Model Predictive Control

Final Mini Project

ME-425

Jiangfan Li, Hao Zhao, Chengkun Li

École polytechnique fédérale de Lausanne

EPFL

Model Predictive Control

Final Mini Project

by

Jiangfan Li, Hao Zhao, Chengkun Li

Student Name	SCIPER Number
Name	Number
Name	Number
Chengkun Li	340485

Instructor: Colin Jones
Institution: École polytechnique fédérale de Lausanne

EPFL

Contents

1 Introduction

1.1. Overview

We consider the problem of investigating the effect of controllers in thrust vector control for combustion engine rockets.

In this project, we set our experiment object as a small-scale rocket prototype (as shown in Fig. 1.1) whose engine is replaced by high-performance racing propeller. We are asked to design controllers to complete tasks under simulated flying conditions involving designing regulators and tracking set points with or without offset. Recursive feasibility and stability are two of the most significant indicators of a reliable controller in this project. However, in practice, thrust vector control for engines is a complicated nonlinear process which has many strict hardware limitations, hence we need to utilize a more advanced engine controller.

1.2. Literature Review

Systematic optimization methods are ideal ways for this project. Several model-based controllers have been used in combustion engine control tasks, such as the Linear Quadratic Regulator (LQR) controller [1], Linear Quadratic Gaussian (LQG) controller [2], Sliding Model Controller (SMC) [3], and Model Predictive Control (MPC) controller [4].

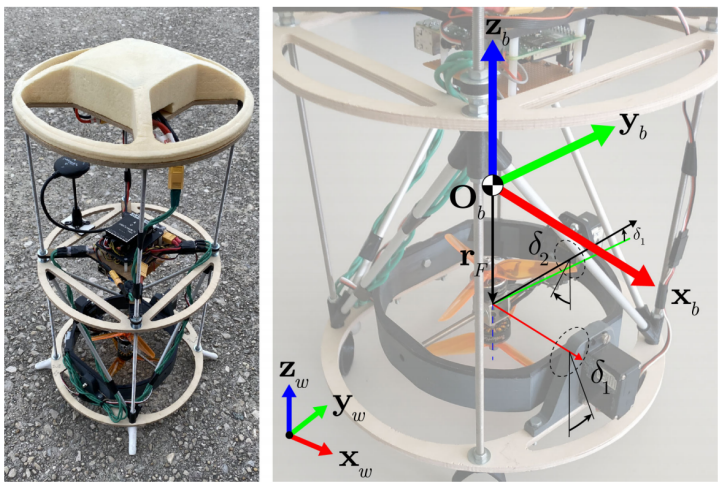

Figure 1.1: Rocket prototype studied in this project

Among these mode-based controllers, MPC is one of the most promising controllers that can deal with the highly constrained nonlinear system. MPC can provide an optimal real-time solution for meeting multi-objective goals while addressing system and operational constraints. New variants of MPC utilize optimization solvers and packages that are suitable for the real-time operation of time-critical systems.

1.3. Problem Statement

Given the MPC model to complete the project, we ask two key questions: (a) how we implement practical MPC controllers in a highly nonlinear system? (b) given a target output reference, how we tune inputs to regulate our system to the reference. And if there is unknown but constant disturbance, i.e., gravity, how we design a robust controller to track the output reference?

The answer to (a) is simplifying the nonlinear system into several linear subsystems and controlling a trimmed and linearized version of the rocket. More specifically, we are going to calculate a local steady-state state and input pair that satisfies $f(x_0, u_0) = 0$. Then we compute the Jacobian matrix with respect to states and inputs, and use it to derive the system dynamics equation according to Euler approximation, $x^+ \approx A(x - x_0) + B(u - u_0)$. Next, we do the tracking task by finding the optimal steady-state state and input pair first, and then adjust the objective function of the MPC optimization problem in a delta form, significantly, with a group of transformed constraints on the feasible set. To tackle the negative influence from unknown disturbance, we model the disturbance into our system dynamics and introduce a linear state and disturbance estimator to observe current states and disturbance. And then we adapt the target condition accordingly to account for disturbance., which makes the realization of (b) feasible.

Usually, in order to understand this control process, modelling the system dynamics from physical principles, which is out of the scope of this course, is a crucial step. In this project, we are going to derive a 12-state dynamic system with the state vector $\boldsymbol{x} = [\boldsymbol{\omega}^T, \boldsymbol{\phi}^T, \boldsymbol{v}^T, \boldsymbol{p}^T]^T$, where $\boldsymbol{\omega}$ denotes the angular velocities about the body axes. $\boldsymbol{\phi}$, $\boldsymbol{v}$ and $\boldsymbol{p}$ represent the Euler angles of the body frame, velocity and position of the world frame. A four-dimensional input vector $\boldsymbol{u} = [\delta_1, \delta_2, P_{avg}, P_{diff}]^T$ regulates states of the model in a nonlinear way, where δ_1 and δ_2 represents the deflection angles of servo 1 and servo 2, $P_{avg} = (P_1 + P_2)/2$ is the average throttle and $P_{diff} = P_2 - P_1$ is the throttle difference between the motors.

1.4. Report Organization

This report is organized into two chapters. In Chapter 1, we present the main goals of the project and reason that MPC controller is suitble in thrust vector control for the mini rocket in our project. Meanwhile, we briefly discuss practical strategies about transforming complex nonlinear system into several linear models and how to realize reliable tracking function with or without disturbance.

In Chapter 2, we propose implementation details about the design of MPC regulators and MPC tracking controllers with or without offset using the linearized version of system. This section also compares the performance of linear MPC controller and Nonlinear MPC controller in the tracking task. Finally, we summarize our work in the last deliverable.

2

Deliverables

2.1. Deliverable 2-1

In this deliverable, we aim to decompose our linearized system into 4 subsystems such that we could divide the task of controlling the rocket system into four sub tasks and assign each to an individual controller.

First, we find the steady state of the system and linearize the system at this steady state `(us, xs)`, and then we look at matrix A acquired from the linearization step (in Eq. (2.1)),

$$A = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{matrix} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 9.81 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -9.81 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{array} \tag{2.1}$$

the non zero entries in matrix A are entries $(4,1),(5,2),(6,3),(8,4),(7,5),(10,7),(11,8),(12,9)$, which means state pairs in these tuples are interrelated, therefore, the interrelated pairs be clustered into 4 groups:

- $4,1,8,11 \rightarrow \alpha, \omega_x, v_y, y$
- $5,2,7,10 \rightarrow \beta, \omega_y, v_x, x$
- $6,3 \rightarrow \gamma, \omega_z$
- $12,9 \rightarrow z, v_z$

Likewise, nonzero entries in matrix B (as shown in Eq. (2.2)) indicate the relationships between

states and inputs,

$$B = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{pmatrix} -55.68 & 0 & 0 & 0 \\ 0 & -55.68 & 0 & 0 \\ 0 & 0 & 0 & -0.104 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 9.81 & 0 & 0 \\ -9.81 & 0 & 0 & 0 \\ 0 & 0 & 0.173 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{matrix} \tag{2.2}$$

therefore, we can get the following 4 independent relationships (4 pairs of input/states):

- `sys_x`: $\delta_2 \rightarrow \omega_y, v_x \rightarrow \beta, \omega_y, v_x, x$
- `sys_y`: $\delta_1 \rightarrow \omega_x, v_y \rightarrow \alpha, \omega_x, v_y, y$
- `sys_z`: $P_{avg} \rightarrow v_z \rightarrow z, v_z$
- `sys_roll`: $P_{diff} \rightarrow \omega_z \rightarrow \gamma, \omega_z$

From an intuitive physical perspective, the independence between these four subsystems can be explained as: *the overall controlling effect can be regard as the superposition of the controlling effects of these 4 subsystems*, which is supported by the fact that **each input** can independently determine **a set** of states, for example, deflection angle of servo 1 determines the angular velocity about body x axis, rotation angle on x to transform world to body frame, speed and position on y axis.

2.2. Deliverable 3-1

In order to ensure the recursive constraint satisfaction in this project, we compute a positive invariant set from the perspective of LQR control law and set up this invariant set as the terminal set that forces the N-th step of the trajectory to fall within it. The MPC problem can be formulated as follows,

$$\begin{array}{ll} \min\limits_{x,u} & \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i + x_N^T P x_N \\ \text{s.t.} & x_{i+1} = A x_i + B u_i \\ & C x_i + D u_i \leq b \\ & x_N \in \mathcal{X}_f \\ & x_0 = x \end{array} \tag{2.3}$$

This terminal constraint brings the recursive feasibility into our MPC problem. The brief proof of it follows: firstly, let $[u_0^*, u_1^*, \ldots, u_{N-1}^*]$ be the optimal control sequence computed at x. At the next state x^+, there is at least a closed-loop trajectory whose first N-1 steps apply the previous inputs $[u_1^*, u_2^*, \ldots, u_{N-1}^*]$. Because x_N is in $\mathcal{X}_f$, the N-th input can be calculated as $\kappa_f\left(x_N^*\right)$. The new feasible input sequence $[u_1^*, u_2^*, \ldots, u_{N-1}^*, \kappa_f\left(x_N^*\right)]$ shows that the feasible set is not empty and thus ensures the recursive constraint satisfaction of our MPC method.

The translation and rotation within a certain range (e.g., 5 meters shift along x, y, z, and 45° rotation for roll) works with a relatively simple setup of parameters. As shown in the following table 2.1, R is set to 1 to strengthen the control effect, and in order to better guide the trajectory to the regulation point,

Table 2.1: Parameters for each subsystem in deliverable 3-1

Parameter	Controller x	Controller y	Controller z	Controller roll
Q	$10 \times I_4$	$10 \times I_4$	$10 \times I_2$	$10 \times I_2$
R	1	1	1	1
H	5	5	5	5

penalties Q equals to $10 \times I_{nx}$, where nx is the number of states in each sub-system. And there is no need to separate various penalty coefficients and tune each of them in a fine-grained way.

The parameters used in this deliverable are listed in

We calculate the terminal invariant set under the scheme of LQR method, where the optimal controller $u^* = Kx$. Thus, the controlled invariant set can be computed as an autonomous system. In an iterative way, we calculate the pre-set of current feasible set and keep the intersection of both areas. Finally, this method will return a maximum invariant set as our terminal set $\mathcal{X}_f$. Plots of terminal invariant set for each of the dimensions are shown in Figs. 2.1 to 2.4.

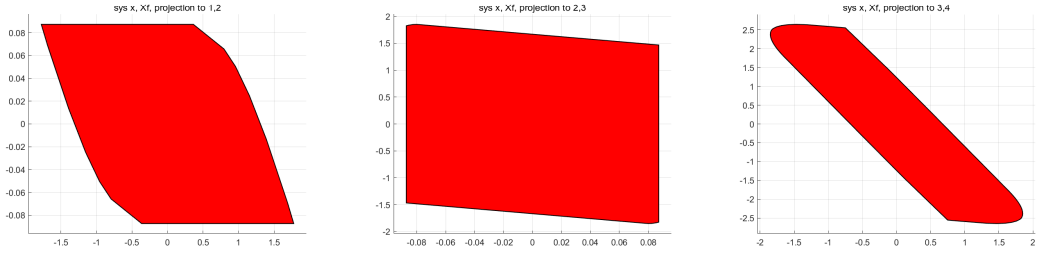

Figure 2.1: Terminal invariant set for sub-system x

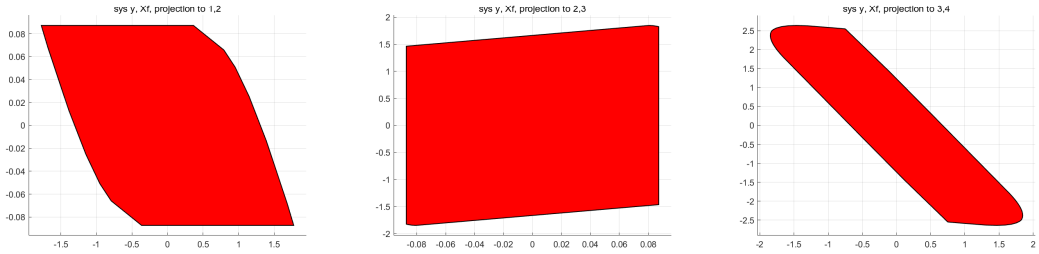

Figure 2.2: Terminal invariant set for sub-system y

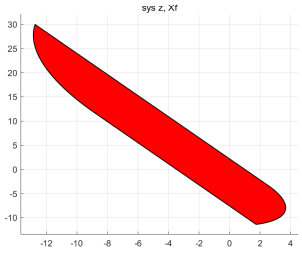

Figure 2.3: Terminal invariant set for sub-system z

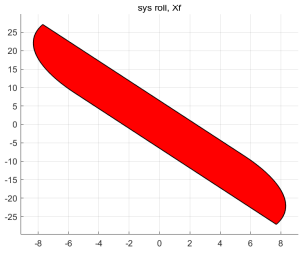

Figure 2.4: Terminal invariant set for sub-system roll

Plots for each dimension starting stationary at 5 meters from the origin (for x, y, z) or stationary at $45°$ for roll are shown in Figs. 2.5 and 2.6,

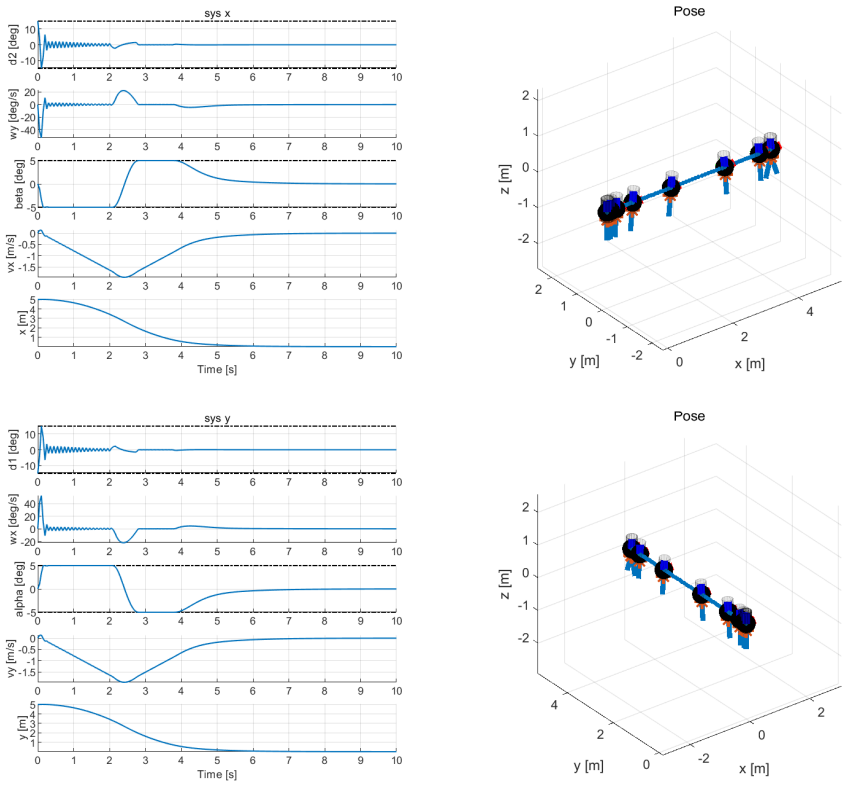

Figure 2.5: Control performance of subsystems x and y

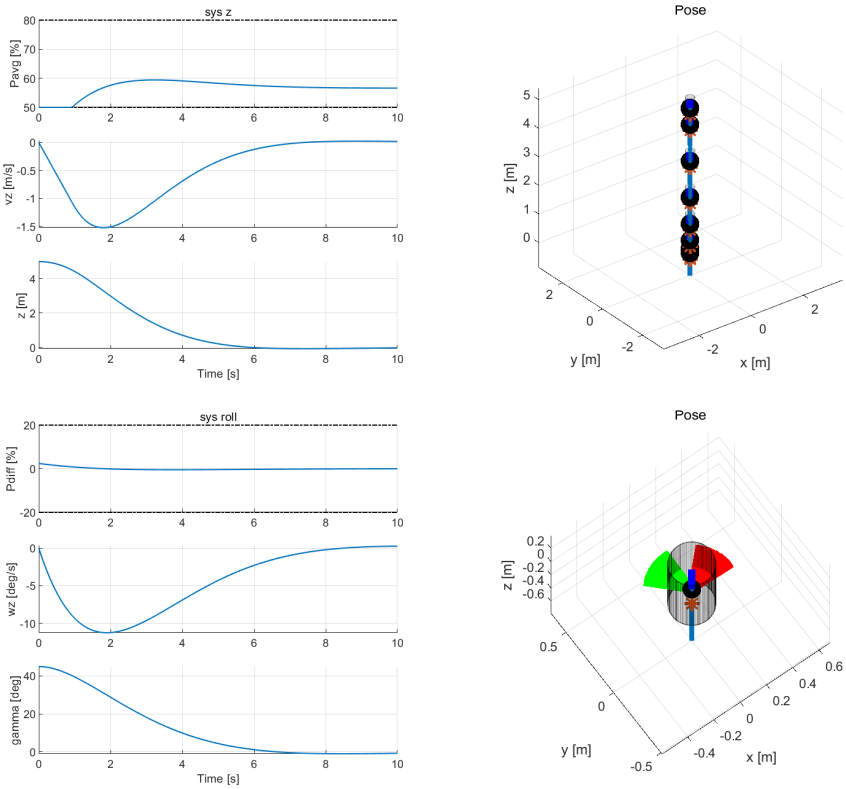

Figure 2.6: Control performance of subsystems z and roll

2.3. Deliverable 3-2

In this deliverable we aim to design the tracking system of our rocket so that the rocket could perform set point tracking. We adopt the delta-formulation for tracking, the problem is converted into a regulation problem as shown in Eq. (2.4)

$$\begin{array}{ll} \min & \sum_{i=0}^{N-1} \Delta x_i^T Q \Delta x_i + \Delta u_i^T R \Delta u_i + V_f\left(\Delta x_N\right) \\ \text{s.t.} & \Delta x_0 = \Delta x \\ & \Delta x_{i+1} = A\Delta x_i + B\Delta u_i \\ & H_x \Delta x_i \leq k_x - H_x x_s \\ & H_u \Delta u_i \leq k_u - H_u u_s \end{array} \tag{2.4}$$

To get the deltas of states and input we first compute the steady state and input for the reference output under constraints as shown in Eq. (2.5) where the terminal cost is deterimined by terminal LQR.

$$\begin{array}{ll} \min & u_s^T R_s u_s \\ \text{s.t.} & \begin{bmatrix} I - A & -B \\ C & D \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} 0 \\ r \end{bmatrix} \\ & H_x x_s \leq k_x \\ & H_u u_s \leq k_u \end{array} \tag{2.5}$$

For each controller, we prioritize the states over the inputs, that is setting the parameter Q relatively large compared to R. On top of that we also choose a relatively long horizon to ensure feasibility. The parameters are the same as Table 2.1 and the performance of the tracking system is shown in Figs. 2.7 and 2.8.

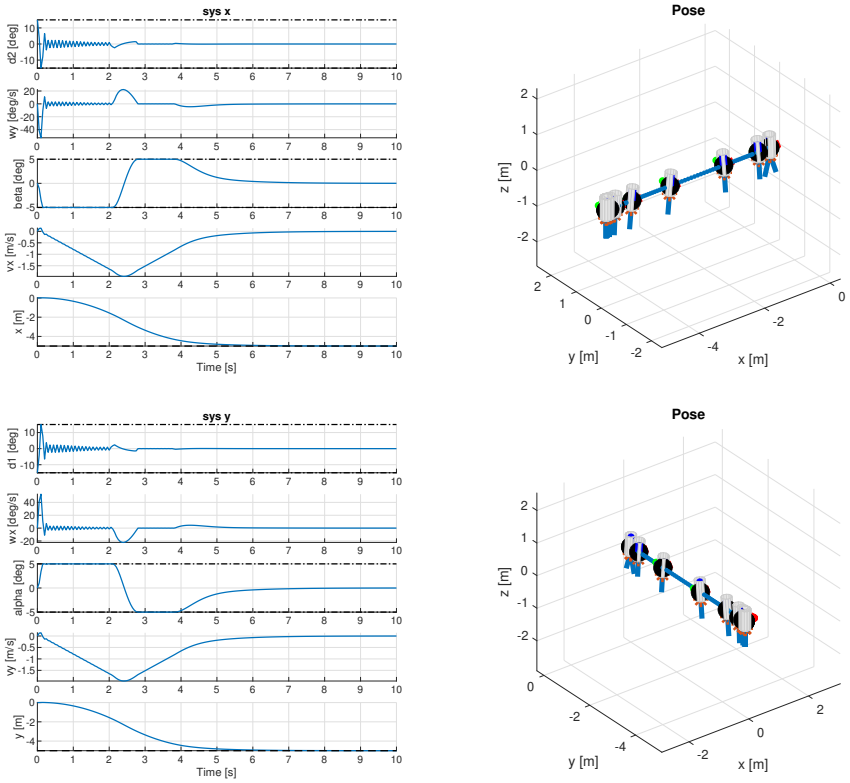

Figure 2.7: Tracking performance of subsystems x and y

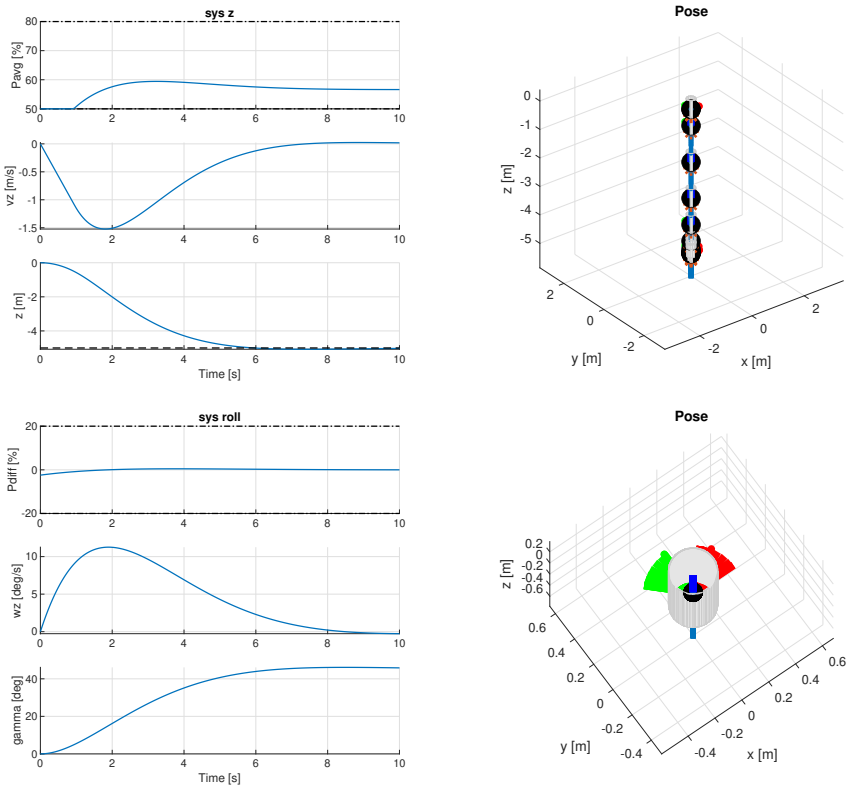

Figure 2.8: Tracking performance of subsystems z and roll

2.4. Deliverable 4-1

In this deliverable, we aim to combine our four controllers as well as their controlling effect in order to tracking a given reference trajectory.

To do so, we first adopt the parameters we acquired in Section 2.2 (Table 2.1), the tracking performance (Fig. 2.9) can not be described as *good*. As we can see, α and β violate the angle limitation a bit. Tracking on z axis and γ is not swift.

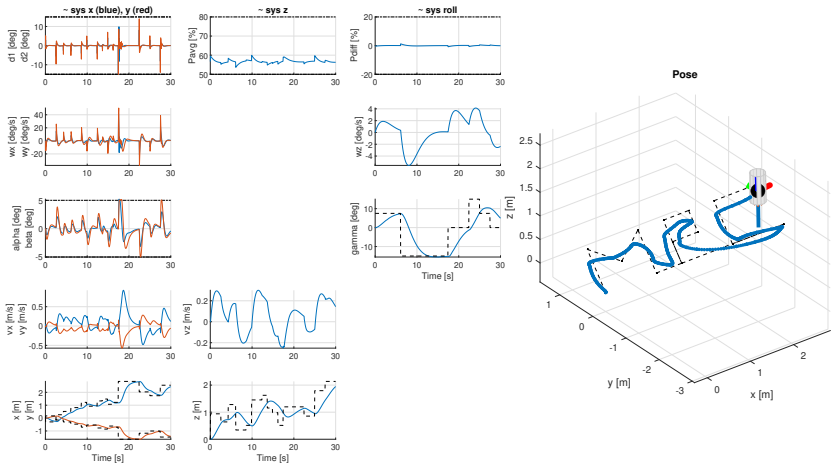

Figure 2.9: Tracking performance using default parameters

Therefore, we modify the parameters for better performance. R is decreased to allow stronger control. Penalties for position are increased for better tracking trajectory. Also, more punishment for ω_x and ω_y to avoid violation of angle limits. After several round of tuning, we get Fig. 2.10 using parameters as shown in Table 2.2.

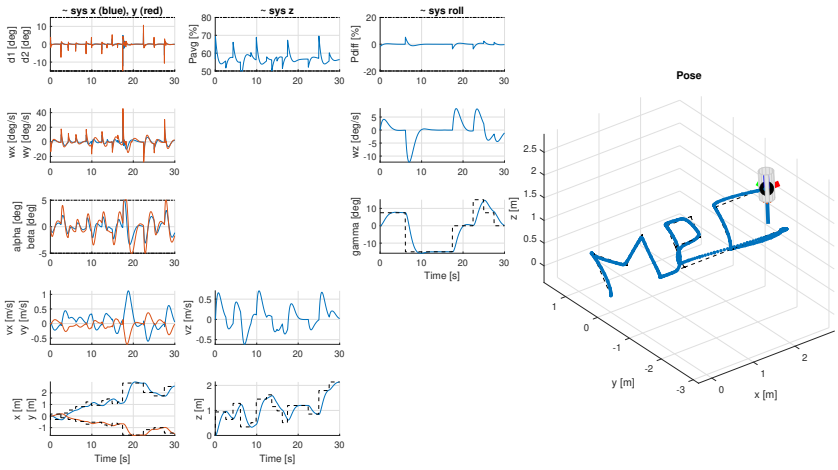

Figure 2.10: Tracking performance using parameters after tweaking

Table 2.2: Parameters for deliverable 4-1

Parameter	Value
H	5
Q_x	$diag(40, 10, 1, 20)$
R_x	0.1
Q_y	$diag(40, 10, 1, 20)$
R_y	0.1
Q_z	$diag(1, 20)$
R_z	0.1
Q_{roll}	$diag(1, 20)$
R_{roll}	0.1

2.5. Deliverable 5-1

In this deliverable, we aim to achieve offset-free tracking when a variable load is imposed on our rocket system.

To do so, we modified the dynamics of mpc controller for the z subsystem by incorporating the disturbances d_k as Eq. (2.6),

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + \underbrace{B_d d_k}_{\text{disturbance}} \\ y_k &= Cx_k + Du_k + C_d d_k \\ d_{k+1} &= d_k \end{aligned} \tag{2.6}$$

where in our system we set $B_d = B, C = [0, 1]^T, D = 0, C_d = 0$. The idea is that the state transition is determined by current state, system input and system disturbance. Since the disturbance is unknown

and we have no prior knowledge on the units of the input, we could set matrix B_d as a random matrix that satisfies the observer rank requirement. As for matrix C_d, since we assume the output of the z subsystem is the velocity of on z direction, therefore the matrix C_d is set as 0.

As for controller parameters, we tweaked a little bit on the basis of Section 2.2, the current parameters for each controller is shown in Table 2.3.

Table 2.3: Parameters for deliverable 5-1

Parameter	Value
H	5
Q_x	$diag(40, 8, 1, 20)$
R_x	0.1
Q_y	$diag(40, 8, 1, 20)$
R_y	0.1
Q_z	$diag(1, 10)$
R_z	0.1
Q_{roll}	$diag(1, 10)$
R_{roll}	0.1

In terms of observer parameters tweaking, we place the poles of the system $A+LC$ to $[0.1, 0.2, 0.3]^T$ to ensure fast estimation process, even though it will bring relatively higher overshoot to the estimation. (as shown in Fig. 2.11).

$$\begin{bmatrix} \hat{x}_{k+1} \\ \hat{d}_{k+1} \end{bmatrix} = \begin{bmatrix} A & B_d \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ \hat{d}_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k + \begin{bmatrix} L_x \\ L_d \end{bmatrix} \left(C\hat{x}_k + C_d\hat{d}_k + Du_k - y_k \right) \tag{2.7}$$

where $\hat{x}, \hat{d}$ are estimates of the state and disturbance. And the error dynamics now is,

$$\begin{aligned} \begin{bmatrix} x_{k+1} - \hat{x}_{k+1} \\ d_{k+1} - \hat{d}_{k+1} \end{bmatrix} &= \begin{bmatrix} A & B_d \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ d_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k \\ &\quad - \begin{bmatrix} A & B_d \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ \hat{d}_k \end{bmatrix} - \begin{bmatrix} B \\ 0 \end{bmatrix} u_k - \begin{bmatrix} L_x \\ L_d \end{bmatrix} \left(C\hat{x}_k + C_d\hat{d}_k - Cx_k - C_d d_k \right) \\ &= \left(\begin{bmatrix} A & B_d \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} L_x \\ L_d \end{bmatrix} \begin{bmatrix} C & C_d \end{bmatrix} \right) \begin{bmatrix} x_k - \hat{x}_k \\ d_k - \hat{d}_k \end{bmatrix} \end{aligned} \tag{2.8}$$

The performance of our rocket system with/without offset-tracking is shown in Fig. 2.12, we then test several disturbances with different magnitude, namely, 1.8 and 2.0, the results are shown in Figs. 2.13 and 2.14.

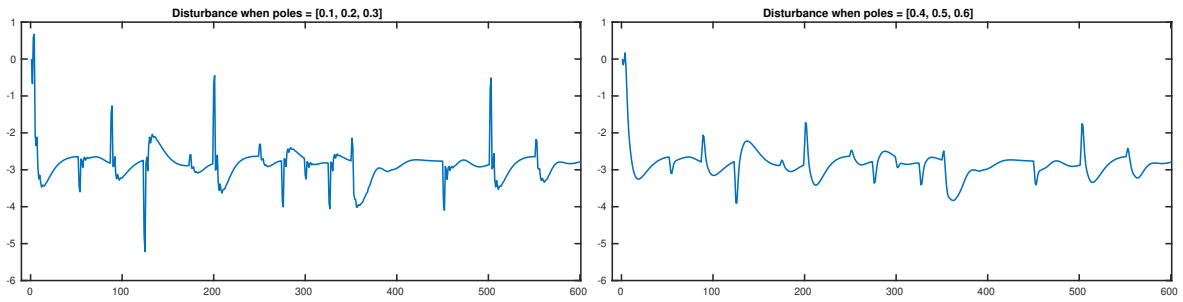

Figure 2.11: Estimated disturbance when the load=1.783 under different pole placements of the estimator

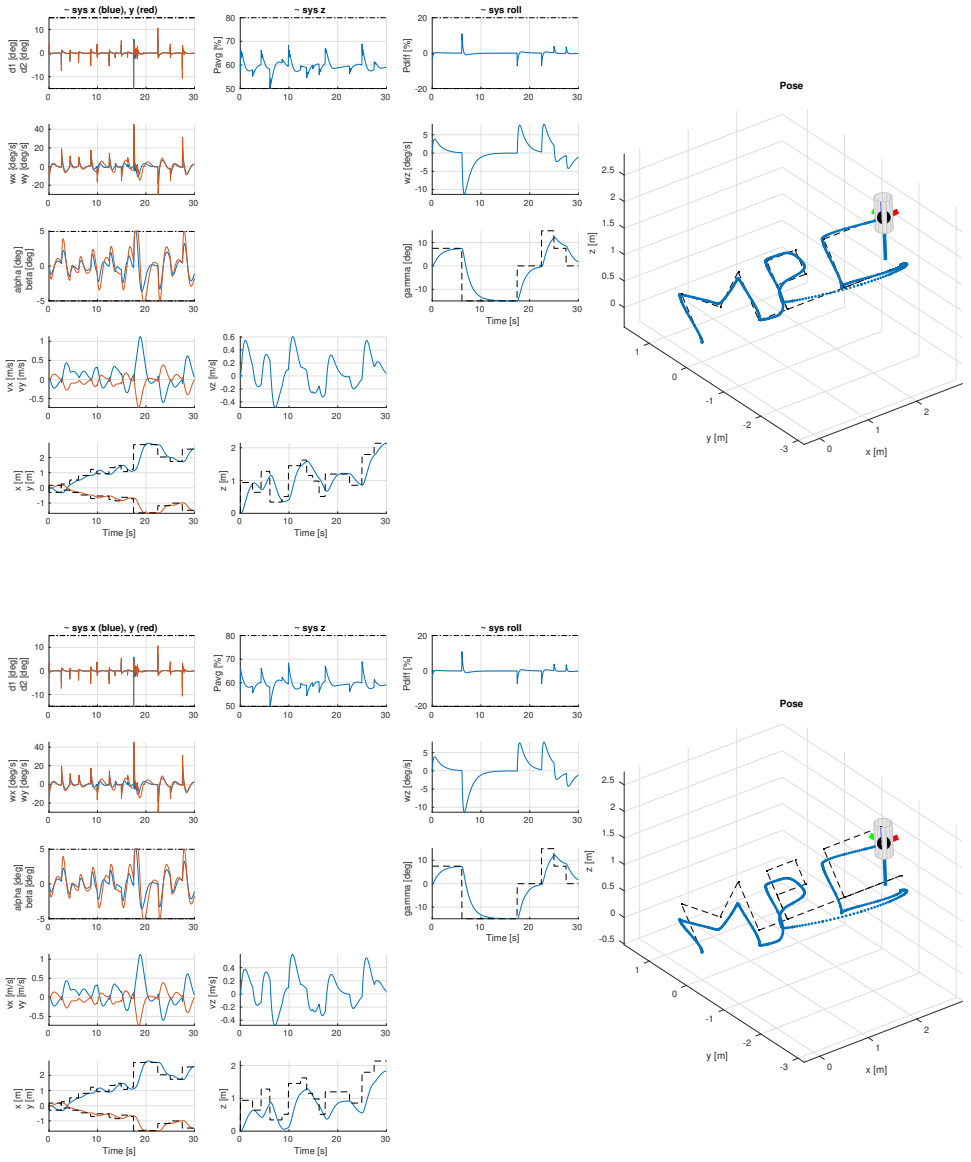

Figure 2.12: The rocket system under variance load with/without off-set free tracking

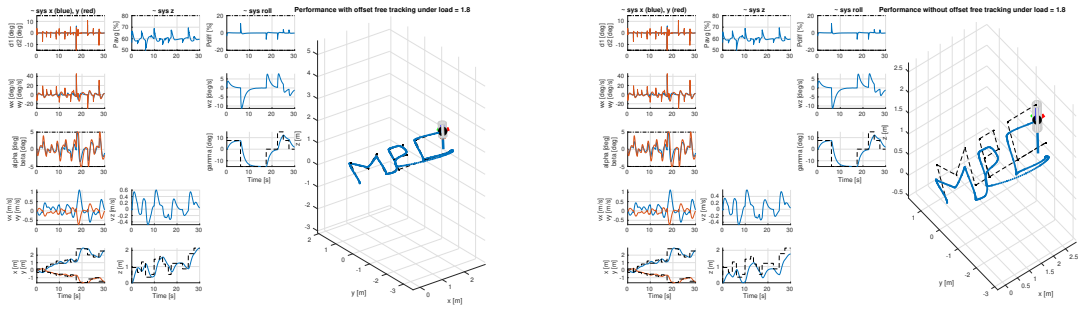

Figure 2.13: Rocket system with/without offset-free tracking under load = 1.8

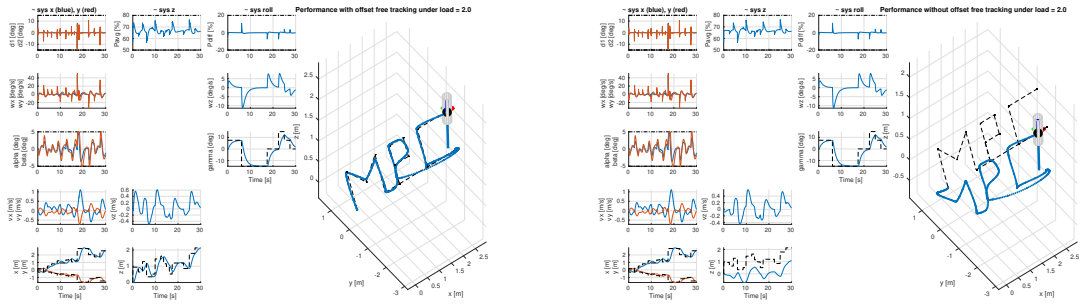

Figure 2.14: Rocket system with/without offset-free tracking under load = 2.0

2.6. Deliverable 6-1

In this deliverable, we aim to design a Nonlinear MPC (NMPC) controller for the overall control of our mini rocket.

First, we use the similar penalty parameters we used in Table 2.2, namely,

$$\begin{aligned} Q &= diag(40, 40, 1, 10, 10, 20, 1, 1, 1, 20, 20, 20) \\ R &= diag(0.1, 0.1, 0.1, 0.1) \end{aligned}$$

We set up the problem with CasADi optimization solver with optimization goal and constraints shown in Eq. (2.9).

$$\begin{aligned} \min \quad & \textstyle\sum_{i=0}^{N-1}(x_i - x_s)^T Q(x_i - x_s) + (u_i - u_s)^T R(u_i - u_s) + V_f(x_N - x_s) \\ \text{s.t.} \quad & x_0 = x \\ & x_{i+1} = f_{discrete}(x_i, u_i) \\ & H_x x_i \leq k_x \\ & H_u u_i \leq k_u \end{aligned} \tag{2.9}$$

In terms of terminal implementation, we designed two ways to achieve a similar result. One way is to create steady state optimization variables and impose terminal state constraint on them and giving them to the optimizer to solve, the other way is to design a terminal cost V_f via LQR by linearizing the system at the steady point. While the rocket moves along the predefined trajectory, i.e. different reference states, we get different steady points along the way, which means we need to compute the terminal cost online. To accelerate calculation, we choose the linearization method which linearized the system near the steady point, and discretize it. The parameters are the same Q, R we chose as mentioned for the terminal LQR controller. Terminal set is empty so we drop the terminal set constraint.

We use Runge-Kutta 4 as the discrete integration algorithm. u are constrained as what in the previous sections. As for x, previously in linear MPC, we need to constrain them near the trim point to meet the assumption that the system functions will be similar to that of trim point. Since NMPC covered the whole state space, we don't need these constraints anymore now. However, to avoid singularity, we set $|\beta| \leq 85°$.

Nonlinear calculation is more computational expensive, so we decrease the time step to 0.1s. But it still cost much more time to calculate comparing to linear one. It takes about 109s to compute a 30-second trajectory, but linear MPC takes only about 23s. We found there's no obvious difference between the performance of 5-second-horizon one and 1-second-horizon one (Fig. 2.16), and 1-second-horizon one takes about 18s. So we decide to use horizon of 1 second.

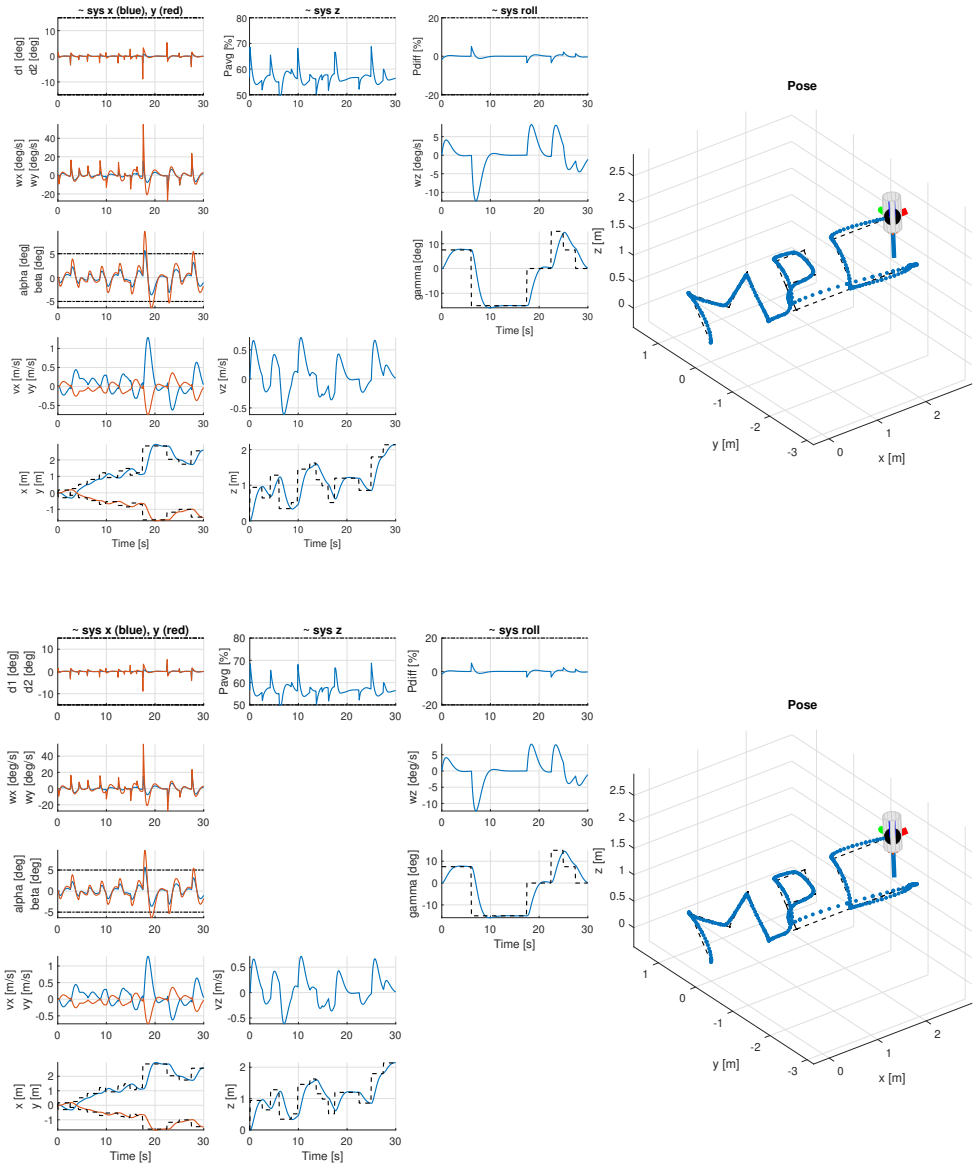

Figure 2.15: NMPC performance with Horizon of 5 seconds(top) and of 1 seconds(bottom)

When the reference is far away from trim point, for example, if we use trajectory with maximum roll of $50°$, linear controller will perform much worse(we re-tune the Q matrices a little to make it feasible). It is because that when the rocket's rolling angle is quite big, the difference of the system function between the that of trim point and of current state is not neglectable, therefore, out of the controller's expectation, the system will not act as the controller predicted.

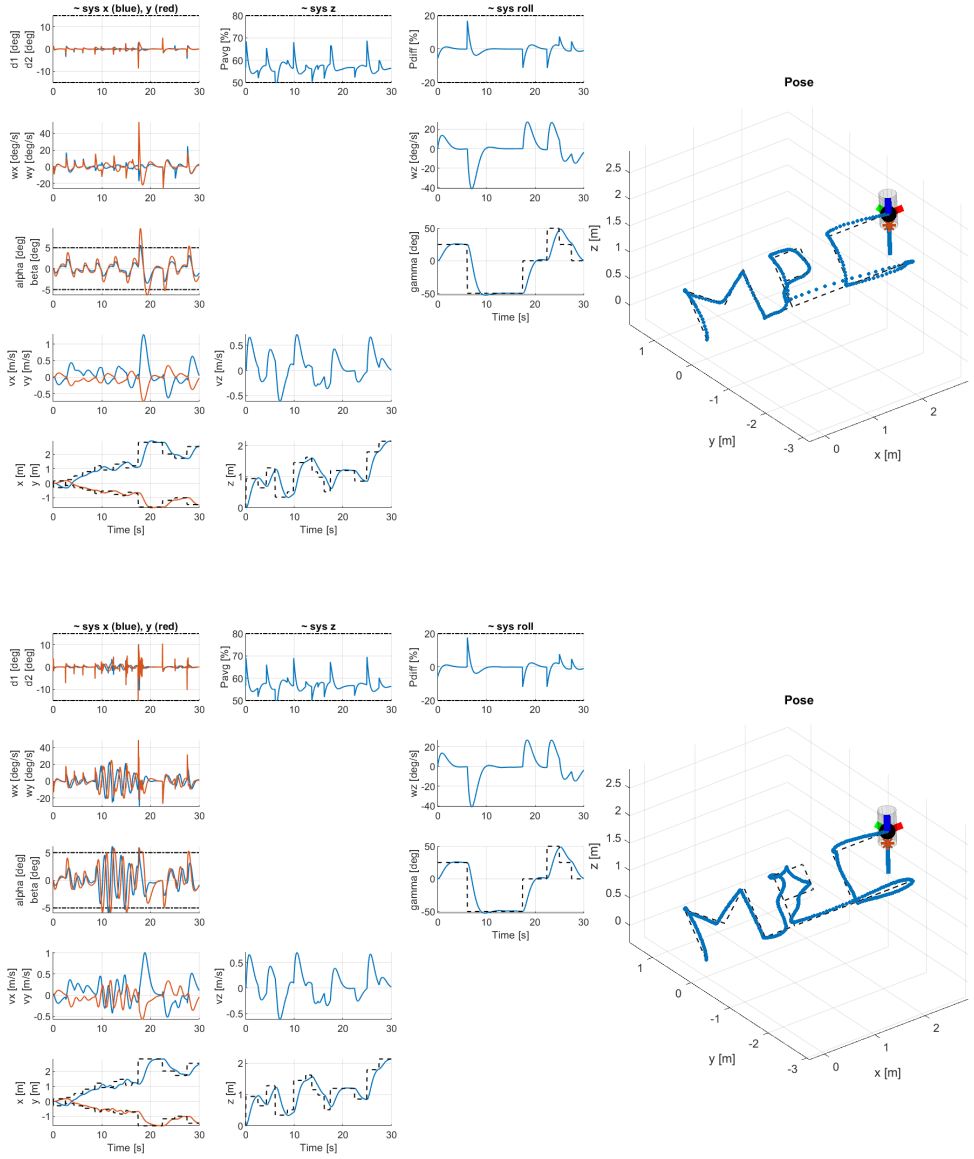

Figure 2.16: Performance of Nonlinear MPC (top) and Liner MPC (bottom) with $|\gamma| \leq 50^\circ$

Comparing to linear controllers, Nonlinear MPC controller 1) has larger feasible region since it does not need the system to function within a short distance from the steady state, and it works directly on the entire states so it's applicable for the whole state space. 2) is much more computationally expensive. To solve nonlinear optimization problem, we have to discretize the system functions, calculate the Jacobian matrices, and deal with the 4 tangled subsystems at meantime. These have to be done online. 3) can work without linearizing in advance and decomposing or merging in the controlling process. 4) can be tricky and challenging to debug if numerical issues or local optima exist.

References

[1] José David López, Jairo José Espinosa, and John Ramiro Agudelo. “LQR control for speed and torque of internal combustion engines”. In: *IFAC Proceedings Volumes* 44.1 (2011), pp. 2230–2235.

[2] Roland Pfeiffer et al. “System identification and LQG control of variable-compression HCCI engine dynamics”. In: *Proceedings of the 2004 IEEE International Conference on Control Applications, 2004.* Vol. 2. IEEE. 2004, pp. 1442–1447.

[3] Mohammad Reza Amini et al. “Discrete adaptive second order sliding mode controller design with application to automotive control systems with model uncertainties”. In: *2017 American Control Conference (ACC)*. IEEE. 2017, pp. 4766–4771.

[4] Behrouz K Irdmousa et al. “Data-driven modeling and predictive control of combustion phasing for RCCI engines”. In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 1617–1622.