ME 425 - Model Predictive Control

Mini-Project

Thrust Vector Control

Mehmet Furkan Doğan[1],
Ali Fuat Şahin[1], and
Mustafa Emre Gürsoy[1]

[1]École Polytechnique Fédérale de Lausanne

Prof. Colin Jones

Contents

Part 1 Introduction

During this project, we focused on creating an MPC system for a rocket prototype. This involved designing an MPC regulator, incorporating tracking mechanisms, and implementing an estimator to address potential offsets caused by disturbances. Our testing encompassed a non-linear simulation, highlighting the challenges of employing linear MPC in a non-linear system. We concluded the project with the design and implementation of a non-linear MPC using CASADI, followed by a comparative analysis of its performance against the linear MPC.

Part 2 Linearization

Deliverable 2.1

Our system model consists of twelve states and four inputs:

$$\mathbf{X} = \begin{bmatrix} \omega_x & \omega_y & \omega_z & \alpha & \beta & \gamma & v_x & v_y & v_z & x & y & z \end{bmatrix}^T \tag{1}$$

$$\mathbf{U} = \begin{bmatrix} \delta_1 & \delta_2 & P_{avg} & P_{diff} \end{bmatrix}^T \tag{2}$$

The system is to be linearized around a point and a steady-state position and steady-state input are to be found. After linearizing our system around the trim point $\mathbf{X_t} = \mathbf{0}$ and $\mathbf{U_t} = \begin{bmatrix} 0 & 0 & 56.667 & 0 \end{bmatrix}^T$, we can observe four subsystems that are completely independent of each other from the overall state matrix.

But even from a physical/mechanical perspective before doing any linearization, we can intuitively say that, for the small angles of β and α, which we do the linearization around, certain components of motion can be described by subcomponents of the state matrix. These relations can be summarized as:

X: Movements along the X axis can be described by four states: ω_y, v_x, x, and β. The variables ω_y and β are effected by the input δ_1. This is only because a small difference in δ_1 causes the thrust vector to change direction when we are close to the linearization point.

Y :Movement in Y axis is quite similar: ω_x, v_y, y and α are the variables affecting the motion in Y. The input δ_2 is the driving force for this axis and directly influences the variables α and ω_x. Similarly, this is only because a small difference in δ_2 causes the thrust vector to change direction when we are close to the linearization point.

Z :In the Z direction, the motion is described by v_z and z, and they are proportional to the P_{avg}. In reality, δ_1 and δ_2 affect the movement in Z direction. However, close to the linearization point, these differences are small, and the linearized system is close to reality.

Roll :In roll motion, the variables are ω_z and γ. The input variable influencing the motion is P_{diff}. Similarly, the magnitude of the P_{diff} might be constrained by the magnitude of P_{avg}. In our case, close to the linearization point, this constraint does not apply since we are far from the boundaries.

In a nonlinear system, each of the subsystems is interacting. In our case, this separation makes sense since the states are assumed to be close to the linearization point.

Part 3 Design of MPC Controllers for Each Sub-System

Deliverable 3.1

3.1.1 Recursive Feasibility

Recursive constraint satisfaction in MPC involves the use of a terminal cost and a terminal constraint to ensure that system constraints are satisfied over an infinite time horizon. To achieve this, a terminal set is carefully chosen to guarantee feasibility. A terminal cost serves as a measure of the system's performance at the end of the prediction horizon whereas the terminal constraint ensures the system remains in a safe region at the end of the prediction horizon.

The chosen terminal set is designed to be invariant under the Linear Quadratic Regulator (LQR) control law. In this case, the terminal set is selected as the maximum invariant set for the closed-loop system $x^+ = (A + BK)x$ to ensure that the system is feasible under LQR control law at the end of each time horizon.

Then, the terminal cost was introduced as a continuous Lyapunov function inside the terminal set to ensure stability. Terminal cost is determined as the optimal LQR cost in this case and the terminal weight is calculated from the solution to the discrete-time algebraic Ricatti equation.

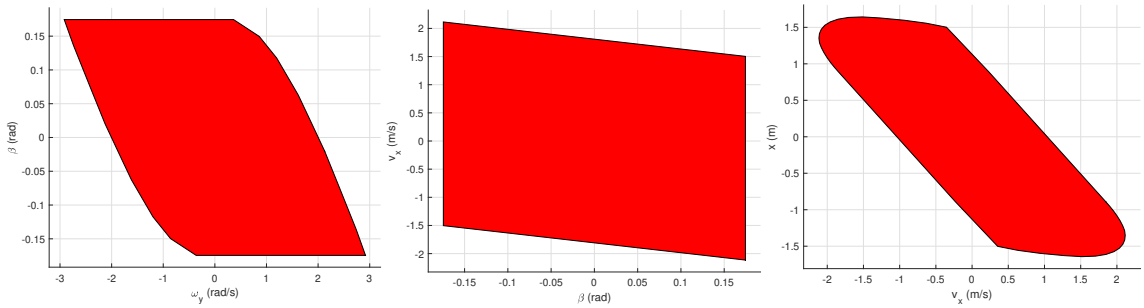
The designed terminal sets are dependent on the state and input constraints. In the mpc_x controller, the only state constraint is on β and there is an input constraint of 15°. Similarly, in the mpc_y controller only state constraint is on α and there is an input constraint of 15°. These constraints shape the terminal set since in the design the maximum invariant set for the LQR solution, which does not consider the constraints, is selected. Lastly, mpc_z and mpc_roll controllers do not have any state constraints; therefore, the terminal set shape is determined by the input constraints.

3.1.2 Parameter Tuning

Finally, for each one of the controllers, the horizon length influences the region of attraction; however, a larger horizon leads to a higher computational cost. Therefore, it is optimized by keeping the problem feasible and minimizing the time complexity of the algorithm. To do that the results are observed by changing the time horizon of the MPC controllers.

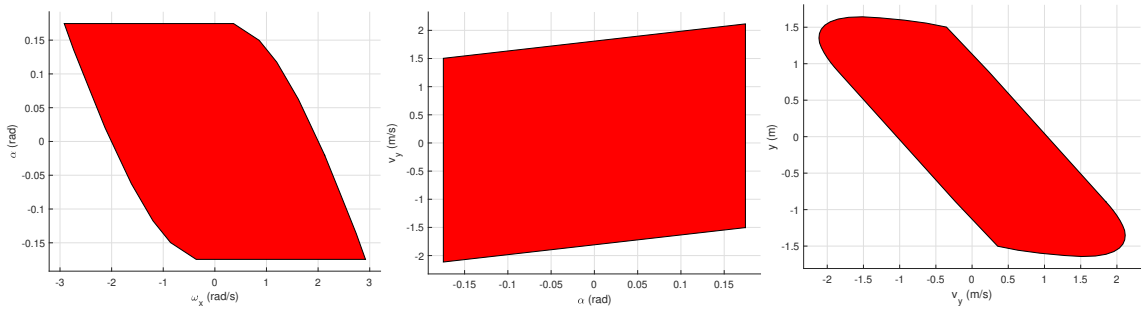
The input and state weights are manipulated according to the desired results. An intuitive interpretation would be that larger state weights lead to smaller settling time and larger input weights lead to minimizing effort while reaching the solution. However, large input weights lead to non-zero steady-state error for the systems since the problem is optimized based on the optimum input performance rather than satisfying final state requirements completely. For the weight matrices in all controllers, the state weights are increased to achieve the desired settling time. Also, the weights of the angular speeds are increased a bit to have smoother maneuvers. The input weight matrices are kept small. However, as one can observe from the Figures 5 and 7, this causes abrupt changes in the inputs. In this case, we did not try to avoid it since we thought servo motors would be fast enough to respond. However, if this was not the case, one could avoid these abrupt changes in the inputs by increasing the input weights.

Open and closed loop plots can be seen as follows. It can be observed that the open loop and the closed loop results are almost the same since there are no disturbances in the simulation.



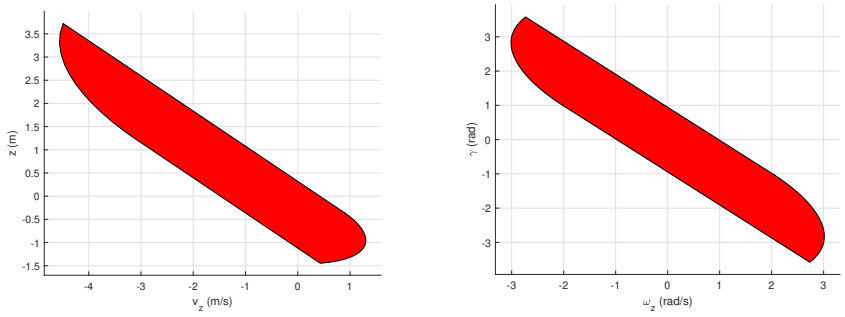
(a) $\mathcal{X}_f(\omega_y, \beta)$ (b) $\mathcal{X}_f(\beta, v_x)$ (c) $\mathcal{X}_f(v_x, x)$

Figure 1: Terminal invariant set for x dimension



(a) $\mathcal{X}_f(\omega_x, \alpha)$ (b) $\mathcal{X}_f(\alpha, v_y)$ (c) $\mathcal{X}_f(v_y, y)$

Figure 2: Terminal invariant set for y dimension



(a) $\mathcal{X}_f(v_z, z)$ (b) $\mathcal{X}_f(\omega_z, \gamma)$

Figure 3: Terminal invariant set for z dimension and roll

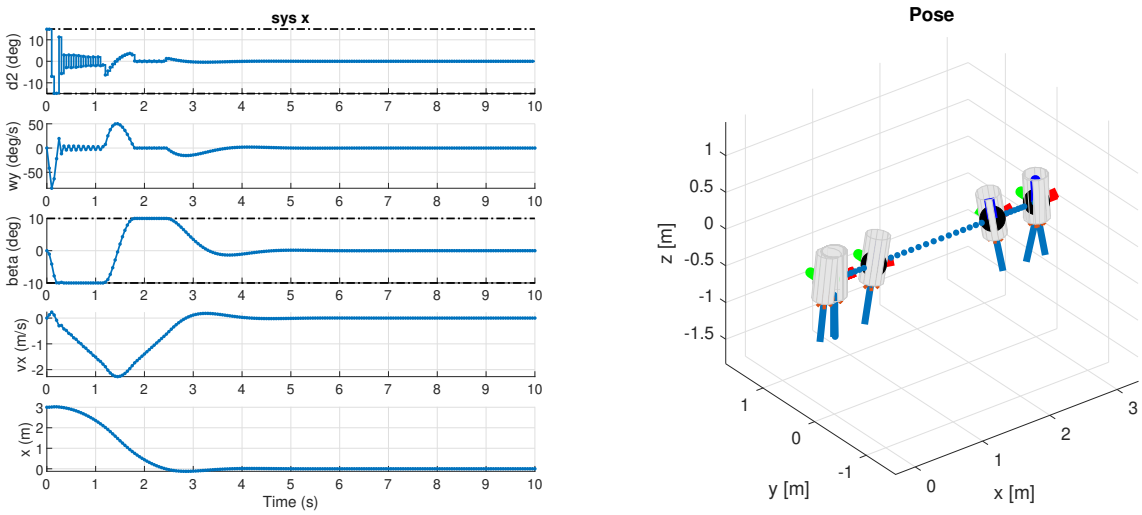


Figure 4: Open loop behaviour of the subsystem for x dimension starting stationary at 3 meters from the origin

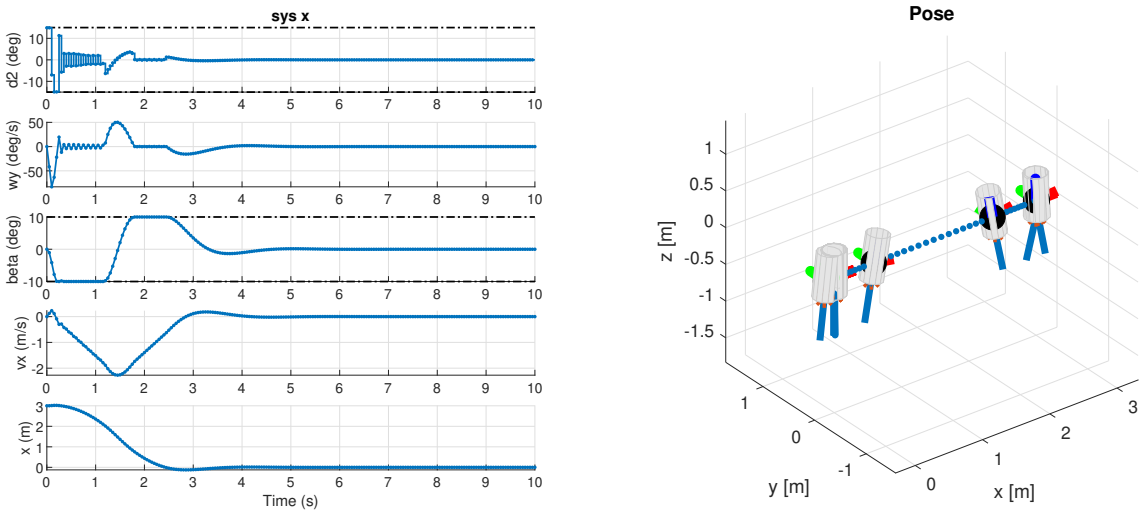


Figure 5: Closed loop behaviour of the subsystem for x dimension starting stationary at 3 meters from the origin

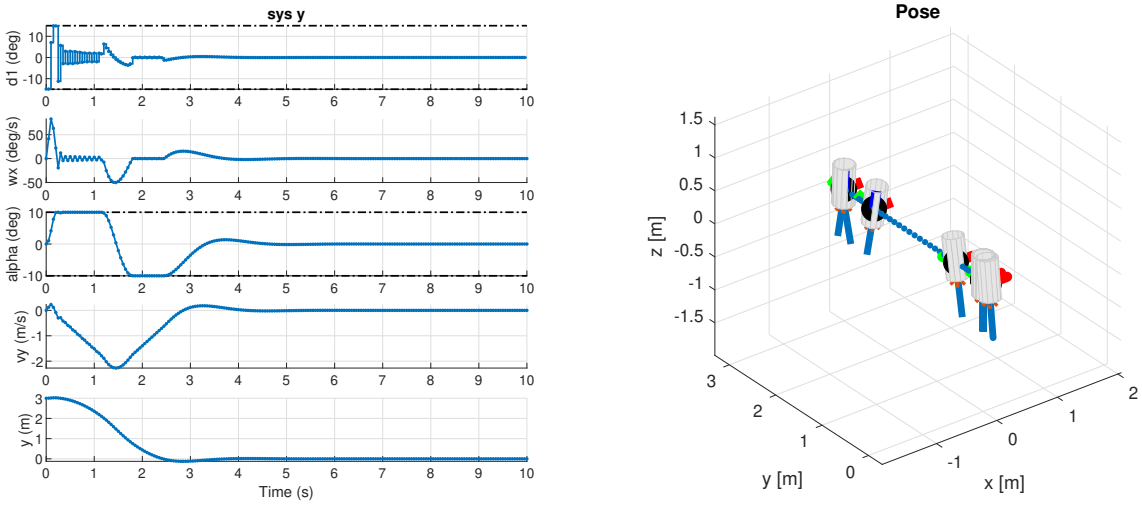


Figure 6: Open loop behaviour of the subsystem for y dimension starting stationary at 3 meters from the origin

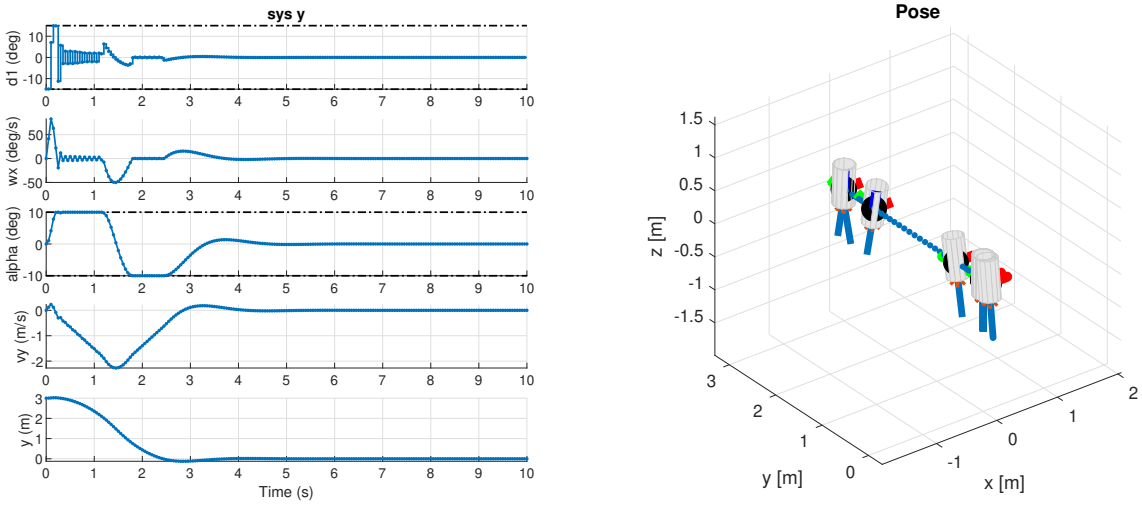


Figure 7: Closed loop behaviour of the subsystem for y dimension starting stationary at 3 meters from the origin

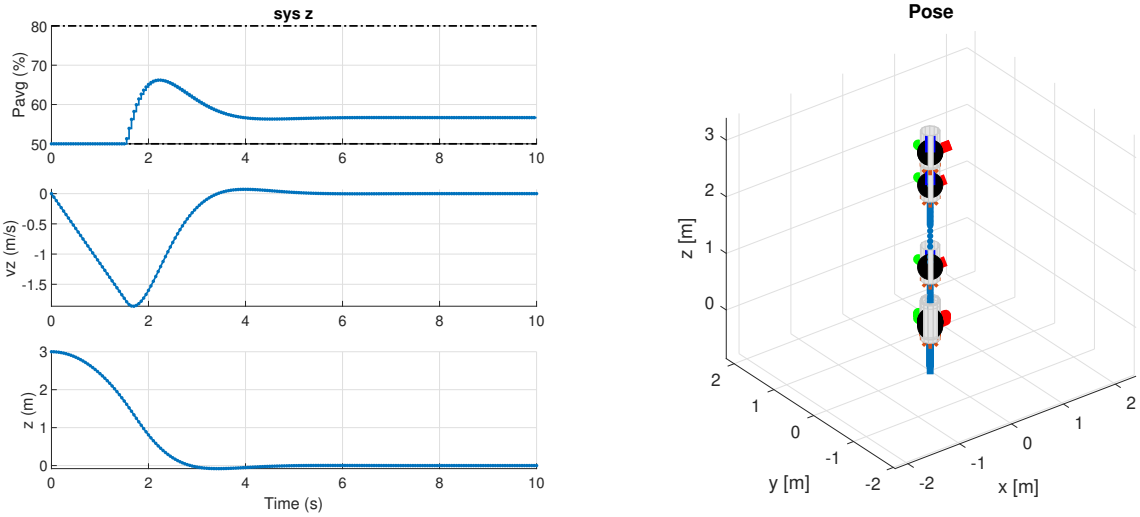


Figure 8: Open loop behaviour of the subsystem for z dimension starting stationary at 3 meters from the origin

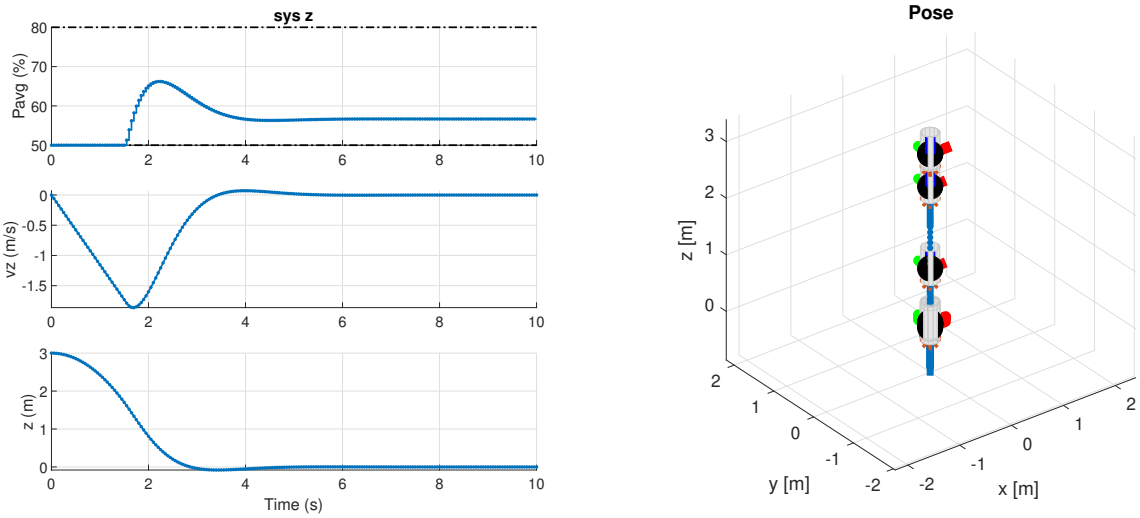


Figure 9: Closed loop behaviour of the subsystem for z dimension starting stationary at 3 meters from the origin

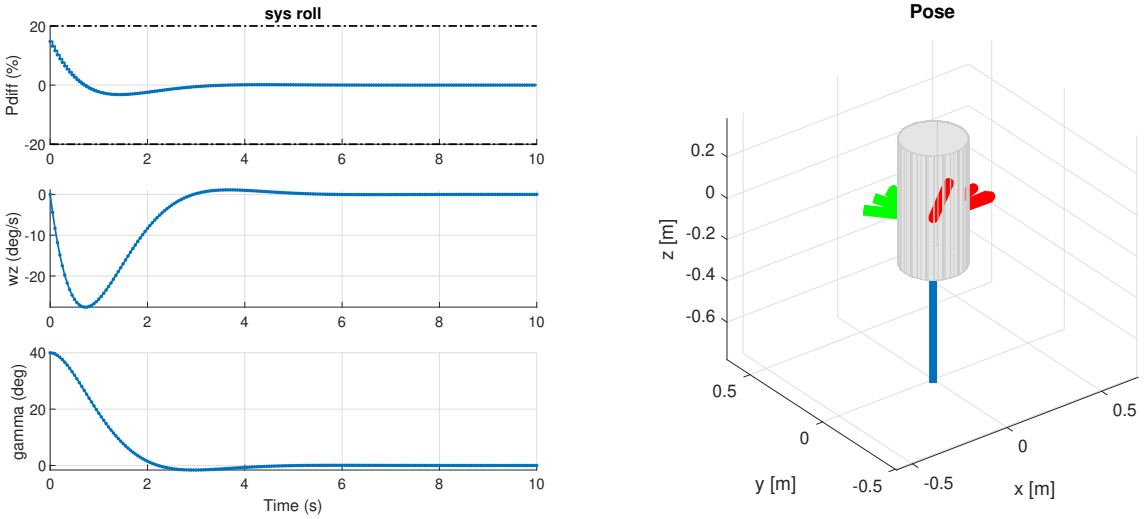


Figure 10: Open loop behaviour of the subsystem for roll starting stationary at 30°

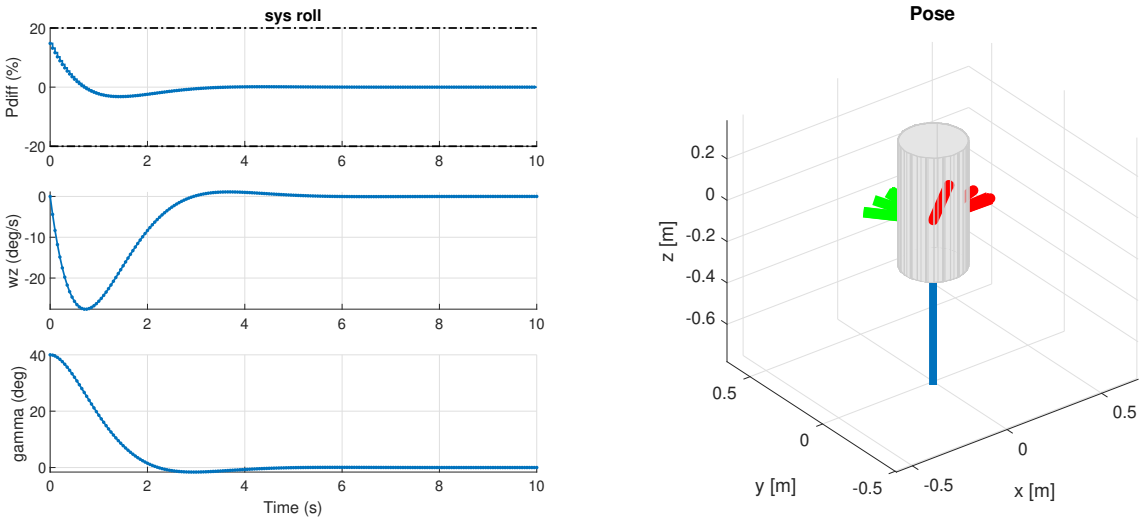


Figure 11: Closed loop behaviour of the subsystem for roll starting stationary at 30°

Deliverable 3.2

3.2.1 Steady-State Target Problem

Firstly, a steady-state target is calculated in the presence of input and state constraints. Here, an optimization problem is defined as follows.

$$
\begin{aligned}
& min_{u_s} u_s^T R u_s \\
s.t. \quad & x_s = A x_s + B u_s \\
& r = C x_s \\
& H_x x_s \leq k_x \\
& H_u u_s \leq k_u
\end{aligned}
$$

Solving the given optimization problem returns the steady state and the input needed to remain at that state. After computing the steady-state target, we can now formulate the MPC.

3.2.2 Delta-Formulation for Tracking

The only difference between the reference tracking part and the previous MPC problem we have solved is that now we are shifting the states and inputs to the calculated target by subtracting them in the calculation. Also, rather than dropping the terminal set, we have implemented a shifted version of the terminal set based on the steady-state target. Then, the controllers are modified and a delta formulation for the problem is written to track a given reference signal.

$$
\begin{aligned}
& min \big[(x_N - x_s)^T Q_f (x_N - x_s) + \sum_{k=0}^{N-1} (x_k - u_s)^T Q (x_k - x_s) + (u_k - u_s)^T R (u_k - u_s) \big] \\
& s.t. \quad (x^+ - x_s) = A(x - x_s) + B(u - u_s) \\
& F x \leq f \\
& M x \leq m \\
& x \in \chi_f + x_s
\end{aligned} \tag{3}
$$

Tuned parameters from the previous section are kept the same since the problem is similar and the requirements for the states and the control inputs are the same. Therefore, tuned parameters from the previous section are used while plotting the results.

Solving the given MPC problem leads to the following results. It can be observed that the open loop and the closed loop results are almost the same since there are no disturbances in the simulation.

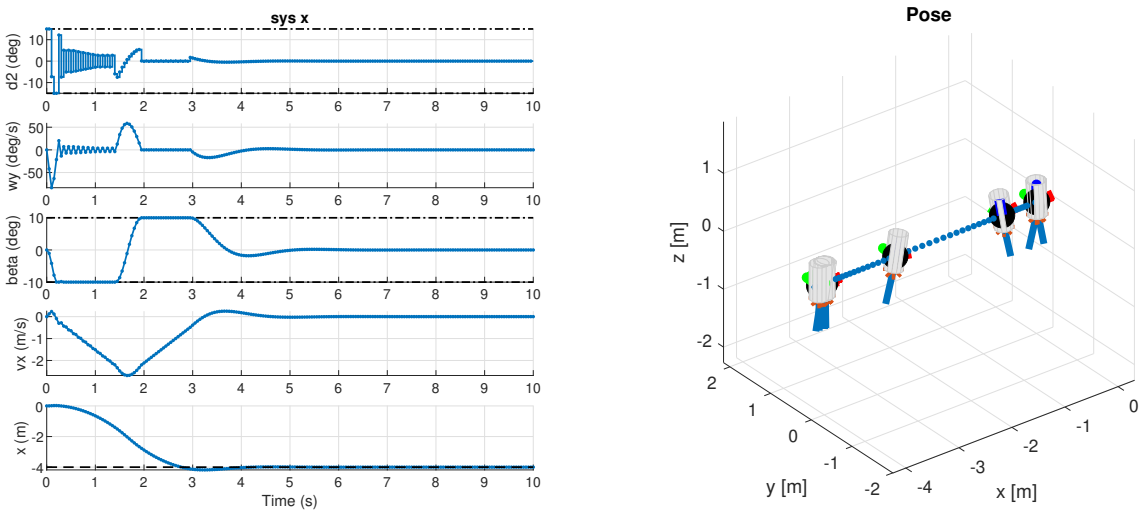


Figure 12: Open loop behaviour of the subsystem for x dimension starting stationary at the origin and tracking a reference to -4 meters

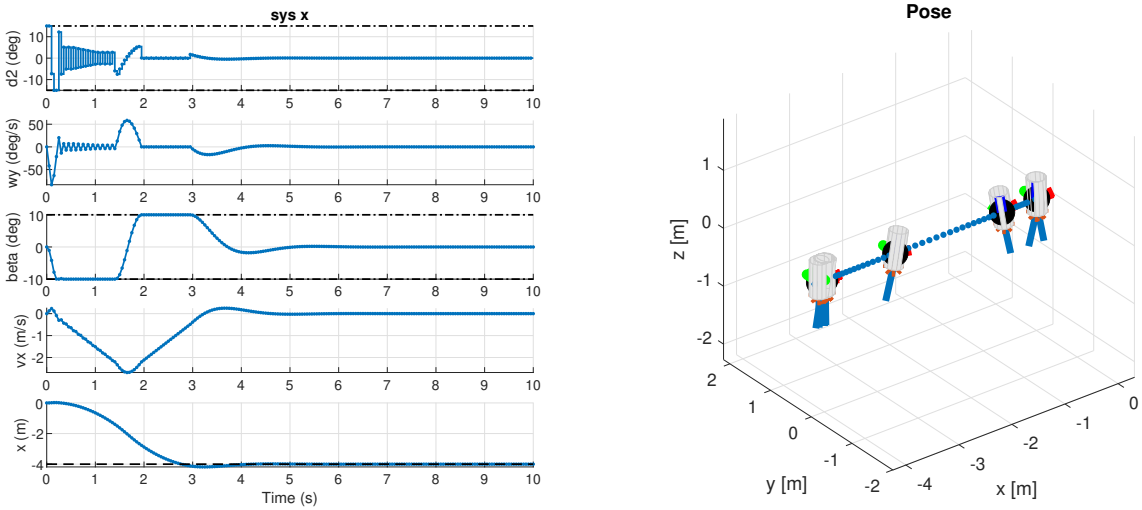


Figure 13: Closed loop behaviour of the subsystem for x dimension starting stationary at the origin and tracking a reference to -4 meters

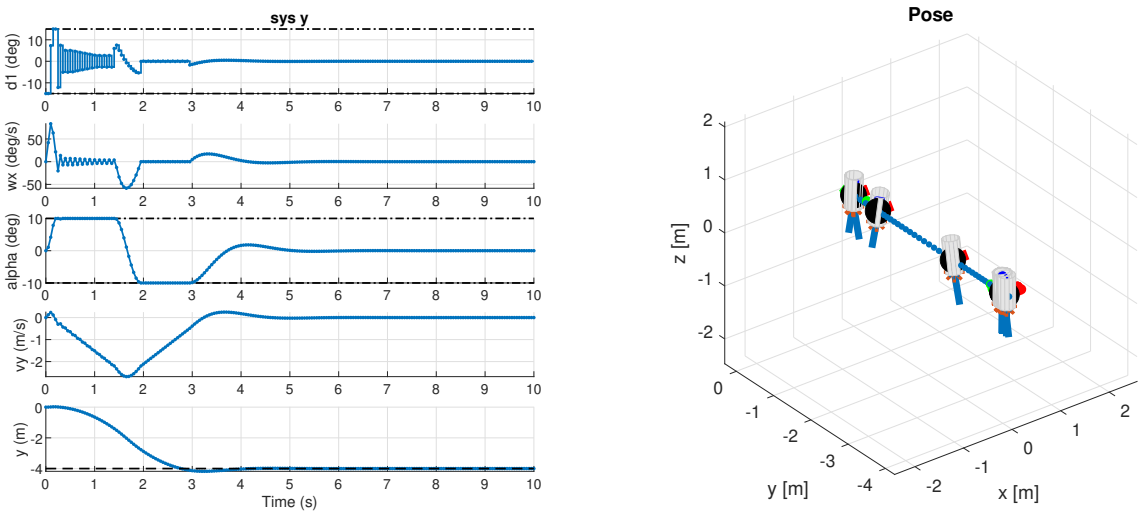


Figure 14: Open loop behaviour of the subsystem for y dimension starting stationary at the origin and tracking a reference to -4 meters

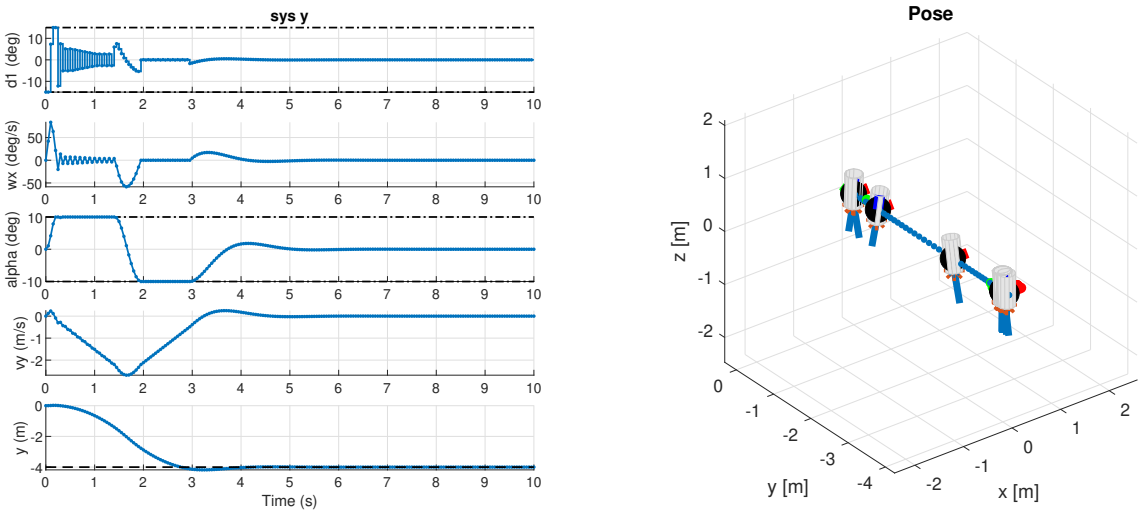


Figure 15: Closed loop behaviour of the subsystem for y dimension starting stationary at the origin and tracking a reference to -4 meters

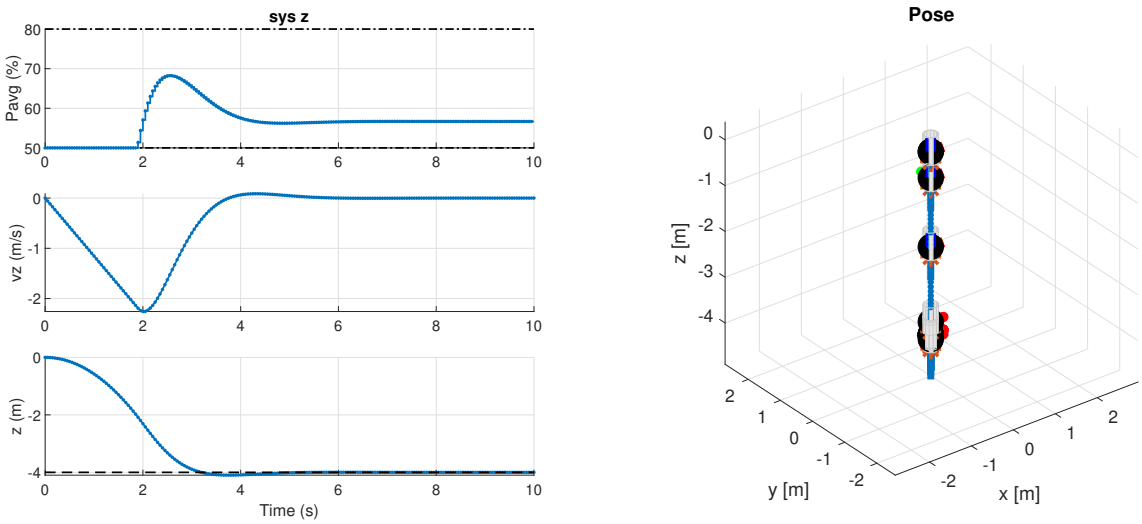


Figure 16: Open loop behaviour of the subsystem for z dimension starting stationary at the origin and tracking a reference to -4 meters

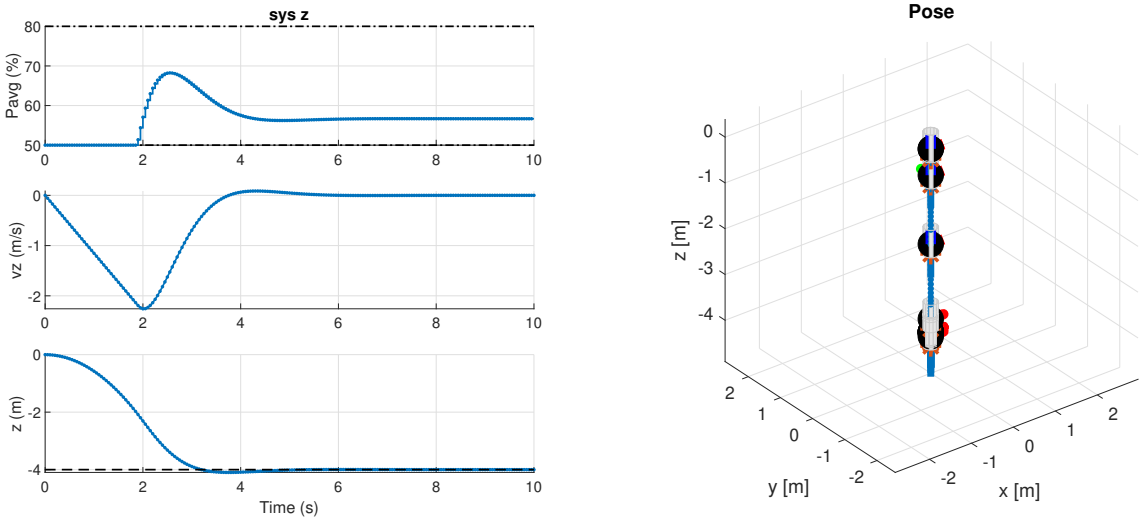


Figure 17: Closed loop behaviour of the subsystem for z dimension starting stationary at the origin and tracking a reference to -4 meters

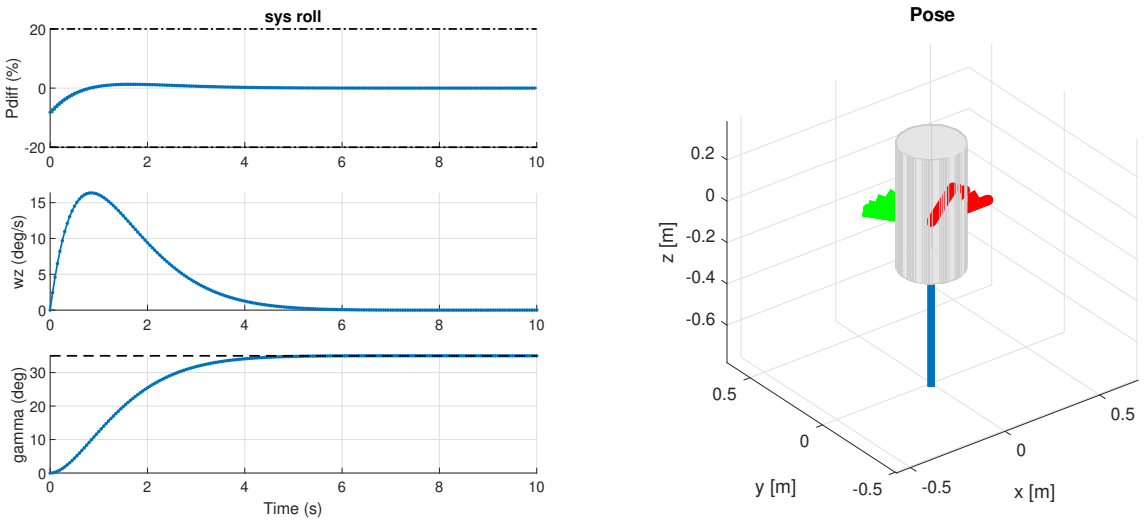


Figure 18: Open loop behaviour of the subsystem for roll starting stationary at the origin and tracking a reference to 35°

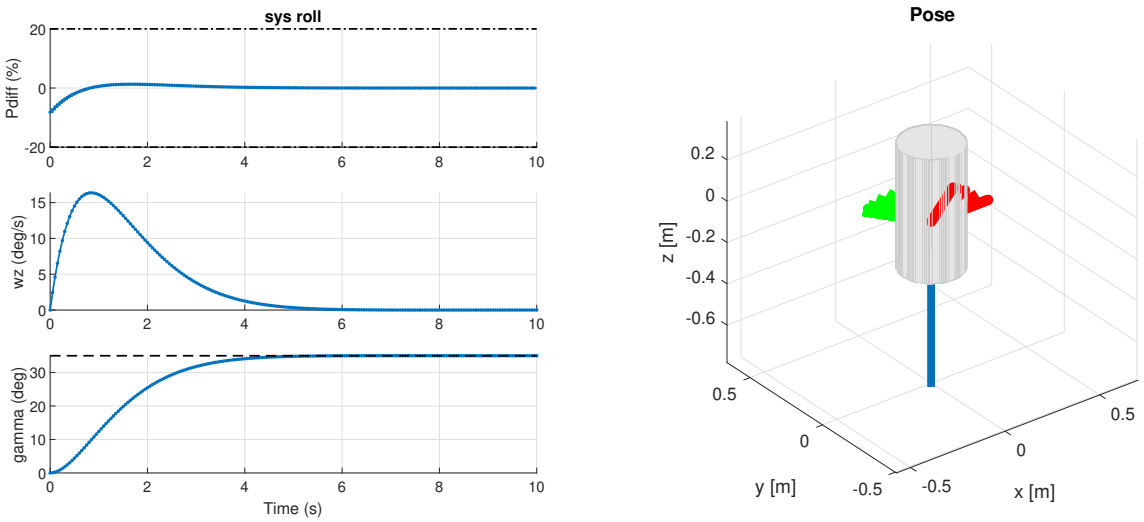


Figure 19: Closed loop behaviour of the subsystem for roll starting stationary at the origin and tracking a reference to 35°

Part 4 Simulation with Nonlinear Rocket

Deliverable 4.1

In the beginning, the tracking performance of the controller mpc_z was not good enough. Therefore, we have changed the input and state weights to achieve a better tracking performance. Firstly, we have increased the state weights and reduced the input weights a bit. This created some abrupt changes in the input; however, we expect it to happen since there are sudden changes in the

directions in the reference path.

We attempted to enhance the controller's aggressiveness by further amplifying the weights associated with position variables. However, this adjustment led to the optimization problem becoming infeasible beyond a certain time horizon. This challenge arises from the inherent limitations of the rocket's maneuverability in the linear approximation, preventing it from executing extremely rapid movements within the constraints imposed by its narrow state boundaries. We tried to work around this issue by increasing the time horizon, yet it increased the computation time to a considerable extent, and it was not dramatically better. It can also be observed in the previous parts that increasing the weights of the position variables only causes a small overshoot. In this case, this overshooting caused significant problems; therefore, we increased the weights associated with the speed states to avoid overshooting.

Our cost matrices are as follows:

$$Q_x = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 20 \end{bmatrix}, Q_y = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 20 \end{bmatrix}, Q_z \begin{bmatrix} 1 & 0 \\ 0 & 20 \end{bmatrix}, Q_{roll} \begin{bmatrix} 1 & 0 \\ 0 & 60 \end{bmatrix} \tag{4}$$

$$R_x = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}, \quad R_y = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}, \quad R_z = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}, \quad R_{roll} = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix} \tag{5}$$

The resulting plots are as follows.

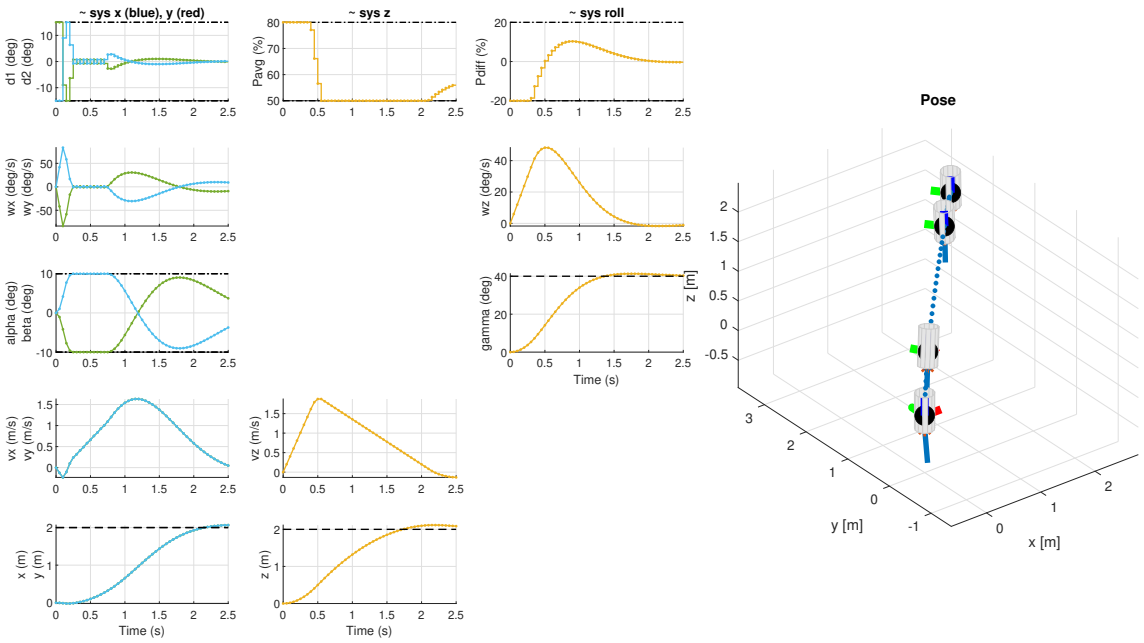


Figure 20: Closed loop behaviour of the nonlinear system with the combined linear MPC tracking the reference path $\begin{bmatrix} x_{ref} \\ y_{ref} \\ z_{ref} \end{bmatrix} = \begin{bmatrix} 2\text{m} \\ 2\text{m} \\ 2\text{m} \end{bmatrix}$ and reference roll angle $\gamma_{ref} = 40°$

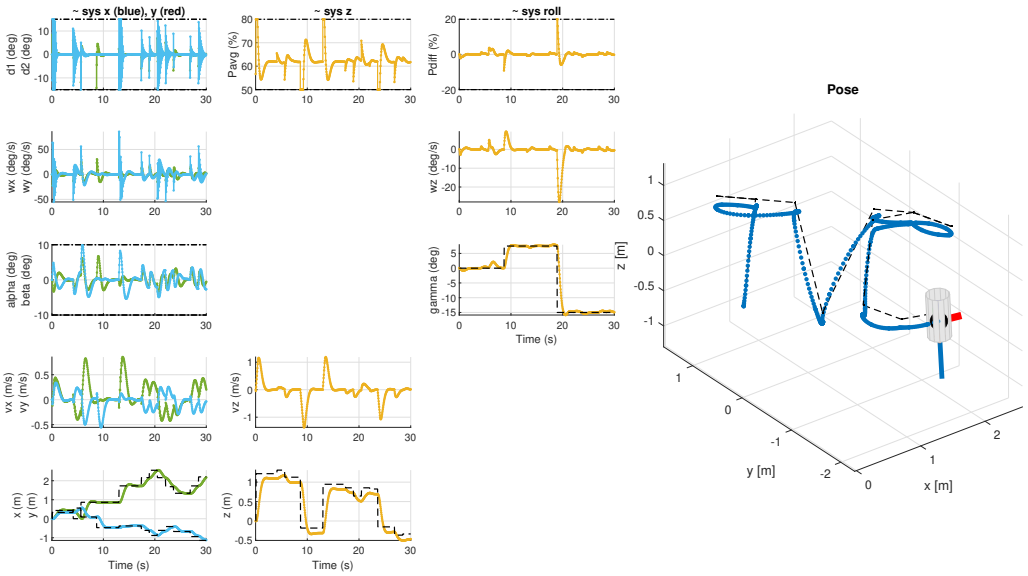


Figure 21: Closed loop behaviour of the nonlinear system with the combined linear MPC tracking the reference path and reference roll angle (TVC)

Part 5 Offset-Free Tracking

Deliverable 5.1

5.1.1 Design Procedure

Up to this point, we assumed that the system parameters are known and constant. However, in reality, this would never be true due to measurement errors and the usage of fuel over time. To simulate this uncertainty, we have tested our controller with $\approx 25\%$ higher initial mass (2.13 kg). Mass offset can be included in system dynamics as a disturbance in the z-direction.

$$x^+ = Ax + Bu + B_d d \tag{6}$$
$$d^+ = d \tag{7}$$
$$y = Cx + C_d d \tag{8}$$
$$C_d = 0, B_d = B \tag{9}$$

To reject the disturbance and make our controller offset-free, we have used a state estimator to estimate the state and the disturbance of the system. The dynamics of the state observer are expressed in an augmented model (10).

$$\begin{bmatrix} \hat{x}_{k+1} \\ \hat{d}_{k+1} \end{bmatrix} = \begin{bmatrix} A & Bd \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ \hat{d}_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k + \begin{bmatrix} L_x & L_d \end{bmatrix} (C\hat{x}_k + C_d\hat{d}_k - y_k) \tag{10}$$

To reject the disturbance we want the error of the state estimator to be stable. Error dynamics is

expressed in equation 11.

$$\begin{bmatrix} x_{k+1} - \hat{x}_{k+1} \\ d_{k+1} - \hat{d}_{k+1} \end{bmatrix} = \left(\begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} + \begin{bmatrix} L_x \\ L_d \end{bmatrix} \begin{bmatrix} C & C_d \end{bmatrix} \right) \begin{bmatrix} x_k - \hat{x}_k \\ d_k - \hat{d}_k \end{bmatrix} = (\bar{A} + L\bar{C}) \begin{bmatrix} x_k - \hat{x}_k \\ d_k - \hat{d}_k \end{bmatrix} \tag{11}$$

With the introduction of the disturbance, we also have a new condition for the selection of the steady-state point for tracking:

$$x_s = Ax_s + Bu_s + B_d d_s \tag{12}$$
$$y_s = Cx_s + C_d d_s = r \tag{13}$$

We have adapted the target selector accordingly with the new conditions on the steady-state point.

5.1.2 Choice of Tuning Parameters

To make the error dynamics stable, we chose the L_x and L_d such that the eigenvalues of the $\bar{A}+L\bar{C}$ are in the unit circle. We achieved this by utilizing the `place` command in MATLAB. We have observed that placing the eigenvalues closer to the origin results in a faster estimator, hence a more aggressive controller as expected. However, this aggressiveness might cause infeasibility of the problem as can be seen in 5.2.

In addition, we have modified the weight matrices. When there is a mass mismatch the rocket either needs a larger or a smaller throttle. Larger weight matrix for P_{avr} allows the controller to track a larger range of weight mismatches. Therefore, we have increased the input weight for mpc_z to keep the system feasible in a larger range of disturbances.

5.1.3 Controller Behaviour

Without the offset-free implementation, our controller could not achieve the reference value in the Z direction. This can be seen in the Figure 22. However, with the offset-free controller, the system can correct itself and reach the reference value in Z, which can be seen in Figure 23. In this case, an aggressive controller is selected; therefore, a very small displacement from the steady-state position can be observed in the Z direction. This is achieved by selected pole positions closer to the origin.

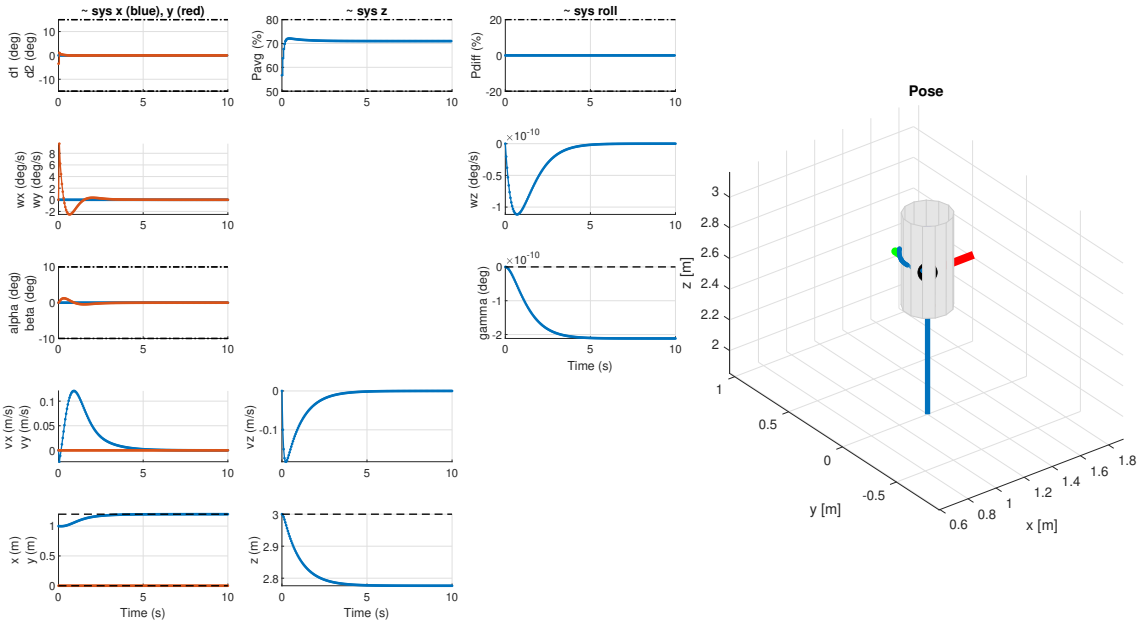


Figure 22: Closed loop behaviour of the nonlinear system with the combined linear MPC tracking the reference path and reference roll angle (TVC), with $\approx 25\%$ higher initial mass (2.13 kg) and without an offset-free tracking implementation

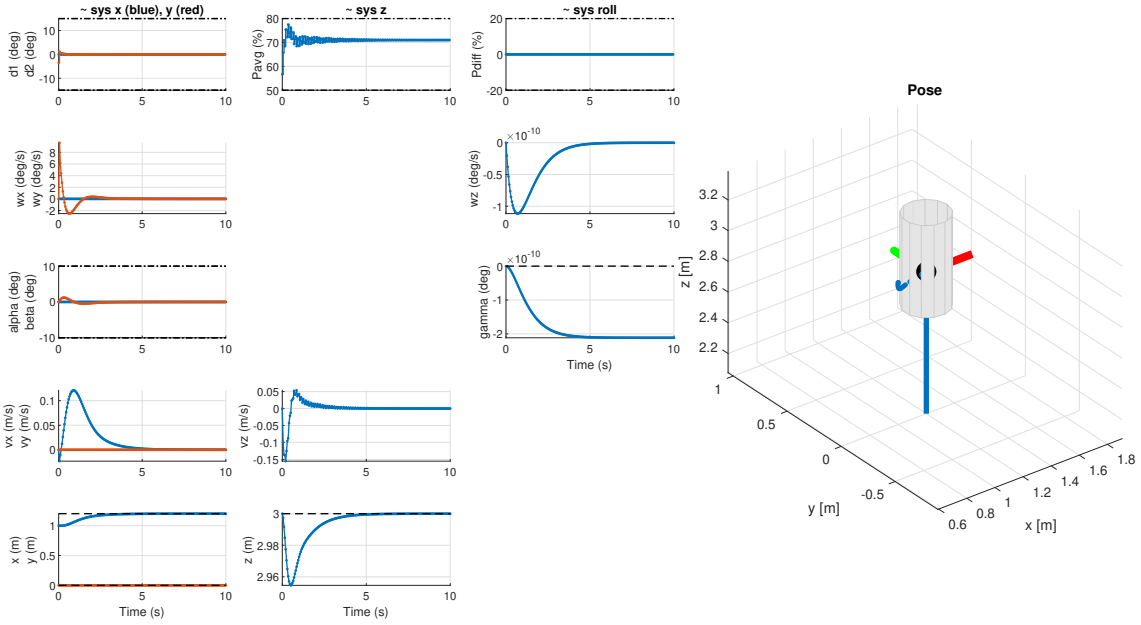


Figure 23: Closed loop behaviour of the nonlinear system with the combined linear MPC tracking the reference path and reference roll angle (TVC), with $\approx 25\%$ higher initial mass (2.13 kg) and with the offset-free tracking with poles of the estimator at 0.1, 0.2, and 0.3

Deliverable 5.2

5.2.1 Impact of Having a Thrust-Dependent Mass Decrease

The offset-free tracking controller couldn't correct for thrust-dependant mass decrease that can be seen in Figure 24.

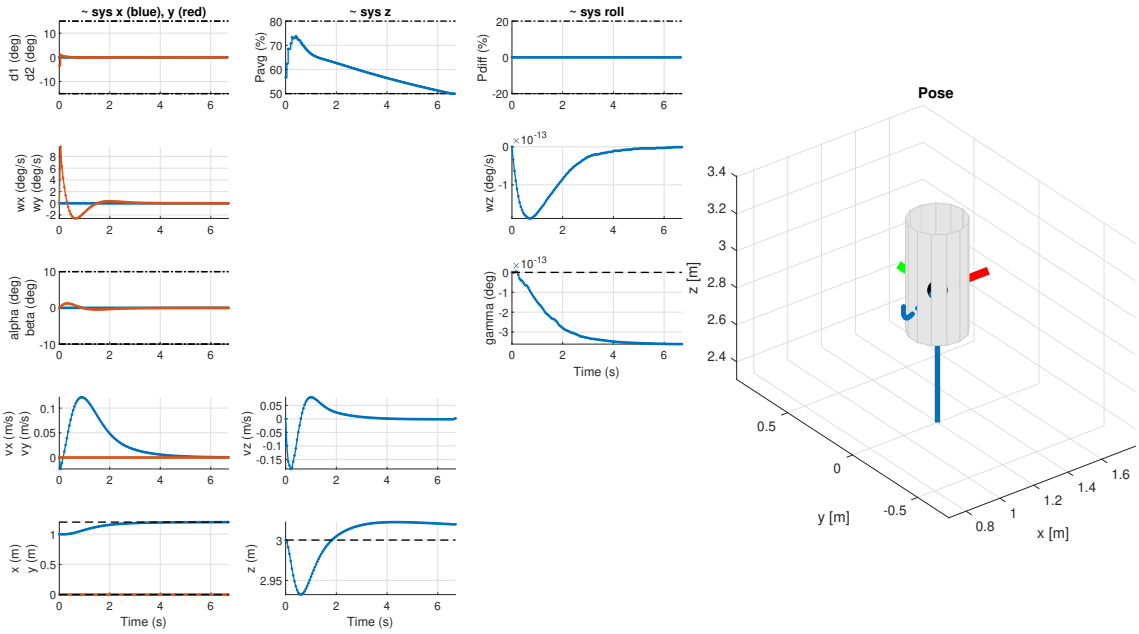


Figure 24: Closed loop behaviour of the nonlinear system with the combined linear MPC tracking the reference path and reference roll angle (TVC), with ≈ 25% higher initial mass (2.13 kg) and mass rate of -0.27 kg/s with the offset-free tracking with poles of the estimator at 0.1, 0.2, and 0.3

5.2.2 Offset Free Tracking for Changing Mass

The initial offset is caused by the initial input applied to the system. Initially, the input is applied according to the steady-state input at the linearization of the system. Therefore, till the system compensates for the disturbance in the mass, the rocket falls a bit from the desired position.

To have an offset-free control, one can include a model for the disturbance according to the applied input. To have an adaptive estimator for the disturbance, a Kalman filter might be integrated. Also, estimator gain can be modified to better compensate for the disturbances. The smaller the decay rate; that is, the closer the eigenvalues are to the origin, the less overshoot we have in the system. Therefore, when the mass of the system is lower than the model, it is better to use larger eigenvalues that have a slower response.

5.2.3 Distinct Behaviours Along the Simulation

After some time in the simulation, we can observe that the input is limited by the previous throttle limit. This is because the lower limit of the input is defined according to the 1.7kg model to prevent the rocket from falling too fast. However, in this case, when the mass of the model decreases too much, we can observe that the lowest applicable input lifts the rocket but we cannot apply lower

inputs due to previously defined physical limits of the system that prevent the rocket of 1.7kg from falling too fast.

5.2.4 Even Longer Simulation Behaviours

Increasing the simulation time to 20 seconds, we can observe that the rocket falls after some time since the fuel runs out. After fuel runs out, the MPC solution gives the maximum possible input to the system but there is no thrust and the rocket falls. Additionally, the problem is only feasible with suitable eigenvalue selections since the smaller eigenvalues cannot compensate for the larger mass displacement when the mass of the rocket is lower than modeled. Therefore, one has to select larger eigenvalues that satisfy the stability to stay in the feasible region throughout the simulation. In our case, we have selected 0.97, 0.98, and 0.99. A more aggressive controller leads to an infeasibility in the problem since it creates an overshoot and the input constraints cannot be satisfied for the linearized problem.

Part 6 Nonlinear MPC

Deliverable 6.1

6.1.1 Problem Definition and Design Choices

To better adapt to the nonlinear dynamics of the system, we have also designed a nonlinear MPC. In the NMPC case, the problem is formulated as follows.

$$\begin{aligned} \min J = \Bigg[(\mathbf{x}_N - \mathbf{x}_s)^T Q_f (\mathbf{x}_N - \mathbf{x}_s) + \sum_{k=0}^{N-1} (\mathbf{x}_k - \mathbf{u}_s)^T Q (\mathbf{x}_k - \mathbf{x}_s) + (\mathbf{u}_k - \mathbf{u}_s)^T R (\mathbf{u}_k - \mathbf{u}_s) \Bigg] \\ \text{s.t.} \quad (\mathbf{\Delta x}^+) = \mathbf{\Delta x} + f_{discrete}(\Delta x, \Delta u)\Delta t \\ F\mathbf{x} \leq \mathbf{f} \\ M\mathbf{x} \leq \mathbf{m} \\ \mathbf{x} \in \chi_f + \mathbf{x}_s \end{aligned} \tag{14}$$

In the nonlinear case, there is no need to constrain the angles α and β to small values since they are valid for all the state-space. However, since the system is singular when $\beta = 90°$, we should have a constraint of $|\beta| < 75°$.

To have better tracking performance we significantly increased the weights for the position states and the roll angle. The weights for input and states are as follows.

```
Q = diag([10 10 10 1 1 100 10 10 10 300 300 300]);
R = 0.01*eye(nu);
```

The performance of our controller can be seen in Figure 25. The input changes abruptly since the reference trajectory is hard for the robot. This can be solved by introducing velocity constraints for the inputs. Also, another solution may be increasing the simulation time.

6.1.2 Linear and Nonlinear MPC Comparison

The main advantage of using the NPMC over linear MPC is that it covers the whole state space and we do not need to put constraints on the Euler angles due to linearization. Therefore, the system

is free to do more agile movements which leads to better path following performance. In addition, with NMPC, the system is not divided into subparts; thus, the response is more realistic since the system dynamics are coupled. With the NMPC, we can increase the weights on the states much more than the linear case, hence we can achieve a much more aggressive controller but the coupled dynamics of the system lead to a more difficult tuning process. In the linear MPC, increasing the weights that much will result in infeasibility in the optimization since nimble motions are restricted by state constraints. Finally, NMPC is more adaptable to any system modifications since we do not need to divide the system into subparts and perform linearization for each of them.

6.1.3 Plots Showing the Performance of NMPC

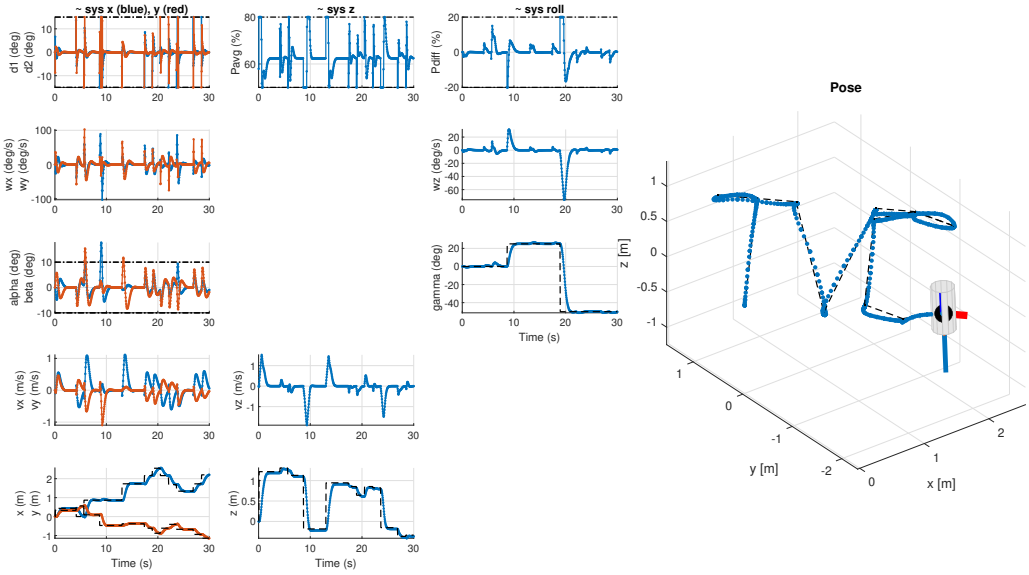


Figure 25: Closed loop behaviour of the nonlinear system with nonlinear MPC implemented using CASADI, tracking the reference path and reference roll angle (TVC)

Deliverable 6.2

6.2.1 Observations on The Closed-Loop Performance

The performance of the controller depends on how aggressive the design is. A slower controller better compensates for the delay. It can be observed that the performance of the controller starts to decrease with a delay of 25ms. At a delay greater than or equal to 25ms the inputs to the system oscillate and there are abrupt changes. However, after increasing the delay to 50ms the system becomes unstable, and even though the initial behaviour is similar to what is seen for delays 50m-100ms, after some time the system becomes completely unstable. Therefore, an uncompensated delay of 50ms causes instability in the system.

6.2.2 Plots for Delay Compensated NMPC

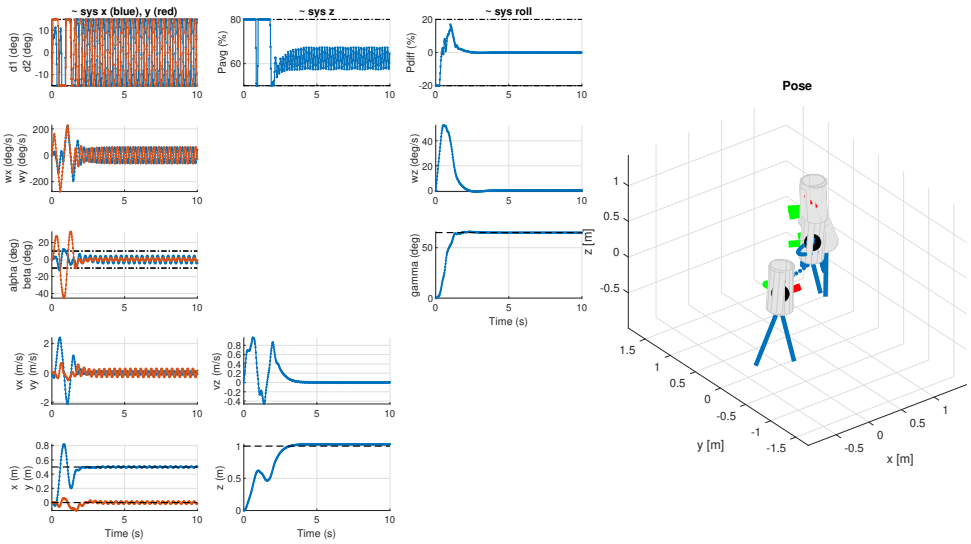


Figure 26: Closed loop behaviour of the nonlinear system with nonlinear MPC tracking the reference path $\begin{bmatrix} x_{ref} \\ y_{ref} \\ z_{ref} \end{bmatrix} = \begin{bmatrix} 0.5\text{m} \\ 0\text{m} \\ 1\text{m} \end{bmatrix}$ and reference roll angle $\gamma_{ref} = 65°$. The system has a 125 ms delay (delay = 5) where the expected delay is 100 ms.

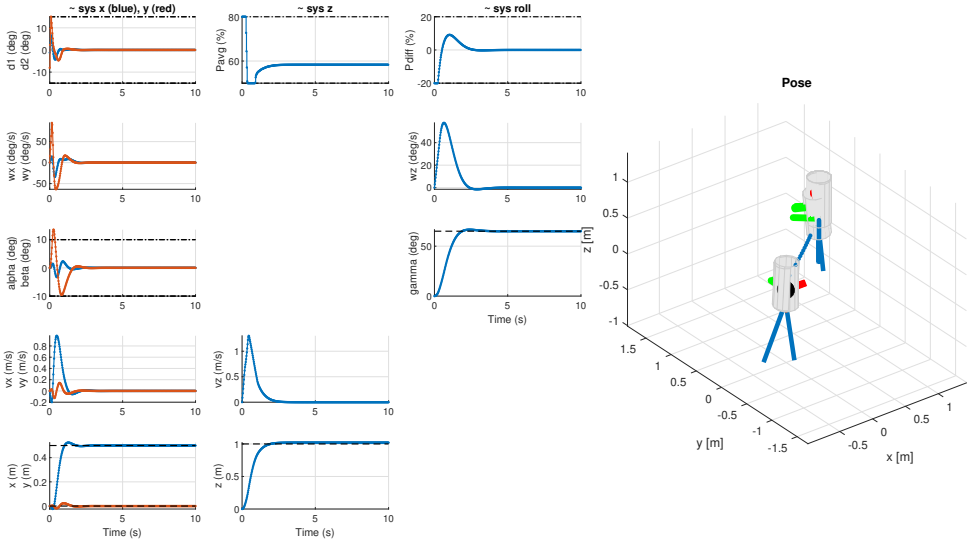


Figure 27: Closed loop behaviour of the nonlinear system with nonlinear MPC tracking the reference path $\begin{bmatrix} x_{ref} \\ y_{ref} \\ z_{ref} \end{bmatrix} = \begin{bmatrix} 0.5\text{m} \\ 0\text{m} \\ 1\text{m} \end{bmatrix}$ and reference roll angle $\gamma_{ref} = 65°$. The system has a 125 ms delay (delay = 5) which is fully compensated by the delay compensation implementation.