Ecole Polytechnique Fédérale de Lausanne

Model Predictive Control
Mini-Project
Fall 2023

Group BM

Leonie Gasser 315151
Diana Bejan 325029
Simona Herren 295810

Professor Colin Jones

Contents

1 Introduction

For our mini-project in the Model Predictive Control course given by Prof. Colin Jones, we are asked to create different controllers for a rocket governed by two servo motors. The system is non-linear. However, we look at a linearized model in Deliverables 2, 3 and 4. In Deliverable 5, the same linearized model is used to add offset-free tracking. Deliverable 6 is the final control that works with the original nonlinear system.

2 Linearization

2.1 Deliverable 2.1 - Division in four sub-systems

From an intuitive point of view, when the rocket is not tilted and no angular velocities are given, we understand that lateral, vertical and horizontal movements do not affect each other. Indeed, by changing the lateral position, the vertical and horizontal positions stay the same, but are laterally displaced. Moreover, turning the rocket along its axis does not influence the other positions as they are taken from the world reference, not the body reference. We thus understand that we can control these four elements separately.

Knowing that the matrix C is an identity matrix, we can confirm our intuition by looking at the linearized matrices A and B:

$$A = \begin{array}{c} \\ wx \\ wy \\ wz \\ \alpha \\ \beta \\ \gamma \\ vx \\ vy \\ vz \\ x \\ y \\ z \end{array} \begin{array}{c} \begin{array}{cccccccccccc} wx & wy & wz & \alpha & \beta & \gamma & vx & vy & vz & x & y & z \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 9.81 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -9.81 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$B = \begin{array}{c} \\ wx \\ wy \\ wz \\ \alpha \\ \beta \\ \gamma \\ vx \\ vy \\ vz \\ x \\ y \\ z \end{array} \begin{array}{c} \begin{array}{cccc} \delta_1 & \delta_2 & Pavg & Pdiff \end{array} \\ \begin{pmatrix} -55.68 & 0 & 0 & 0 \\ 0 & -55.68 & 0 & 0 \\ 0 & 0 & 0 & -0.104 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 9.81 & 0 & 0 \\ -9.81 & 0 & 0 & 0 \\ 0 & 0 & 0.173 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k)$$

- δ1 affects wx and vy, who in turn affect alpha and y (and vy again)
- δ2 affects wy and vx, who in turn affect beta and x (and vx again)
- Pavg affects vz, who in turn affects z
- Pdiff affects wz, who in turn affects γ

As none of these four systems overlap, the separation into four sub-systems is valid.

3 Design of the sub-systems

3.1 Deliverable 3.1

3.1.1 Design procedure

We designed four finite horizon MPC controllers with terminal sets. The terminal invariant sets are used to ensure recursive feasibility: once in the set, we always stay in the set. Thus we are "safe" for a longer horizon than the one we base the controllers on. We also have a cost function that needs to be minimised for all sets. Between the first and the N-1st step we use the costs Q and R, while the last step uses the terminal cost Q_f. These costs penalise the distance from the actual values to the desired values. We wish to reach the origin in less than 7 seconds when starting stationary at 3 meters from the origin (for x,y and z) or stationary at 40° roll.
The cost function looks as follows:

$$\begin{aligned} \min_{u} \quad & \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i + x_N^T Q_f x_N \\ \text{s.t.} \quad & x_i \in \mathbb{X}, i \in 1,2,...,N-1 \\ & x_N \in \mathbb{X}_f \\ & u_i \in \mathbb{U}, i \in 1,2,...,N-1 \\ & x_{i+1} = Ax_i + Bu_i \end{aligned} \tag{1}$$

Table 1 gives an overview of the inputs, states and constraints of each sub-system.

Sub-system	Inputs	States	Constraints
sys x	$u = \delta_2$	$x = [\omega_y, \beta, v_x, x]$	$abs(\beta) \leq 10$
sys y	$u = \delta_1$	$x = [\omega_x, \alpha, v_y, y]$	$abs(\alpha) \leq 10$
sys z	$u = P_{avg}$	$x = [v_z, z]$	$50\% \leq P_{avg} \leq 80\%$
sys roll	$u = P_{diff}$	$x = [\omega_z, \gamma]$	$-20\% \leq P_{avg} \leq 50\%$

Table 1: Inputs, states and constraints of each sub-system

3.1.2 Tuning of the parameters

Q and R
To tune Q and R, we use identity matrices multiplied by weights $\overline{Q}$ and $\overline{R}$. These weights are chosen experimentally to meet the requirements of the problem.

$$Q = \overline{Q}. * eye(nx) \quad R = \overline{R}. * eye(nu)$$

Roll controller:
In the case of the roll controller, $\overline{R}$ is less than 1 to make the control action as aggressive as possible. A $\overline{R} = 0.1$ yields similar results but takes much longer to compute. This is why we chose $\overline{R} = 0.2$. $\overline{Q}$ is high, with a value of 200. This strongly penalizes the distance between the current states and the origin. This makes the rocket reach the origin as fast as possible. A higher value of $\overline{Q}$ did not make the rocket reach the origin faster. We therefore stuck to $\overline{Q} = 200$.

$$\overline{\mathbf{Q}}_{\mathbf{roll}} = \begin{bmatrix} 200 & 0 \\ 0 & 200 \end{bmatrix} \quad \overline{\mathbf{R}}_{\mathbf{roll}} = 0.2$$

X and y controller:
Setting both $\overline{Q}$ and $\overline{R}$ to the identity matrix results in significant oscillations during the first couple of seconds. To reduce the oscillations, $\overline{Q}$ is kept equal to the identity, but $\overline{R}$ is increased to 150. Ideally,

to get a smooth closed-loop plot, $\overline{R} = 150$. However, this significantly increases the run time of the program. Reducing $\overline{Q}$ has the same effect.

$$\overline{\mathbf{Q}}_\mathbf{x} = \overline{\mathbf{Q}}_\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \overline{\mathbf{R}}_\mathbf{x} = \overline{\mathbf{R}}_\mathbf{y} = 150$$

Z controller: Choosing the following weights was enough to satisfy all the problem requirements:

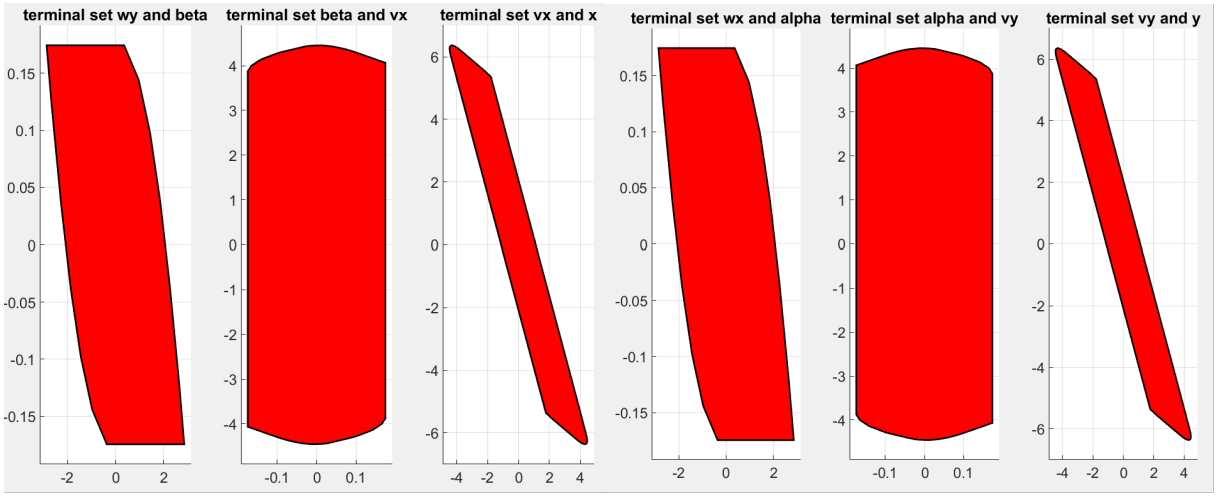
$$\overline{\mathbf{Q}}_\mathbf{z} = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \quad \overline{\mathbf{R}}_\mathbf{z} = \mathbf{1}$$

Terminal components
The terminal components Ff, ff, Q_f and the terminal invariant set were calculated using LQR computed by the Multi-Parametric Toolbox. We noticed that this is a bit faster than using the dlqr function.

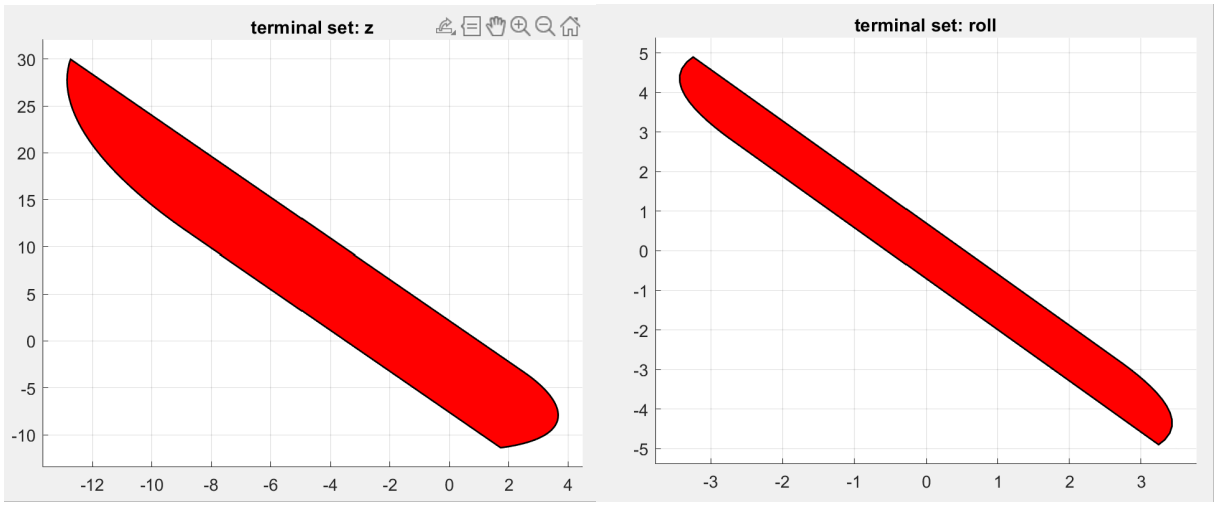
Horizon length
The horizon length H is given in seconds rather than steps N. This value is converted to steps in the creation of each controller $N = H/T_s$. As the settling time must be no more than 7s when starting 3m from the origin (for x, y, z) or stationary at 40° for roll, H must be at least 7 seconds. We noticed that this value is too low and yields an unsatisfactory response. In general, a short horizon can lead to a loss of stability. We decided to choose H such that the open- and closed-loop plots look similar, and have ended up using H=30 seconds.

3.1.3 Plots Deliverable 3.1

Terminal sets



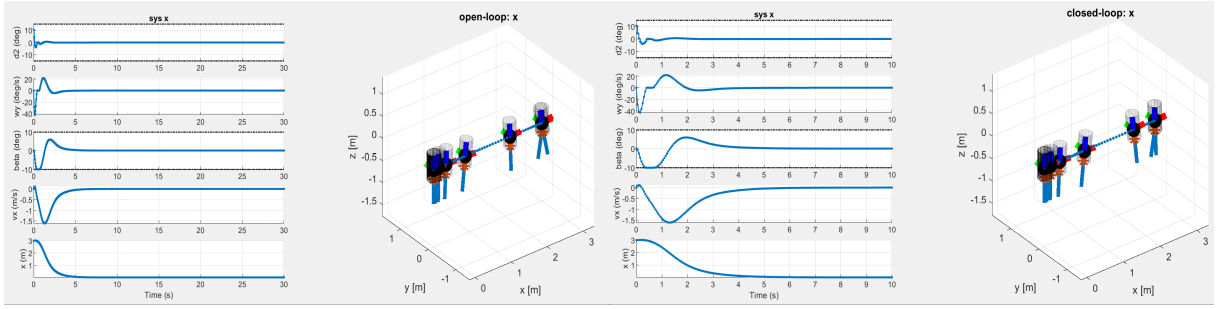
(a) X (b) Y



(c) Z (d) Roll

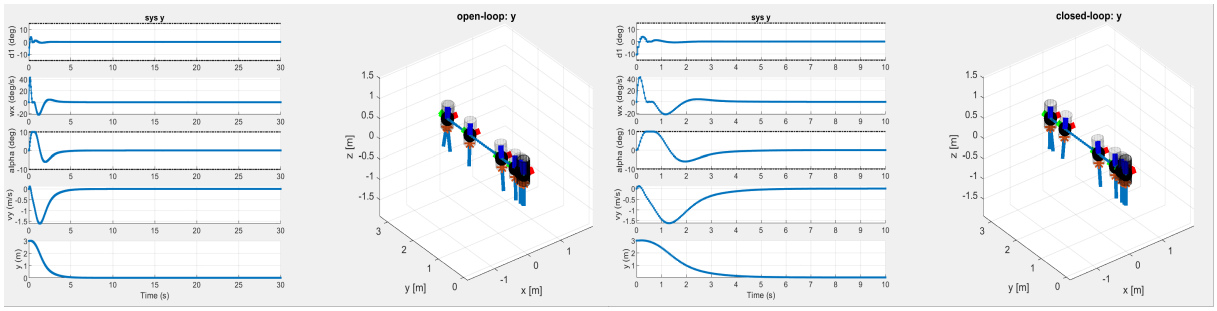
Figure 1: Terminal set plots

Open-loop and closed-loop plots for each sub-system



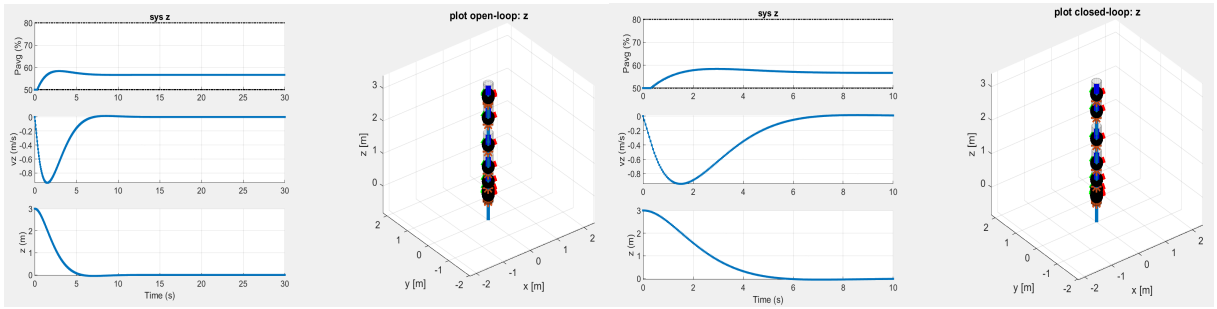
(a) Open-loop (b) Closed-loop

Figure 2: X controller plots



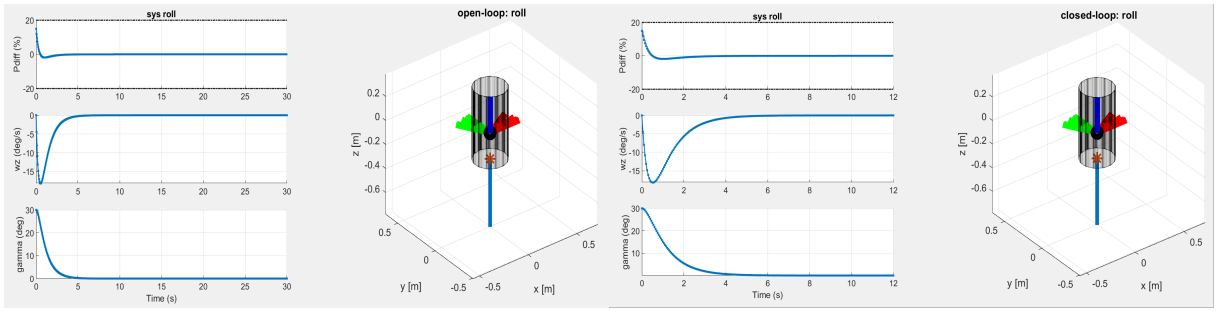
(a) Open-loop (b) Closed-loop

Figure 3: Y controller plots



(a) Open-loop (b) Closed-loop

Figure 4: Z controller plots



(a) Open-loop (b) Closed-loop

Figure 5: Roll controller plots

3.2 Deliverable 3.2 - Reference tracking

For this section, we need to track a reference. To do so, we adapt our references to the linearisation point *xs*, *us*. The function *setup_steady_state_target(mpc)* has been completed to find the adapted reference points.
Because dim(C) = [1,∼], Q is a 1x1 matrix. Thus, minimizing $(Cx_s - r)'Q_s(Cx_s - r)$ is the same as minimizing $(Cx_s - r)$. We therefore defined $Q = eye(1)$.

The *steady-state target (xs, us)* has to satisfy both state and input constraints. The set point closest to the reference r is calculated using the following equations:

$$
\begin{aligned}
&\min (Cx_s - r)' Q_s (Cx_s - r) \\
\text{s.t.:} \quad &x_y = Ax_s + Bu_x \\
&H_x \cdot x_s \leq k_x \\
&H_u \cdot u_s \leq k_u
\end{aligned}
$$

The function *setup_controller(mpc, Ts, H)* has been changed: Instead of penalizing the distance to the origin, it penalizes the distance to the reference points given. We mostly keep the same tuning parameters as for the previous step as the performance was deemed satisfactory. To reduce the oscillation at the start of system x, we use $\mathbf{\overline{R}_z = 300}$.

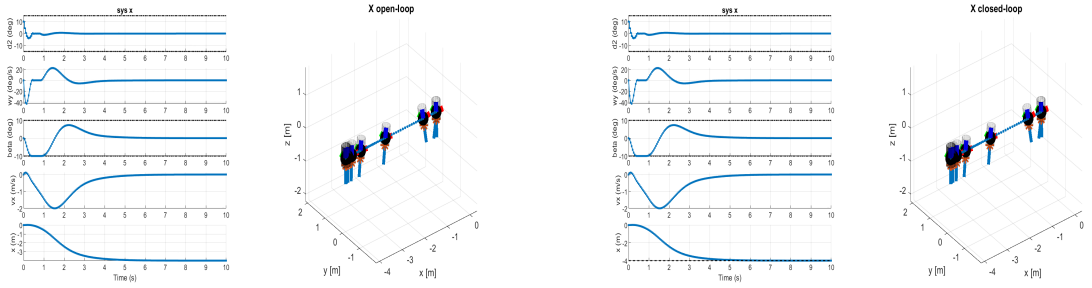
Using the delta formulation, we get the equations below for the MPC problem tracking a given reference.

$$
\begin{aligned}
&\min \sum_{i=0}^{N-1} \Delta x_i^T Q \Delta x_i + \Delta u_i^T R \Delta u_i + V_f(\Delta x_N) \\
\text{s.t.:} \quad &\Delta x_0 = \Delta x \\
&\Delta x_{i+1} = A \Delta x_i + B \Delta u_i \\
&H_x \Delta x_i \leq k_x - H_x x_s \\
&H_u \Delta u_i \leq k_u - H_u u_s
\end{aligned}
$$

with $\Delta x = u - x_s, \Delta u = u - u_s$ and $V_f(\Delta x_N) = \Delta x_N' Q_f \Delta x_N$ where Qf is calculated with the MATLAB function dlqr.
As suggested in the assignment, we dropped the requirement of invariance: $\Delta x_N \in \chi_f$.
The horizon is $H = 10$. Lowering the horizon improved the run-time of the MATLAB file.

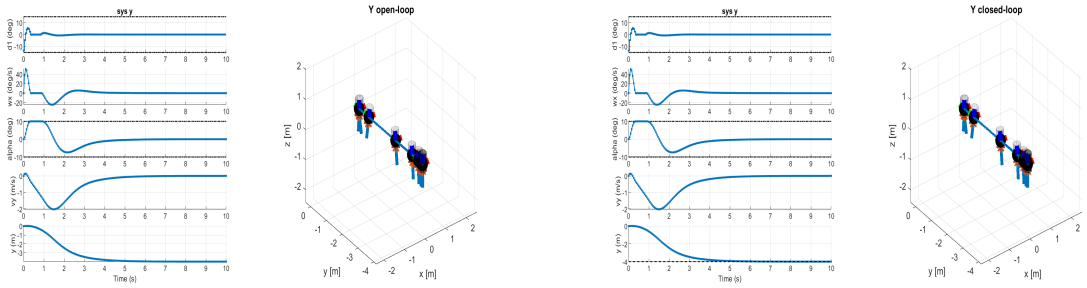
3.2.1 Plots Deliverable 3.2

Open-loop and closed-loop plots for each sub-system



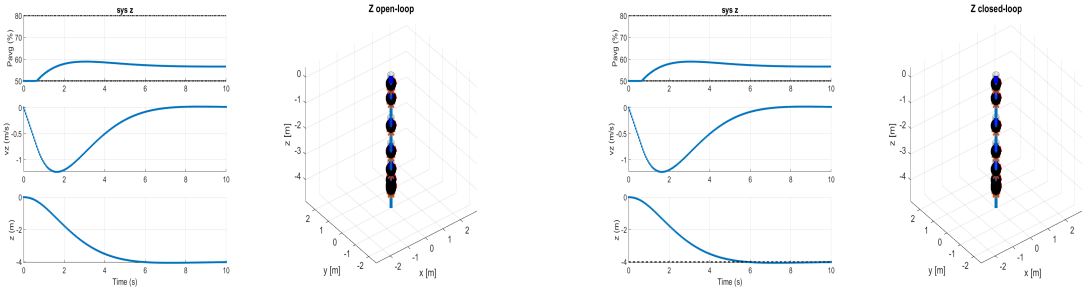
(a) Open-loop (b) Closed-loop

Figure 6: X controller plots



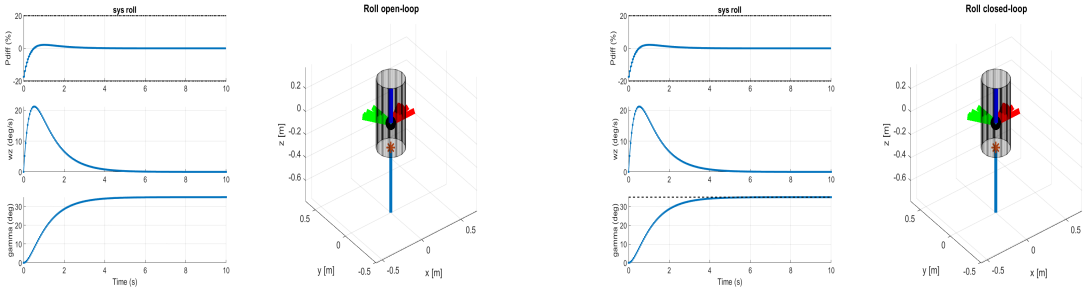
(a) Open-loop (b) Closed-loop

Figure 7: Y controller plots



(a) Open-loop (b) Closed-loop

Figure 8: Z controller plots



(a) Open-loop (b) Closed-loop

Figure 9: Roll controller plots

4 Putting together the sub-systems

4.0.1 Deliverable 4.1

Using the same tuning as in Deliverable 3.1 leads to an unsatisfactory performance. As you can see in Figure 10, the controller is not tracking z correctly. Furthermore, aggressive behaviour of wx and wy can be observed.

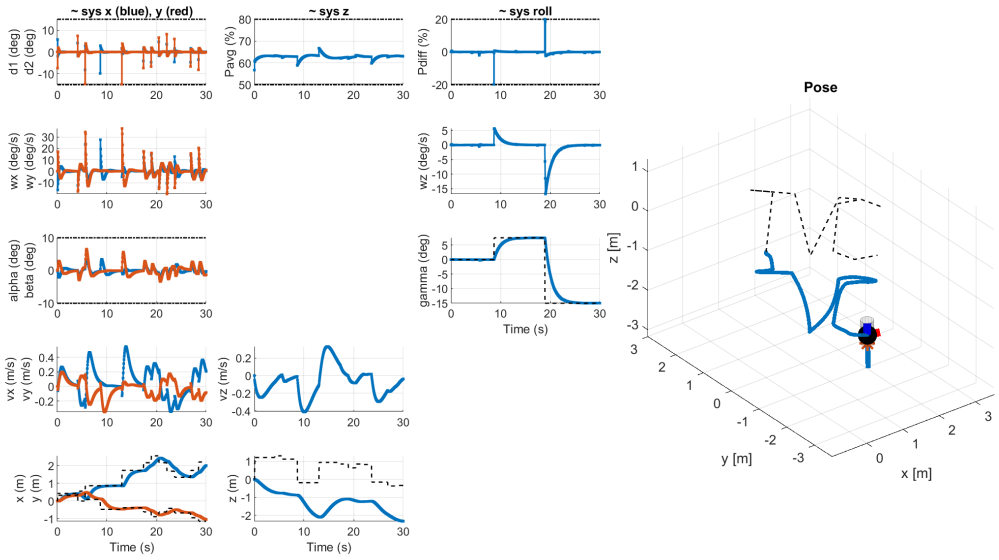


Figure 10: no tuning - Merged lin. MPC in nonlinear simulation

As suggested in the assignment, higher weights are assigned to the angular velocities (wx, wy,wz) to avoid too aggressive manoeuvres. To improve the tracking of z, high weights were necessary. The weights have been determined through trial and error. Below you can find the state vectors of the sub-systems and the corresponding weights (Q). $R = 1$ for all sub-systems, except $R_{roll} = 0.2$.

$$\mathbf{x_x} = \begin{bmatrix} wy \\ beta \\ vx \\ x \end{bmatrix} Q_x = \begin{bmatrix} 35 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$$

$$\mathbf{x_y} = \begin{bmatrix} wx \\ alpha \\ vy \\ y \end{bmatrix} Q_y = \begin{bmatrix} 30 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$$

$$\mathbf{x_z} = \begin{bmatrix} vz \\ z \end{bmatrix} Q_{roll} = \begin{bmatrix} 1000 & 0 \\ 0 & 1000 \end{bmatrix}$$

$$\mathbf{x_{roll}} = \begin{bmatrix} wz \\ gamma \end{bmatrix} Q_{roll} = \begin{bmatrix} 400 & 0 \\ 0 & 200 \end{bmatrix}$$

The tracking performance using the tuning above is illustrated in Figure 11.

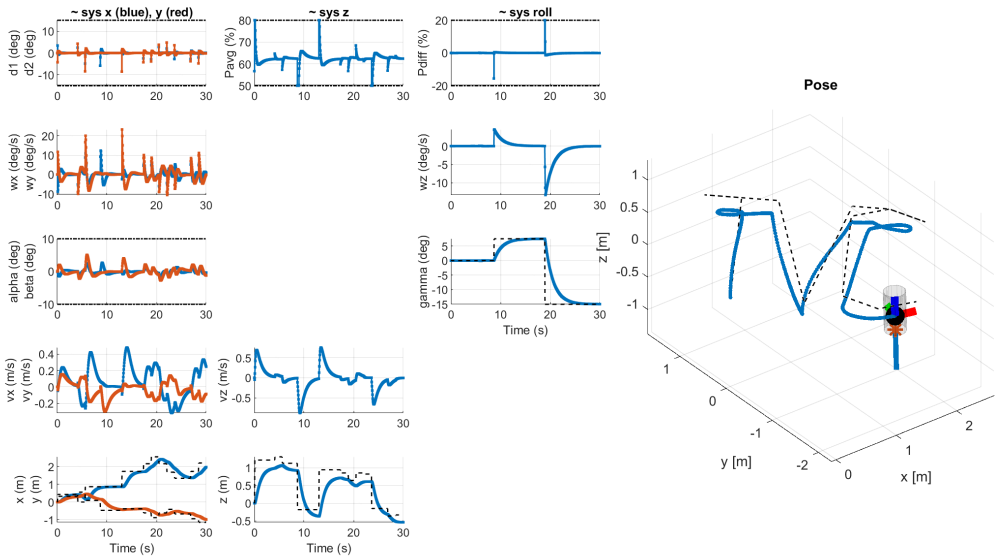


Figure 11: normal tuning - Merged lin. MPC in nonlinear simulation

In Figure 12, you can observe that open-loop trajectory nicely tracks the following reference:

$$\textbf{reference} = \begin{bmatrix} x = 2 \\ y = 2 \\ z = 2 \\ gamma = 40^\circ \end{bmatrix}$$

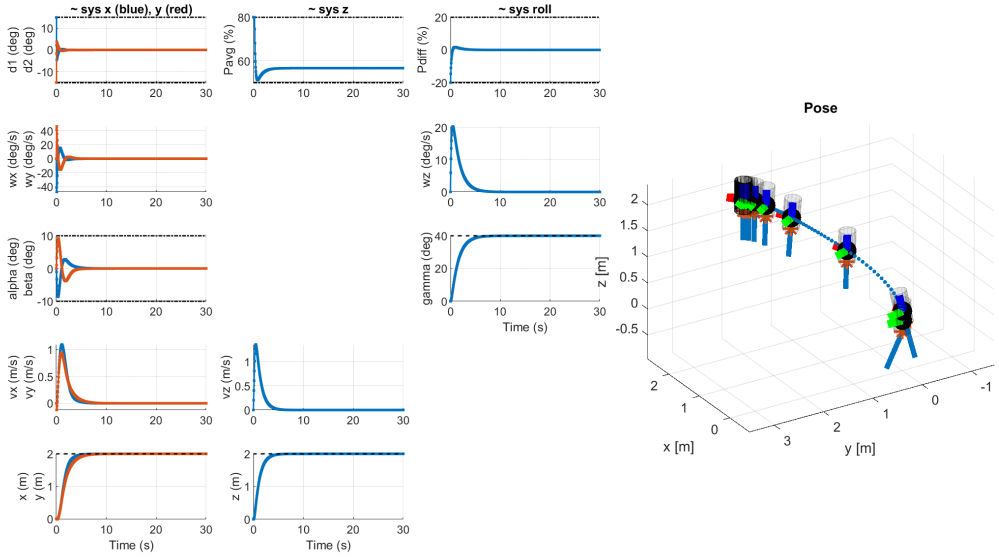


Figure 12: normal tuning - optimal open-loop trajectory

Unfortunately, even with ”normal” tuning, meaning adjusting Q and R to reasonable values, the tracking misses most corners of ”TVC”. We therefore implement a more aggressive tuning with the following weights:

$$\mathbf{x_x} = \begin{bmatrix} wy \\ beta \\ vx \\ x \end{bmatrix} Q_x = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix} R_x = 0.1$$

$$\mathbf{x_y} = \begin{bmatrix} wx \\ alpha \\ vy \\ y \end{bmatrix} Q_y = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix} R_y = 0.1$$

$$\mathbf{x_z} = \begin{bmatrix} vz \\ z \end{bmatrix} Q_z = \begin{bmatrix} 1000 & 0 \\ 0 & 1000 \end{bmatrix} R_z = 0.001;$$

$$\mathbf{x_{roll}} = \begin{bmatrix} wz \\ gamma \end{bmatrix} Q_{roll} = \begin{bmatrix} 10 & 0 \\ 0 & 1000 \end{bmatrix} R_{roll} = 0.1;$$

In Figure 13, you can observe that ”TVC” is tracked very accurately. Furthermore, we can see that alpha and beta take bigger values than in Figure 11 and gamma follows the reference a lot more precisely. To allow higher angles, we reduced their respective weight in Q. The rocket being able to have bigger angles (alpha, beta and gamma) allows it to track the reference more precisely.

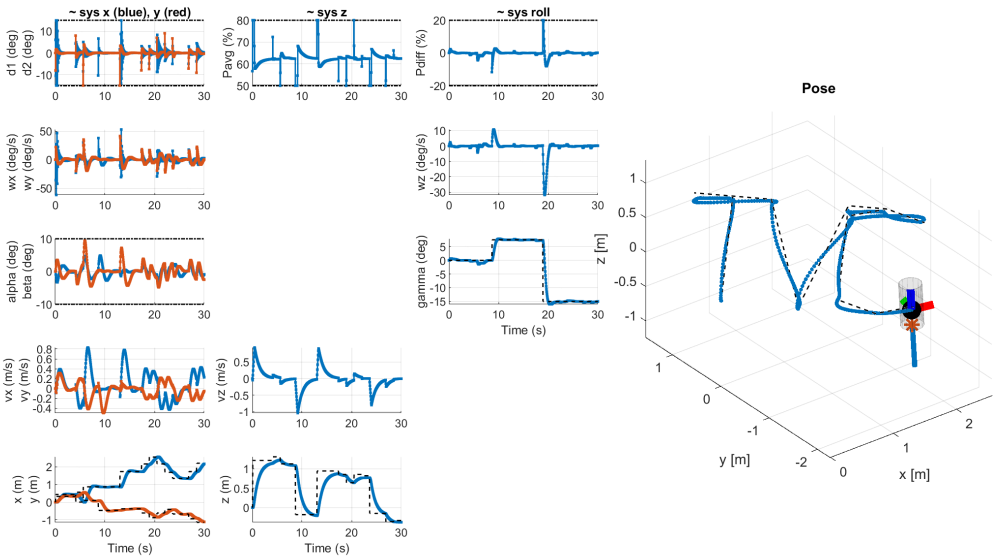


Figure 13: best tuning - Merged lin. MPC in nonlinear simulation

However, in Figure 14, we can see that the plots of the open-loop trajectory are a lot more aggressive (e.g. sys z).

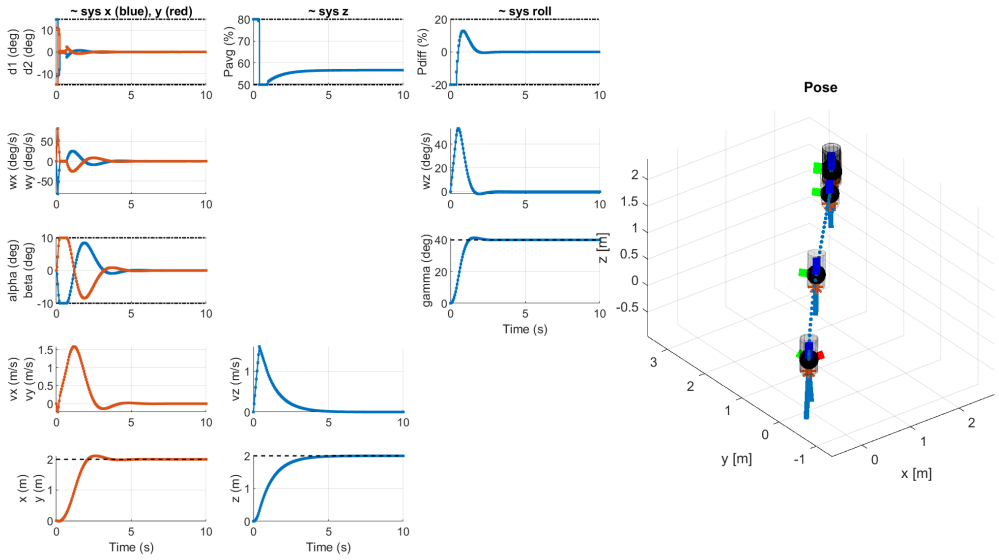


Figure 14: best tuning - optimal open-loop trajectory

5 Offset-Free Tracking

5.1 Deliverable 5.1

To include a disturbance for offset-free tracking, we "trick" the system by increasing the mass of the rocket by 25% after the creation of the sub-systems. This means that the disturbance is constant.
We model the disturbance as follows:

$$x(k+1) = Ax(k) + Bu(k) + Bd(k)$$

$$d(k+1) = d(k)$$

$$y(k) = Cx(k)$$

5.1.1 Design procedure

To guide us in our design we use the augmented estimated system.

$$\begin{bmatrix} \hat{x}(k+1) \\ \hat{d}(k+1) \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{x}(k) \\ \hat{d}(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(k) + L(C\hat{x}(k) - y(k)$$

$$\begin{bmatrix} \hat{x}(k+1) \\ \hat{d}(k+1) \end{bmatrix} = \bar{A} \begin{bmatrix} \hat{x}(k) \\ \hat{d}(k) \end{bmatrix} + \bar{B}u(k) + L(C\hat{x}(k) - y(k))$$

Next, we look at the error dynamics of the system

$$\begin{bmatrix} x(k+1) - \hat{x}(k+1) \\ d(k+1) - \hat{d}(k+1) \end{bmatrix} = \bar{A} \begin{bmatrix} x(k) - \hat{x}(k) \\ d(k) - \hat{d}(k) \end{bmatrix} + L[C0] \begin{bmatrix} x(k) - \hat{x}(k) \\ d(k) - \hat{d}(k) \end{bmatrix} = (\bar{A} + L\bar{C}) \begin{bmatrix} x(k) - \hat{x}(k) \\ d(k) - \hat{d}(k) \end{bmatrix}$$

To ensure stability, the norm of the eigenvalues must be less than one in the complex plane. We chose to use eigenvalues with a moderate norm after having tried all types of eigenvalues. These come with a faster convergence to the reference value. The eigenvalues we've chosen are 0.4, 0.5 and 0.6.

We are using the following Q and R matrices that have been slightly adapted from the "normal" tuning of the previous sections, with R = 1 except $R_{roll} = 0.2$. The modifications are in bold.

$$\mathbf{x_x} = \begin{bmatrix} wy \\ beta \\ vx \\ x \end{bmatrix} Q_x = \begin{bmatrix} 35 & 0 & 0 & 0 \\ 0 & \mathbf{10} & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$$

$$\mathbf{x_y} = \begin{bmatrix} wx \\ alpha \\ vy \\ y \end{bmatrix} Q_y = \begin{bmatrix} 30 & 0 & 0 & 0 \\ 0 & \mathbf{20} & 0 & 0 \\ 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & \mathbf{20} \end{bmatrix}$$

$$\mathbf{x_z} = \begin{bmatrix} vz \\ z \end{bmatrix} Q_{roll} = \begin{bmatrix} \mathbf{500} & 0 \\ 0 & \mathbf{500} \end{bmatrix}$$

$$\mathbf{x_{roll}} = \begin{bmatrix} wz \\ gamma \end{bmatrix} Q_{roll} = \begin{bmatrix} 400 & 0 \\ 0 & 200 \end{bmatrix}$$

5.1.2 Plots Deliverable 5.1

The following plots have been simulated for 8 seconds from

$$x0 = \begin{pmatrix} zeros(1,9)' \\ 1 \\ 0 \\ 3 \end{pmatrix}, ref = \begin{pmatrix} 1.2 \\ 0 \\ 3 \\ 0 \end{pmatrix}.$$

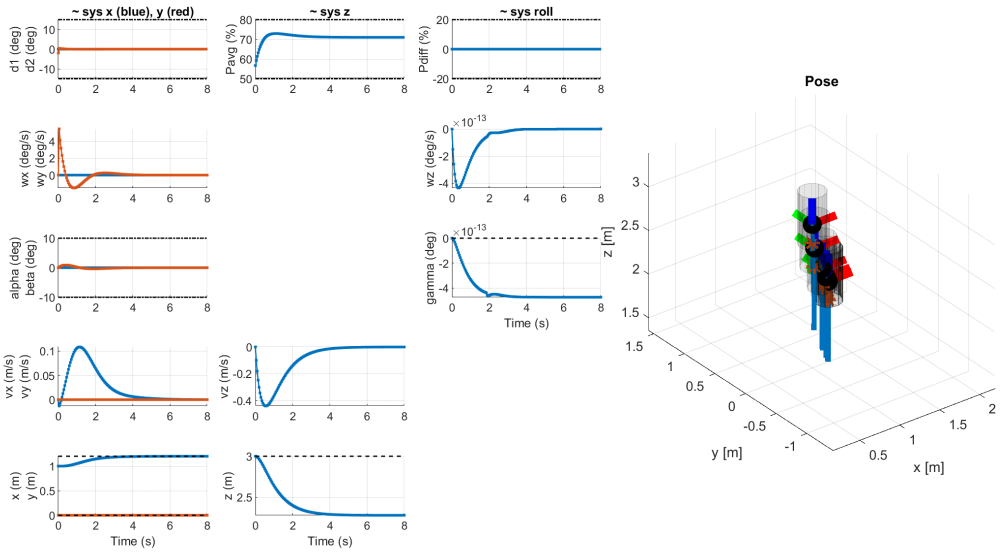


Figure 15: Plot without offset-free tracking

We observe that in Figure 15, the z-position stabilises under 2.5 meters. This happens because the system believes the mass to be 1.7 kg. The force required to stabilise this mass is lower than for the actual mass we've given the rocket, namely 2.13 kg. Hence, the rocket stabilises at a lower height than desired but still in under 7 seconds. We also notice that the shape of gamma is unexpected. However, given the amplitude, this can still be considered as "zero".
The small dents in the plots are negligible because they have a very small amplitude.

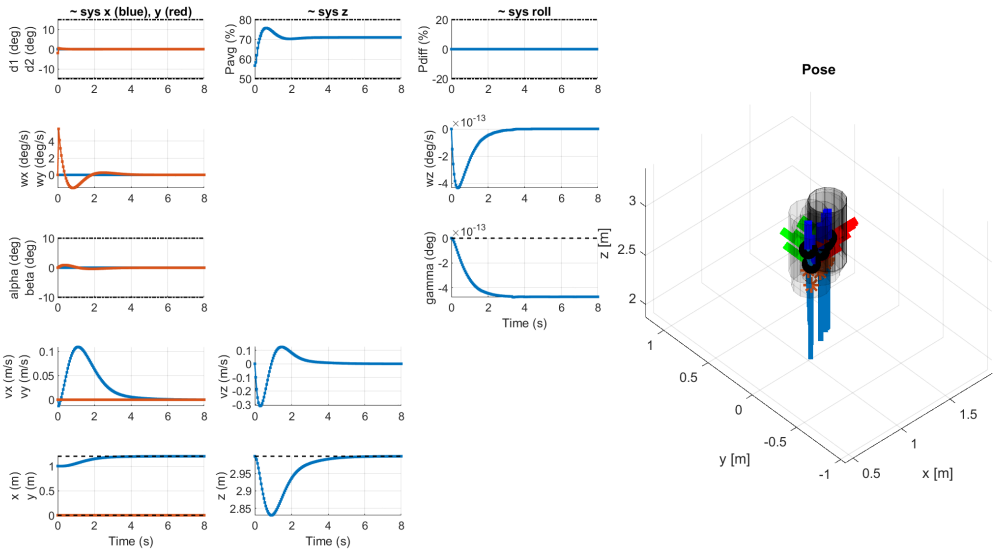


Figure 16: Plot with offset-free tracking

The second plot, Figure 16, shows the offset-free tracking in action. First, a dip is shown. This is because the simulation starts without knowing about the disturbance. The disturbance is then accounted for and the z-position is corrected back to 3 meters and the stabilisation happens in less than 7 seconds. Similarly to the plot in Figure 15, the plot of gamma is considered "zero" despite the look of it.

5.2 Deliverable 5.2

In this section, the rocket's mass decreases as fuel is consumed. Fuel constitutes half of the mass of the rocket, so the mass stabilises after all the fuel is consumed. This mass rate, just like the "real" mass is modified after the system is initialised and the controllers calculated.

Figure 17 shows how the simulation goes. We observe that with the time-varying mass, the offset-free tracking is not so effective. Indeed, we start at 3 meters in z and first descend. We then see an overshoot and never quite redescend back to 3 meters. The simulation is abandoned after 6.85 seconds because of infeasibility. As in Deliverable 5.1, the small dents in the plots are negligible because they have a very small amplitude.

Here is a table summarizing key moments of this simulation:

Time [s]	F=m*g [N]	Pavg [%]	F [N]	Situation
0	2.13g	57.7	1.7g	Beginning of the simulation
0.5	1.99g	73.9	2.21g	Maximum Pavg reached
0.8	1.91g	71.7	2.15g	Descent stopped, vz=0
1.8	1.64g	62.2	1.86g	3 meters reached again
3.5	1.18g	57.8	1.73g	Overshoot maximum, vz = -0
3.94	1.06g	56.6	1.69g	Fuel completely depleted
6.55	1.06g	50	1.5g	Lower limit of Pavg reached
6.85		49.4	1.48g	Simulation abandoned, infeasibility

Table 2: Summary table of simulation events

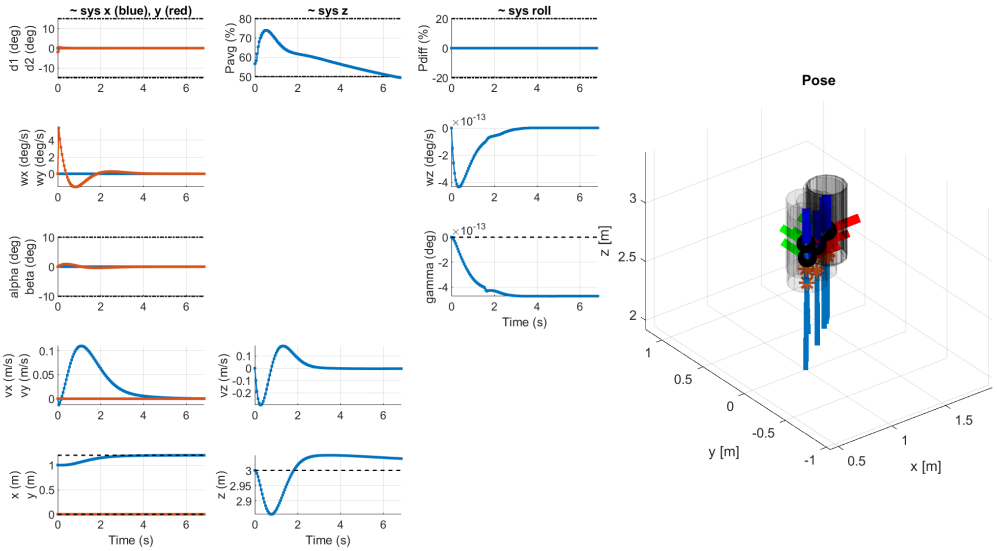


Figure 17: Plot of offset-free tracking with a thrust-dependent mass decrease during flight

F is calculated as seen in the assignment document. This amounts to F=0.03*Pavg*g around the linearization point. Given our calculations of F exerted by the servos, we see that to get F = 1.06g N we need Pavg to be 35.3%, which is less than the limits of the controller, and thus the physical limits of the rocket. By trying to match this value, Pavg dips under 50% and causes the rocket to fall and the control to be infeasible.

For better performance, we would need to have a control that does not consider the disturbance to be constant so that it can adapt. We could, for example, use Robust MPC. In this case, the problem can be written as:

$$x(k+1) = Ax(k) + Bu(k) + w$$

where w is bounded. There exists a set O^W that is robust positive invariant and works like X_f in the disturbance-free problem. The boundedness criterion is met in our case as the mass cannot logically vary forever. It is first of all bounded below by 0, as mass will not be negative, and infinite mass is not physically possible. Big masses might be attained, but given the size of the rocket, it is not something that we considered.

6 Nonlinear MPC

6.1 Deliverable 6.1

6.1.1 Design procedure

For the design of the nonlinear problem, we first get the system dynamics from the rocket module and then discretize using Runge-Kutta (4th order) as seen in exercise 7. Then, the box-constraints are established. The next step is the terminal cost where we consider the difference between the reference and the associated states as well as the control-action. The weights were determined largely by trial and error. We found that we had the best results when the weights of the reference following are around 200 for the control weights being an identity matrix. Additionally, we have quite a large offset for z. Therefore we increased its weight until we no longer saw any benefit, which left it at 500. Interestingly, we found that for the TVC reference, we have the smallest offset in the z-axis for a horizon of 3 seconds. For longer horizons, it settles at a constant offset of ∼0.3 meters lower than the TVC reference after around 5 seconds. We found this rather surprising as we would expect a better performance for a longer horizon. These parameters lead to "aggressive" control but the best tracking performance.

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \\ gamma \end{bmatrix} \quad Q = \begin{bmatrix} 200 & 0 & 0 & 0 \\ 0 & 200 & 0 & 0 \\ 0 & 0 & \mathbf{500} & 0 \\ 0 & 0 & 0 & 200 \end{bmatrix} \quad u = \begin{bmatrix} d_1 \\ d_2 \\ P_{avg} \\ P_{diff} \end{bmatrix} \quad R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

6.1.2 Plots Deliverable 6.1

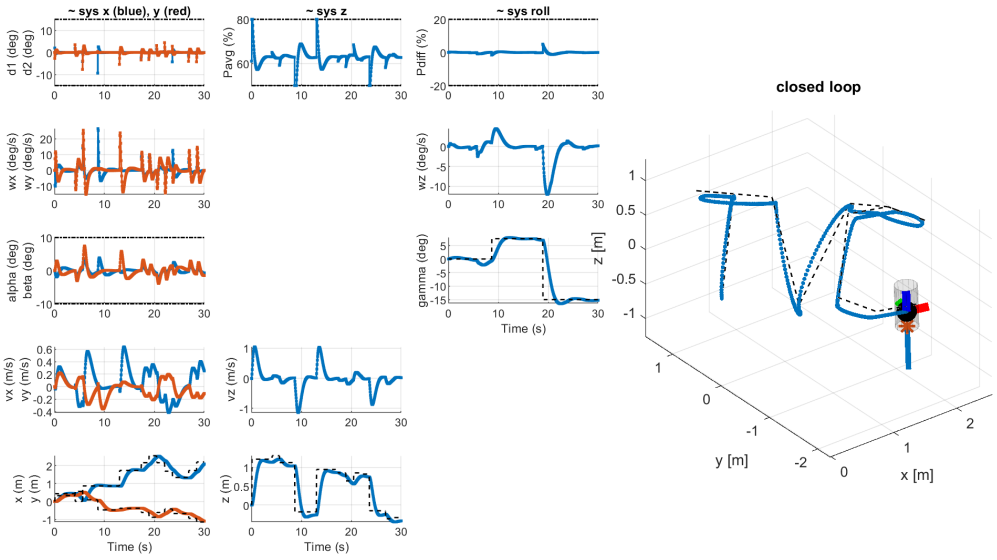


Figure 18: Plot of offset-free tracking with a 3-second horizon

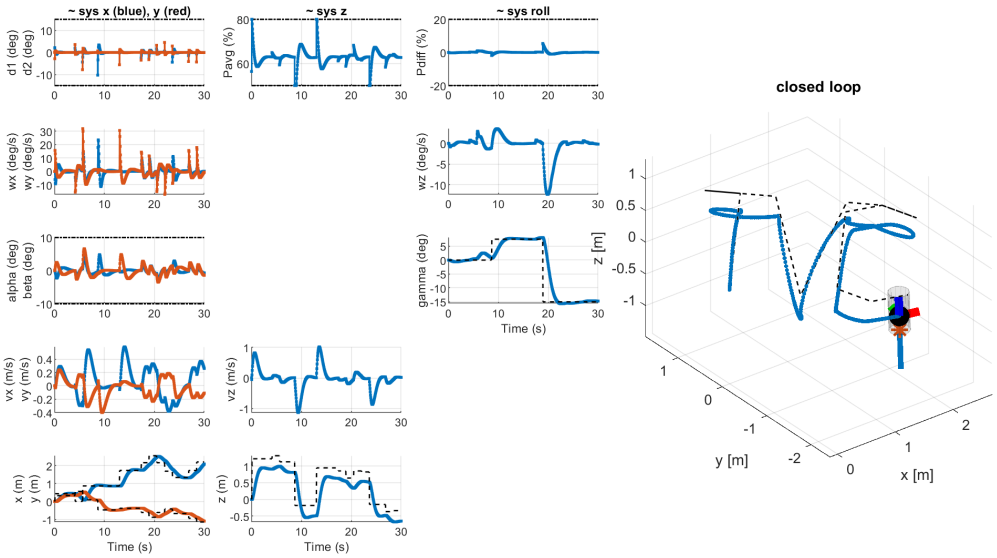


Figure 19: Plot of tracking with an offset with a 5-second horizon

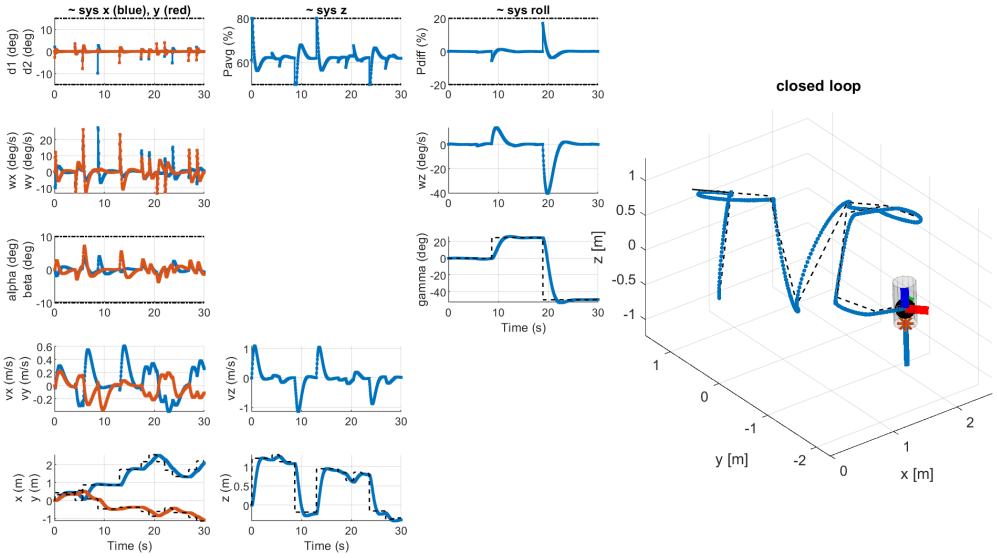


Figure 20: Plot of offset-free tracking of a 50-degree roll reference with a 3-second horizon

6.1.3 The pros and cons of our nonlinear controller

The big advantage of NMPC is that it is more accurate and enables the control of highly non-linear systems. For our performance, we can profit from the fact that we are not constrained to our linearization approximations that make up the feasible sets. This leads to a bigger working area. The biggest drawback is how computationally intense it is and how long the calculations take. We have to use a very short horizon, whereas our linear controller uses a much bigger horizon of 30 seconds. Using this short 3-second horizon as a comparison value, non-linear calculations take 76 seconds, compared to 26 seconds for the linear controller (calculated on the same computer). Both offer a similar performance when it comes to following the trajectory.

In conclusion, for our project, the linear controller offers sufficient precision and it is much more computationally efficient.

6.2 Deliverable 6.2

6.2.1 How much delay is needed for adverse effects?

For all the measurements in this part, the horizon is 4 seconds for best performance without delay. It settles on the reference with no offset at around 4 seconds. We speculate the specified weight could be a contributing factor.
For the performance without delay compensation, starting at 75 milliseconds of delay there is a first drop in performance where we have some oscillation, especially in speeds, that no longer gets attenuated. The performance goes downhill from there with a further increase of delay. Instability, here, is interpreted as the rapidly growing oscillations appearing at 100ms of delay. The problem becomes infeasible at 175 milliseconds of delay. The implemented delay compensation can correct this issue as you can see in the graphs below.

6.2.2 Plots Deliverable 6.2

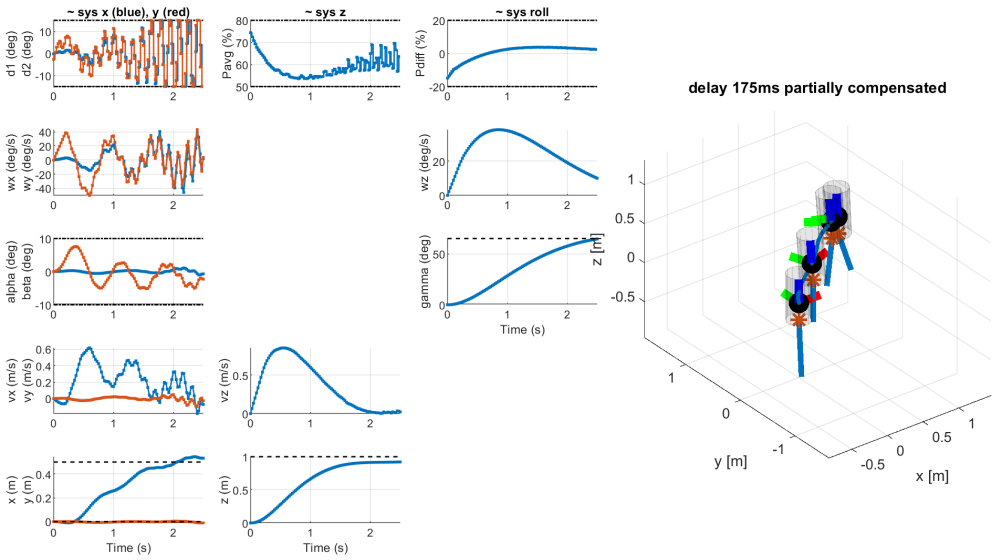


Figure 21: Plot of a partially compensated delay (125 of 175 ms)

In the case of partially compensated delay, we can observe that the states have a bit of oscillation but they follow the trajectory rather well. The control however is very aggressive and oscillates a lot, especially the $\delta 1$ and $\delta 2$.

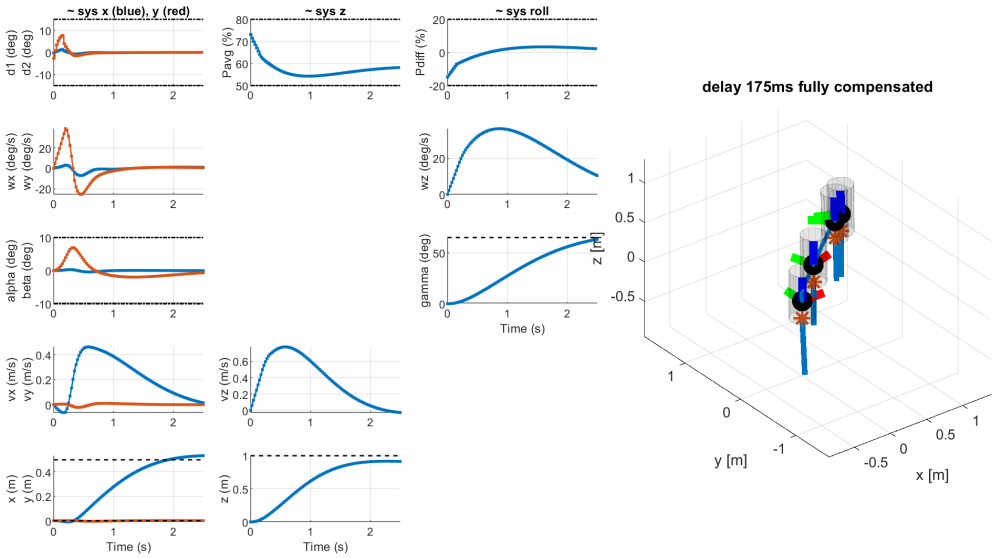


Figure 22: Plot of a fully compensated delay (175 of 175 ms)

The NMPC with fully compensated delay has a very smooth control action as well as a good trajectory.