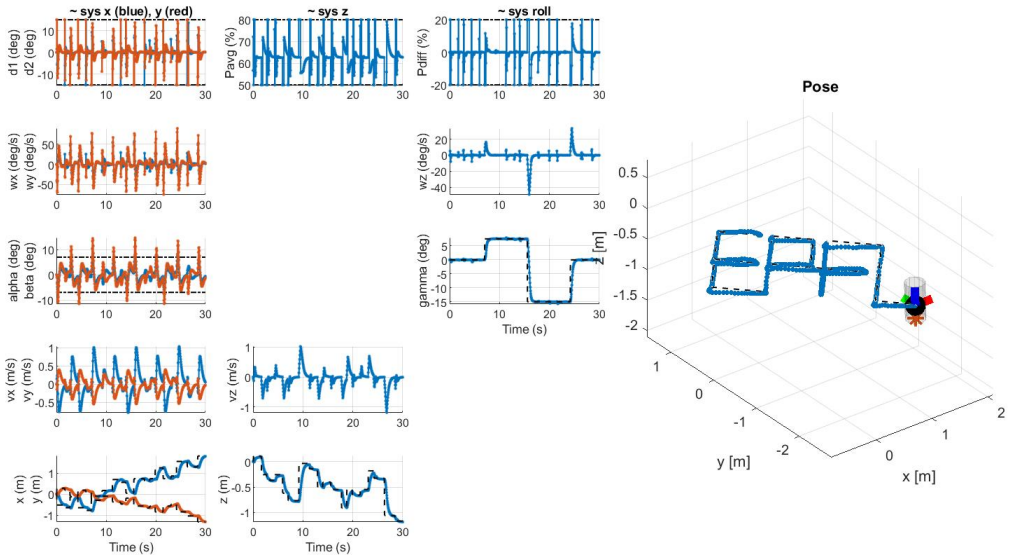


Figure 29: NMPC tracking results for a maximal roll of $\gamma = 15°$

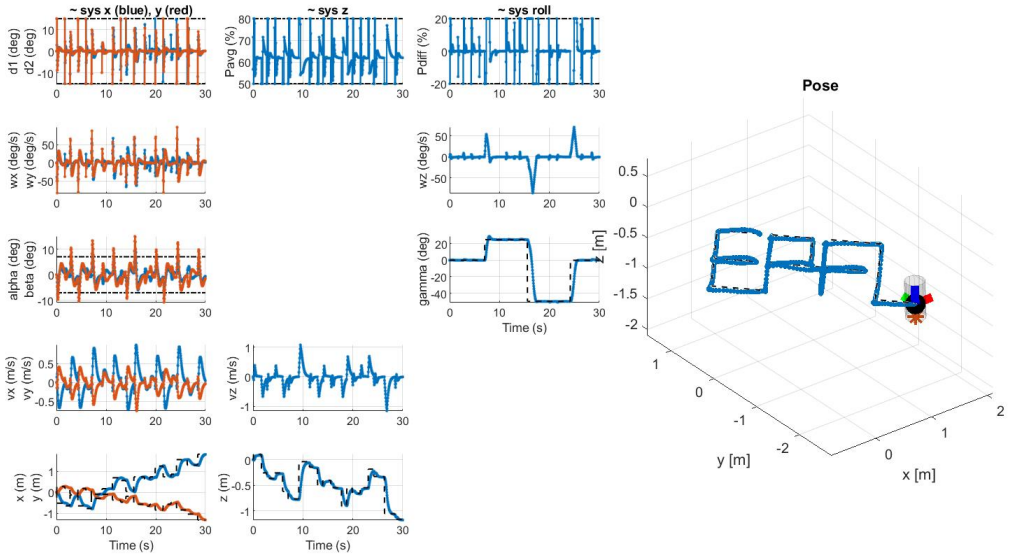


Figure 30: NMPC tracking results for a maximal roll of $\gamma = 50°$

Both controllers successfully track the EPFL reference trajectory in the simulations.

The NMPC controller allows to cover the whole state space (except for β) in contrast to the linear MPC controller which requires more constraints. However, for NMPC, the computational cost is increased due to the nonlinear model. With a horizon of 3, the computation time is around 60 seconds, while it takes only 40 seconds for a horizon of 4 in the case of linear MPC. Therefore we can see that the use of NMPC may be limited to slower systems that can accept long computation times, whereas linear MPC would allow control on much faster systems such as rockets. Additionally, The NMPC can be tricky to use if there are singularities in the model (in our case, we must add a constraint on state β to avoid a singularity

at $\beta = 90°$). Overall, when the maximum roll is set to 15°, our linear MPC controller provides satisfying tracking with the benefit of a low computational cost, which makes it most suitable for the control of the rocket.

As for the results when the maximal roll is set to 50°, here is the performance of the linear controller from section 4.1:

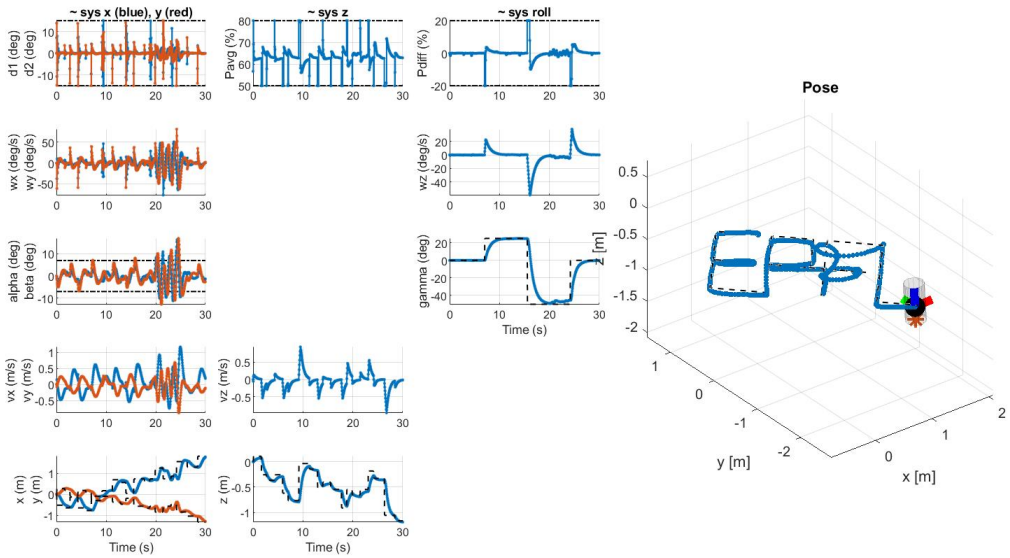


Figure 31: Linear MPC tracking results for a maximal roll of $\gamma = 50°$

Compared to the performance of the NMPC controller shown on figure 30, the tracking results are much worse on the "F" letter and satisfactory elsewhere. This requires a retuning of the linear controller (or the usage of another controller altogether), whereas the nonlinear controller managed to provide satisfying performances for both the 15° and 50° case. This could be due to the fact that the linear controller separated the system into 4 subsystems, and that the lack of interaction between them led to unexpected behaviors. This problem is non-existent for the nonlinear case.

7 Conclusion

To conclude this project, we learned how to develop an MPC controller for the control of a rocket prototype and compared the effectiveness of using a linear versus non-linear approach for this model. From the provided state-space model of the prototype, we first linearized and separated our model into four independent sub-systems, which would later facilitate the tuning of the whole system. Next we designed an MPC controller for each sub-system, which required defining constraints, computing a terminal set, and adjusting parameters such as cost functions Q and R to obtain desired performance while maintaining stability. We expanded this to control the non-linear model of the rocket, as well as compensate for mass changes (offset-free tracking for the z-controller). Finally, we designed a non-linear MPC controller and compared its performances to those of the previous linear MPC, providing us a better understanding of the concepts and limitations of MPC.