MODEL PREDICTIVE CONTROL

ME-425

An MPC Controller to Land a Rocket Under Thrust

Professors:

PROF. COLIN JONES

Group AI:

ZHANG Xinyue (385865)

RAUBER Jean Bernard (330764)

MAHAPATRA Shreesh (404681)

Lausanne, Switzerland, January 11th 2026

Contents

1 Introduction

This project is conducted as part of the EPFL course ME-425: Model Predictive Control and focuses on the design and implementation of Model Predictive Control (MPC) strategies for a thrust-controlled rocket. Starting from the nonlinear rocket dynamics, the system is linearized around a hover equilibrium and decomposed into four weakly coupled subsystems. For each subsystem, stabilizing MPC controllers with state and input constraints are designed, ensuring recursive feasibility through appropriate terminal ingredients.
The control framework is then extended to reference tracking and position control, and the performance of the linear MPC controllers is evaluated on the full nonlinear rocket model. To address modeling errors caused by mass variations, an offset-free MPC scheme with disturbance estimation is introduced for the vertical dynamics. Furthermore, a robust tube MPC controller is developed to guarantee constraint satisfaction during the landing phase under bounded disturbances. Finally, a nonlinear MPC (NMPC) controller is implemented and its performance is compared with the previously developed linear and robust controllers.

2 Deliverable 2: Linearization

2.1 Deliverable 2.1

From a physical perspective, each actuator influences only one dominant motion mode when the rocket is close to the upright hover configuration. This is why the linearized dynamics can be separated into four independent subsystems.
The full state vector used in this project is

$$\mathbf{x} = [\omega_x,\ \omega_y,\ \omega_z,\ \alpha,\ \beta,\ \gamma,\ v_x,\ v_y,\ v_z,\ x,\ y,\ z]^\top,$$

and the input vector is

$$\mathbf{u} = [\delta_1,\ \delta_2,\ P_{\text{avg}},\ P_{\text{diff}}]^\top.$$

In this project, the trim point for linearization is not fixed at the origin. Instead, we may specify a reference state $\mathbf{x}_{\text{ref}}$ at which the rocket should hover, i.e., a state $(\mathbf{x}_s, \mathbf{u}_s)$ satisfying $f(\mathbf{x}_s, \mathbf{u}_s) = 0$. In the implementation, we use

```
x_ref = np.array([0.0] * 9 + [1.0, 0.0, 3.0])
xs, us = rocket.trim(x_ref)
```

which instructs the trimming algorithm to find an equilibrium at the position $(x, y, z) = (1, 0, 3)$ where the rocket remains upright and stationary.
The trim computation returns

$$\mathbf{x}_s = [0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 0,\ 3]^\top, \qquad \mathbf{u}_s = [0,\ 0,\ 66.67,\ 0]^\top.$$

After computing the trim point, we linearize the nonlinear model around $(\mathbf{x}_s, \mathbf{u}_s)$, and decompose the resulting model into four independent subsystems. The resulting subsystems are shown below:

1. An independent system based on δ_1 input:

$$\dot{x}_y = \begin{bmatrix} \dot{\omega}_x \\ \dot{\alpha} \\ \dot{v}_y \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -9.81 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \omega_x \\ \alpha \\ v_y \\ y \end{bmatrix} + \begin{bmatrix} -65.47 \\ 0 \\ -9.81 \\ 0 \end{bmatrix} \delta_1 \tag{1}$$

2. An independent system based on δ_2 input:

$$\dot{x}_x = \begin{bmatrix} \dot{\omega}_y \\ \dot{\beta} \\ \dot{v}_x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 9.81 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \omega_y \\ \beta \\ v_x \\ x \end{bmatrix} + \begin{bmatrix} -65.47 \\ 0 \\ 9.81 \\ 0 \end{bmatrix} \delta_2 \tag{2}$$

3. An independent system based on P_{avg} input:

$$\dot{x}_z = \begin{bmatrix} \dot{v}_z \\ \dot{z} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v_z \\ z \end{bmatrix} + \begin{bmatrix} 0.1471 \\ 0 \end{bmatrix} P_{\text{avg}} \tag{3}$$

4. An independent system based on P_{diff} input:

$$\dot{x}_{\text{roll}} = \begin{bmatrix} \dot{\omega}_z \\ \dot{\gamma} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \omega_z \\ \gamma \end{bmatrix} + \begin{bmatrix} 0.104 \\ 0 \end{bmatrix} P_{\text{diff}} \tag{4}$$

These input–state relationships follow from the geometry of the propulsion system: each actuator changes the thrust vector or torque along a distinct axis. Although the nonlinear system contains weak coupling (e.g., tilting the thrust slightly reduces the vertical thrust component), such effects are second order and disappear after linearizing about the hover equilibrium. As a result, each input excites only a specific subset of states, making the linearized system naturally decompose into four independent subsystems.

3 Deliverable 3: MPC Velocity Controllers for Each Sub-System

In the Section 2 we already linearized the whole rocket system and seperated it into 4 independent sub-systems for x, y, z, and roll. Now in Section 3, we will design a recursively feasible, stabilizing MPC controller for reaching close to the landing platform.

3.1 Deliverable 3.1: MPC Controller for Stabilization

3.1.1 Design Procedure

When designing an MPC controller that ensures recursive constraint satisfaction, the key requirement is that if the optimization problem is feasible at the current time step, it must remain feasible for all future time steps. This property is referred to as recursive feasibility, and achieving it requires splitting the infinite-horizon control problem into two parts.

From step 0 to step N, we solve a finite-horizon MPC problem with explicit state and input constraints. From step N to infinity, instead of continuing the constrained optimization indefinitely, we replace the tail of the horizon with a stabilizing terminal control law and an associated terminal invariant set.

To guarantee stability, the terminal cost is chosen as a Lyapunov function

$$V_f(x_N) = x_N^\top Q_f x_N,$$

where Q_f is obtained from the solution of the Discrete Algebraic Riccati Equation (DARE) associated with the unconstrained LQR problem. Solving the LQR also yields the stabilizing terminal feedback law

$$u = Kx.$$

To ensure recursive feasibility under this terminal control law, we compute the maximum invariant set X_f, defined as the largest set satisfying

$$x \in X_f \implies (A + BK)x \in X_f,$$

while respecting all state and input constraints.
Imposing the terminal constraint $x_N \in X_f$ guarantees that once the state enters this set, the terminal feedback law $u = Kx$ can drive the system to the equilibrium without violating any constraints.
This results in the standard finite-horizon MPC formulation used in our project:

$$\begin{aligned} J^\star(x_0) = \min_{x_0,\dots,x_N,\, u_0,\dots,u_{N-1}} \quad & \sum_{i=0}^{N-1} \left(x_i^\top Q x_i + u_i^\top R u_i\right) + x_N^\top Q_f x_N \\ \text{s.t.} \quad & x_{i+1} = Ax_i + Bu_i, \\ & Fx_i \le f, \quad i = 0,\dots,N-1, \\ & Mu_i \le m, \quad i = 0,\dots,N-1, \\ & x_N \in X_f. \end{aligned}$$

The same design procedure is used for each subsystem (x, y, z, and roll), with subsystem-specific constraints summarized in Table 1.

Subsystem	State constraints	Input constraints
MPC x controller	$\lvert\beta\rvert \le 10^\circ = 0.1745$ rad	$\lvert\delta_2\rvert \le 15^\circ = 0.26$ rad
MPC y controller	$\lvert\alpha\rvert \le 10^\circ = 0.1745$ rad	$\lvert\delta_1\rvert \le 15^\circ = 0.26$ rad
MPC z controller	—	$40\% \le P_{\text{avg}} \le 80\%$
MPC roll controller	—	$\lvert P_{\text{diff}}\rvert \le 20\%$

Table 1: State and input constraints for each MPC subsystem.

3.1.2 Choice of Tuning Parameters

The MPC controller requires tuning several key parameters: the state-weight matrix Q, the input-weight matrix R, the prediction horizon H, and the terminal components (K, Q_f). Each of these parameters influences the performance, stability, and feasibility of the closed-loop system.

State and Input Weight Matrices Q and R The matrix Q penalizes deviations in the state variables and therefore affects the convergence speed of the subsystem. Larger entries in Q force the states to converge more rapidly, but may introduce overshoot or oscillatory behaviour. The matrix R penalizes the magnitude of the

control inputs. Increasing R results in smoother, less aggressive control actions, whereas decreasing R produces faster but potentially more oscillatory responses.

Prediction Horizon H The horizon length determines how far into the future the MPC predicts. The number of prediction steps is given by

$$N = \frac{H}{T_s},$$

where T_s is the sampling time. If H is too small, the solver may be unable to reach the terminal set X_f in N steps, leading to infeasibility. Conversely, a very large horizon increases computation time without significant improvement in performance.

Terminal Components (K, Q_f) The terminal feedback gain K and terminal cost matrix Q_f are computed by $K,\ Q_f = \texttt{dlqr}(A,\ B,\ Q,\ R)$ in Python. The terminal set X_f is then constructed as a positively invariant set under the terminal controller $u = Kx$, while satisfying all state and input constraints. Enforcing the terminal constraint $x_N \in X_f$ ensures that the MPC problem remains feasible at every future time step, and is therefore a key component in guaranteeing recursive feasibility and closed-loop stability.

Selected Parameters The complete set of parameters used for all four subsystems is summarised in Table 3. These choices reflect the relative importance of the states, the desired convergence performance, and the actuator limitations specific to each subsystem.

Subsystem	Q	R
MPC x controller	$\mathrm{diag}(0.1,\ 0.8,\ 0.02)$	100
MPC y controller	$\mathrm{diag}(0.1,\ 0.8,\ 0.02)$	100
MPC z controller	100	3.5
MPC roll controller	$50I_2$	0.1

Table 2: Summary of chosen parameters for all MPC subsystems.

3.1.3 Terminal Invariant Set

The terminal invariant set X_f for each subsystem is computed based on the property of positive invariance. Consider the closed-loop autonomous system

$$x_{k+1} = A_{\mathrm{cl}} x_k, \qquad A_{\mathrm{cl}} = A + BK.$$

A set C is positively invariant for this system if and only if

$$x \in C \ \Rightarrow\ A_{\mathrm{cl}} x \in C,$$

which can be equivalently written as

$$C \subseteq \mathrm{pre}(C),$$

where the pre-set is defined as

$$\mathrm{pre}(C) = \{\, x \mid A_{\mathrm{cl}} x \in C \,\}.$$

This leads to the following fixed-point iteration for computing the maximal invariant set:

Conceptual Algorithm to Compute Invariant Set
$\Omega_0 \leftarrow X$
loop
$\Omega_{i+1} \leftarrow \text{pre}(\Omega_i) \cap \Omega_i$
if $\Omega_{i+1} = \Omega_i$ **then return** $\Omega_\infty = \Omega_i$
end loop

In practice, we initialize the iteration with the polyhedron defined by the state and input constraints expressed in the reduced subsystem coordinates, and apply A_{cl} recursively until the set converges.

For the x- and y-subsystems, the reduced state vector has dimension three (angular rate, attitude angle, and velocity; position states are not included in these subsystems). Therefore their terminal sets $X_f \subset \mathbb{R}^3$ are visualised through 2D projections $\text{proj}_{(0,1)}(X_f)$ and $\text{proj}_{(1,2)}(X_f)$, shown in Fig. 1 and Fig. 2.

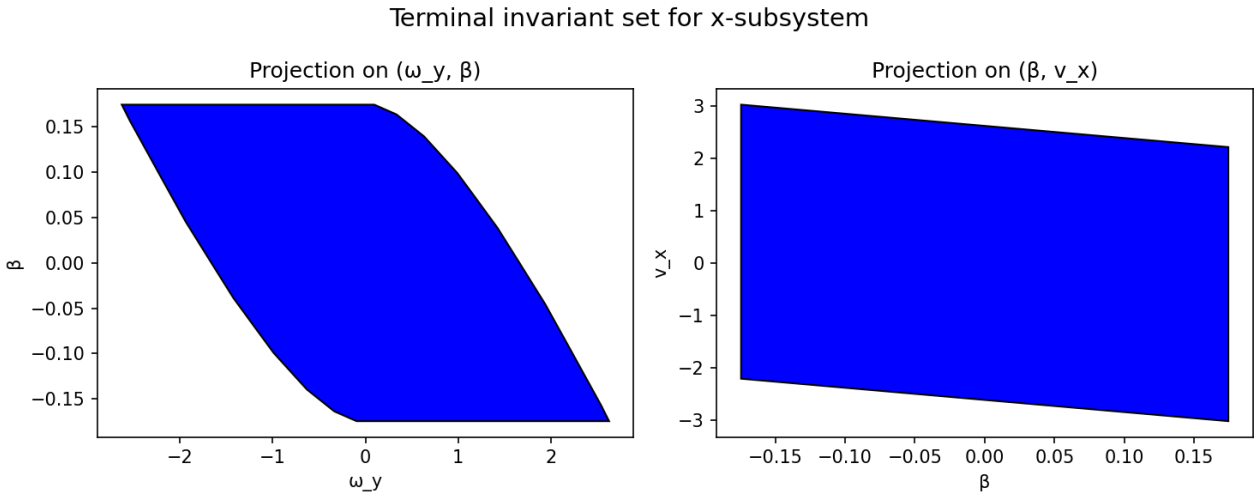

Figure 1: Terminal invariant set for the x-subsystem.

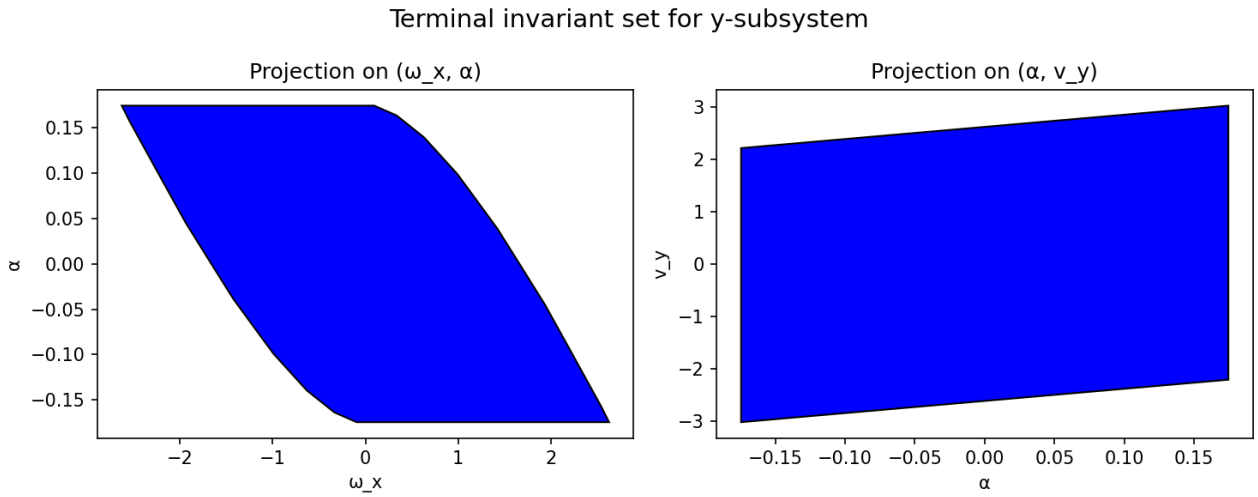

Figure 2: Terminal invariant set for the y-subsystem.

For the z subsystems, the reduced state dimension is one, and thus their invariant sets reduce to intervals on the real line. Our computation yields

$$-2.5440 \;\le\; v_z \;\le\; 5.0879,$$

For the roll-subsystem, the reduced state vector has two dimensions(angular rate and attitude angle), its terminal set is shown in Fig. 3.

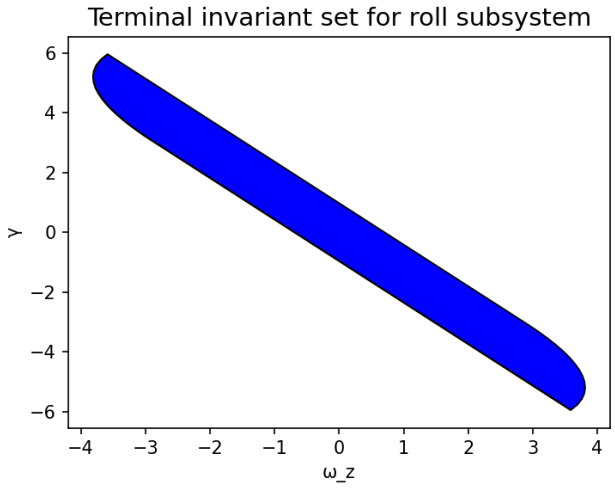

Figure 3: Terminal invariant set for the roll-subsystem.

These terminal invariant sets represent the maximal admissible sets for the final prediction state x_N such that the terminal controller $u = Kx$ satisfies both the state and input constraints for all future time steps.

3.1.4 Open-loop and Closed-loop Plots

We simulate the four MPC controllers under the prescribed initial conditions and obtain the trajectories shown in Fig. 4. With the chosen tuning parameters, all subsystems converge smoothly to the origin within the required settling time of 7 s, and all state and input constraints remain satisfied throughout the simulation.

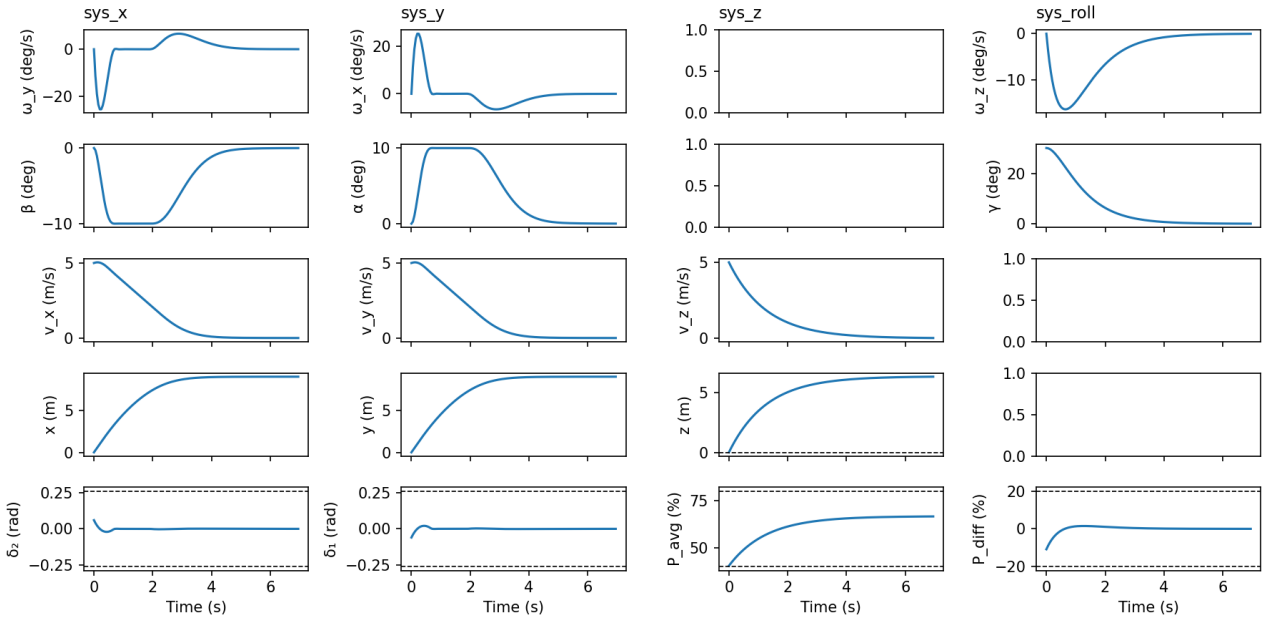

Figure 4: Closed-loop trajectories for all subsystems.

3.2 Deliverable 3.2: MPC Controller for Tracking

3.2.1 Design Procedure

In Deliverable 3.1, MPC controllers were designed to stabilize each subsystem at the origin. In this section, we extend these controllers to track constant references. For the subsystems (x, y, and z), the reference is a constant velocity, while for the roll subsystem the reference is a constant attitude angle. Since all references are constant, reference tracking can be achieved by reformulating the MPC problem in terms of deviation variables.
Let x_{ref} and u_{ref} denote the steady-state reference state and input and we define the deviation variables

$$\tilde{x}_k = x_k - x_{\text{ref}}, \qquad \tilde{u}_k = u_k - u_{\text{ref}}.$$

Substituting these expressions into the system dynamics yields

$$\tilde{x}_{k+1} = A\tilde{x}_k + B\tilde{u}_k,$$

which has the same structure as the stabilization dynamics considered in Deliverable 3.1.
As a result, the MPC formulation derived for stabilization can be directly applied to the tracking problem by penalizing deviations from the reference in the cost function. The finite-horizon optimization problem is therefore given by

$$\begin{aligned} J^\star(x_0) = \min_{\tilde{x}_0,\dots,\tilde{x}_N,\, \tilde{u}_0,\dots,\tilde{u}_{N-1}} \quad & \sum_{i=0}^{N-1} \left(\tilde{x}_i^\top Q \tilde{x}_i + \tilde{u}_i^\top R \tilde{u}_i\right) + \tilde{x}_N^\top Q_f \tilde{x}_N \\ \text{s.t.} \quad & \tilde{x}_{i+1} = A\tilde{x}_i + B\tilde{u}_i, \\ & x_i \in X, \quad u_i \in U, \\ & \tilde{x}_N \in X_f. \end{aligned}$$

The terminal cost matrix Q_f and the terminal feedback law $\tilde{u} = K\tilde{x}$ are identical to those used in Deliverable 3.1.

3.2.2 Choice of Tuning Parameters

The tuning parameters of the MPC controllers for reference tracking are based on those used in Deliverable 3.1. However, when extending the controllers to track nonzero velocity references, a direct reuse of all parameters before was found to result in overly aggressive control actions, particularly in the vertical (z) subsystem. To address this, the input-weight matrix R of the z subsystem is increased to penalize excessive thrust usage and to ensure smoother and more robust control actions. At the same time, the state-weight matrix Q is adjusted to place a higher emphasis on velocity states, improving damping and reducing oscillatory behavior during transients. The final set of tuning parameters used in Deliverable 3.3 is summarized in Table 3.

Subsystem	Q	R
MPC x controller	$\text{diag}(0.1,\ 0.8,\ 0.02)$	100
MPC y controller	$\text{diag}(0.1,\ 0.8,\ 0.02)$	100
MPC z controller	175	5
MPC roll controller	$50I_2$	0.1

Table 3: Summary of chosen parameters for all MPC subsystems.

3.2.3 Open-loop and Closed-loop Tracking Performance

Fig. 5 shows the closed-loop trajectories of all subsystems. The velocities converge smoothly to their respective reference values, while the roll angle tracks the desired constant attitude. Throughout the entire simulation, all state and input constraints remain satisfied.

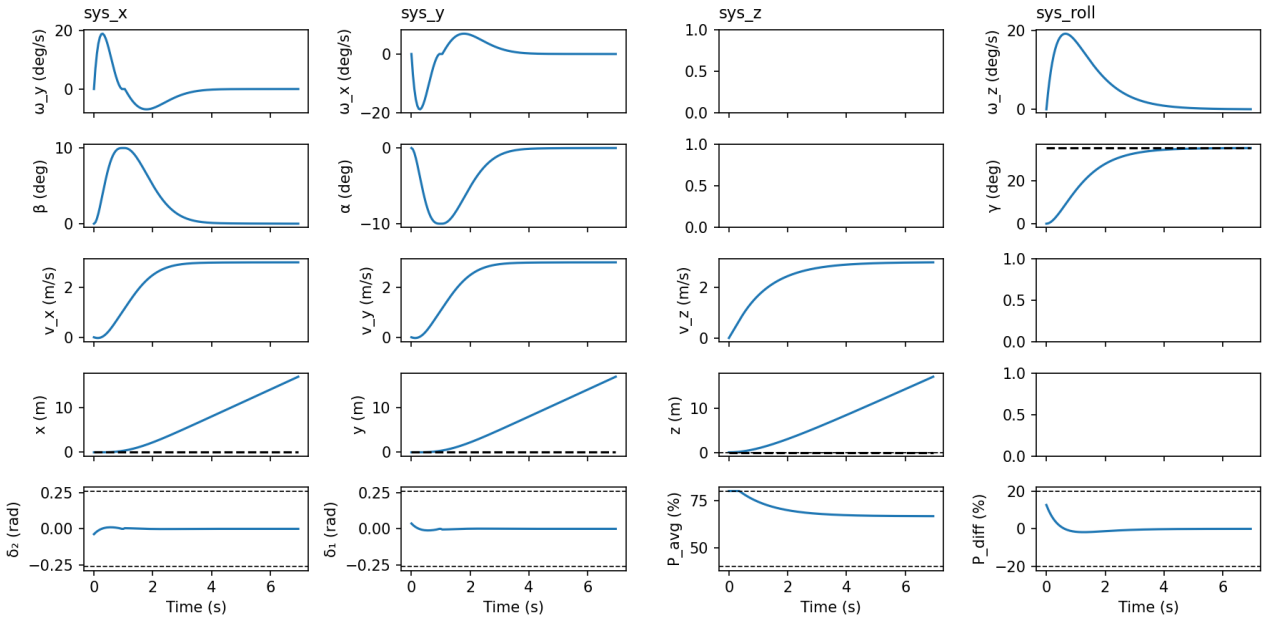

Figure 5: Closed-loop trajectories for all subsystems.

3.3 Deliverable 3.3: PID Position Tracking Controller

3.3.1 Design Procedure

In this section, the control architecture is extended to enable position tracking by introducing a PID-based outer-loop controller. The objective is to guide the rocket from a high initial altitude to a stationary hovering position, from which a controlled landing can subsequently be executed.

3.3.2 Choice of Tuning Parameters

Subsystem	Q	R
MPC x controller	$\mathrm{diag}(2,\ 0.8,\ 0.02)$	100
MPC y controller	$\mathrm{diag}(2,\ 0.8,\ 0.02)$	100
MPC z controller	50	10
MPC roll controller	$\mathrm{diag}(50,\ 100)$	0.1

Table 4: Summary of chosen parameters for all MPC subsystems.

When directly reusing the tuning parameters from Deliverable 3.2, it was observed that the z subsystem tended to generate control inputs that were too small during the initial transient, leading to violations of the lower bound on the average power constraint P_{avg}. To mitigate this issue, the state-weight matrix Q of the z subsystem was slightly reduced, while the input-weight matrix R was increased to penalize excessive variations in thrust. In addition, the prediction horizon H was moderately increased, allowing the controller to better anticipate future

dynamics and produce smoother, more constraint-aware control actions. The final set of tuning parameters used in Deliverable 3.3 is summarized in Table 4.

3.3.3 Open-loop and Closed-loop Tracking Performance

Figure 6 shows the resulting closed-loop trajectories for all subsystems. With the chosen tuning parameters, all subsystems converge smoothly to the target, achieving the desired operating condition within approximately 20 s.

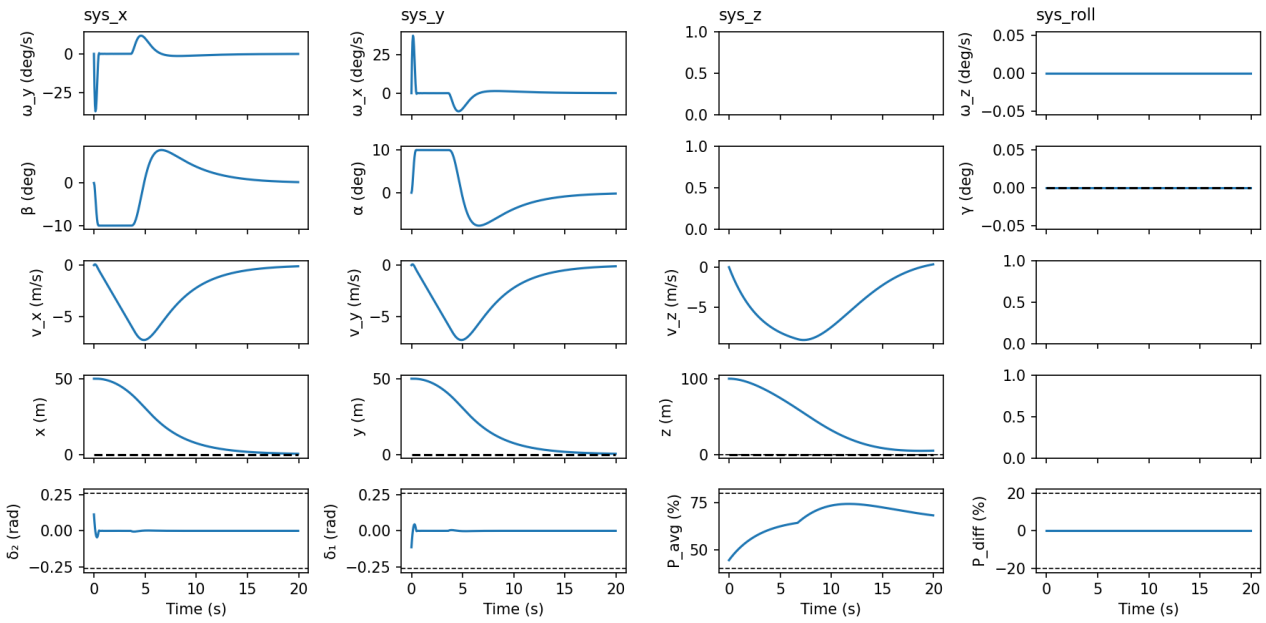

Figure 6: Closed-loop trajectories for all subsystems.

4 Deliverable 4: Simulation with Nonlinear Rocket

In this part, the four controllers designed in the previous sections are evaluated on the nonlinear rocket model. The control objective is identical to that of Deliverable 3.3, namely to guide the rocket from an initial position at high altitude to a stationary hovering point.

4.1 Deliverable 4.1: Nonlinear Simulation Results

4.1.1 Choice of Tuning Parameters

Subsystem	Q	R
MPC x controller	$\mathrm{diag}(69,\ 0.01,\ 0.05)$	1
MPC y controller	$\mathrm{diag}(55,\ 0.015,\ 0.035)$	1
MPC z controller	20	1
MPC roll controller	$\mathrm{diag}(1.8,\ 0.1)$	0.001

Table 5: Summary of chosen parameters for all MPC subsystems.

When applying the linear MPC controllers to the nonlinear rocket model, aggressive maneuvers were observed to cause constraint violations. To mitigate this effect, higher weights were assigned to the angular velocity states

in the x-, y-, and roll-subsystems, resulting in smoother attitude responses and less aggressive tilting maneuvers. The final set of tuning parameters used in Deliverable 4.1 is summarized in Table 5.

4.1.2 Open-loop and Closed-loop Tracking Performance

Figure 7 shows the resulting closed-loop trajectories for all subsystems. With the adjusted tuning parameters, the rocket successfully completes the maneuver and reaches the desired hovering condition. The velocities converge to zero while the position states reach their reference values. The state and input constraints are respected throughout the simulation.

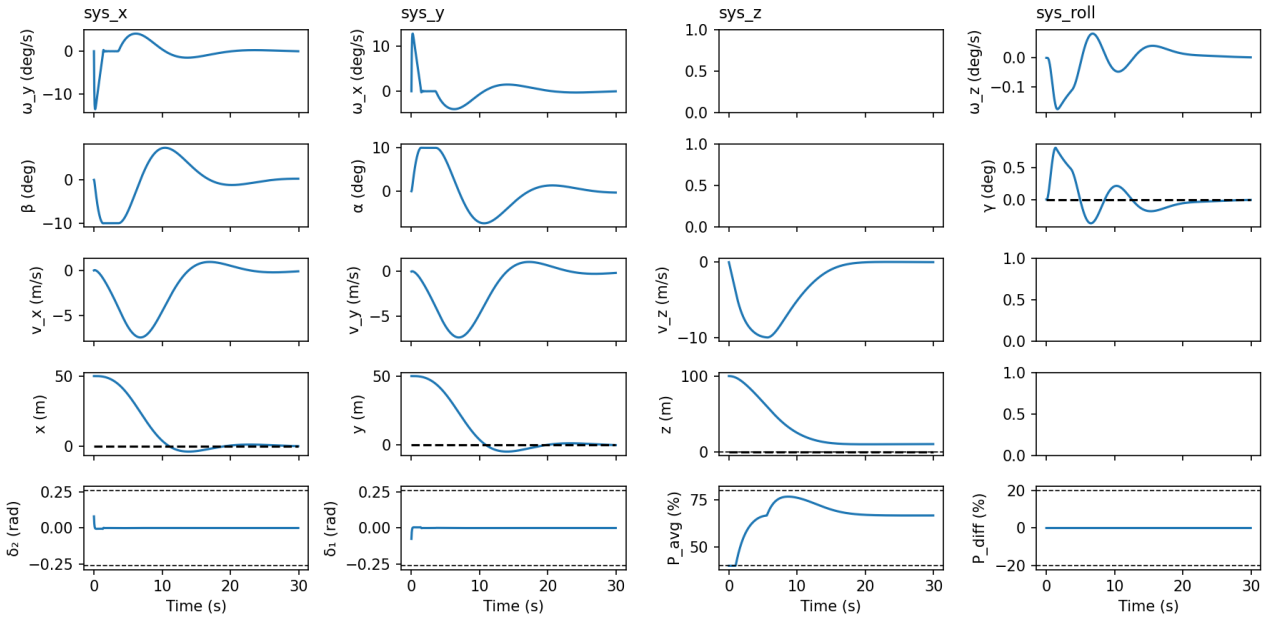

Figure 7: Closed-loop trajectories for all subsystems.

5 Deliverable 5: Offset free tracking

In the previous section, the controller was designed assuming an accurate model of the rocket dynamics. However, in practice, fuel consumption alters the rocket mass, leading to a mismatch between the true system dynamics and the model used for control design. This can be modeled as a constant disturbance. In this section, we extend the z-controller to achieve offset-free tracking in the presence of such modeling errors. The mass uncertainty is modeled as an unknown constant disturbance acting on the z-dynamics. To compensate for this disturbance, we augment the controller with a disturbance estimator and incorporate the estimated disturbance into the control law. The performance of the proposed controller is evaluated by comparing it with the controller from Part 4 under significant mass mismatch. We consider the discrete-time dynamics in the z-direction with an additive disturbance:

$$x_{k+1} = Ax_k + Bu_k + B_d d,$$

where d represents an unknown, constant disturbance. This disturbance captures the effect of incorrect mass modeling, which alters the effective gravitational and thrust terms.

5.1 Deliverable 5.1: Offset-Free Tracking Controller Design

5.1.1 Design Procedure

To compensate the steady-state bias induced by the constant disturbance introduced in Section 5, a disturbance estimator is integrated into the z-subsystem MPC controller. The disturbance is treated as an unknown input bias acting through the input channel and is estimated online.

Rather than designing a full augmented-state observer with matrix gains, a lightweight engineering estimator is implemented based on a one-step-ahead prediction and an integral disturbance update. Using the previous corrected state estimate $\hat{x}_k$, the previously applied input u_{k-1}, and the current disturbance estimate $\hat{d}_k$, a one-step prediction is computed as

$$\hat{x}_{k+1|k} = A\hat{x}_k + B\big(u_{k-1} + \hat{d}_k\big).$$

The prediction error (innovation) is then given by

$$e_k = x_k - \hat{x}_{k|k-1}.$$

where x_k is treated as a measurement.

The disturbance estimate is updated using a scalar integral gain:

$$\hat{d}_{k+1} = \hat{d}_k + k_d\, e_k,$$

where k_d determines the adaptation speed. The state estimate is subsequently corrected using the innovation:

$$\hat{x}_k \leftarrow \hat{x}_{k|k-1} + e_k.$$

The updated disturbance estimate $\hat{d}$ is injected into the MPC prediction model, enabling the optimizer to compensate steady-state input bias.

5.1.2 Choice of Tuning Parameters

The final set of tuning parameters used in Deliverable 5.1 is summarized in Table 6. And scalar gain k_d is tuned to achieve fast offset rejection without inducing overly aggressive control action, here we use $k_d = 3$.

Subsystem	Q	R
MPC x controller	diag(6100, 1, 7)	100
MPC y controller	diag(5500, 1, 3.5)	100
MPC z controller	50	0.08
MPC roll controller	diag(5000, 5000)	0.001

Table 6: Summary of chosen parameters for all MPC subsystems.

5.1.3 Disturbance Estimation Discussion

After an initial transient, the estimated disturbance will converge to a constant nonzero value, indicating successful identification of the mass-induced modeling error. For a constant rocket mass, the disturbance is approximately constant and small fluctuations in the estimate will arise due to observer dynamics and state

estimation errors. If fuel consumption were enabled, the disturbance would become time-varying because of the changing rocket mass, and hence the modeling error will evolve over time.

5.1.4 Tracking Performance for offset-free controller

Figure 8 shows the closed-loop response of the controller from Part 4 with the offset-free controller proposed in this section. The offset-free controller successfully compensates for the disturbance.

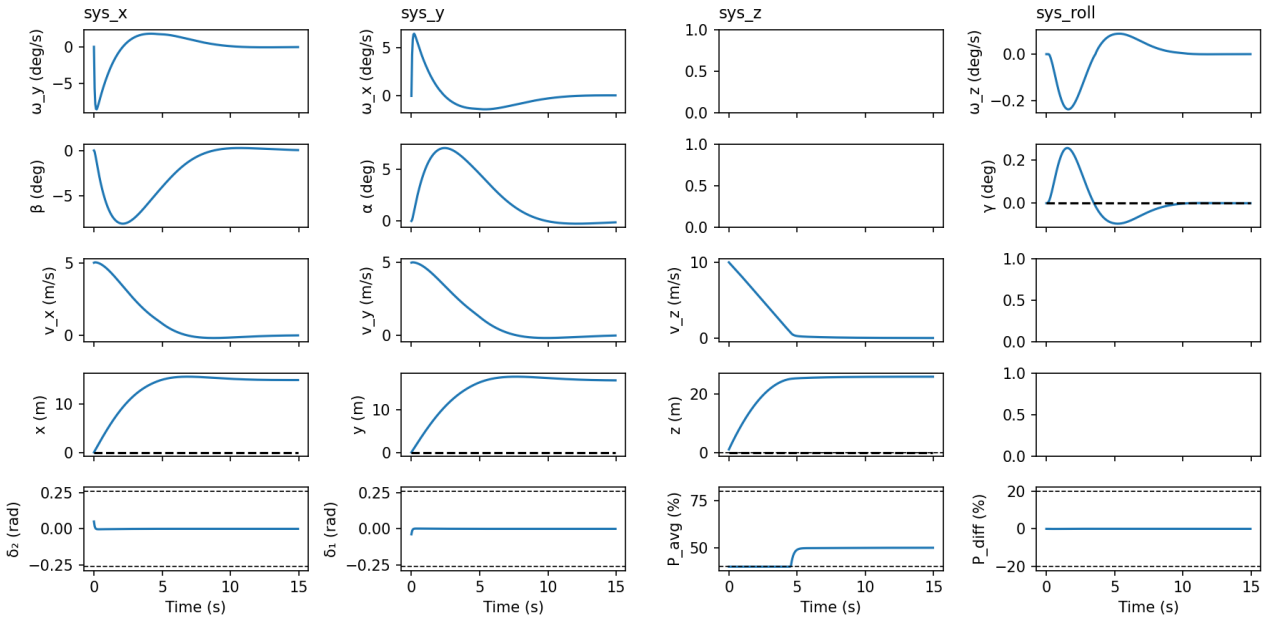

Figure 8: Position and Velocity for offset-free controller under mass mismatch.

5.2 Deliverable 5.2: Simulation with Time-Varying Mass

In this section, the offset-free controller developed in Section 5.1 is evaluated under more realistic conditions, where the rocket mass varies over time due to fuel consumption.

5.2.1 Choice of Tuning Parameters

The final set of tuning parameters used in Deliverable 5.2 is summarized in Table 7.

Subsystem	Q	R
MPC x controller	diag(6100, 1, 7)	100
MPC y controller	diag(5500, 1, 3.5)	100
MPC z controller	50	0.1
MPC roll controller	diag(5000, 5000)	0.001

Table 7: Summary of chosen parameters for all MPC subsystems.

5.2.2 Tracking Performance

Figure 9 illustrates the trajectories of the system under time-varying mass due to fuel consumption. Compared to the constant-mass case in Section 5.1, the changing mass introduces noticeable transient effects in the vertical dynamics, which are further analyzed in the following subsections.

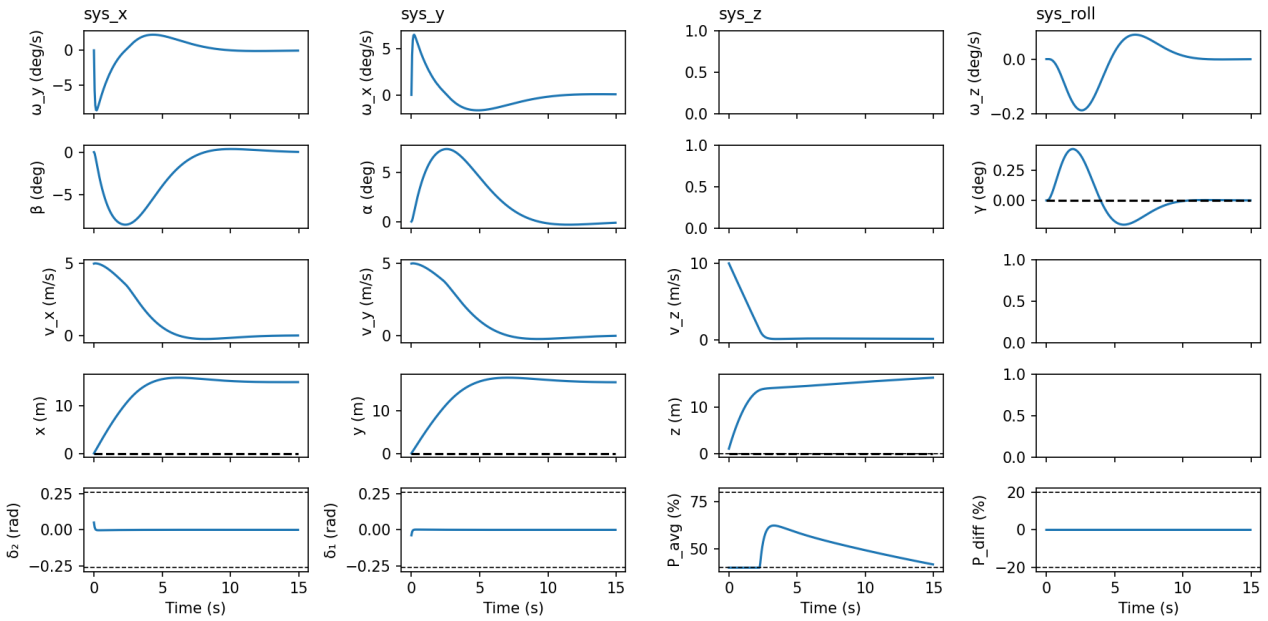

Figure 9: Closed-loop position, velocity, and control input under time-varying mass due to fuel consumption.

5.2.3 Initial Tracking Offset Despite Disturbance Estimation

During the first few seconds of the simulation, a noticeable tracking offset in height can be observed, even though a disturbance estimator is employed. This behavior arises because the estimator was designed under the assumption of a constant disturbance. With time-varying mass, the disturbance acting on the z-dynamics changes continuously as fuel is consumed. As a result, the estimator lags behind the true disturbance, leading to a temporary mismatch between the compensated model and the actual system dynamics. This transient mismatch manifests as a tracking offset until the estimator partially adapts. To achieve offset-free tracking in the presence of changing mass, the estimator could be modified to explicitly model the disturbance as time-varying through the following approaches:

- modeling the disturbance as a slowly varying state, e.g., $d_{k+1} = d_k + w_k$ with process noise,
- augmenting the estimator with fuel mass as an additional state,
- employing an adaptive or gain-scheduled observer that accounts for mass variation.

5.2.4 Observed Trajectory Behaviors

As the rocket mass decreases, the controller rapidly drives the vertical velocity v_z close to zero, causing the optimal control input to decrease in order to maintain $v_z = 0$. Consequently, the optimal input gradually approaches the lower bound on P_{avg}, eventually leading to a violation of the minimum thrust constraint and termination of the simulation.

6 Deliverable 6: Robust Tube-MPC for landing

In this section, a robust tube MPC controller is designed for the landing phase, where the rocket is required to move from $(x, y, z, \gamma) = (3, 2, 10, 30^\circ)$ to $(1, 0, 3, 0^\circ)$ while satisfying the ground safety constraint $z \geq 0$

under bounded disturbances. For the z-subsystem, deviation variables $\Delta x_z = x_z - x_{s,z}$ and $\Delta u_z = u_z - u_{s,z}$ are introduced, leading to the uncertain dynamics

$$\Delta x_z^+ = A_d \Delta x_z + B_d \Delta u_z + B_d w, \qquad w \in [-15, 5].$$

6.1 Deliverable 6.1: Z-direction Position controller

6.1.1 Design Procedure

We implement a robust tube MPC controller for the z-subsystem by combining (i) a nominal MPC defined on the disturbance-free dynamics and (ii) an ancillary state-feedback law that keeps the true disturbed trajectory inside a bounded tube around the nominal trajectory.

Ancillary feedback and error dynamics. An LQR gain K is computed for the z-subsystem using the weighting matrices Q and R. This gain is used as an ancillary feedback law to stabilize the deviation between the true and nominal trajectories. The resulting closed-loop error dynamics are given by

$$e^+ = (A_d + B_d K)e + B_d w, \qquad w \in \mathcal{W},$$

which are stable by construction and admit a bounded robust positively invariant set.

Minimal robust positively invariant set. To bound the tracking error e, we compute the minimal robust positively invariant (mRPI) set $\mathcal{E}$ for the above error dynamics:

$$\mathcal{E} \approx \sum_{i=0}^{\infty} (A_d + B_d K)^i \big(B_d \mathcal{W}\big),$$

Constraint tightening. Robust constraint satisfaction is enforced by tightening the nominal constraints using $\mathcal{E}$. Let $\mathcal{X}$ denote the original state constraint set induced by $z \geq 0$ (expressed in deviation coordinates), and let $\mathcal{U}$ denote the original input constraint set for Δu_z. We compute the tightened sets

$$\tilde{\mathcal{X}} = \mathcal{X} \ominus \mathcal{E}, \qquad \tilde{\mathcal{U}} = \mathcal{U} \ominus K\mathcal{E},$$

so that enforcing $z_k \in \tilde{\mathcal{X}}$ and $v_k \in \tilde{\mathcal{U}}$ guarantees satisfaction of the original constraints for the true system.

Terminal set. To ensure recursive feasibility, a terminal set $\mathcal{X}_f$ is computed as the maximal positively invariant set for the nominal closed-loop dynamics under the tightened constraints.

Online optimization and control law. At each time step, a nominal MPC problem is solved to optimize the nominal state and input trajectories $\{z_k\}_{k=0}^{N}$ and $\{v_k\}_{k=0}^{N-1}$ under the tightened constraints, with the initial tube condition $\Delta x_{z,0} - z_0 \in \mathcal{E}$. The applied control input is then obtained by combining the nominal input with the ancillary feedback,

$$\Delta u_{z,0} = v_0 + K(\Delta x_{z,0} - z_0),$$

and mapped back to the absolute thrust command using the steady-state input $u_{s,z}$.

6.1.2 Choice of Tuning Parameters

The selected parameters are summarized in Table 8.

Subsystem	Q	R
MPC x controller	diag(100, 200, 20, 50)	10
MPC y controller	diag(100, 200, 20, 50)	10
MPC z controller	diag(100, 350)	1
MPC roll controller	diag(10, 30)	0.01

Table 8: Summary of chosen parameters for all MPC subsystems.

6.1.3 Invariant and Terminal Sets

Figure 10 shows the minimal invariant set $\mathcal{E}$ of the error dynamics together with the terminal set $\mathcal{X}_f$ used in the nominal MPC. The vertices of the tightened input constraint $\tilde{\mathcal{U}}$ are :

$$\tilde{\mathcal{U}} = [-8.209, 4.878]$$

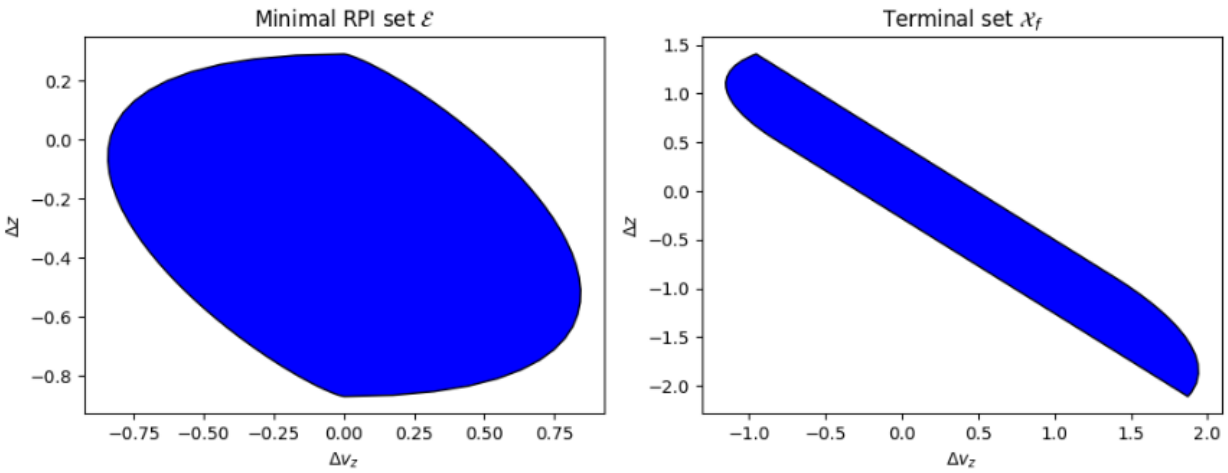

Figure 10: Minimal robust positively invariant set $\mathcal{E}$ and terminal set $\mathcal{X}_f$ for the z-subsystem.

6.1.4 Closed-Loop Performance Under Disturbances

The robust tube MPC controller is evaluated under different disturbance scenarios using the provided simulator. Closed-loop simulations are performed with both `random` and `extreme` disturbance profiles. Figures 11 and 12 show the closed-loop trajectories of the z-subsystem states and inputs. In both cases, the controller robustly satisfies the constraint $z \geq 0$ and drives the system to a neighborhood of the target height within the specified settling time.

6.2 Deliverable 6.2: Nominal MPC Design for x, y, and roll

In this section, the robust tube MPC controller designed for the z-subsystem in Section 6.1 is extended to a full landing controller by designing nominal MPC position controllers for the x-, y-, and roll-subsystems.
For the x-, y-, and roll-dynamics, nominal MPC controllers are employed without explicit disturbance modeling. Soft state and input constraints are used to maintain feasibility when operating on the nonlinear rocket model.

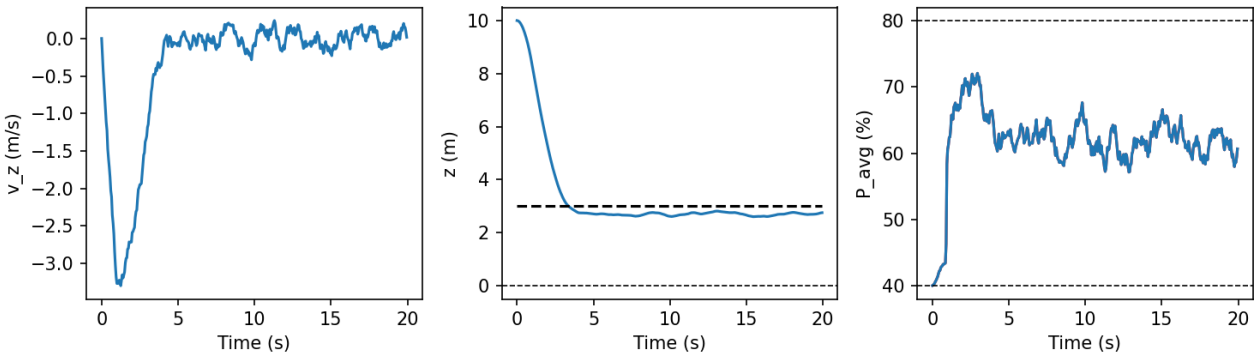

Figure 11: Closed-loop z-subsystem response under random disturbances using robust tube MPC.

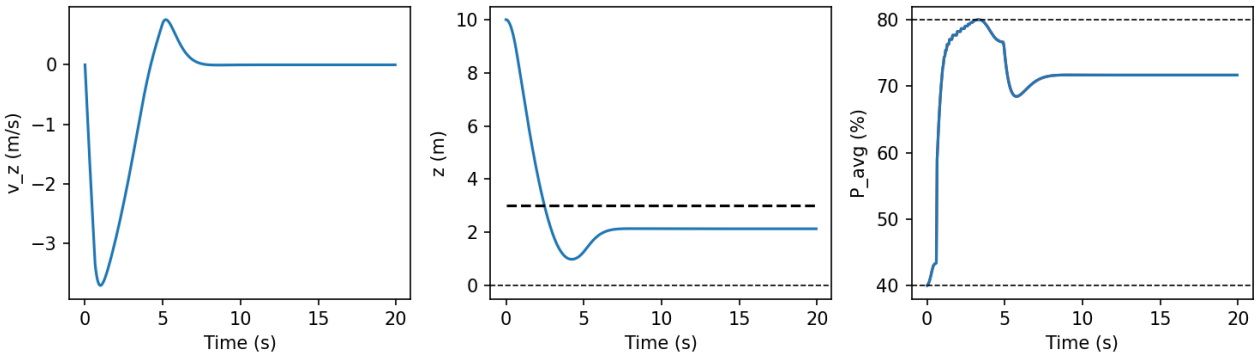

Figure 12: Closed-loop z-subsystem response under extreme disturbances using robust tube MPC.

6.2.1 Nonlinear Simulation Results

Figures 13 shows the closed-loop response of the merged landing controller on the nonlinear rocket model. All position states converge to their target values within 4 s while satisfying the constraint $z \geq 0$ and respecting input limits, demonstrating the effectiveness of the combined robust and nominal MPC design.

7 Deliverable 7: Nonlinear MPC

7.1 Design and tuning of the NMPC

The nonlinear MPC controller was developed starting from the provided CasADi-based NMPC scaffold, which already defined the interface, nonlinear rocket dynamics, and solver structure. The controller formulation was completed by explicitly defining the optimal control problem, including the prediction horizon, decision variables, nonlinear dynamics constraints, cost function, and state and input constraints. The full nonlinear rocket model was enforced over the prediction horizon using an explicit fourth-order Runge–Kutta (RK4) discretization, ensuring accurate state prediction during aggressive landing maneuvers.

The stage cost was chosen quadratic in the state and input tracking errors with respect to a steady-state trim $(x_{\text{ref}}, u_{\text{ref}})$. State weights were selected to strongly prioritize position accuracy near the landing target, while penalizing attitude and angular rates sufficiently to ensure safe and smooth convergence. In particular, the roll angle γ was weighted significantly higher than other attitude states to avoid residual roll drift after touchdown. Input weights were tuned to limit excessive actuator usage while still allowing sufficient authority for fast landing. To improve robustness and reduce actuator chattering, a small penalty on the input rate Δu was added,

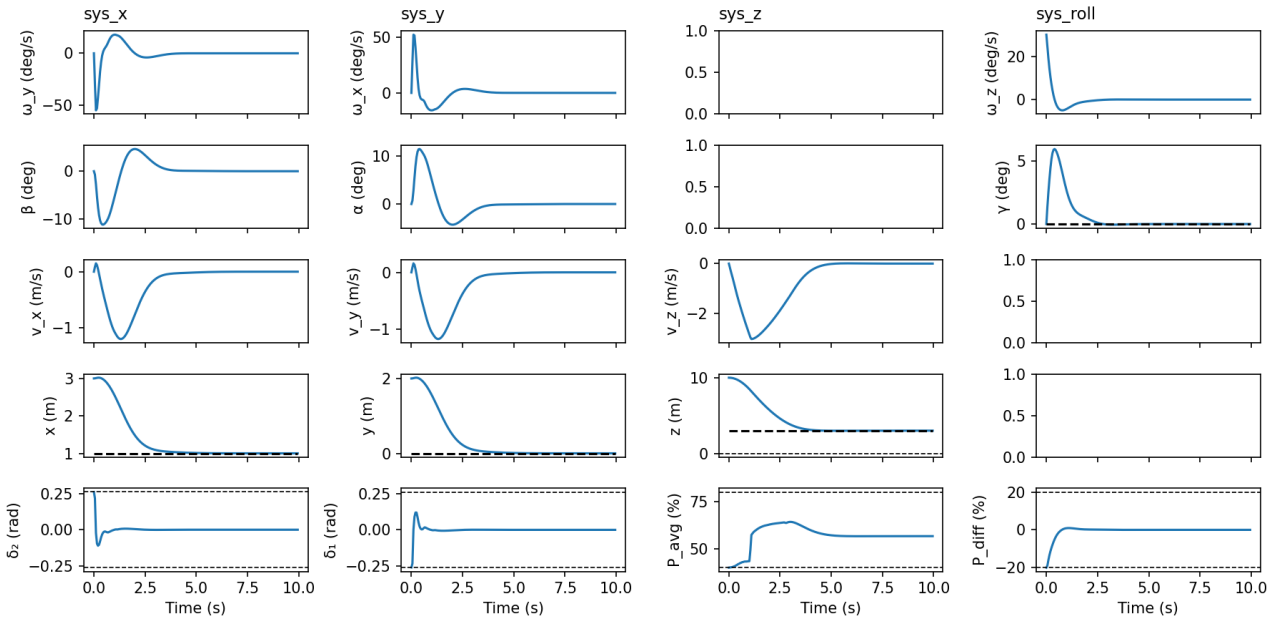

Figure 13: Closed-loop x, y, z and γ trajectories during the landing maneuver

which smooths control actions without noticeably degrading performance.

To compensate for the short prediction horizon, a terminal cost based on a continuous-time LQR design around the trim point was included. The terminal weight matrix was obtained by solving the continuous algebraic Riccati equation using the linearized rocket dynamics. This terminal cost encourages stabilizing behavior near the target state and improves closed-loop convergence without imposing hard terminal constraints.

Several robustness-oriented modifications were incorporated. The roll angle error was wrapped inside the cost function using a smooth atan2 formulation to avoid discontinuities due to angle periodicity. A warm-start strategy based on shifting the previous optimal solution was implemented to improve numerical reliability and convergence speed. Finally, a safe fallback strategy was added to handle rare solver failures by reusing the previously feasible control sequence or the trim input. The resulting NMPC achieves reliable landing within the required settling time while respecting all state and input constraints.

7.2 Open and Closed Loop Performance

At each sampling instant, the NMPC computes an open-loop optimal state and input trajectory over the prediction horizon based on the current measured state and the nonlinear rocket model. These open-loop predictions provide a planned landing trajectory assuming perfect model knowledge and no disturbances. In closed loop, only the first control input of the optimal sequence is applied to the system, and the optimization is repeated at the next sampling instant using updated state measurements.

Simulation results show that the open-loop predictions closely match the closed-loop trajectories, with small discrepancies occurring mainly during the initial, aggressive descent phase where actuator saturation is active. These deviations are effectively corrected by the receding-horizon feedback mechanism, and the closed-loop system converges smoothly to the landing target. The rocket reaches the desired position and attitude within the required settling time of approximately 4 s while respecting all state and input constraints. This illustrates the ability of the NMPC to generate accurate predictions while maintaining robustness through closed-loop re-optimization.

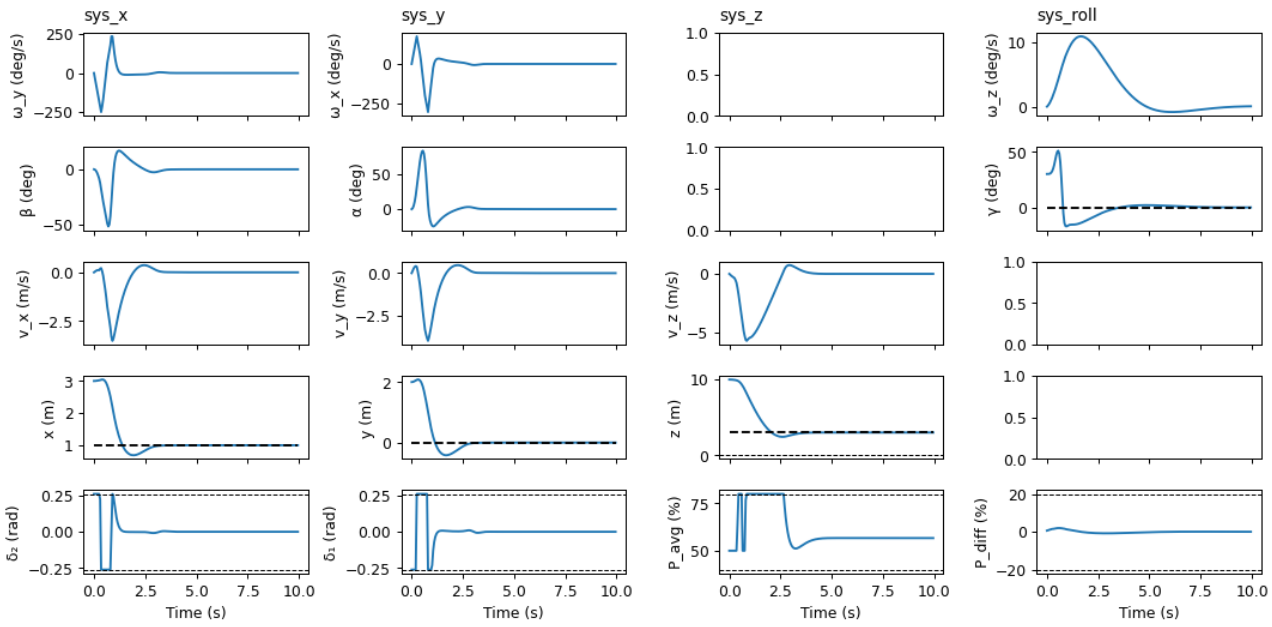

Figure 14: Closed-loop x, y, z and γ trajectories during the landing maneuver using the NMPC

7.3 Comparison with Robust + 3 nominal MPCs from 6.2

Compared to the robust MPC combined with three nominal MPCs, the NMPC directly optimizes the full nonlinear dynamics and therefore captures cross-couplings more accurately and achieves faster, less conservative convergence in nominal conditions. This is visible in the landing maneuver, where the NMPC exhibits more aggressive transients and control actions, while the robust MPC shows smoother and more damped responses due to constraint tightening and disturbance margins. A key advantage of the robust MPC design is its guaranteed constraint satisfaction and robustness to model mismatch, in particular for the safety-critical altitude constraint $z \geq 0$. In contrast, the NMPC enforces constraints only on the predicted nonlinear model and can violate them in the presence of uncertainties, while also requiring higher computational effort and more careful tuning. Overall, the robust MPC provides safer and more reliable performance under uncertainty, whereas the NMPC offers improved nominal performance and a unified control formulation at the cost of reduced robustness guarantees.