ME-425 Model Predictive Control

Mini-Project 2025

MPC Controller for Rocket Landing

Group AL

TLIGUI Yasmine *(SCIPER: 341215)*
EL QABLI Rim *(SCIPER: 340997)*
FILALI Ismail *(SCIPER: 345459)*

January 11, 2026

Contents

1 Introduction

This report presents the design, implementation, and evaluation of Model Predictive Control (MPC) strategies for a rocket landing task based on the ME-425 mini-project framework. The objective is to achieve regulation, tracking, and landing maneuvers while respecting actuator limits, linearization validity constraints, and safety constraints near the ground.

The objective of this report is to address multiple control challenges for the rocket system by developing and comparing different controller architectures. We tackle notably: linear MPC controllers for velocity regulation, NMPC (Nonlinear MPC) controllers to exploit the full system dynamics, position tracking via a cascaded PID-MPC architecture, as well as robustness techniques such as Tube MPC and offset-free tracking to handle disturbances and model uncertainties. Each approach is designed to meet specific requirements in terms of performance, robustness, and computational feasibility, thus enabling a comprehensive comparative evaluation of the different control strategies.

2 Linearization

2.1 System Overview

The rocket system is a 12-state nonlinear dynamical system described by the state vector:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\omega}^T & \boldsymbol{\phi}^T & \boldsymbol{v}^T & \boldsymbol{p}^T \end{bmatrix}^T \tag{1}$$

where $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T$ are the angular velocities (rad/s), $\boldsymbol{\phi} = [\alpha, \beta, \gamma]^T$ are the Euler angles (rad), $\boldsymbol{v} = [v_x, v_y, v_z]^T$ are the linear velocities (m/s), and $\boldsymbol{p} = [x, y, z]^T$ are the positions (m).

The input vector consists of four control signals:

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

where δ_1 and δ_2 are servo deflection angles (± 15 max), P_{avg} is the average throttle (10-90%), and P_{diff} is the differential throttle ($\pm 20\%$).

2.2 Deliverable 2.1: System Decomposition

2.2.1 Trimming and Linearization

To design a linear MPC controller, we first compute the equilibrium point (trim point) where the rocket hovers stationary. Using the trim function, we obtain:

Steady-state values around origin:

$$\boldsymbol{x}_s = \begin{bmatrix} 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \end{bmatrix}^T, \boldsymbol{u}_s = \begin{bmatrix} 0\ 0\ 66.67\ 0 \end{bmatrix}^T \tag{3}$$

This corresponds to a vertical rocket at rest, requiring $P_{avg} = 66.67\%$ to counteract gravity with no servo deflection or differential throttle.

Steady-state values around landing point: For the landing phase at $x = 1$m, $z = 3$m:

$$\boldsymbol{x}_s = \begin{bmatrix} 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 3 \end{bmatrix}^T, \boldsymbol{u}_s = \begin{bmatrix} 0\ 0\ 66.67\ 0 \end{bmatrix}^T \tag{4}$$

The required throttle remains the same (gravity compensation), but the position changes.

Linearizing the nonlinear dynamics $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$ around the trim point yields:

$$\dot{\boldsymbol{x}} \approx \mathbf{A}(\boldsymbol{x} - \boldsymbol{x}_s) + \mathbf{B}(\boldsymbol{u} - \boldsymbol{u}_s) \tag{5}$$

where $\mathbf{A} \in \mathbb{R}^{12\times12}$ and $\mathbf{B} \in \mathbb{R}^{12\times4}$ are the linearized system matrices.

2.2.2 Independent Subsystems

Analysis of the linearized matrices reveals that the system can be decomposed into four independent subsystems due to the block-diagonal structure:

$$
\mathbf{A} = \left[\begin{array}{c|cccccccccccc}
 & \omega_z & \gamma & \omega_y & \beta & v_x & x & \omega_x & \alpha & v_y & y & v_z & z \\
\hline
\omega_z & 0 & 0 & . & . & . & . & . & . & . & . & . & . \\
\gamma & 1 & 0 & . & . & . & . & . & . & . & . & . & . \\
\omega_y & . & . & 0 & 0 & 0 & 0 & . & . & . & . & . & . \\
\beta & . & . & 1 & 0 & 0 & 0 & . & . & . & . & . & . \\
v_x & . & . & 0 & 9.8 & 0 & 0 & . & . & . & . & . & . \\
x & . & . & 0 & 0 & 1 & 0 & . & . & . & . & . & . \\
\omega_x & . & . & . & . & . & . & 0 & 0 & 0 & 0 & . & . \\
\alpha & . & . & . & . & . & . & 1 & 0 & 0 & 0 & . & . \\
v_y & . & . & . & . & . & . & 0 & -9.8 & 0 & 0 & . & . \\
y & . & . & . & . & . & . & 0 & 0 & 1 & 0 & . & . \\
v_z & . & . & . & . & . & . & . & . & . & . & 0 & 0 \\
z & . & . & . & . & . & . & . & . & . & . & 1 & 0
\end{array}\right]
\qquad
\mathbf{B} = \left[\begin{array}{c|cccc}
 & P_{diff} & \delta_2 & \delta_1 & P_{avg} \\
\hline
\omega_z & 0.104 & . & . & . \\
\gamma & 0 & . & . & . \\
\omega_y & . & -65.5 & . & . \\
\beta & . & 0 & . & . \\
v_x & . & 9.8 & . & . \\
x & . & 0 & . & . \\
\omega_x & . & . & -65.5 & . \\
\alpha & . & . & 0 & . \\
v_y & . & . & -9.8 & . \\
y & . & . & 0 & . \\
v_z & . & . & . & 0.147 \\
z & . & . & . & 0
\end{array}\right]
$$

where (.) denotes zero. Observing this matrix, we have:

$$
\mathbf{A} = \left[\begin{array}{c|c|c|c}
\mathbf{A}_{\text{roll}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline
\mathbf{0} & \mathbf{A}_{\text{x}} & \mathbf{0} & \mathbf{0} \\ \hline
\mathbf{0} & \mathbf{0} & \mathbf{A}_{\text{y}} & \mathbf{0} \\ \hline
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{\text{z}}
\end{array}\right]
\begin{array}{l}
\leftarrow (\omega_z, \gamma) \\
\leftarrow (\omega_y, \beta, v_x, x) \\
\leftarrow (\omega_x, \alpha, v_y, y) \\
\leftarrow (v_z, z)
\end{array}
\quad
\mathbf{B} = \left[\begin{array}{c|c|c|c}
\mathbf{B}_{\text{roll}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline
\mathbf{0} & \mathbf{B}_{\text{x}} & \mathbf{0} & \mathbf{0} \\ \hline
\mathbf{0} & \mathbf{0} & \mathbf{B}_{\text{y}} & \mathbf{0} \\ \hline
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{\text{z}}
\end{array}\right]
\begin{array}{l}
\leftarrow (P_{diff}) \\
\leftarrow (\delta_2) \\
\leftarrow (\delta_1) \\
\leftarrow (P_{avg})
\end{array}
\tag{6}
$$

where all off-diagonal blocks $\mathbf{0}$ are zero matrices.

This block-diagonal structure means we can decompose the linearized system into four independent subsystems around the hover equilibrium without any ambiguity:

$$\dot{\boldsymbol{x}}_i = \mathbf{A}_i \boldsymbol{x}_i + \mathbf{B}_i \boldsymbol{u}_i, \quad i \in \{\text{roll}, \text{x}, \text{y}, \text{z}\} \tag{7}$$

There is no coupling between subsystems: the evolution of states in one subsystem does not affect any states in the other subsystems. This allows us to design MPC controllers for each subsystem independently, significantly reducing computational complexity:

Subsystem roll: Differential throttle P_{diff} to roll angle γ. States: $\boldsymbol{x}_{\text{roll}} = [\omega_z, \gamma]^T$ (2 states). This is a double integrator: angular acceleration → angular velocity → angle.

Subsystem x: Thrust vector angle δ_2 to position x. States: $\boldsymbol{x}_{\text{x}} = [\omega_y, \beta, v_x, x]^T$ (4 states).

Subsystem y: Thrust vector angle δ_1 to position y. States: $\boldsymbol{x}_{\text{y}} = [\omega_x, \alpha, v_y, y]^T$ (4 states).

Subsystem z: Average throttle P_{avg} to height z. States: $\boldsymbol{x}_{\text{z}} = [v_z, z]^T$ (2 states). Simple double integrator: acceleration → velocity → position.

2.2.3 Interpretation of Independence

The ability to decompose the linearized rocket dynamics into four independent subsystems stems from both mathematical properties and fundamental physical principles.

Mathematical Justification The block-diagonal structure arises from linearization around hover. At trim, the rocket is vertical with $\alpha = \beta = \gamma = 0$, $\boldsymbol{\omega} = \mathbf{0}$, and $\boldsymbol{v} = \mathbf{0}$. Nonlinear coupling terms like $\sin(\alpha)\sin(\beta)$ or $\omega_x\omega_y$ vanish at this point. The symmetric two-axis gimbal creates orthogonal control: δ_1 controls only the x_b-axis moment, δ_2 only the y_b-axis moment. At hover, thrust is purely vertical with $\cos(\delta) \approx 1$, decoupling z-dynamics from horizontal tilts.

Physical Interpretation The rocket's cylindrical symmetry around the z-axis creates identical x and y dynamics. Vertical thrust at hover makes tilt effects on vertical acceleration second-order: $F_z = F\cos(\delta) \approx F(1-\delta^2/2)$, decoupling z from δ_1, δ_2. Roll angle γ rotates the body around the vertical axis without affecting force direction or moment arms to first order, remaining decoupled.

Validity and Limitations This decomposition is valid only near hover. For large deviations: large angles introduce coupling via $\sin(\alpha)\cos(\beta)$ terms; high angular velocities create gyroscopic effects $\omega \times J\omega$; large γ rotates horizontal thrust components; and significant $|\delta|$ reduces vertical thrust projection.

Therefore, MPC controllers based on these linearized subsystems must enforce state constraints ($|\alpha|$ and $|\beta| \leq 10$) to maintain validity.

3 Design MPC Velocity Controllers for Each Subsystem

Following the system decomposition established in Section 2, we design independent MPC controllers for each of the four subsystems. All controllers share a common design methodology and use discrete-time dynamics with sampling period $T_s = 0.05$ s (20 Hz).

3.1 Common Controller Architecture

3.1.1 Optimization Problem Formulation

For each subsystem $i \in \{\mathrm{x, y, z, roll}\}$, we have a stabilizing MPC problem, the optimization at time k is:

$$\begin{aligned}\min_{\boldsymbol{u}_{i,0:N-1}} \quad & \sum_{j=0}^{N-1} \left[(\boldsymbol{x}_{i,j} - \boldsymbol{x}_{r,i})^T \mathbf{Q}_i (\boldsymbol{x}_{i,j} - \boldsymbol{x}_{r,i}) + (\boldsymbol{u}_{i,j} - \boldsymbol{u}_{r,i})^T \mathbf{R}_i (\boldsymbol{u}_{i,j} - \boldsymbol{u}_{r,i}) \right] \\ & + (\boldsymbol{x}_{i,N} - \boldsymbol{x}_{r,i})^T \mathbf{P}_i (\boldsymbol{x}_{i,N} - \boldsymbol{x}_{r,i}) \qquad (8) \\ \text{s.t.} \quad & \boldsymbol{x}_{i,j+1} = \mathbf{A}_{d,i}(\boldsymbol{x}_{i,j} - \boldsymbol{x}_{s,i}) + \mathbf{B}_{d,i}(\boldsymbol{u}_{i,j} - \boldsymbol{u}_{s,i}) + \boldsymbol{x}_{s,i}, \quad \boldsymbol{x}_{i,0} = \boldsymbol{x}_i(k) \qquad (9) \\ & \boldsymbol{x}_{i,j} \in \mathcal{X}_i, \quad \boldsymbol{u}_{i,j} \in \mathcal{U}_i, \quad \boldsymbol{x}_{i,N} \in \mathcal{X}_{f,i} \qquad (10)\end{aligned}$$

where $\boldsymbol{x}_{r,i}$ is the reference trajectory, $\boldsymbol{x}_{s,i}$ and $\boldsymbol{u}_{s,i}$ are the linearization (trim) points, $\mathbf{Q}_i$ and $\mathbf{R}_i$ are the stage cost weighting matrices for states and inputs respectively, $\mathbf{P}_i$ is the terminal cost matrix, N is the prediction horizon, $\mathcal{X}_i$ and $\mathcal{U}_i$ are the state and input constraint sets, and $\mathcal{X}_{f,i}$ is the terminal constraint set ensuring stability.

Note on regulation vs tracking: The MPC formulation remains identical for both regulation and tracking; only the reference values differ: Regulation (Deliverable 3.1): No explicit reference is provided. The controller defaults to the trim point ($\boldsymbol{x}_{r,i} = \boldsymbol{x}_{s,i}$, $\boldsymbol{u}_{r,i} = \boldsymbol{u}_{s,i}$) and drives the system back to hover from non-zero initial conditions. Tracking (Deliverables 3.2 and 3.3): Explicit references are provided ($\boldsymbol{x}_{r,i} \neq \boldsymbol{x}_{s,i}$). The controller tracks the constant non-zero setpoints.

3.1.2 Ingredients for Stability

The finite-horizon MPC requires terminal ingredients to guarantee both asymptotic stability and recursive feasibility. The stage cost $(\boldsymbol{x}_{i,j} - \boldsymbol{x}_{r,i})^T \mathbf{Q}_i (\boldsymbol{x}_{i,j} - \boldsymbol{x}_{r,i}) + (\boldsymbol{u}_{i,j} - \boldsymbol{u}_{r,i})^T \mathbf{R}_i (\boldsymbol{u}_{i,j} - \boldsymbol{u}_{r,i})$ already serves as a Lyapunov function since $\mathbf{Q}_i \succ 0$ and $\mathbf{R}_i \succ 0$ are positive definite. However, truncating the infinite-horizon cost at step N requires a terminal cost to bound the tail. Asymptotic stability alone is insufficient, we must also guarantee that if the MPC problem is feasible at time k, it remains feasible at time $k+1$ (recursive feasibility). This is achieved through a terminal constraint $\boldsymbol{x}_{i,N} \in \mathcal{X}_{f,i}$, where $\mathcal{X}_{f,i}$ is a control-invariant set. By induction, if the MPC problem is feasible at time $k = 0$, it remains feasible for all future times, guaranteeing recursive feasibility. The combination of this conditions guarantees that the MPC controller is stable and converges.

3.1.3 Constraints

The state and input constraints are derived from physical limitations, safety requirements, and linearization validity. Because our linearization is approximate, we must place constraints on the maximum angles: $|\alpha| \leq 10 = 0.1745$ rad, $|\beta| \leq 10 = 0.1745$ rad. Note that the linearized roll subsystem is valid for any roll angle γ. However, the linearizations of the x and y subsystems become less accurate as γ moves away from the linearization point. In addition to mechanical input constraints ($|\delta_1|, |\delta_2| \leq 15$), experiments with the prototype have shown that the rocket descends too quickly when less than 40% average throttle is given. For safety, we require: $40\% \leq P_{avg} \leq 80\%$. Due to the Deliverable 3.3 conditions with PID, we tightened the lower bound to 50% for additional safety margin during large-displacement maneuvers.

Table 1: State and input constraints for each subsystem

Subsystem	Constraint	Lower	Upper
x, y	α, β	-10	$+10$
	δ_1, δ_2	-15	$+15$
	ω, v	$-\infty$	$+\infty$
z	P_{avg}	50%	80%
	v_z	$-\infty$	$+\infty$
roll	P_{diff}	-20%	$+20\%$

3.1.4 Prediction Horizon Selection

We determine the minimum feasible horizon through grid search across $N \in \{10, 15, \ldots, 160\}$ ($H \in [0.5, 8.0]$ s with step 0.25 s). The prediction horizon N fixes the convergence rate and the region of attraction. If too short, the controller cannot reach the terminal set within N steps while satisfying constraints, causing infeasibility. Longer horizons enable more gradual trajectories but increase computational cost. We select the minimal common horizon $H = 1.75$ s ($N = 35$ steps) for Deliverables 3.1 and 3.2 because it balances feasibility with computational efficiency. For position tracking (Deliverable 3.3), the prediction horizon is extended to $H = 6.0\,\text{s}$ ($N = 120$ steps) to account for the cascaded PID-MPC architecture.

Table 2: Minimum feasible prediction horizons

Subsystem	N_{min}	H_{min}
x (vel. reg.)	35	1.75
y (vel. reg.)	35	1.75
z (vel. reg.)	30	1.50
roll (angle reg.)	10	0.50
Common min.	**35**	**1.75**
Selected (3.1, 3.2)	**35**	**1.75**
Extended (3.3)	**120**	**6.00**

3.2 Parameter Tuning: Iterative Design Process

3.2.1 Initial Tuning Strategy

The weight matrices $\mathbf{Q}$ and $\mathbf{R}$ determine the trade-off between tracking performance and control effort. The ratio Q/R controls convergence speed: high ratios prioritize fast state correction at the cost of aggressive actuation, while low ratios favor smooth, energy-efficient control. Within $\mathbf{Q}$, relative weights encode state priorities, higher weights penalize deviations more heavily, forcing tighter tracking of critical states near constraint boundaries. The input penalty R controls actuation smoothness: low R allows rapid control changes but may cause oscillations, while high R enforces gradual, smooth control at the cost of slower response. We began with constraint-based scaling: critical angles received $Q_{\text{angle}} = 100$ to enforce linearization validity ($|\alpha|, |\beta| \leq 10$), velocities received moderate weight $Q_{\text{velocity}} = 10$ for balanced tracking, angular velocities received $Q_\omega = 1$ to allow necessary transient dynamics, and inputs received baseline penalty $R = 1.0$.

3.2.2 Identified Problems and Solutions

Testing revealed three issues requiring tuning refinement:

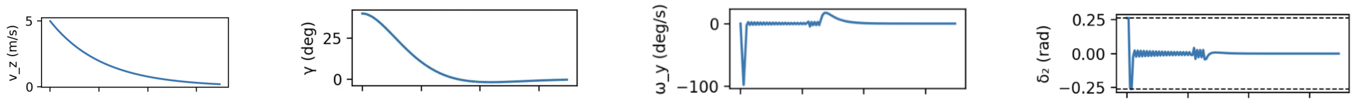


Figure 1: Tuning issues: (a) Z-axis before tuning, (b) Roll before tuning, (c) ω_y oscillations, (d) δ_2 oscillations

Problem 1: Slow Vertical Velocity Convergence Vertical control via thrust modulation has limited authority near hover, and the low Q/R ratio ($10/1 = 10$) made the optimizer conservative with throttle variations. Increasing $Q_{v_z} = 50$ and reducing $R_z = 0.3$ (giving Q/R ≈ 167) encourages aggressive use of the throttle range $[40\%, 80\%]$, achieving faster settling time.

Problem 2: Roll Overshoot When $Q_\gamma \gg Q_{\omega_z}$ (ratio 100:1), the optimizer prioritizes angle reduction over velocity limitation, causing momentum buildup and overshoot. Increasing $Q_{\omega_z} = 100$ and $Q_\gamma = 200$ (ratio 2:1) forces simultaneous minimization of both angle error and angular velocity, achieving critical damping.

Problem 3: High-Frequency Oscillations in x/y Control Initial tuning with $Q = (1, 100, 10)$ and $R = 1$ exhibited oscillations visible in both angular velocities (ω_y, ω_x) and servo deflections (δ_1, δ_2), particularly when angles approached constraint boundaries. This is caused by low R and constraint saturation creating numerical sensitivity. Increasing R to 20 enforces smooth control, while reducing $Q_\beta = 15$ and $Q_v = 0.2$ prevents over-constraining the optimization, with high R, lower state weights being sufficient for tracking. This balanced tuning eliminates oscillations while maintaining feasibility and acceptable convergence speed.

3.2.3 Final Tuning Parameters

Subsys.	State Variables	Q_1	Q_2	Q_3	R
x	(ω_y, β, v_x)	1	15	0.2	20
y	(ω_x, α, v_y)	1	15	0.2	20
z	(v_z)	50	—	—	0.3
roll	(ω_z, γ)	80	200	—	1

3.3 Terminal Invariant Sets

The computed terminal invariant sets for each subsystem are visualized through 2D projections:

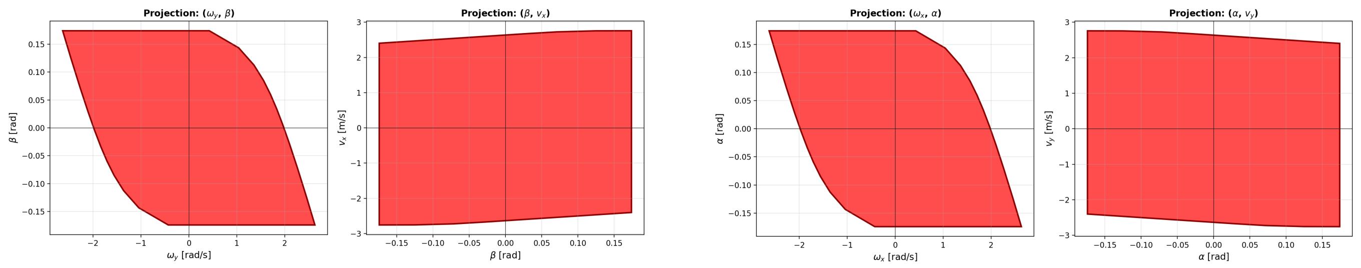


Figure 2: Terminal sets for x (left) and y (right) subsystems

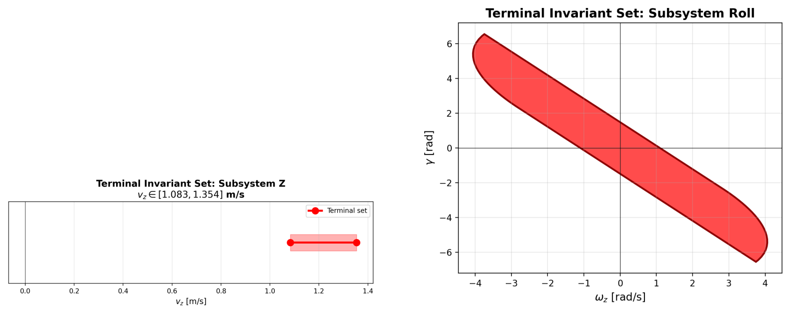


Figure 3: Terminal sets for z (left) and roll (right) subsystems

Algorithm Used

The terminal invariant sets are computed using the maximal controlled invariant set algorithm:

```
1: Initialize: $\Omega_0 \leftarrow \mathbb{X}$
2: loop
3:     $\Omega_{i+1} \leftarrow \text{pre}(\Omega_i) \cap \Omega_i$
4:     if $\Omega_{i+1} = \Omega_i$ then
5:         return $\mathcal{O}_\infty = \Omega_i$
6:     end if
7: end loop
```

where $\text{pre}(\Omega)$ denotes the one-step controllable predecessor set.

These sets guarantee that once the state enters $\mathcal{X}_f$, it can be kept inside indefinitely while satisfying all constraints.

3.4 Deliverable 3.1: MPC Regulators

3.4.1 Test Scenario

We evaluate regulation performance by commanding the system to return to hover from challenging initial conditions: x/y-axis: $v_x = v_y = 5$ m/s (strong horizontal velocity), z-axis: $v_z = 5$ m/s (strong vertical velocity), Roll: $\gamma = 40$ (large roll angle). The objective is to drive all states to zero (hover at origin: $\boldsymbol{x}_r = \boldsymbol{x}_s$) within the 7-second settling time requirement.

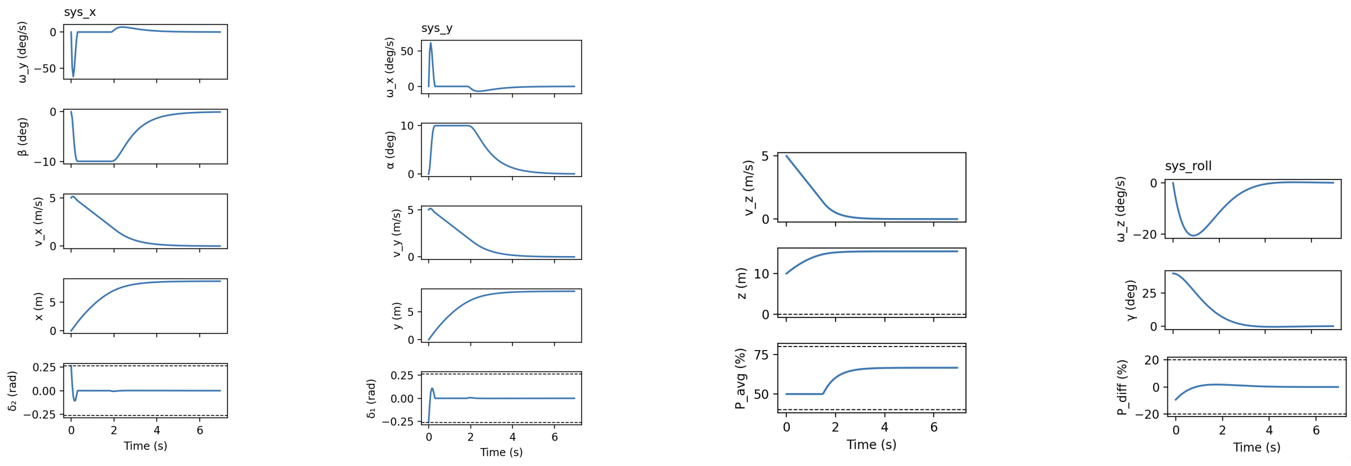


Figure 4: MPC regulation results in closed-loop

In closed-loop, all subsystems converge to hover within the 7 seconds. Pitch and roll angles (β, α) briefly saturate at ± 10 during initial deceleration, as tilting provides the primary mechanism for horizontal control. Small transient spikes appear in angular velocities (ω) and servo deflections (δ) at $t = 0$ due to the aggressive initial corrections required from large velocity errors, these settle within 0.5 seconds without compromising constraint satisfaction or convergence. Positions drift freely since only velocities are regulated. Average throttle P_{avg} briefly reaches the 50% lower bound during vertical deceleration before stabilizing at hover thrust (67%). Differential throttle P_{diff} shows minimal activity, confirming roll stability. The tuning achieves the design objectives: high R ensures smooth x/y regulation, high Q/R compensates weak thrust authority for z, and balanced weights provide critical damping for roll.

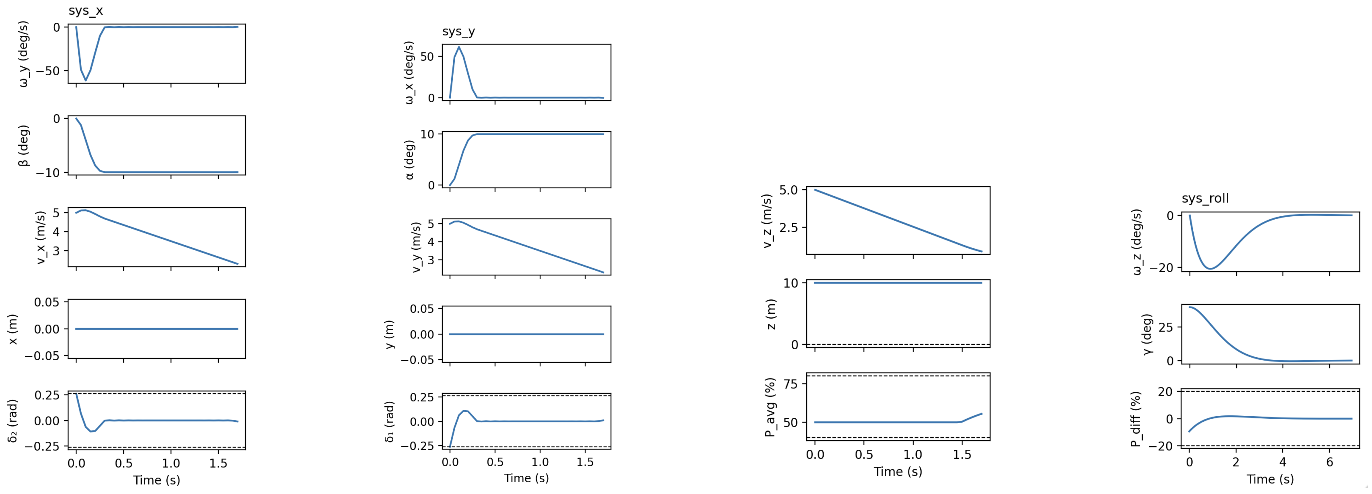


Figure 5: MPC regulation results in open-loop

In the initial open-loop prediction, during finite horizon $N = 35$ ($H = 1.75$ s = simulation time for this first optimization problem), the MPC computed an optimal trajectory where all velocities decreased linearly toward zero. It reflects the idealized prediction without feedback correction and highlights the necessity of closed-loop receding horizon control for continuous correcting deviations and a good and smooth convergence The roll subsystem already converges to zero within the open-loop prediction window, as its faster dynamics and the 1.75 s horizon are sufficient to complete the regulation task.

3.5 Deliverable 3.2: MPC Tracking Controllers

3.5.1 Test Scenario

We evaluate velocity tracking performance with constant non-zero references: Velocity references: $v_{x,r} = v_{y,r} = v_{z,r} = 3$ m/s, $\gamma_r = 35$. The MPC controllers use the same tuning parameters as regulation but now track non-zero reference velocities instead of returning to hover.

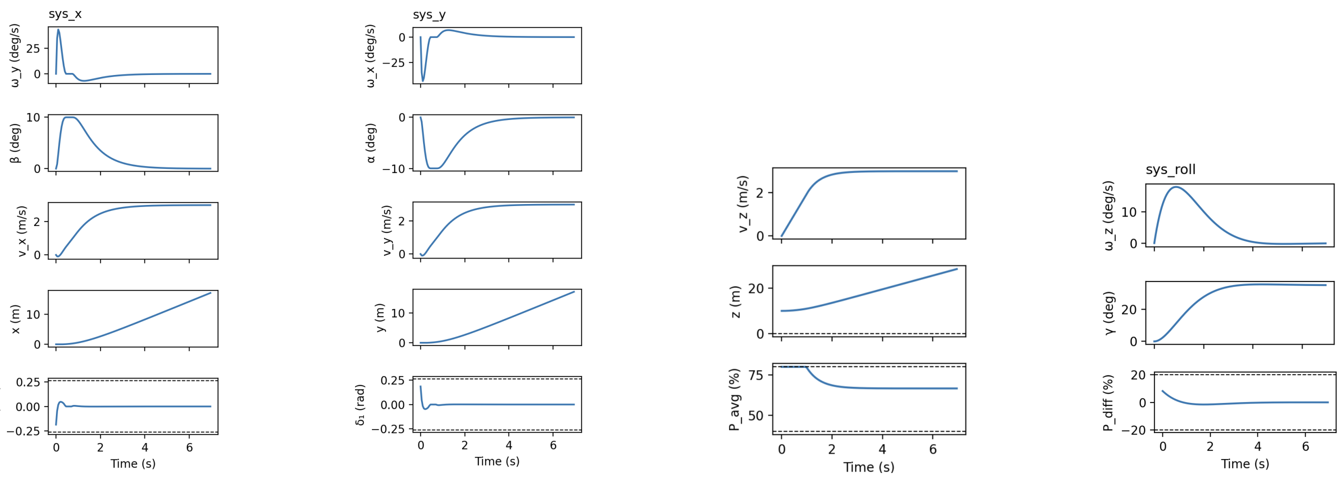


Figure 6: MPC tracking results in closed-loop

All subsystems track their references with negligible steady-state error. Compared to regulation (3.1), the control behavior is inverted: angles saturate in opposite directions (tilting to accelerate rather than decelerate), and P_{avg} saturates at the upper bound (80%) for upward acceleration instead of the lower bound (40%) for deceleration. Notably, the transient spikes in angular velocities (ω) and servo deflections (δ) at $t = 0$ are significantly reduced compared to regulation, this is because tracking from hover requires smaller initial corrections than regulating from $v = 5$ m/s, explaining also why alpha and beta saturates less.

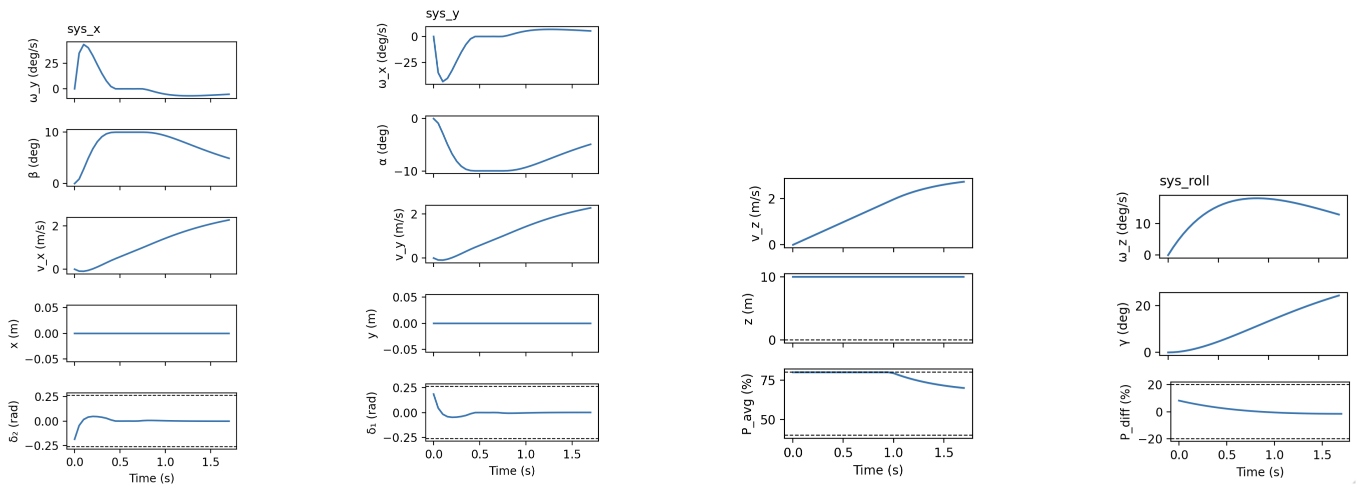


Figure 7: MPC tracking results in open-loop

Once again, the open-loop prediction exhibits rushed, overly aggressive decisions driven by the linear velocity profiles computed over the finite horizon. Unlike regulation where roll converged within 1.75 s, the tracking task with $\gamma_r = 35$ is more demanding-even roll fails to fully converge within the prediction window. The angular velocity (ω) and servo deflection (δ) transients are less sharp compared to the closed-loop curves, also showing that the initial open-loop control strategy is not maintained over time, closed-loop feedback continuously refines the maneuvers

3.6 Deliverable 3.3: Position Tracking with Cascaded PID-MPC

3.6.1 Control Architecture

For position tracking, we implement a cascaded control structure where an outer PID loop generates velocity references for the inner MPC velocity controllers. The outer PID loop operates at the same sampling rate ($T_s = 0.05$ s) and provides continuous velocity commands to the MPC layer.

3.6.2 Test Scenario

Initial position: $(50, 50, 100)$ m, Target position: $(0, 0, 10)$ m. This represents a large-displacement maneuver with significant altitude change, testing the cascaded controller's ability to handle long trajectories while respecting constraints.

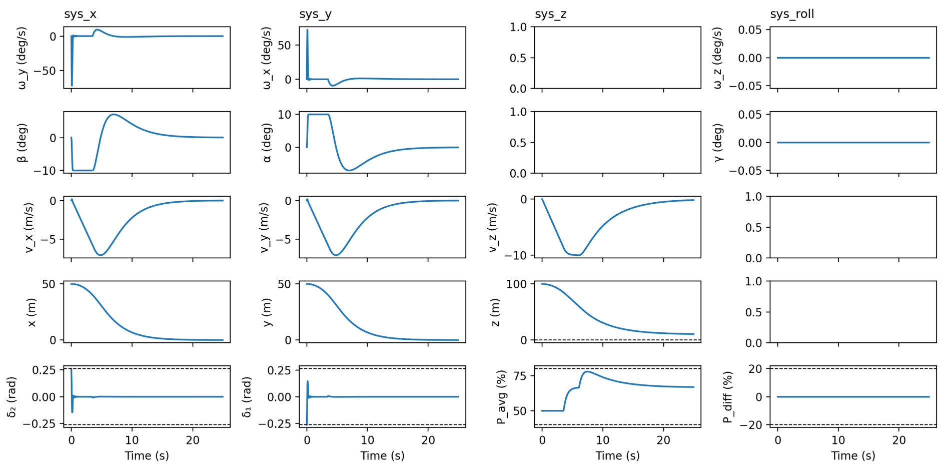


Figure 8: Position tracking using cascaded PID-MPC

The cascaded controller reaches the target in 20 seconds. Roll remains at zero as no correction is needed for this symmetric maneuver. The vertical axis exhibits aggressive behavior: v_z reaches -10 m/s to close the 90 m altitude gap, creating a characteristic dip in P_{avg} below hover thrust during rapid descent before stabilizing. The cascaded controller successfully reaches the target in approximately 20 seconds. Roll remains at zero as it is not part of the cascaded control, the PID only generates references for x, y, z velocities, while roll independently maintains $\gamma = 0$. The vertical axis exhibits aggressive behavior: P_{avg} initially start at the minimum, 50%, to accelerate downward, causing v_z to reach -10 m/s. As the rocket approaches the target altitude, P_{avg} increases to $\approx 75\%$ to brake the descent. Once v_z converges to zero at the target altitude, P_{avg} stabilizes at hover thrust (67%). Horizontal velocities (v_x, v_y) reach around -7 m/s, reflecting the PID's proportional response to the 50 m position errors. The cascaded architecture effectively achieves position tracking despite the MPC having no position weights in the cost. The outer PID closes the position loop while the inner MPC ensures smooth, constraint-compliant velocity tracking. The extended prediction horizon ($H = 6.0$ s) is critical here, it allows the MPC to anticipate the time-varying PID commands and plan constraint-compliant trajectories over the long maneuver. All constraints and objectives are satisfied throughout, demonstrating effective coordination between the outer PID loop and inner MPC velocity controllers.

4 Nonlinear Simulation

4.1 Deliverable 4.1: Parameter Adaptation

The tuning parameters developed for the linearized system (Deliverable 3) provided a robust baseline for nonlinear simulation. However, testing revealed the need for minor adjustments to account for nonlinear rocket simulator. Increasing the angular velocity weight from $Q_\omega = 1$ to $Q_\omega = 30$ improved convergence speed during position tracking maneuvers in x and y velocity controllers and a higher damping on angular rates that become significant during the aggressive maneuvers at the start of the non-linear simulation. The roll subsystem exhibited oscillations and overshoot when all four subsystems operated simultaneously.. This behavior arises from model-plant mismatch: the MPC controllers use linearized, decoupled models while the nonlinear rocket simulator exhibits coupling. When pitch and yaw angles (β, α) deviate significantly, nonlinear cross-coupling terms perturb roll dynamics beyond what the linearized model predicts. We attempted to suppress oscillations by adjusting Q and R, observing changes in frequency and amplitude but unable to fully eliminate transients. Isolated tests confirmed the roll controller performs well when decoupled, validating that coupling rather than poor tuning causes the issue. We still increased $Q_\gamma = 5000$ to aggressively penalize deviations from $\gamma = 0$, making the controller highly resistant to coupling disturbances, and increased $R = 20$ to enforce smoother differential throttle control.

Table 3: Tuning parameters for nonlinear simulation

Subsys.	**States**	**Q (diag)**	**R**
x	(ω_y, β, v_x)	$(30, 15, 0.2)$	20.0
y	(ω_x, α, v_y)	$(30, 15, 0.2)$	20.0
z	(v_z)	(50)	0.3
roll	(ω_z, γ)	$(100, 5000)$	20.0

4.2 Test Scenario

Position tracking from $(50, 50, 100)$ m to $(0, 0, 10)$ m using the full nonlinear rocket dynamics with cascaded PID-MPC control.

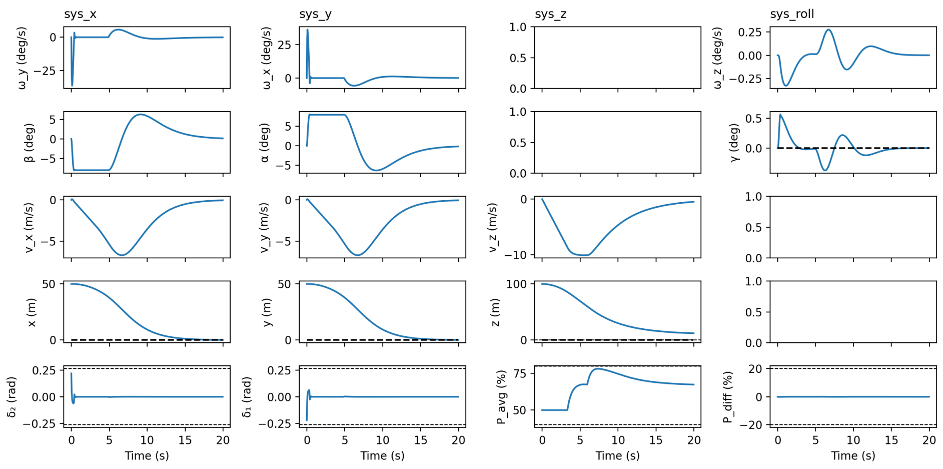


Figure 9: Nonlinear simulation: Position tracking

The nonlinear simulation successfully tracks the target position within 20 s while satisfying all constraints. The overall trajectory closely matches the linear simulation (Deliverable 3.3), confirming that the linearization-based MPC design captures the dominant system dynamics in this operating regime. Increasing the angular velocities state weights Q_ω allowed us to reduce the initial amplitude of the initial ω and δ oscillations compared to the linear case, improving transient behavior and allowed quicker convergence toward the desired coordinates. Small persistent oscillations are observed in the roll subsystem (γ ranging between -0.5 and $+0.5$, ω_z between ± 0.25 deg/s), arising from model-plant mismatch. The linearized MPC cannot anticipate nonlinear coupling effects present in the simulator during aggressive multi-axis maneuvers. This highlights both the effectiveness and the limitations of the full linearization-based approach.

5 Offset-Free Tracking

5.1 Motivation

In practice, the rocket mass deviates from the nominal value used for linearization. This mass mismatch creates a disturbance in the dynamics: for the same throttle input, a lighter rocket accelerates more than the model predicts, while a heavier rocket accelerates less. This mismatch causes steady-state tracking offsets, preventing the velocity from converging to the reference. We distinguish two scenarios: Deliverable 5.1: Mass mismatch is constant (no fuel consumption, `fuel_rate = 0`), Deliverable 5.2: Mass varies with time due to fuel consumption (`fuel_rate = 0.1`). In Deliverable 5.1, the disturbance is constant and can be estimated perfectly. In Deliverable 5.2, the time-varying mass creates a changing disturbance that challenges the observer designed for constant disturbances.

5.2 Augmented Model and Observer Design

The z-subsystem dynamics are augmented to include an unknown constant disturbance d:

$$\begin{bmatrix} x_{k+1} \\ d_{k+1} \end{bmatrix} = \begin{bmatrix} A_d & B_d \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ d_k \end{bmatrix} + \begin{bmatrix} B_d \\ 0 \end{bmatrix} u_k + \begin{bmatrix} c \\ 0 \end{bmatrix}, \quad y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ d_k \end{bmatrix} \tag{11}$$

where $c = (I - A_d)x_s - B_d u_s$ is the affine term from linearization.

A Luenberger observer estimates both state and disturbance:

$$\begin{bmatrix}\hat{x}_{k+1}\\ \hat{d}_{k+1}\end{bmatrix} = \begin{bmatrix}A_d & B_d\\ 0 & 1\end{bmatrix}\begin{bmatrix}\hat{x}_k\\ \hat{d}_k\end{bmatrix} + \begin{bmatrix}B_d\\ 0\end{bmatrix}u_k + \begin{bmatrix}c\\ 0\end{bmatrix} + L(y_k - \hat{y}_k) \tag{12}$$

5.2.1 Observer Gain Design via LQR Dual

The observer gain $L = [L_x, L_d]^T$ determines how aggressively the estimates are corrected based on measurement errors. We use the LQR dual approach because it provides a systematic, optimal way to design L by converting the observer design problem into an equivalent optimal control problem. The LQR dual method offers several advantages: Optimality: Minimizes a quadratic cost balancing estimation error vs. noise sensitivity. Stability guarantee: Ensures stable error dynamics (all eigenvalues inside unit circle). Tuning transparency: Design parameters Q_{obs} and R_{obs} have clear physical interpretations. Numerical robustness: Built-in Riccati solver handles conditioning issues. The LQR dual exploits the duality between estimation and control: designing an observer for (A, C) is equivalent to designing an LQR controller for (A^T, C^T). We compute:

$$K_{\text{obs}} = \text{dlqr}(A_{\text{aug}}^T, C_{\text{aug}}^T, Q_{\text{obs}}, R_{\text{obs}}), \quad L = K_{\text{obs}}^T \tag{13}$$

5.2.2 Observer Tuning Parameters

The design matrices Q_{obs} and R_{obs} control the observer behavior. Process Noise Covariance $Q_{\text{obs}} = \text{diag}(Q_x, Q_d)$: This matrix penalizes errors in the predicted state and disturbance: $Q_x = 1.0$: Weight on state estimation error. Moderate value balances convergence speed and transient overshoot. Higher values make the observer trust the model more, slower correction from measurements. $Q_d = 100.0$: Weight on disturbance estimation error. High value prioritizes rapid disturbance convergence. This is critical because the disturbance directly causes tracking offset-fast estimation minimizes transient offset duration. The ratio $Q_d/Q_x = 100$ reflects that disturbance estimation is the bottleneck: the state v_z is directly measured (low uncertainty), but the disturbance d is unobserved and must be inferred indirectly from measurement residuals. Measurement Noise Covariance $R_{\text{obs}} = 0.1$: This scalar penalizes reliance on noisy measurements. Lower R_{obs} trust measurements more, with faster corrections but higher noise sensitivity and a higher R_{obs} trust model predictions more, so we have smoother estimates but slower convergence. We chose $R_{\text{obs}} = 0.1$ (moderate trust) to balance responsiveness with robustness.

5.3 Target Selector and MPC Formulation

At steady-state, the system must satisfy:

$$x_s = A_d x_s + B_d u_s + B_d d + c, \quad y_s = x_s = r \tag{14}$$

The target selector computes the steady-state input that compensates for the disturbance:

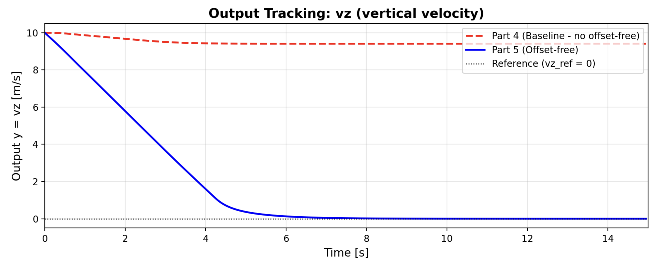
$$u_s = \frac{(1 - A_d)r - B_d\hat{d} - c}{B_d} \tag{15}$$

The MPC problem tracks this disturbance-compensated target (x_s, u_s) using the nominal dynamics. Following the project instructions, we drop the terminal set constraint as the time-varying target breaks recursive feasibility guarantees. We kept the same weights from Deliverable 4: $Q = 50$, $R = 0.3$, and terminal cost P from discrete LQR. These parameters were already tuned for good velocity tracking performance and require no modification for offset-free operation.

5.4 Deliverable 5.1: Constant Mass Mismatch

5.4.1 Test Scenario

Actual mass: 1.5 kg (constant, `fuel_rate = 0`), Initial condition: $\text{pos}_0 = [0, 0, 1]$ m, $v_0 = [5, 5, 10]$ m/s, Reference: $v_{\text{ref}} = [0, 0, 0]$ m/s.



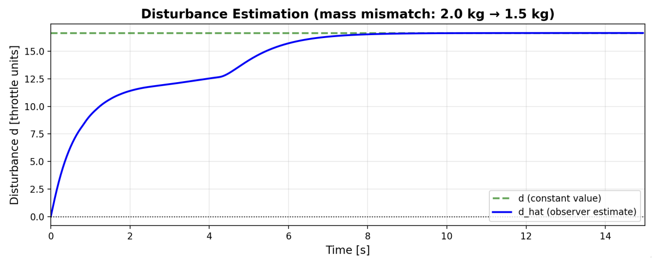


Figure 10: Left: v_z tracking (Part 4 vs 5.1). Right: Disturbance estimation

The z subsystem in Part 5.1 reaches its target despite the presence of a mass mismatch. This mismatch originates from modeling changes: the controller is designed with the nominal mass while the simulation plant mass is 1.5 kg. Since the mass is constant in this case, this difference generates a constant unmodeled gravitational force, which manifests as a persistent disturbance on the vertical dynamics. The offset-free controller's observer detects and estimates this constant disturbance. The estimator error is maximal at the start and becomes smaller with time. We observe in the right plot (Figure 10) that the estimate $\hat{d}$ starts with $\hat{d}_0 = 0$ (no prior knowledge) and converges to approximately 16.67 m/s^2 within 8 seconds. Once this disturbance is estimated and therefor the estimator error becomes 0, the controller can actively compensate for it in its control law. The left plot (Figure 10) illustrates the impact of this compensation. Part 4, without an offset-free mechanism, exhibits significant steady-state error: the vertical velocity v_z stabilizes at approximately 9.5 m/s instead of converging to the reference of 0 m/s. In contrast, Part 5.1 equipped with the offset-free controller perfectly tracks the reference and achieves $v_z = 0$, demonstrating the effectiveness of disturbance rejection through estimation.

5.5 Deliverable 5.2: Time-Varying Mass

5.5.1 Test Scenario

Initial mass: 2.0 kg, fuel consumption: 0.1 kg/s, Other conditions: same as Deliverable 5.1.

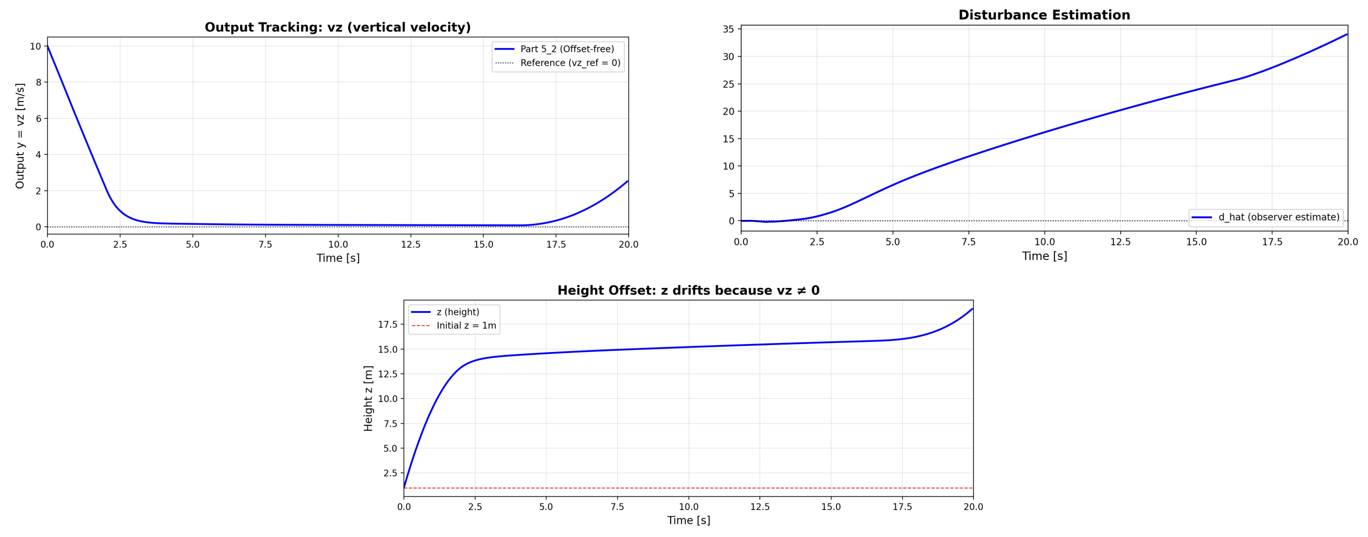


Figure 11: Time-varying mass: (top left) v_z tracking, (top right) Disturbance estimation, (bottom) Height drift

In Part 5.2, the rocket experiences a time-varying mass disturbance, which reveals several interesting phenomena in the offset-free controller's behavior. Despite the presence of a disturbance estimator, a significant height tracking offset is observed during the first seconds of the simulation. This occurs because the estimator requires time to converge: initially, the disturbance estimate $\hat{d}$ is close to zero, preventing the controller from compensating for the mass mismatch. In this scenario, the vertical velocity v_z converges, but not to zero. Instead, it stabilizes at a small positive value +0.1022 m/s, causing the height z to keep increasing slowly over time.

Unlike the constant-mass case, the disturbance d is no longer constant: the effective gravitational disturbance evolves continuously as the mass decreases. As a result, the disturbance estimator can only approximate d locally (for instance as an affine function) and cannot perfectly track its time variation Δd, which leads to a persistent steady-state error. This time-varying disturbance is therefore more difficult to cancel exactly, and the controller cannot fully compensate it using a static feedforward term. To recover convergence to $v_z = 0$ and achieve offset-free position regulation in the presence of varying mass, several improvements are possible: (i) introduce an integral action on the position z rather than on velocity, so that accumulated errors are actively corrected; (ii) design an adaptive observer that estimates both the disturbance d and its rate of change Δd, allowing faster compensation of time-varying effects; or (iii) explicitly include mass dynamics in the model and estimate the mass online, enabling direct prediction of disturbance variations instead of reacting to them.

In the time-varying mass case, an additional effect appears toward the end of the simulation: the vertical velocity v_z starts increasing again instead of converging to zero. As fuel is consumed, the rocket mass decreases continuously. As a result, the required average thrust P_{avg} to compensate gravity also decreases, keeping v_z constant after convergence, until the control input P_{avg} reaches its lower bound and saturates. At this point, the mass continues to decrease and therefore the gravitational force keeps decreasing as well, while the thrust force remains constant due to the lower saturation of P_{avg}. Before saturation, thrust and weight could compensate each other, but once P_{avg} is saturated, the thrust becomes larger than the weight. Consequently, the rocket is pulsed vertically and the controller can no longer maintain v_z close to zero.

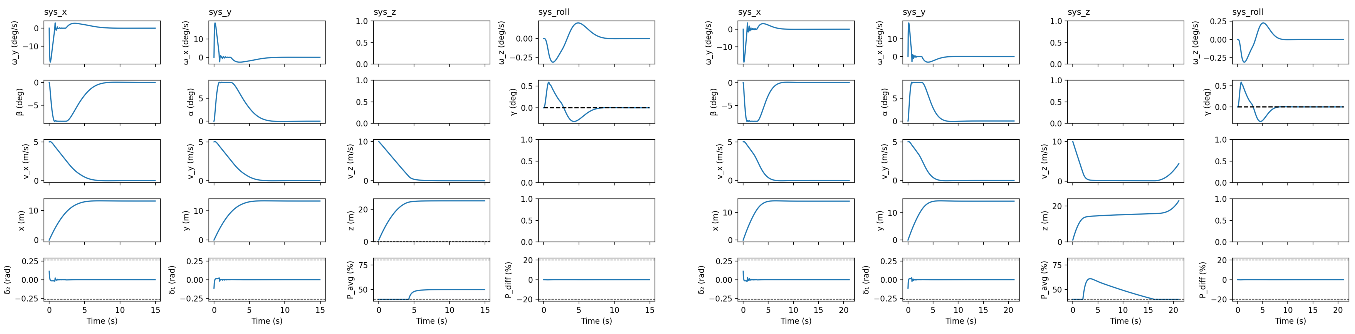


Figure 12: Closed-loop overview Deliverable 5.1 (left) and Closed-loop overview Deliverable 5.2 (right)

Despite the absence of offset-free tracking guarantees, the v_x, v_y, and roll subsystems continue to converge perfectly to their references. This is because the mass mismatch between the model and the actual system primarily affects the vertical (z) controller through gravitational force modeling errors. In the horizontal dynamics, the mass appears in both the thrust and drag terms, largely canceling out its effect on the x and y equations of motion. Similarly, the roll subsystem depends on differential thrust and inertial properties rather than absolute mass, making it insensitive to mass discrepancies.

6 Robust Tube MPC for Landing

In this section, we design a robust controller for the landing phase, where the rocket must descend and translate to a capture configuration ("chopsticks") while remaining safe near the ground. The landing objective is:

$$(x, y, z, \gamma) : (3,\ 2,\ 10,\ 30°)\ \longrightarrow\ (1,\ 0,\ 3,\ 0°),$$

with an additional near-ground safety constraint

$$z \geq 0. \tag{16}$$

6.1 Deliverable 6.1: Tube MPC for z-dimension

6.1.1 Subsystem Extraction and Uncertain Delta-Dynamics

Part 6.1 focuses on a robust tube-MPC design for the z-subsystem only. Around the new trim point, the subsystem states and input are:

$$\boldsymbol{x}_z = \begin{bmatrix} v_z \\ z \end{bmatrix}, \qquad u_z = P_{\text{avg}}.$$

Let $(\boldsymbol{x}_{s,z}, u_{s,z})$ be the trimmed equilibrium extracted from $(\boldsymbol{x}_s, \boldsymbol{u}_s)$. As in the project statement, we define delta-variables

$$\Delta\boldsymbol{x}_z := \boldsymbol{x}_z - \boldsymbol{x}_{s,z}, \qquad \Delta u_z := u_z - u_{s,z}.$$

After discretization with sampling time T_s, the uncertain linear delta-system is:

$$\Delta\boldsymbol{x}_{z,k+1} = \mathbf{A}_d\,\Delta\boldsymbol{x}_{z,k} + \mathbf{B}_d\,\Delta u_{z,k} + \mathbf{B}_d\,w_k, \qquad w_k \in \mathcal{W} := [-15,\ 5]. \tag{17}$$

The additive term $\mathbf{B}_d w$ models input-channel mismatch in the vertical acceleration: any uncertainty in thrust-to-acceleration mapping (mass variations, aerodynamics, actuator bias) appears as an effective throttle disturbance. The set $\mathcal{W}$ is asymmetric: the model may overestimate thrust reduction more severely than thrust increase, which matches typical descent conditions where under-actuation/braking uncertainty is more critical. The safety requirement becomes, in delta form, $z_k = z_s + \Delta z_k \geq 0 \iff \Delta z_k \geq -z_s$. Tube MPC will enforce a tightened nominal constraint so that the true trajectory satisfies $z \geq 0$ robustly under all admissible disturbances.

6.1.2 Tube MPC Structure

We use the standard tube MPC decomposition:

$$\Delta\boldsymbol{x}_{z,k} = \boldsymbol{z}_k + \boldsymbol{e}_k,$$

where $\boldsymbol{z}_k$ is the nominal (disturbance-free) tube center state and $\boldsymbol{e}_k$ is the bounded deviation due to disturbances and model mismatch. We apply the composite input:

$$\Delta u_{z,k} = v_k + \mathbf{K}\,\boldsymbol{e}_k = v_k + \mathbf{K}\,(\Delta\boldsymbol{x}_{z,k} - \boldsymbol{z}_k), \tag{18}$$

where v_k is the nominal MPC decision variable (computed by a QP), and $\mathbf{K}$ is a fixed stabilizing ancillary feedback gain that keeps the deviation $\boldsymbol{e}_k$ bounded. Importantly, even with full state measurement, $\boldsymbol{e}_k$ is not assumed to be zero: $\boldsymbol{z}_k$ is chosen by the optimizer to be within the tube around the measured state, and the feedback term $\mathbf{K}(\Delta\boldsymbol{x}_{z,k} - \boldsymbol{z}_k)$ provides the robust correction. The nominal state evolves without disturbances: $\boldsymbol{z}_{k+1} = \mathbf{A}_d\,\boldsymbol{z}_k + \mathbf{B}_d\,v_k$. Substituting control law into uncertain dynamics yields error dynamics: $\boldsymbol{e}_{k+1} = (\mathbf{A}_d + \mathbf{B}_d\mathbf{K})\,\boldsymbol{e}_k + \mathbf{B}_d\,w_k = \mathbf{A}_K\,\boldsymbol{e}_k + \mathbf{B}_d\,w_k$. If $\mathbf{A}_K$ is Schur-stable, then $\boldsymbol{e}_k$ remains bounded for all time under bounded w_k.

6.1.3 Choice of Stabilizing Gain K

We compute $\mathbf{K}$ from a discrete-time LQR design on $(\mathbf{A}_d, \mathbf{B}_d)$ to ensure a fast, well-damped vertical response. We selected LQR weights that strongly penalize altitude error while also damping vertical speed: $\mathbf{Q}_K = \text{diag}(q_{v_z},\, q_z)$, $\mathbf{R}_K = [r_u]$. In practice, we set $q_z \gg q_{v_z}$ to prioritize reaching $z = 3$ quickly, while q_{v_z} prevents aggressive overshoot. The resulting gain yields a stable $\mathbf{A}_K$ and a compact invariant error bound (smaller tube, less constraint tightening).

6.1.4 Robust Positive Invariant Set $\mathcal{E}$

The minimal robust positively invariant (mRPI) set $\mathcal{E}$ for error dynamics satisfies:

$$\mathbf{A}_K\mathcal{E} \oplus \mathbf{B}_d\mathcal{W} \subseteq \mathcal{E}.$$

For linear stable systems, the mRPI set can be written as the infinite Minkowski sum:

$$\mathcal{E} = \bigoplus_{i=0}^{\infty} \mathbf{A}_K^i \mathbf{B}_d \mathcal{W}.$$

We approximate this set using a finite truncation with a residual bound, as typically implemented in tube MPC toolchains. The truncation depth is chosen so that the residual contribution is negligible compared to the state tolerances used for landing (and so that the polytope remains numerically well-conditioned). A smaller $\mathcal{E}$ directly increases feasibility: it reduces constraint tightening (larger admissible nominal set) and reduces conservatism. However, shrinking $\mathcal{E}$ requires stronger feedback (larger $\|\mathbf{K}\|$), which increases $\mathbf{K}\mathcal{E}$ and may tighten input constraints more. Our tuning balances these two effects to preserve both state safety ($z \geq 0$) and enough control authority to meet the 4s settling requirement.

6.1.5 Constraint Tightening

Let the original delta-constraints for the z-subsystem be: $\Delta \boldsymbol{x}_z \in \mathcal{X}$, $\Delta u_z \in \mathcal{U}$. In particular, since $u_z = P_{\text{avg}}$, if $P_{\text{avg}} \in [40, 80]$ (percent), then: $\mathcal{U} = [40 - u_{s,z},\ 80 - u_{s,z}]$.

To guarantee robust satisfaction under disturbances, tube MPC solves the nominal optimization under the tightened sets:

$$\tilde{\mathcal{X}} := \mathcal{X} \ominus \mathcal{E}, \qquad \tilde{\mathcal{U}} := \mathcal{U} \ominus \mathbf{K}\mathcal{E}. \tag{19}$$

This ensures: $\boldsymbol{z}_k \in \tilde{\mathcal{X}}$, $v_k \in \tilde{\mathcal{U}} \Rightarrow \Delta \boldsymbol{x}_{z,k} = \boldsymbol{z}_k + \boldsymbol{e}_k \in \mathcal{X}$, $\Delta u_{z,k} = v_k + \mathbf{K}\boldsymbol{e}_k \in \mathcal{U}$ for all $w_k \in \mathcal{W}$. The ground constraint is a state constraint on z, hence part of $\mathcal{X}$. Since $\Delta z_k = z_k^{\text{nom}} + e_{z,k}$, robust satisfaction of $z_s + \Delta z_k \geq 0$ for all $e \in \mathcal{E}$ is guaranteed by enforcing the tightened (nominal) constraint $z_s + z_k^{\text{nom}} \geq -\min_{e \in \mathcal{E}} e_z$, which is exactly the z-component of $\tilde{\mathcal{X}} = \mathcal{X} \ominus \mathcal{E}$.

6.1.6 Terminal Set $\mathcal{X}_f$ and Recursive Feasibility

To guarantee recursive feasibility and stabilizing behavior, we enforce a terminal constraint on the nominal state: $\boldsymbol{z}_N \in \mathcal{X}_f$, where $\mathcal{X}_f$ is a positively invariant set for the nominal closed-loop under the local control law $v = \mathbf{K}\boldsymbol{z}$. We compute $\mathcal{X}_f$ as the maximal positive invariant subset of $\tilde{\mathcal{X}}$ under $\mathbf{A}_K$ and $\tilde{\mathcal{U}}$ using the standard backward-reachability iteration. The iteration converges in finitely many steps for polytopic sets.

6.1.7 Tuning Procedure and Final Parameters

The tube MPC controller must satisfy three requirements: maintain $z \geq 0$ under worst-case disturbances and preserve sufficient control authority for the two-phase descent maneuver (accelerate down, then brake) without taking too long to settle to the final value. We tune the controller using a sequential procedure. First, we design the ancillary LQR feedback gain $\mathbf{K}$ by selecting weights $(\mathbf{Q}_K, \mathbf{R}_K)$ that yield fast, well-damped error dynamics. This produces a stable closed-loop matrix $\mathbf{A}_K = \mathbf{A}_d + \mathbf{B}_d\mathbf{K}$ and determines the size of the minimal robust invariant set $\mathcal{E}$. Second, we compute the tightened constraint sets $\tilde{\mathcal{X}}$ and $\tilde{\mathcal{U}}$, and verify they remain feasible from the initial condition $\Delta z = 7$ m. Finally, we select the MPC horizon N and stage weights $(\mathbf{Q}, \mathbf{R})$ to meet the settling-time specification while respecting the tightened constraints.

Our initial tuning used weights $q_{v_z} = 200$, $q_z = 500$, and $r = 0.5$, prioritizing tight velocity control to prevent overshoot and excessively negative v_z velocities. This configuration achieved excellent nominal performance (very fast settling time) and good convergence to $z = 3$ m under random disturbances. However, under extreme disturbances, the controller failed: the rocket descended too aggressively, v_z became excessively negative, and despite saturating at $P_{\text{avg}} = 80\%$, the controller lacked sufficient authority to brake in time, violating the ground constraint $z < 0$. Since $z \geq 0$ is our critical safety constraint, we increased its relative importance by adjusting the ratio q_z/q_{v_z}. Reducing q_{v_z} to 50 (while keeping $q_z = 500$) resolved the extreme-case failure. Through iterative tuning, we observed a three-way trade-off between settling time, convergence precision, and robustness under extreme disturbances. The final parameters ($q_z = 500$, $q_{v_z} = 50$, $r = 0.5$) achieve a 5 seconds settling under no noise and maintain $z \geq 0$ under random disturbances. We use identical weights for LQR and MPC ($\mathbf{Q}_K = \mathbf{Q}$, $\mathbf{R}_K = \mathbf{R}$) to simplify tuning, and a 6-second horizon to accommodate the full descent maneuver.

6.1.8 Invariant Set Plots and Tightened Input Vertices

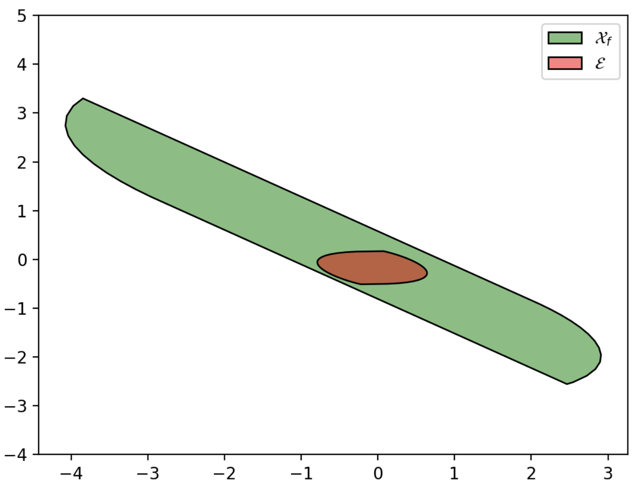


Figure 13: Invariant sets for Tube MPC z-controller: $\mathcal{E}$ and $\mathcal{X}_f$

Our code outputs the following tightened nominal endpoints: $\tilde{\mathcal{U}} = [\tilde{u}_{\min}, \tilde{u}_{\max}] = [-7.2247,\ 1.7762]$, where these values are expressed in percentage points around the trim input $u_{s,z}$ (i.e., in Δu_z coordinates for the nominal variable v).
Therefore, the corresponding tightened absolute nominal throttle interval is: $P_{\text{avg}}^{\text{nom}} = u_{s,z} + v_k \in [u_{s,z} + \tilde{u}_{\min},\ u_{s,z} + \tilde{u}_{\max}] = [49.4419,\ 58.4428]\%$. With $u_{s,z} = 56.6667\%$, this nominal interval is strictly contained in the physical bounds $P_{\text{avg}} \in [40, 80]\%$. The remaining margin is precisely what allows the additive feedback correction $\mathbf{K}e_k$ to act while ensuring the true applied input respects the physical limits under all $w \in \mathcal{W}$.

6.1.9 Closed-Loop Performance Under No-Noise, Random, and Extreme Disturbances

We test the controller using the provided simulator modes: 'no noise' ($w_k \equiv 0$ baseline nominal performance), 'random' (bounded random draws $w_k \in [-15, 5]$ at each step), and 'extreme' (worst-case persistent disturbance applied with sign that most challenges constraint satisfaction).

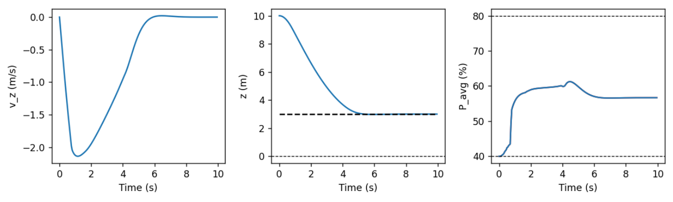


Figure 14: Robust Tube MPC (z only): no disturbance

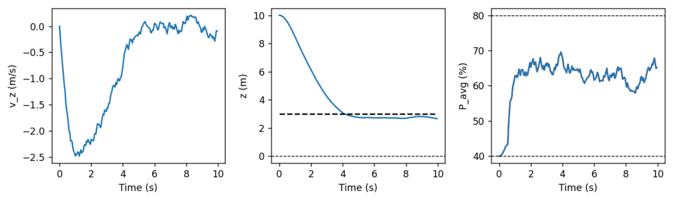


Figure 15: Robust Tube MPC (z only): random disturbances

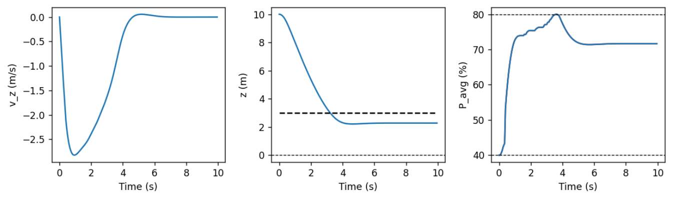


Figure 16: Robust Tube MPC (z only): extreme disturbance

Across all three modes, the controller: Maintains strict safety ($z \geq 0$) due to constraint tightening and tube invariance. Produces the expected two-phase descent: P_{avg} first increases (fast downward acceleration), then creases towards the upper bound (braking) to settle at $z = 3$. Exhibits oscillations in v_z and z under random disturbances because w_k varies randomly between -15 and 5 at each time step, forcing continuous corrective action. In contrast, the extreme case produces smooth trajectories since the disturbance remains constant at $w = -15$ throughout the descent. Shows different steady-state convergence: the extreme case converges to $z \approx 2.45$ m (offset from the target $z = 3$ m due to the persistent worst-case disturbance $w = -15$), while the random case converges closer to the target at $z \approx 2.80$ m. The larger steady-state error under extreme disturbances is expected, the constant worst-case input mismatch creates a persistent offset that the tube MPC accommodates while maintaining feasibility and safety. The no-noise plot validates nominal performance and tuning, while the extreme disturbance case is most informative for robustness: it validates that even adversarial mismatch cannot drive the true altitude below zero nor prevent convergence to the target neighborhood.

6.2 Deliverable 6.2: Merged Controller (Nominal x/y/roll + Robust z)

In Part 6.2, we design nominal MPC position controllers for sys_x, sys_y, and sys_{roll}, then merge them with the robust tube MPC controller for sys_z designed in Part 6.1. The merged controller is applied to the full nonlinear rocket model.

6.2.1 Nominal Position MPC Design for x, y, and Roll

The three subsystems (from the linearized decomposition) are:

$$\text{sys}_x : \ \boldsymbol{x}_x = [\omega_y,\ \beta,\ v_x,\ x]^\top,\ u_x = \delta_2,$$

$$\text{sys}_y : \ \boldsymbol{x}_y = [\omega_x,\ \alpha,\ v_y,\ y]^\top,\ u_y = \delta_1,$$

$$\text{sys}_{\text{roll}} : \ \boldsymbol{x}_\gamma = [\omega_z,\ \gamma]^\top,\ u_\gamma = P_{\text{diff}}.$$

We re-linearize around the same landing trim $(\boldsymbol{x}_s, \boldsymbol{u}_s)$, extract each subsystem $(\mathbf{A}_{d,i}, \mathbf{B}_{d,i})$, and formulate nominal MPC tracking the position targets $x \to 1$, $y \to 0$, $\gamma \to 0$, with soft constraints and no terminal constraint (as requested). These three subsystems are simulated on the nonlinear plant, and the linear model mismatch is larger than in the isolated subsystem simulations. Hard constraints may cause infeasibility during aggressive transients (especially for α, β near their bounds). Soft constraints maintain closed-loop continuity: violations (if any) are penalized heavily but do not break feasibility, which is crucial in a multi-axis landing maneuver.

6.2.2 Tuning Choices and Final Parameters

The main tuning goal in Part 6.2 is fast convergence (settling time $\leq$ 4s) from $(3, 2, 10, 30°)$ to $(1, 0, 3, 0°)$ on the nonlinear model. The main practical trade-offs are: Larger position weights achieve faster translation but more gimbal saturation. Larger input weight produces smoother gimbal commands but slower settling. Strong slack penalty ensures constraint-respecting behavior, but may create aggressive inputs to avoid any violation.

Table 4: Final tuning parameters for nominal landing MPC controllers (Part 6.2)

Controller	$\mathbf{Q}_i$ **(diag)**	$\mathbf{R}_i$	ρ_i
sys_x (δ_2)	$(10, 80, 6, 350)$	8	10^5
sys_y (δ_1)	$(10, 80, 6, 350)$	8	10^5
sys_{roll} (P_{diff})	$(30, 600)$	5	10^5

We kept a common horizon $N = 30$ ($H = 1.5$s) for the nominal controllers to keep the merged controller computationally lightweight. The robust z tube MPC uses a longer horizon ($N = 120$) to robustly plan the full accelerate/brake descent while guaranteeing tightened constraint satisfaction near the ground.

6.2.3 Controller Fusion

The class `MPCLandControl` merges the four independent controllers by: Evaluating each subsystem controller on the current full state $\boldsymbol{x}$ (each controller extracts its own subsystem state). Returning the combined input vector in the rocket input ordering: $\boldsymbol{u} = \begin{bmatrix}\delta_1 & \delta_2 & P_{\text{avg}} & P_{\text{diff}}\end{bmatrix}^\top$. In our implementation, the robust z-controller provides P_{avg} while the three nominal controllers provide δ_1, δ_2, and P_{diff}.

6.2.4 Closed-Loop Landing Results on the Nonlinear Model

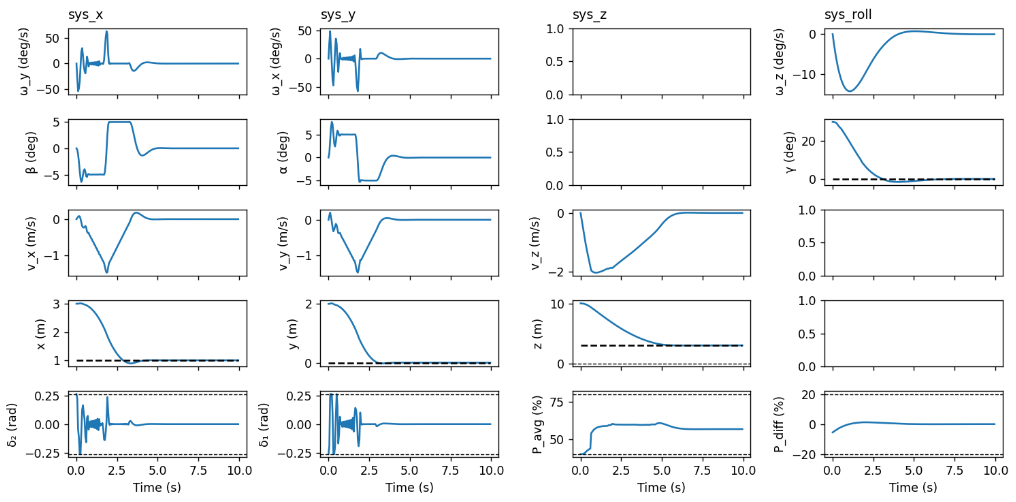


Figure 17: Nonlinear landing simulation with merged controller (Part 6.2)

The merged controller achieves the landing objective within the required settling time: Translation: x decreases from 3 to 1 and y from 2 to 0 by commanding transient tilts (α, β) through gimbal deflections δ_1, δ_2 with a settling time around 3s. Descent: z decreases from 10 to 3 in a fast accelerate/brake profile controlled by the robust tube MPC; throughout the maneuver, $z \geq 0$ is satisfied. Attitude: γ converges to $0°$ via P_{diff} while the translation controllers are active, showing that the decomposition remains effective in this operating region. Inputs and validity: δ_1, δ_2 remain within $\pm 15°$ and P_{avg} within $[40, 80]\%$. Angle constraints (for linearization validity) are mostly respected; any small transient excess is handled by the soft-constraint formulation without feasibility loss. Importantly, using a robust z-controller while keeping nominal $x/y/\gamma$ controllers is a practical compromise: vertical safety near ground is guaranteed under mismatch, while lateral motion remains fast and computationally inexpensive.

7 Nonlinear MPC

In Part 6, the landing controller relied on linearized, decoupled subsystem models (with a robust tube design only for z). In this section, we implement a nonlinear MPC (NMPC) controller using CASADI that operates on the full nonlinear rocket model and outputs all four inputs simultaneously: $\boldsymbol{u} = \begin{bmatrix}\delta_1 & \delta_2 & P_{\text{avg}} & P_{\text{diff}}\end{bmatrix}^\top$. The NMPC formulation naturally captures coupling between translation and attitude, which becomes relevant during aggressive near-ground maneuvers.

7.1 Deliverable 7.1: NMPC Controller Design

7.1.1 Constraints and Numerical Safety

The rocket model uses Euler angles (α, β, γ), which have a singularity at $\beta = 90°$. Even though this is not expected in normal landing, to prevent numerical issues we impose: $|\beta| \leq 80°$. We also enforce the ground constraint: $z \geq 0$. Finally, we enforce the physical input constraints: $|\delta_1| \leq 15°$, $|\delta_2| \leq 15°$, $40\% \leq P_{\text{avg}} \leq 80\%$, $|P_{\text{diff}}| \leq 20\%$.

7.1.2 NMPC Discretization and Decision Variables

Let the full state be $\boldsymbol{x} = \begin{bmatrix}\boldsymbol{\omega}^\top & \boldsymbol{\phi}^\top & \boldsymbol{v}^\top & \boldsymbol{p}^\top\end{bmatrix}^\top \in \mathbb{R}^{12}$, and the continuous dynamics $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$. NMPC uses a discrete-time prediction model; we discretize the continuous-time model with an explicit Runge–Kutta scheme (RK4) using step T_s, yielding the discrete mapping $\boldsymbol{x}_{k+1} = F(\boldsymbol{x}_k, \boldsymbol{u}_k)$, where $F(\cdot)$ denotes one RK4 integration step. We adopt a multiple-shooting transcription with decision variables: $\boldsymbol{X} = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N\}$, $\boldsymbol{U} = \{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{N-1}\}$. Multiple shooting improves numerical robustness compared to single shooting, and it enables effective warm-starting by shifting previous optimal trajectories.

7.1.3 NMPC Objective (Stage + Terminal Cost)

The landing objective is to reach the trim around $(x, y, z, \gamma) = (1, 0, 3, 0)$ with near-zero velocities and small angles. We use a quadratic tracking cost with input-rate penalty $\Delta\boldsymbol{u}_k = \boldsymbol{u}_k - \boldsymbol{u}_{k-1}$ to reduce gimbal chatter and improve numerical robustness. Because NMPC horizons must remain short for computational reasons, the terminal cost $\mathbf{Q}_f$ is crucial. We compute $\mathbf{Q}_f$ once from an LQR design at the landing trim: Linearize the continuous dynamics at $(\boldsymbol{x}_s, \boldsymbol{u}_s)$ and discretize to $(\mathbf{A}_d, \mathbf{B}_d)$. Solve the discrete-time Riccati equation with weights consistent with $\mathbf{Q}$ and $\mathbf{R}$. Set $\mathbf{Q}_f = \mathbf{P}$ (DARE solution). This gives a strong "tail approximation" and noticeably improves convergence without increasing N.

7.1.4 Tuning Parameters

We tuned NMPC to satisfy the same settling requirement as Part 6.2 while maintaining solver speed and avoiding oscillatory gimbal commands. The final parameters are:

Table 5: Final tuning parameters for NMPC (Part 7.1)

Parameter	**Selected value**
Horizon	$N = 30$ steps ($H = 1.5$ s)
State weights $\mathbf{Q}$	$\mathrm{diag}(5, 5, 5, 60, 80, 120, 8, 8, 25, 250, 250, 600)$
Input weights $\mathbf{R}$	$\mathrm{diag}(30, 30, 0.6, 2.0)$
Input-rate weights $\mathbf{R}_{\Delta u}$	$\mathrm{diag}(200, 200, 0.05, 5)$
Terminal weight	$\mathbf{Q}_f = \mathbf{P}$ from LQR at landing trim

The diagonal structure of $\mathbf{Q}$ is chosen to prioritize position capture by weighting (x, y, z) most strongly, while weights on velocities and angular rates provide damping and reduce overshoot. The angle-related weights (with a stronger emphasis on β) help keep attitudes moderate and improve numerical robustness near the Euler singularity. Finally, the input-rate penalty $\mathbf{R}_{\Delta u}$ is included to suppress high-frequency gimbal motion (chatter) while maintaining fast convergence. The largest weights are placed on (x, y, z), with z the highest to ensure a tight altitude capture. Euler angles are weighted to keep attitudes moderate and avoid the Euler singularity region; β is more heavily weighted than α because it is directly constrained and is critical for numerical stability. Velocities and angular velocities are weighted to add damping and avoid aggressive oscillations. High penalties on δ_1, δ_2 and their rates reduce gimbal chatter. Throttle P_{avg} is penalized moderately so the optimizer still uses saturation when needed for accelerate/brake, while P_{diff} remains smooth to prevent roll oscillations.

7.1.5 CASADI Implementation Details

We implement NMPC in `nmpc_land.py` using CASADI `Opti`: Multiple shooting with RK4 discretization for improved robustness and consistent constraint enforcement along the trajectory. Warm-start by shifting the previously optimal state and input trajectories, which significantly reduces IPOPT iterations and improves reliability. Solver settings chosen to reduce overhead and stabilize convergence (e.g., `expand=True`, low print level, limited iterations). Single linearization for terminal cost: the LQR-based $\mathbf{Q}_f$ is computed once at the landing trim and reused at every NMPC solve.

7.1.6 Open-Loop Sanity Check

Before closing the loop, we verify that the open-loop optimal trajectory from a representative initial state is physically reasonable (smooth approach, feasible inputs, no ground or β constraint violations).

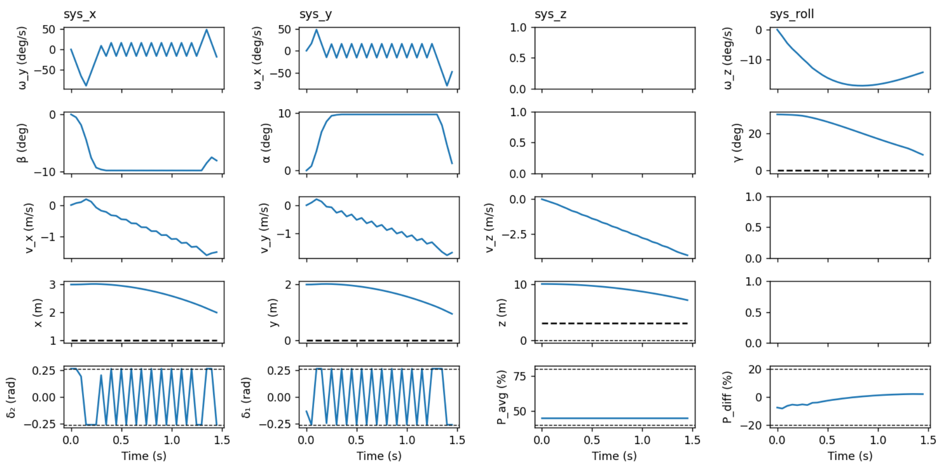


Figure 18: NMPC open-loop optimal trajectory check

7.1.7 Closed-Loop NMPC Landing Performance

We then simulate NMPC in closed loop on the nonlinear plant and evaluate the closed-loop trajectory and inputs.

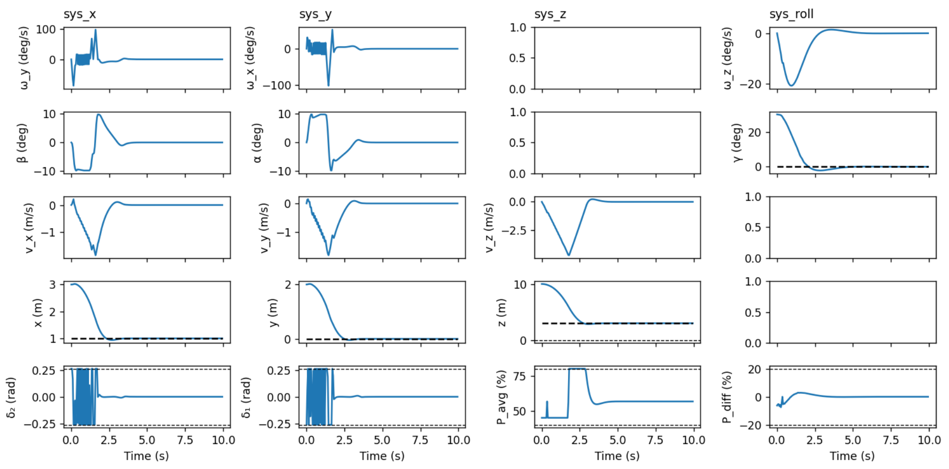


Figure 19: Closed-loop NMPC landing performance on nonlinear rocket model

NMPC achieves fast landing because it plans over the full coupled dynamics: The optimizer directly trades off lateral translation and attitude motion, reducing overshoot that may appear when combining independent linear controllers. Input-rate penalties significantly reduce high-frequency oscillations of δ_1, δ_2, improving smoothness near touchdown. Constraints $z \geq 0$ and $|\beta| \leq 80°$ are enforced explicitly throughout the prediction horizon and remain satisfied in closed loop.

7.1.8 Comparison: NMPC vs. (Tube MPC z + Nominal MPC $x/y/\gamma$)

Both accomplish the landing maneuver from $(x, y, z, \gamma) = (3, 2, 10, 30°)$ to $(1, 0, 3, 0°)$ within the required $\leq$ 4s, while satisfying $z \geq 0$ and respecting actuator bounds. The main difference is the transient control activity (especially gimbal commands) and the associated angular-rate peaks.

Table 6: Key empirical differences observed in closed-loop plots

Metric	**NMPC**	**Merged linear MPC**
Settling time (x, y, z)	All positions reach target neighborhood in $\approx$ 3s with smooth convergence profiles.	Similar settling time in $\approx$ 3s with comparable trajectory smoothness.
Tilt angles (α, β)	Larger transient excursions (approximately $\pm 10°$), still far from safety constraint $\|\beta\| \leq 80°$.	More conservative tilt angles (± 5), consistent with linear model validity region.
Angular velocities (ω_x, ω_y)	Sharper transient spikes reaching up to $\sim$ 100 deg/s, aligned with aggressive gimbal switching behavior.	Lower peak values ($\sim$ 50 deg/s), reflecting smoother attitude control transients.
Gimbal deflections (δ_1, δ_2)	Clear early-time chattering behavior: rapid switching near $\pm 15°$ constraints for approximately 2 seconds.	Oscillations present but significantly reduced (fewer rapid sign changes, faster decay to steady state).
Average throttle (P_{avg})	Does not remain at physical minimum initially: it rises toward upper bound to brake vertical motion, then settles near hover trim.	More pronounced oscillations: near 40% initially (acceleration phase), then near 80% (braking phase), settling near hover trim.
Robustness guarantees	Good practical performance in simulation, but no formal tube guarantee (disturbances handled implicitly by nonlinear model).	Formal robustness in vertical channel via tube MPC framework with explicit disturbance bounds.

The dominant empirical difference is the NMPC gimbal chattering and the associated peaks in ω_x and ω_y. This behavior is consistent with the nonconvex nature of NMPC: the optimization may admit multiple locally optimal solutions, and small changes in the measured state or in the warm-start can cause the solver to switch between different control profiles at successive sampling instants. With a relatively short horizon ($H = 1.5\,\text{s}$) and strong position/attitude weights, the optimal solution may therefore alternate between near-saturated gimbal commands unless explicit hard rate constraints are imposed. The quadratic input-rate penalty reduces this effect but does not fully eliminate step-to-step switching when tracking terms dominate.

In contrast, the merged linear MPC strategy solves convex QPs on decoupled linear subsystems. Convexity and consistent warm-starting typically yield more repeatable solutions, which matches the smoother δ_1, δ_2 traces and lower angular-rate peaks observed in our simulations. Overall, NMPC provides coupling-aware optimization over the full nonlinear dynamics and can exploit larger attitude excursions when needed, at the cost of higher transient control activity and greater sensitivity to solver behavior. The merged linear approach is computationally lightweight and produces smoother actuator commands in our tests, while retaining formal robustness margins in the safety-critical z channel through the tube MPC design.

8 Conclusion

This project developed MPC-based controllers for rocket regulation, tracking, and landing, progressing from linearized subsystem designs to robust and nonlinear optimization-based strategies. After trimming and linearizing the nonlinear model, the dynamics were decomposed into four independent subsystems, enabling efficient design of constrained MPC velocity controllers for x, y, and z, and an MPC roll controller. Terminal costs and invariant terminal sets ensured stabilizing behavior and recursive feasibility in the nominal linear setting.

When applied to the nonlinear simulator, tuning adjustments were required to mitigate coupling and model mismatch. Offset-free tracking for the vertical subsystem was achieved by augmenting the model with a disturbance estimate, and robust tube MPC was used in the landing phase to enforce $z \geq 0$ under bounded disturbances. Finally, a full NMPC controller implemented in CasADi provided a coupled alternative with explicit nonlinear constraints.

References

[1] EPFL, *ME-425 Model Predictive Control: Course material and mini-project framework*, course repository and Moodle handouts, 2025. Available at: https://github.com/PREDICT-EPFL/MPC-Course-EPFL/tree/main.