ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

MPC CONTROLLER TO FLY A ROCKET PROTOTYPE

MODEL PREDICTIVE CONTROL (ME-425)

PROF: COLIN JONES

Elisa Ferrara 371064
Edoardo Finamore 377644
Angelo Giovine 368440

12th January 2024

Contents

CHAPTER 1

INTRODUCTION

This report presents an in-depth analysis of a Model Predictive Control (MPC) problem focusing on rocket control, constituting the final mini-project for Professor C. Jones' MPC course at EPFL . The project revolves around a nonlinear model of a rocket, using drone propellers for simplicity. It comprehensively examines various MPC strategies, beginning with a linear controller applied to a linearized model and progressing to its application on the nonlinear model for both regulation and tracking tasks. The report delves into the challenges posed by varying mass, highlighting the critical role of an observer. It culminates with the implementation of a nonlinear MPC, significantly enhancing performance at the expense of increased computational demands.

The report has to be considered together with the Matlab code.

CHAPTER 2

LINEAR MPC

2.1 DELIVERABLE 2.1

Explain from an intuitive physical / mechanical perspective, why this separation into independent subsystems is possible.

In this deliverable, and in most subsequent ones, we divide the system into four smaller subsystems in such a way that each subsystem is independent of the others and can be controlled independently.

Intuitively, we can understand that this division is correctly executed, and any correlation between them is surmountable. This is due to two main reasons:

First of all, each subsystem has a **dedicated input for the specific movement** it controls, be it x, y, or z translation, or γ rotation. For example, in the x and y subsystems, the respective thrust vector angles

(d1 and d2) directly influence lateral movement, while in the z subsystem, the altitude is controlled by the average throttle P_{avg}. Similarly, the γ (in the following also referred to as Roll) rotation in the roll subsystem is managed by the differential throttle P_{diff}. Furthermore, **each axis is controlled independently from the others**: rotations about the x and y axes have their own servos, while for controlling the z-axis and the roll angle, two different propellers are used. The average power and the power difference of these propellers are used as independent quantities, ensuring distinct control over the z-axis movement and the roll angle.

Moreover, after linearizing, we obtain the matrices A, B, C, D of the linearized system. B in particular represents how the four inputs affect the twelve states. We can see that each input affects only the variables of the respective system (highlighted in colors): red for system x,green for system y, blue for system z and yellow for system roll.

B matrix $=$

	$d1$	$d2$	P_{avg}	P_{diff}
wx	-55.68	0	0	0
wy	0	-55.68	0	0
wz	0	0	0	-0.104
α	0	0	0	0
β	0	0	0	0
γ	0	0	0	0
vx	0	9.81	0	0
vy	-9.81	0	0	0
vz	0	0	0.1731	0
x	0	0	0	0
y	0	0	0	0
z	0	0	0	0

We would have had coupling, for example, if the mechanism that tilts the propellers (servos for d1 and d2) is physically linked in a way that moving one servo slightly moves the other, this would introduce coupling between the x and y axes. Another example of coupling could be the presence of just one propeller: in this case in order to increase the altitude z one would unintentionally generate rotation around z axis because of the reaction torque (roll moment). The structure of the system allows to avoid these couplings. However, since the model does never reflects exactly the reality, some mismatches in the assembly of the rocket or some aerodynamic effects could create couplings, in any case negligible.

CHAPTER 3

DESIGN MPC CONTROLLERS FOR EACH SUB-SYSTEM

3.1 DELIVERABLE 3.1: DESIGN MPC REGULATORS

3.1.1 QUESTION 1

Explanation of design procedure that ensures recursive constraint satisfaction.

The MPC problem is called "recursively feasible", if for all feasible states feasibility is guaranteed at every state along the closed-loop trajectory.

In MPC theory there there exist the following theorem for stability of an MPC problem:

Theorem 3.1.1 (Stability of MPC) *The closed-loop system under the MPC control law $u_0^*(x)$ is stable and the system $x^+ = Ax + Bu_0^*(x)$ is invariant in the feasible set X_N if:*

1. *The stage cost is a positive definite function, i.e., it is strictly positive and only zero at the origin.*
2. *The terminal set is invariant under the local control law $k_f(x)$:*

$$x^+ = Ax + Bk_f(x) \in \chi_f \quad \textit{for all } x \in \chi_f.$$

   All state and input constraints are satisfied in χ_f:

$$\chi_f \subseteq X, \quad k_f(x) \in U \textit{ for all } x \in \chi_f.$$

3. *The terminal cost is a continuous Lyapunov function in the terminal set χ_f:*

$$V_f(x^+) - V_f(x) \leq -l(x, k_f(x)) \textit{ for all } x \in \chi_f.$$

In our case, after the linearization of the system and having input and state constraints as polytopes in order to satisfy the hypothesis of the theorem, we chose:

- The terminal controller as $\kappa_f(x) = Kx$, as the optimal unconstrained LQR control law: $K = -(R + B^T P B)^{-1} B^T P A$ where P is the solution of the DARE.
- The terminal weight as $V_f(x) := x^T P x = \sum_{i=0}^{\infty} x_i^T Q x_i + x_i^T K^T R K x_i$, as the optimal LQR cost. This is implemented through the MPT toolbox.

- The terminal set χ_f as the maximum invariant set for the closed-loop system $x^+ = (A + BK)x$ subject to $\chi_f \subseteq X$, $K\chi_f \subseteq U$. Again, we used the MPT toolbox.

Given these three parameters the hypothesis of the theorem are satisfied:

1. The stage cost $l(x, u) = x^T Qx + u^T Ru$ is positive definite because we choose $Q \geq 0$ and $R > 0$.
2. Since the terminal set χ_f has been defined as the largest invariant set for the control law it is invariant under it.
3. We can show that if we choose Q and R as mentioned above then the terminal cost is a continuous Lyapunov function.

The theorem guarantees recursive feasibility and asymptotic stability.

3.1.2 QUESTION 2

Explanation of choice of tuning parameters. (e.g., Q, R, H, terminal components):

As stated in the previous question, Q must be positive semi-definite and R positive definite. Moreover, we were given a maximum settling time of 7s. In order to achieve stability and the latter requirement, we settled the MPC problem parameters as follows:

- **Q and R matrices**. For x and y controllers we did not find any problems neither of non-convergence type nor of infeasibility. We took advantage of this to request a small input. The final choice sees for both the controllers, diagonal matrices in which the entries of R are bigger then the ones on Q in order to penalize more big inputs.

  For altitude and roll angle controllers we used a different approach. Setting Q»R brought to infeasibility of the problem while R«Q made the problem not converging. So the techique used to tune the matrices was starting from identity matrices and increasing Q entries (or decreasing R) to maximize the magnitude of the terminal set respecting the constraint about the settling time. The final values found are summarized in the table 3.1. Increasing the entries of Q makes χ_f smaller and the settling time bigger and vice versa.

	Q	R
System X	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 100 \end{bmatrix}$
System Y	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 100 \end{bmatrix}$
System Z	$\begin{bmatrix} 20 & 0 \\ 0 & 20 \end{bmatrix}$	$\begin{bmatrix} 1 \end{bmatrix}$
System Roll	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 0.1 \end{bmatrix}$

TABLE 3.1
Q and R matrices for the various subsystems

- **H**. The choice of H = 7 is sufficiently high to ensure a sufficiently large region of attraction. We will choose a shorter horizon in case of non-linear MPC to reduce the computational cost.

- **Terminal Component** The terminal component is calculated through the MPT toolbox.

3.1.3 QUESTION 3

Plot of terminal invariant set for each of the dimensions, and explanation of how they were designed and how their respective tuning parameters were used.

Choices of the parameters to generate the terminal sets using MPT toolbox.

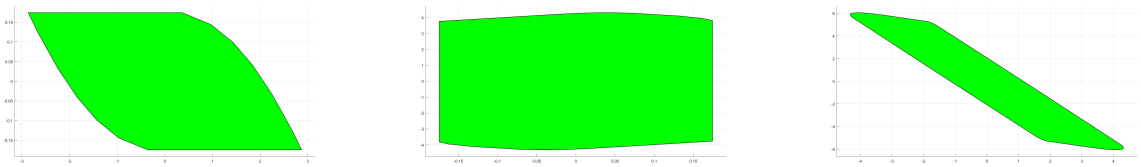
- States constraints. The problem gave constraints exclusively on α and β due to linearization; we set them to be less then 15°in absolute value. All other limits are set to Inf or -Inf.
- Input constraints were given and are summarized in the following table 3.2:

	Lower limit	**Upper limit**
d1	-15°	15°
d2	-15°	15°
$\mathbf{P_{avg}}$	50%	80%
$\mathbf{P_{diff}}$	-20%	20%

TABLE 3.2
Input constraints

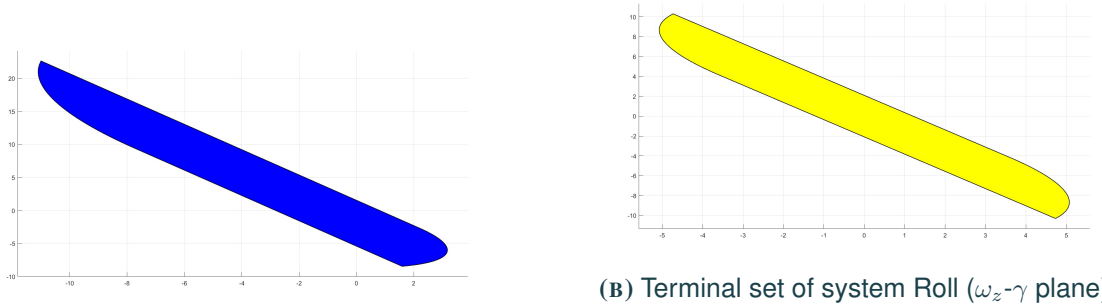
The constraints over the servos are physical. P_{avg} ones instead try to limit the downward acceleration that can occur to the rocket. P_{diff} limit holds the throttle of the rocket in between [0-100]%.

The Terminal set must be a sub-set of the input and state constraints as we can verify in the images below 3.1.



(A) Projection over ω_y-β plane (B) Projection over β-v_x plane (C) Projection over v_x-x plane

FIGURE 3.1
Terminal set projections of system x (or system y)

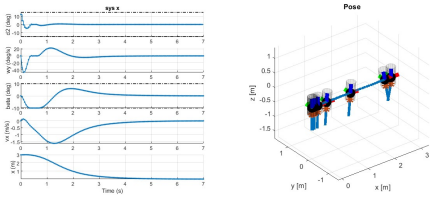


(A) Terminal set of system z (v_z-z plane)

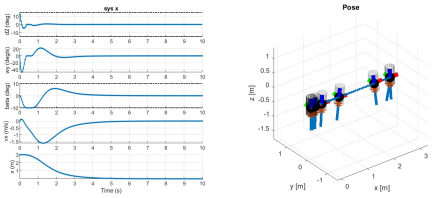
(B) Terminal set of system Roll (ω_z-γ plane)

3.1.4 QUESTION 4

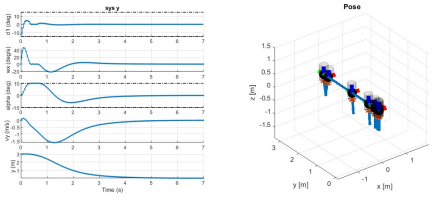
Open-loop and closed-loop plots for each dimension of the system starting stationary at 3 meters from the origin (for x, y and z) or stationary at 30∘ for roll. 3.4



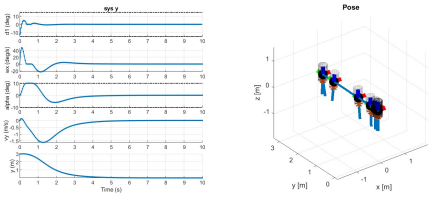
(A) System X: open-loop trajectory



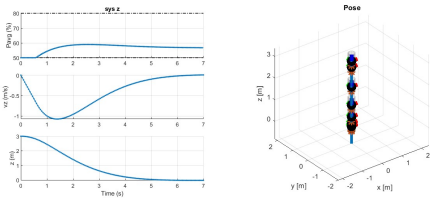
(B) System X: closed-loop trajectory



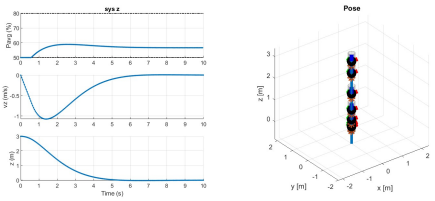
(C) System Y: open-loop trajectory



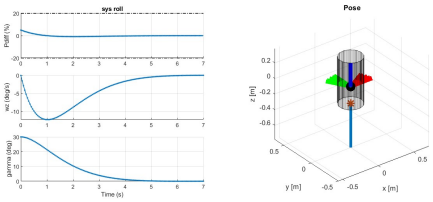
(D) System Y: closed-loop trajectory



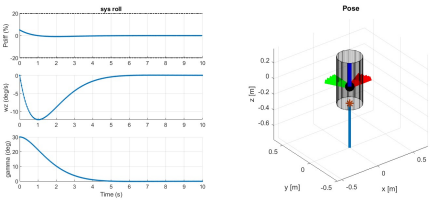
(E) System Z: open-loop trajectory



(F) System Z: closed-loop trajectory



(G) System Roll: open-loop trajectory



(H) System Roll: closed-loop trajectory

FIGURE 3.3
MPC regulator: open-loop and closed loop trajectories for each subsystem

3.2 DELIVERABLE 3.2: DESIGN MPC TRACKING CONTROLLERS

3.2.1 QUESTION 1

Explanation of your design procedure and choice of tuning parameters.

In order to track constant references, it is necessary to compute the steady-state $(\mathbf{x}_s, \mathbf{u}_s)$ corresponding to the reference $\mathbf{ref}$. This computation is achieved by solving a straightforward quadratic programming problem that also incorporates the target condition into its constraints.

Steady-state target problem formulation:

$$\begin{aligned} \min_{\mathbf{u}_s} \quad & \mathbf{u}_s^T R_s \mathbf{u}_s \\ \text{subject to} \quad & \begin{bmatrix} I - A & -B \\ C & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_s \\ \mathbf{u}_s \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{ref} \end{bmatrix} \\ & F\mathbf{x}_s \leq f \quad \text{(state constraints)} \\ & M\mathbf{u}_s \leq m \quad \text{(input constraints)} \end{aligned} \tag{3.1}$$

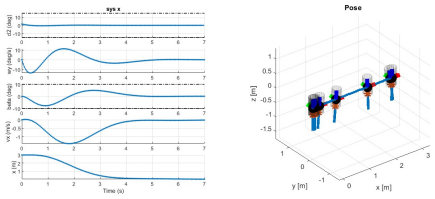
The matrix $\mathbf{R_s}$ could be set equal to 1.

Literature offers an alternative in case there is no solution to the previous problem: a reachable set point closest to the reference is computed. This is achieved by solving an optimization problem that minimizes the expression $(C\mathbf{x}_s - \mathbf{ref})$, however this approach is not implemented in the code.

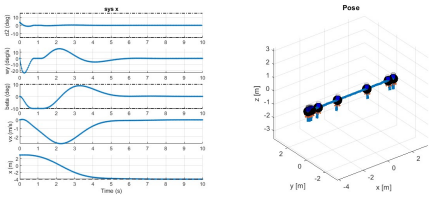
The target steady-state $(\mathbf{x}_s, \mathbf{u}_s)$ is then provided as an input to the newly configured `setup_controller` function. Within this function, the matrices $\mathbf{Q}$ and $\mathbf{R}$ have been altered from their original versions. Specific entries in these matrices are adjusted through a trial-and-error process to optimize performance.

3.2.2 QUESTION 2

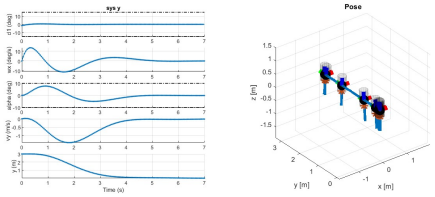
Open-loop and closed-loop plots for each dimension starting at the origin and tracking a reference to -4 meters (for x, y, and z) and to 35 degrees for roll.



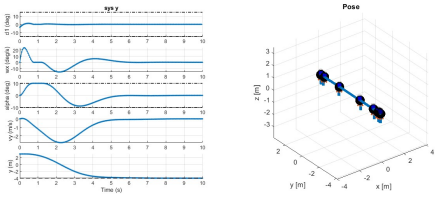
(A) System X: open-loop trajectory



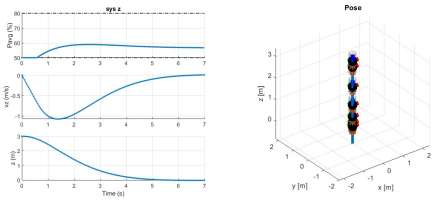
(B) System X: closed-loop trajectory



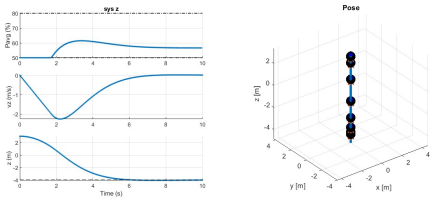
(C) System Y: open-loop trajectory



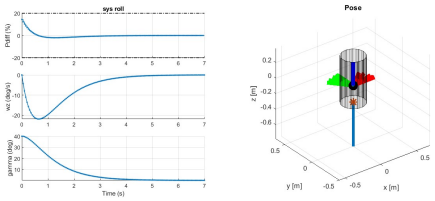
(D) System Y: closed-loop trajectory



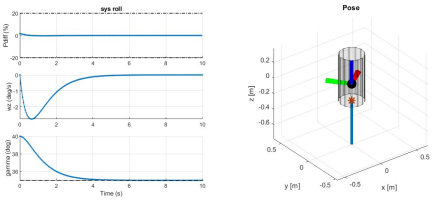
(E) System Z: open-loop trajectory



(F) System Z: closed-loop trajectory



(G) System Roll: open-loop trajectory



(H) System Roll: closed-loop trajectory

FIGURE 3.4
MPC Tracking regulator: open-loop and closed loop trajectories for each subsystem

CHAPTER 4

SIMULATION WITH NONLINEAR ROCKET

4.1 DELIVERABLE 4.1: SIMULATION WITH NONLINEAR ROCKET

A plot of your controllers successfully tracking the reference path and reference roll angle. If your tracking performance is not good, explain how you adapt your tuning to improve it.

4.1.1 QUESTION 1

MATRIX TUNING

A first naive approach used to track the non linear reference is to manually tune entries of matrices $\mathbf{Q}$ and $\mathbf{R}$. The role of $\mathbf{Q}$ is to penalize high absolute values of the states, whereas $\mathbf{R}$ is focused on inputs. It is notable that the relative importance of the states or of the inputs is attributed to the relative weight of the corresponding matrix entries. This means that a change in the entries of $\mathbf{Q}$ implies to vary also values of $\mathbf{R}$. In particular, in order to avoid too aggressive manoeuvres, we decided to set higher weights on inputs. Indeed, high oscillations on inputs are deplorable and a finer tuning is mandatory.

A first result of tuning can be appreciated in the following Figure 4.1.

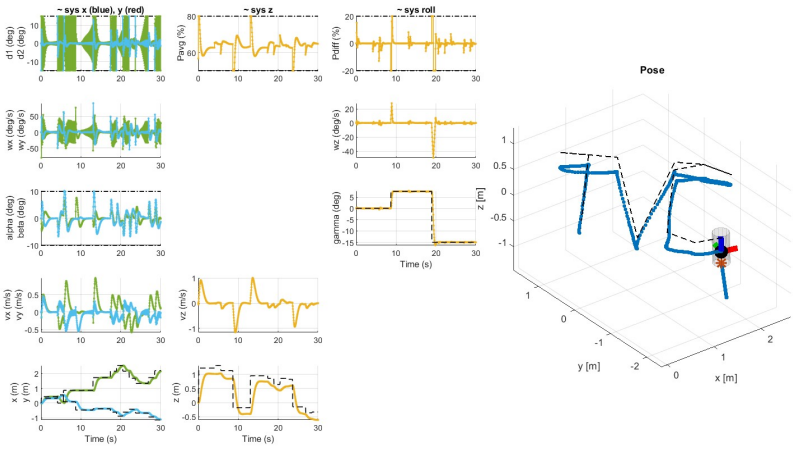


FIGURE 4.1
Simulation plot in case of a nonlinear time-varying trajectory.

From the plot of the simulation, it's evident that tracking is not perfect and a better tuning can be achieved.

The number of parameters and the way they interfere with each other make the manual tuning more difficult. It's possible to notice by looking to the plot of the simulation in Figure 4.2 that a better tracking of the reference could be obtained by reducing the R entry corresponding to P_{avg}, which is related to z coordinate. However, inputs still show oscillatory behaviour. To overcome this issue, slack variables are considered in the MPC problem formulation.

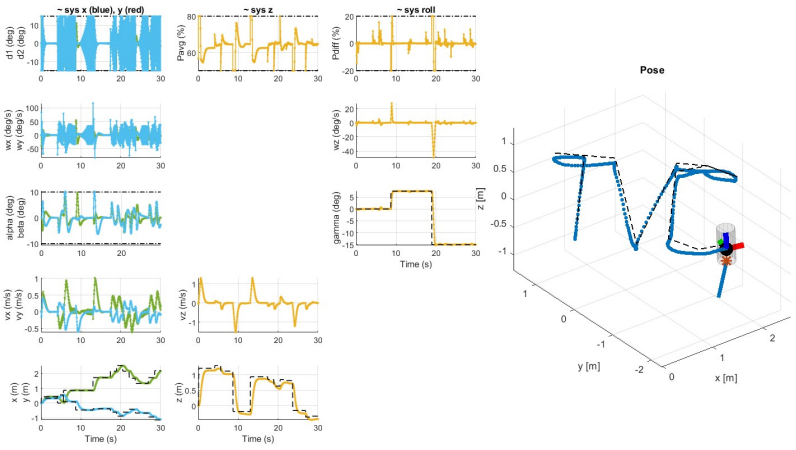


FIGURE 4.2
Simulation plot in case of a nonlinear time-varying trajectory, after matrix tuning.

SOFT CONSTRAINTS IMPLEMENTATION

A different approach that could increase the precision in tracking is to add slack variables. We decided to soften constraints over the different state and input variables that allow doing such a relaxation. In particular, constraints on α and β are due to linearization; a small violation of them gives relevant improvements. We included slack variables also on P_{avg} constraint because it is not a physical limit. Indeed, the limitation in range 50-80% is related to limitation on the downward acceleration that can occur to the rocket. However, we also set a strict constraint (i.e. without slack variables) on P_{avg} to force it to stick to the range 0-100%. We used the single optimization problem formulation integrating a quadratic cost function for epsilon and adding the necessary modifications to the constraints. Increases in entries of S leads to a'hardening' of the soft constraints: reduced size of violation for a longer duration.

The following formulation is valid for each one of the controllers realized for the subsystems x, y, z, Roll. The difference is in the matrix S of quadratic cost and on the position of slack variables on constraints.

$$J^*(x) = \min \Bigg(\sum_{i=0}^{N-1} \Big((x_i - x_{\text{ref}})^T Q (x_i - x_{\text{ref}}) + (u_i - u_{\text{ref}})^T R (u_i - u_{\text{ref}}) + \rho(\epsilon_i) \Big) + \Big((x_N - x_{\text{ref}})^T Q_f (x_N - x_{\text{ref}}) + \rho(\epsilon_N) \Big) \Bigg) \tag{4.1}$$

$$s.t. \qquad x_{i+1} = Ax_i + Bu_i \quad \forall i = 0, 1, \ldots, N-1 \tag{4.2}$$

$$Mu_i \leq m + \epsilon_i \tag{4.3}$$

$$Fx_i \leq f + \epsilon_i \tag{4.4}$$

$$\epsilon_i \geq 0 \tag{4.5}$$

$$F_f(x_N - x_{ref}) \leq f_f \tag{4.6}$$

where $\rho(\epsilon_N) = \epsilon_i^T S \epsilon_i$.

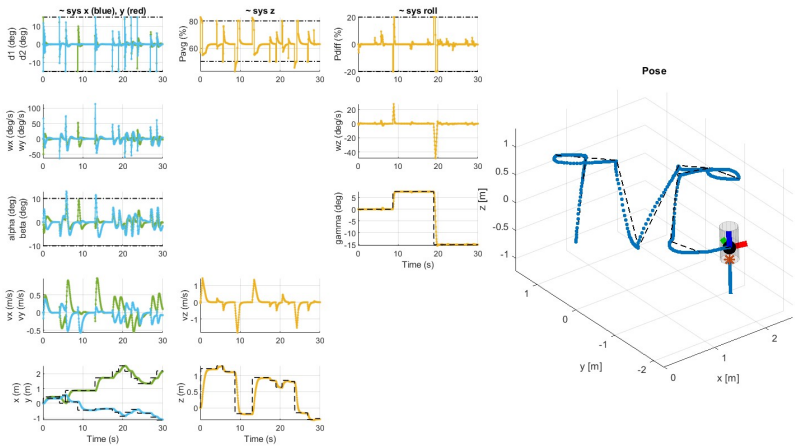


FIGURE 4.3
Simulation plot in case of a nonlinear time-varying trajectory, after matrix tuning and addition of soft constraints.

As expected, one can observe from Figure 4.3 that the introduction of soft constraints allows the controller to violate in some cases the limits for P_{avg}, α and β.

Despite the improvements, perfect tracking cannot be achieved because of linear controllers and a linearized system describing non linear model and tracking reference. This causes of a mismatch between the prediction of the controllers and the real behaviour of the system.

CHAPTER 5

OFFSET-FREE TRACKING

5.1 DELIVERABLE 5.1: CONSTANT MASS

5.1.1 QUESTION 1

Explanation of your design procedure and choice of tuning parameters.

To estimate a constant disturbance d, it is first essential to implement an augmented model defined as follows:

$$\begin{aligned} x_{i+1} &= Ax_i + Bu_i + Bd_i \\ d_{i+1} &= d_i \\ y_i &= Cx_i \end{aligned}$$

Let M be the matrix defined as:

$$M = \begin{bmatrix} A - I & B \\ C & 0 \end{bmatrix} \tag{5.1}$$

Given that the pair (A, C) is observable and the matrix M possesses full column rank, the observability of the augmented state is assured.

With the model now encapsulating the disturbance dynamics, it becomes possible to estimate it using a Luenberger Observer. This method offers the advantages of simplicity in implementation and effectiveness. The observer's poles were determined through a trial-and-error approach to achieve a reasonable overshoot and settling time for the z trajectory (Figure 5.1 and Figure 5.2).

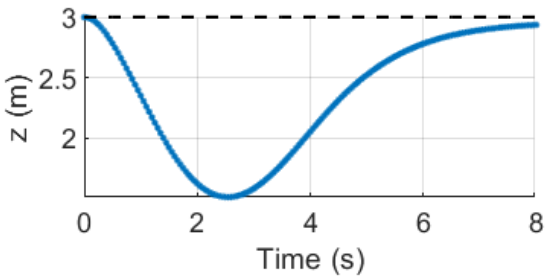


FIGURE 5.1
L poles in 0.90, 0.95, 0.97

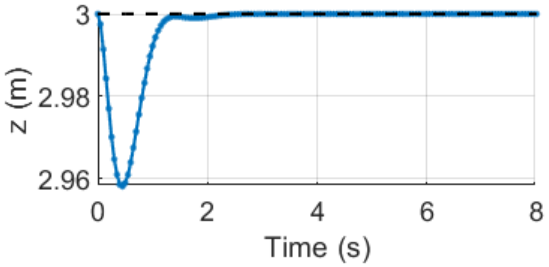


FIGURE 5.2
L poles in 0.35, 0.42, 0.49

Once the disturbance is estimated, the next step involves defining a new Steady-State Target problem

(akin to that presented in Deliverable 3.2), incorporating an adapted target condition into its constraints. New target condition:

$$\begin{bmatrix} I - A & -B \\ C & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_s \\ \mathbf{u}_s \end{bmatrix} = \begin{bmatrix} Bd \\ \mathbf{ref} \end{bmatrix}$$

Finally, the target steady-state and the estimated disturbance, are given as input into the reconfigured `setup_controller` function.

5.1.2 QUESTION 2

Plot showing the impact of changing the mass on your original controller from Part 4, and then another plot showing that your controller now achieves offset-free tracking.

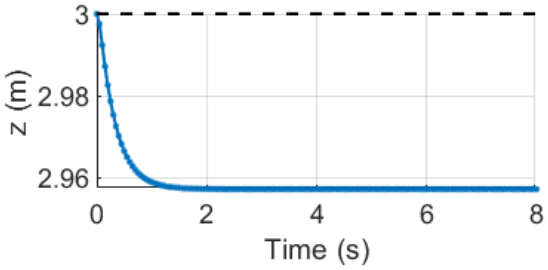


FIGURE 5.3
Controller without Observer, Off-set free tracking not achieved

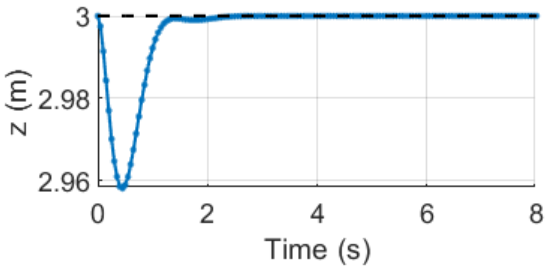


FIGURE 5.4
Controller with Observer, Off-set free tracking achieved

5.2 DELIVERABLE 5.2: THRUST-DEPENDENT MASS

5.2.1 QUESTION 1

Plot showing the impact of having a thrust-dependent mass decrease during flight on your controller from Part 5.1.

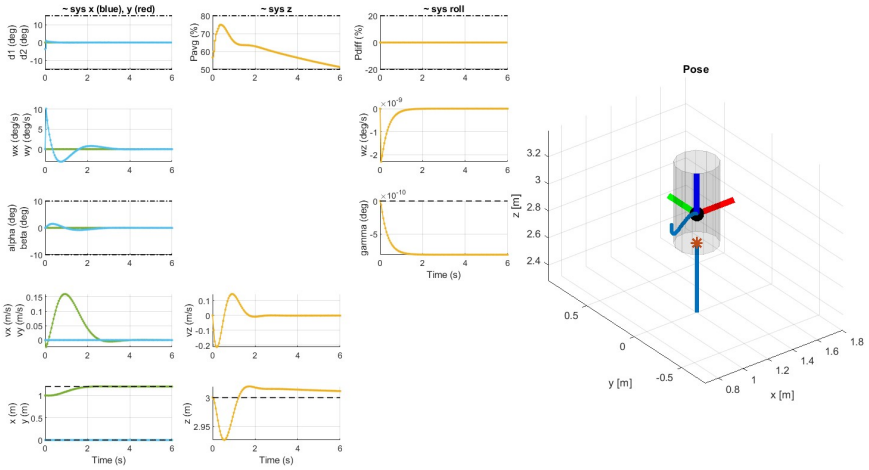


FIGURE 5.5
Plot showing impact of thrust-dependent mass

5.2.2 QUESTION 2

In the first couple of seconds of the simulation, despite the estimator, why is there still a tracking offset in height? Explain how the estimator could be modified to achieve offset-free tracking also for the changing mass case.

Despite the presence of the estimator, an offset is present. The reason for this is that the real mass is lower than the one expected from the observer: fuel is indeed consumed over time. Since the observer is not aware of this change, the controller asks for a higher input than the required one. In order to overcome this issue, it is possible to change the type of the estimator that allows for a time varying disturbance estimation. The reference diagram for this controller is given in the following Figure 5.6.

A disturbance generator (autonomous system) has to be considered and provides at every step of the simulation the value of the disturbance. In this case the disturbance model can be represented by the time evolution of the mass considering the rocket mass rate.

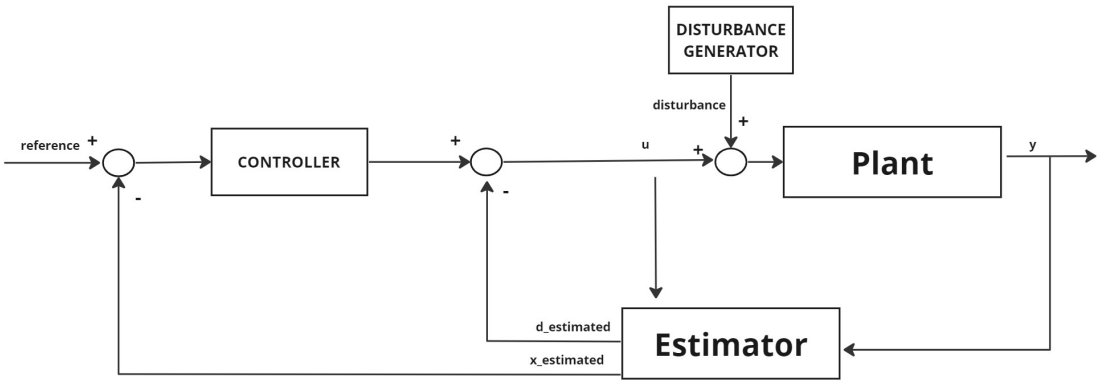


FIGURE 5.6
Block diagram of a possible implementation of a time-varying disturbance estimator for tracking.

5.2.3 QUESTION 3

With time-varying mass, you might observe multiple distinct properties of the trajectory (depending on your tuning). Briefly describe which different behaviors you can see along the simulation. Towards the end of the simulation, what unexpected behavior can you observe, and why?

In this section we are going to discuss the physical reasons behind the behaviour of the rocket in case of a time-varying mass. We distinguish between cases with and without estimators, because of the different phenomena involved.

Note that we removed the input constraints in the `setup_controller` function, since the MPC formulation already has them and they may lead to infeasibility problems.

A sketch of the expected behaviour is shown in the Figure 5.7.

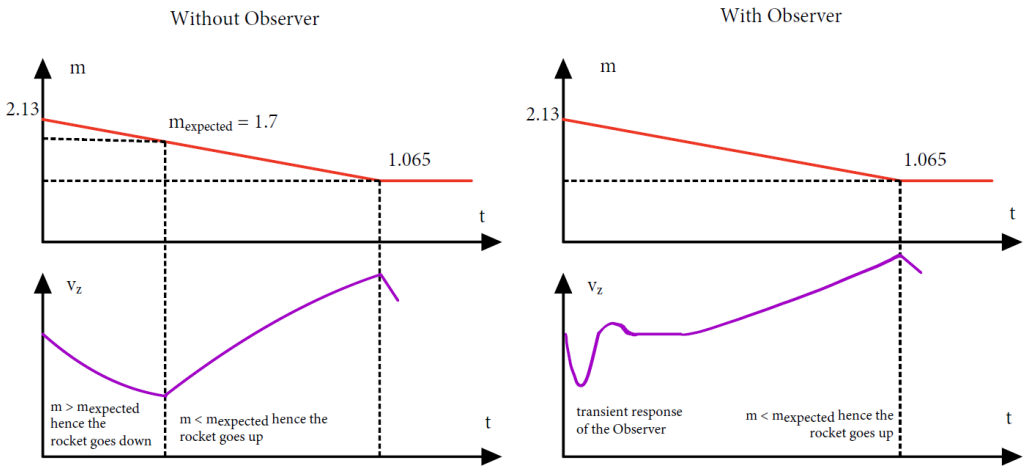


FIGURE 5.7
Qualitative behaviour of the mass and velocity over time considering both the systems with and without the observer. Here is assumed a linear decrease of mass.

CASE WITHOUT OBSERVER

1) In the first seconds of the simulation the estimator is expecting a lighter rocket, which means that the control input calculated by the rocket is lower than the one required for the actual mass. This results in a downward acceleration.

2) There is a moment when the real mass is equal to the expected one and the input from the controller is matching the required one: in this case we have a change in acceleration, which can be observed as a cusp (singularity) in the velocity plot. From this moment, the rocket is lifting, since the real mass is lower than the actual.

The behaviour of the rocket can be observed in Figures 5.8 and 5.9.

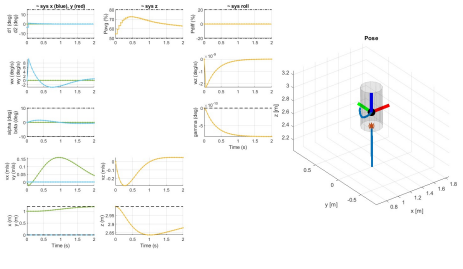


FIGURE 5.8
Plot of the simulation without the observer for a simulation time of 2s to highlight initial behaviour of the system.

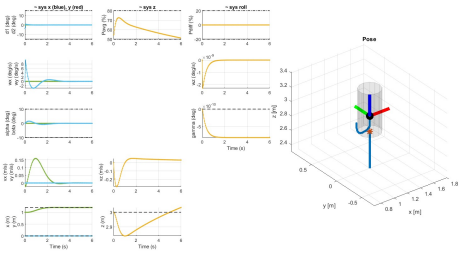


FIGURE 5.9
Plot of the simulation without the observer for a longer simulation time.

CASE WITH OBSERVER

In case the observer is implemented, the real mass is always lower than the actual, which means that the controller is always asking for an higher input than the one required and the rocket reaches an higher steady state than the one desired. This behaviour is observable after the transient response of the observer is finished: the z state is stabilizing on a value higher that the reference.

The behaviour of the rocket can be observed in Figures 5.10 and 5.11.

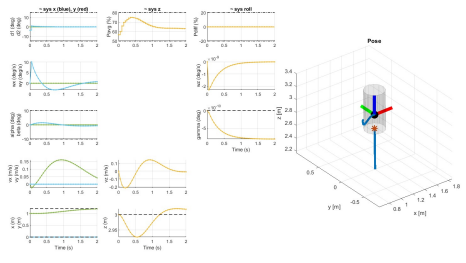


FIGURE 5.10
Plot of the simulation with the observer for a simulation time of 2s to highlight initial behaviour of the system.

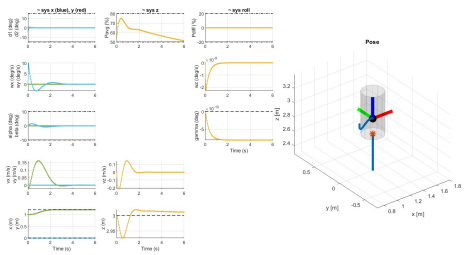


FIGURE 5.11
Plot of the simulation with the observer for a longer simulation time.

5.2.4 QUESTION 4

Bonus question: Simulating for even longer (up to 20 seconds), you can observe up to two more distinct events in the plots of the closed-loop system. What are they, and how can you explain them?

In this section we describe the behaviour of the rocket for a longer time horizon.

CASE WITHOUT OBSERVER

The rocket continues with an upwards acceleration until the fuel is finished. That exact moment is identified by a cusp in the velocity (at around 13s): the acceleration becomes equal to -**g**, where **g** is the gravitational acceleration. The peak of the vertical position is reached afterwards, because of inertia. After this moment the velocity decreases linearly and z quadratically.

The input calculated by the controller is saturated to the lower bound value, because it is not aware that no more fuel is available. Since the rocket is in free fall, at a certain moment the controller asks for a higher input to restore reference, but no fuel is remaining and therefore the input saturates the upper bound.

The behaviour of the rocket can be observed in Figure 5.12.

CASE WITH OBSERVER

Simulating the system for a longer time, the effect of observer cannot be appreciated, because of the high difference between expected and real mass of the rocket. Hence the behaviour of the system with or without the observer is similar.

The behaviour of the rocket can be observed in Figures 5.13.

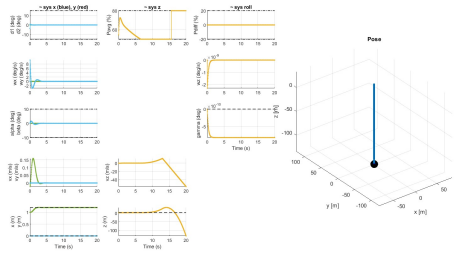


FIGURE 5.12
Plot of the simulation without the observer for a longer simulation time.

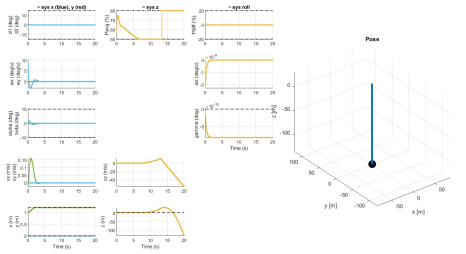


FIGURE 5.13
Plot of the simulation with the observer for a longer simulation time.

CHAPTER 6

NONLINEAR MPC

6.1 DELIVERABLE 6.1: NMPC WITHOUT DELAY

In the case of a nonlinear MPC, we considered the nonlinear model of the system dynamics.

6.1.1 QUESTION 1

Explanation of your design procedure and choice of tuning parameters.

Since a nonlinear dynamics is considered, no more linearization constraints are taken into account: indeed, these constraints were used to respect the approximations made to to model the system around steady state points.

In order to select a reasonable trade off between computational cost and insurance of feasibility, a proper horizon length must be considered. To further reduce it, we have used an approximation of the terminal cost: we used linearization of the system around the trim point. The nonlinear function that expresses the dynamics of the states $\dot{x}$ = f(x,u) is discretized with Runge-Kutta 4 method. This allows us to obtain a linear system, for which we computed the LQR terminal cost and applied it to the nonlinear one. The mathematical formulation of the NMPC problem is similar to MPC and can be solved using CasADi optimizer.

$$J^*(x) = \min\Big(\sum_{i=0}^{N-1}(x_i - x_{\text{ref}})^T Q(x_i - x_{\text{ref}}) + (u_i - u_{\text{ref}})^T R(u_i - u_{\text{ref}}) + (x_N - x_{\text{ref}})^T Q_f(x_N - x_{\text{ref}})\Big) \tag{6.1}$$

$$\text{s.t.} \qquad x_{i+1} = f_{\text{discrete}}(x_i, u_i) \quad \forall i = 0, 1, \ldots, N-1 \tag{6.2}$$

$$Mu_i \leq m \tag{6.3}$$

$$Fx_i \leq f \tag{6.4}$$

where $f_{discrete}$ is the RK4 discretization of f.

Concerning matrix Q tuning, we attributed higher cost on states corresponding to one for which tracking problem is the objective (i.e. x, y, z, γ). This allows a smaller offset from the reference and a better tracking.

6.1.2 QUESTION 2

Discuss the pros and cons of your nonlinear controller vs the linear ones you developed earlier.

While comparing linear MPC and nonlinear MPC, multiple differences can be outlined.

First, a NMPC relies on a nonlinear dynamics, which is closer to the real dynamic behaviour of the rocket, whereas the linear controller considers four decoupled linear versions of the system, which may not be a true representation of the reality. However, in case of a NMPC controller weights are harder to tune due to those coupling effects.

Concerning path trajectory, a more faithful representation of real rocket behaviour results in a better tracking, as can be noticed form the plots below.

On the other hand, the nonlinear controller is very computationally intense, meaning that we can not have a too small sampling time for the discretization of our model, and that we can not have a too large horizon.

Another benefit of the nonlinear controller can be appreciated by comparing the change in dynamics when the roll angle was constrained at 15°and then constrained at 50°. We can see that the LMPC will struggle for large roll angles as it will operates far from the linearization points resulting in more approximations in the model (Figures 6.3 and 6.4).

Considering the case with maximum Roll = 50°, we increased (with respect to controller formulated in Deliverable 4) the slack penalty in order to prevent for extremely high constraint violation, potentially leading to an unpredicted behaviour.

A significant improvement can be noticed thanks to the adoption of the nonlinear model (Figures 6.1 and 6.2). However, in the case with maximum Roll = 50°, a better tuning of the matrices Q and R could reduce the oscillations in the input variables.

6.1.3 QUESTION 3

Plots showing the performance of your controller.

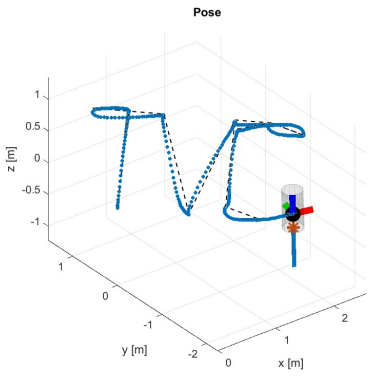


FIGURE 6.1
Plot of the simulation of nonlinear reference considering a linear controller.

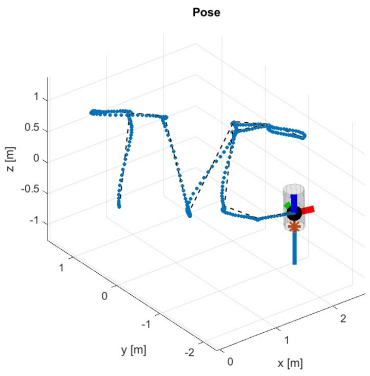


FIGURE 6.2
Plot of the simulation of non linear reference considering a nonlinear controller.

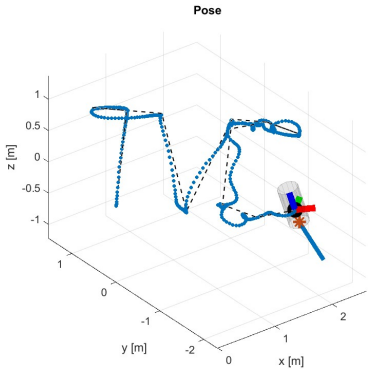


FIGURE 6.3
Plot of the simulation of non-linear reference considering a linear controller with roll angle constrained at 50°.

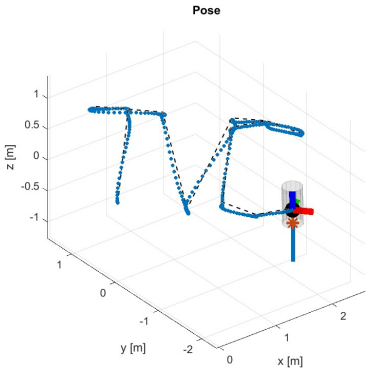


FIGURE 6.4
Plot of the simulation of non linear reference considering a nonlinear controller with roll angle constrained at 50°.

6.2 DELIVERABLE 6.2: NMPC WITH DELAY / DELAY COMPENSATION

6.2.1 QUESTION 1

How much delay is needed to observe a drop in closed-loop performance? How much delay to make the closed-loop system unstable?

For observing a decline in closed-loop performance, a delay of 3 samples is sufficient. However, to reach a point where the closed-loop system becomes unstable, a delay of 4 samples is required if no compensation is applied (Figures 6.5, 6.6) .

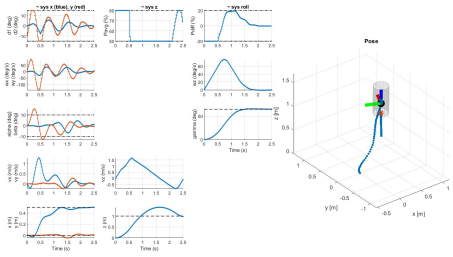


FIGURE 6.5
Performance degradation (compensated delay = 0, actual delay = 3)

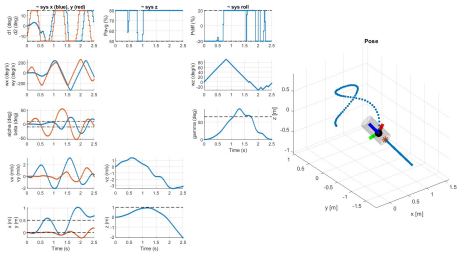


FIGURE 6.6
Instability (compensated delay = 0, actual delay = 4)

6.2.2 QUESTION 2

Plot the closed-loop trajectory for a partially delay-compensating controller (compensated delay < actual delay), and a fully delay-compensating controller (compensated delay = actual delay).

It is notable that as the delay increases, a longer time is needed for the system to settle at the target value. Furthermore, even with a fully compensated controller, the performance of the closed-loop system diminishes as the delay becomes bigger (Figure 6.10).

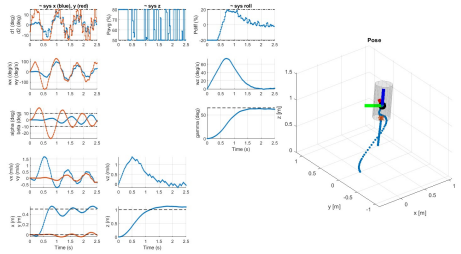


FIGURE 6.7
Partially compensating controller (compensated delay = 4, actual delay = 6)

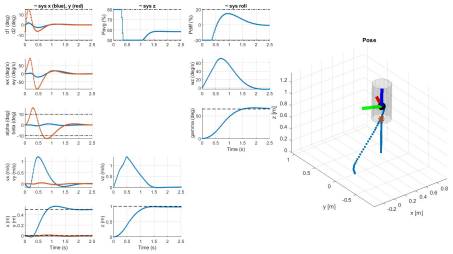


FIGURE 6.8
Fully compensating controller (compensated delay = 6, actual delay = 6)

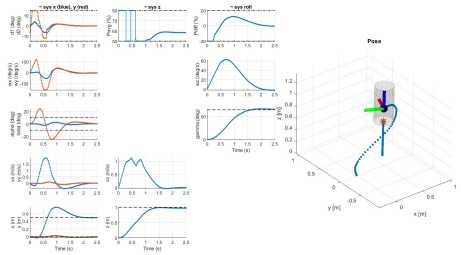


FIGURE 6.9
Fully compensating controller with higher delay (compensated delay = 8, actual delay = 8)

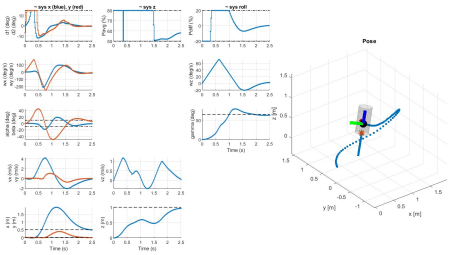


FIGURE 6.10
Fully compensating controller with higher delay (compensated delay = 12, actual delay = 12)

BIBLIOGRAPHY

Jones, Colin (2023). *Model Predictive Control (ME-425)*. Fall semester, École Polytechnique Fédérale de Lausanne (EPFL).

Kwakernaak, Huibert and Raphael Sivan (1972). *Linear Optimal Control Systems*. Library of Congress Cataloging in Publication Data. United States of America: John Wiley & Sons, Inc.