Model Predictive Control
Autumn 2023

Mini Project

Jad Benabdelkader
Alexis Robardet
Davide Lisi

1 Introduction

In this project, we are tasked with designing an MPC controller to control a system. The specific system that we will be controlling is a rocket. The system is described with 12 states with the state vector $\mathbf{x}$ defined as shown below :

$$\mathbf{x} = [\omega^T, \phi^T, \mathbf{v}^T, \mathbf{p}^T] \tag{1}$$

Where ω^T represents the angular velocities around the three body axes, ϕ^T represents the altitude of the rocket's body while $\mathbf{v}^T$ and $\mathbf{p}^T$ represent the volocity and position respectively.

First, we will start by decomposing the whole system into four independent linear sub-systems to then be able to compute a controller for each sub-system. We will then try to merge the four controllers to simulate a non-linear rocket. Finally, we will design one non-linear controller for the whole system without dividing it into sub-systems.

2 Part 2

2.1 Deliverable 2.1 : Linearization

By looking at the model equations presented on the project's data, it appears clear that we can in fact draw clear links between the states and inputs : δ_2 will be linked to the index x of the position vector, δ_1 will be linked to the index y of the position vector, P_{avg} will be linked to the index y of the position vector and finally P_{diff} will be linked to the roll angle γ. These links are in fact crucial as we can use them to decompose the rocket into four different subsystems that do not depend on each other. The first subsystem that we will create, sys_x, will only depend on ω_y, β, v_x and x. Then, sys_y will depend on ω_x, α, v_y and y. The third subsystem will be sys_z which will depend on v_z and z. The last subsystem is sys_{roll} which will depend on ω_z and γ. As we can see from the decomposition, each subsystem is indeed independent. This is in fact pretty intuitive as each input only determines specific states, meaning that the overall system can be seen as the superposition of four different sub-systems which then allows us to consider each sub-system individually instead of considering the whole system at once.

3 Part 3

3.1 Deliverable 3.1 : Design of four MPC controllers

For each subsystem, we will design an MPC controller. To design each controller, we will solve an optimization problem of the following form :

$$\begin{aligned} \min_{x_i \in \mathbb{R}^n, u_i \in \mathbb{R}^p} \quad & \sum_{i=0}^{N-1} l(x_i, u_i) + x_N^T Q_f x_N \\ \text{s.t} \quad & x_{i+1} = Ax_i + Bu_i \\ & Fx_i \leq f \\ & Mu_i \leq m \\ & x_N \in X_f \end{aligned} \tag{2}$$

For the cost function $l(x_i, u_i)$, we will use the cost function of an LQR controller which has the following form :

$$l(x_i, u_i) = x_i^T Q x_i + u_i^T R u_i \tag{3}$$

The parameters R and Q will have to be manually tuned for each controller to ensure reliable results as well as fast performances as each controller will have to stabilize in a relatively fast fashion.

As for the constraints, they will be different for each of the four controllers. The state constraints are modeled in (2) by the inequations $Fx_i \leq f$ while the input constraints are modeled by the inequations $Mu_i \leq m$.

3.1.1 Constraints for the X controller

For the X controller, we want to ensure that δ_2 always stays between -0.26 and 0.26, which means that we will have $M = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ and $m = \begin{bmatrix} 0.26 \\ 0.26 \end{bmatrix}$. As for the state constraints relative to this controller, we want to keep β between -0.1745 and 0.1745 radians at all times. This constraint can be expressed by the matrices $F = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}$ and $f = \begin{bmatrix} 0.1745 \\ 0.1745 \end{bmatrix}$

3.1.2 Constraints for the Y controller

For the Y controller, the constraint matrices are defined in exactly the same way as for the X controller as the constraints that we want to impose on the Y controller are symmetrical to the ones that we are imposing on the X controller.

3.1.3 Constraints for the Z controller

For the Z controller, we want to ensure that P_{avg} always stays between 50 and 80 percent, which means that we will have $M = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ and $m = \begin{bmatrix} 23.33 \\ 6.66 \end{bmatrix}$. There are no state constraints relative to this controller.

3.1.4 Constraints for the roll controller

For the roll controller, we want to ensure that P_{diff} always stays between -20 and 20 percent, which means that we will have $M = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ and $m = \begin{bmatrix} 20 \\ 20 \end{bmatrix}$. There are no state constraints relative to this controller.

3.1.5 Choice of parameters

The most important parameters that will have to be chosen for each controller are the Q and R matrices. These two matrices have to be carefully tuned as they represent a trade-off. High values for Q will lead to more agressive tracking, which favorizes tracking accuracy while a higher R favorizes smoother control actions and enlarges the terminal set. To design these parameters, we started by setting them equal to the identity and then progressively modifying them to improve performance. This trial-and-error process is time-consuming but ensures satisfying results. As for the Horizon length, we decided to set it equal to 5 seconds. A value that is high enough to provide good predicting power but low enough to avoid making the computational cost too large.

For the X controller, we have prioritized good tracking performance, leaving R unchanged at $R = 1$. The trial-and-error process led us to set $Q = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$.

For the Y controller, the fact that the constraints inequations are the same as the X controller led us to set the Q and R parameters in exactly the same way.

For the Z controller, we want the controller to be relatively agressive and so we focus on tuning Q and leave $R = 1$. Through trial-and-error, we obtain $Q = \begin{bmatrix} 1 & 0 \\ 0 & 30 \end{bmatrix}$.

Finally, for the roll controller, we would like to make it that P_{diff} uses its full range and we want to have fast and accurate tracking. We have thus reduced R to $R = 0.01$ and we have set $Q = \begin{bmatrix} 1 & 0 \\ 0 & 15 \end{bmatrix}$.

These choices of parameters will ensure that our controllers stabilize the system at the origin while providing recursive satisfaction of the input and angle constraints with fast settling times.

3.1.6 Terminal Sets

The terminal set is the set in which the state of our system will be at Horizon time. Its main property is that it is an invariant set. In fact, it is the maximum invariant set of our controller. For the X controller, here are the three projections of the terminal set :

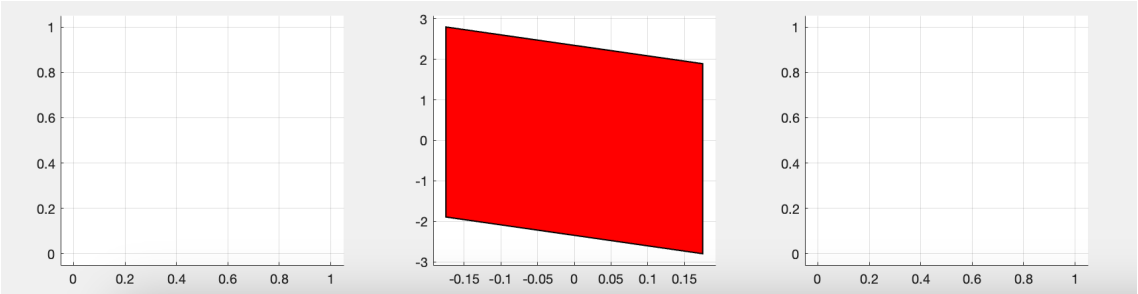

Figure 1: Projections of the X controller's terminal set

Now, for the Y controller, we can see that the three projections of the terminal set are very similar to those for the X controller :

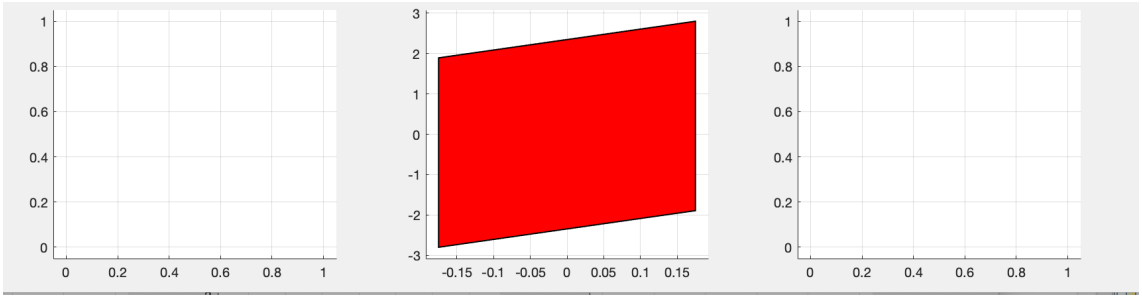

Figure 2: Projections of the Y controller's terminal set

For the Z controller, we find the following terminal set :

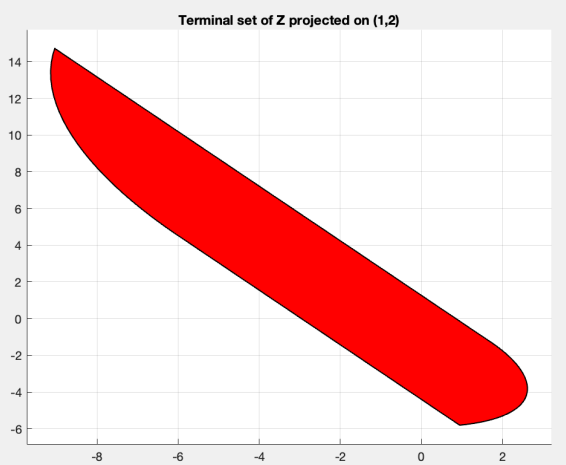

Figure 3: Z controller's terminal set

And finally, we find the following terminal set for the roll controller :

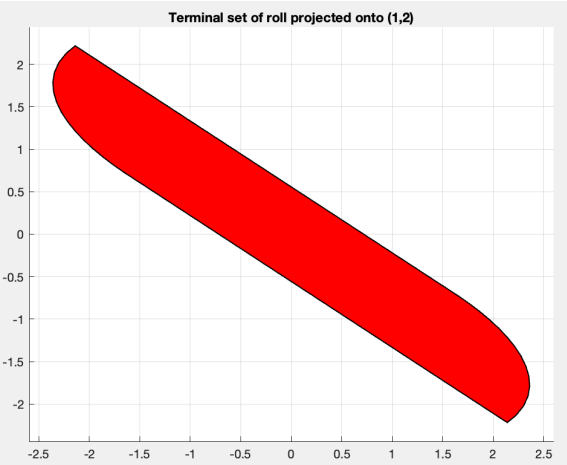

Figure 4: Roll controller's terminal set

3.1.7 Open-Loop Simulation

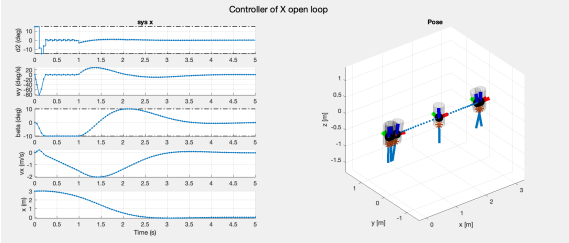

Figure 5: X Open-Loop Simulation

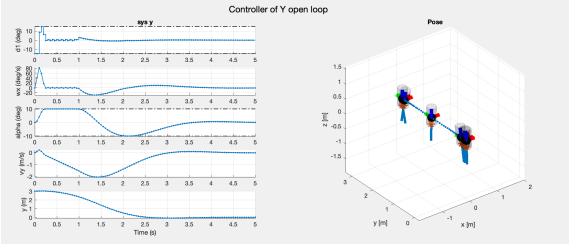

Figure 6: Y Open-Loop Simulation

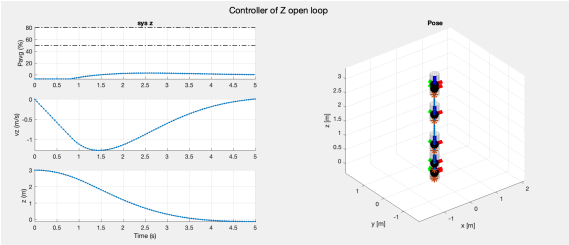

Figure 7: Z Open-Loop Simulation

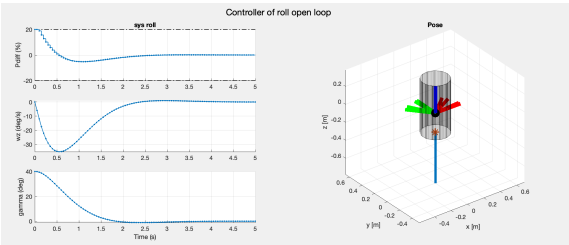

Figure 8: Roll Open-Loop Simulation

As we can see from the open-loop simulation plots, the controllers do stabilize each sub-system at the origin with settling times that are significantly inferior to 7 seconds when starting stationary at 3 meters from the origin (for x, y and z) or stationary at 40 degrees for roll. We will now perform closed-loop simulation to confirm the preliminary results obtained with open-loop simulation.

3.1.8 Closed-Loop Simulation

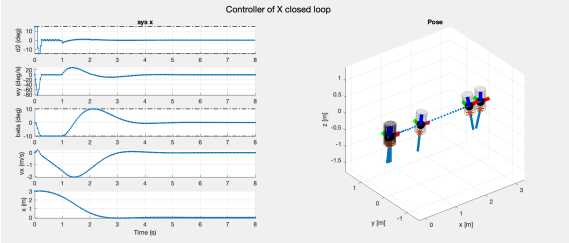

Figure 9: X Closed-Loop Simulation

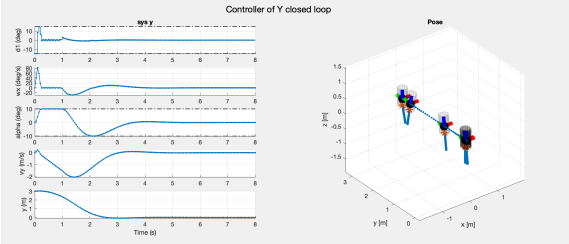

Figure 10: Y Closed-Loop Simulation

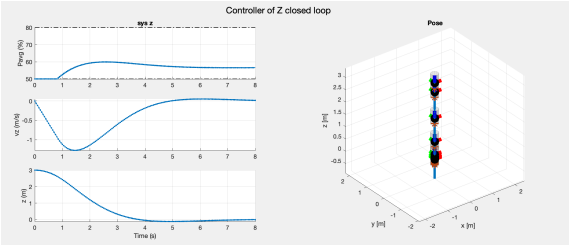

Figure 11: Z Closed-Loop Simulation

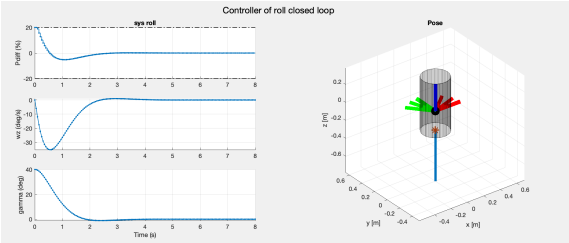

Figure 12: Roll Closed-Loop Simulation

We can see that the closed-loop simulation plots do in fact confirm the results that we have obtained through open-loop simulation. This allows us to validate the design of our controllers.

3.2 Deliverable 3.2 : Design of MPC Tracking Controllers

Our goal is now to extend each of our controllers so that they can track a given constant reference. This task is actually quite simple as it does not require for us to modify the optimization problem that has to be solved. The only thing that needs to be done is to implement the delta formulation for our variables.

$$\begin{aligned} \Delta x &= x - x_s \\ \Delta u &= u - u_s \end{aligned} \tag{4}$$

The parameters that we will use for simulation are tuned in exactly the same way as in Deliverable 3.1 as we have found that using the same parameters was enough to have robust performances.

3.2.1 Open-Loop Tracking Simulation

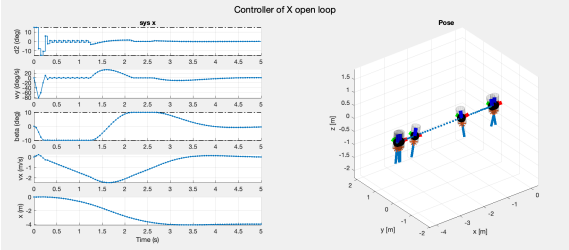

Figure 13: X Open-Loop Simulation in Tracking

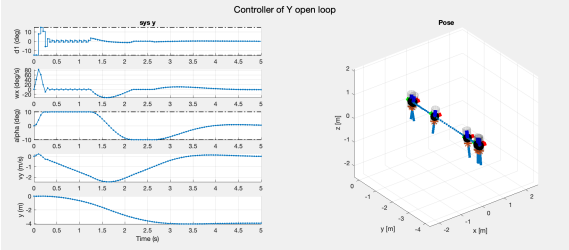

Figure 14: Y Open-Loop Simulation in Tracking

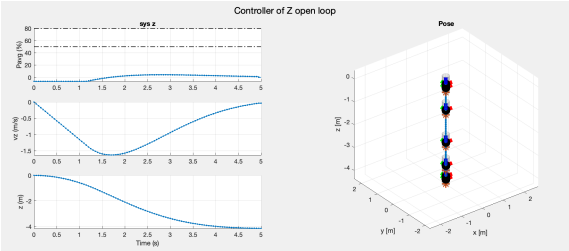

Figure 15: Z Open-Loop Simulation in Tracking

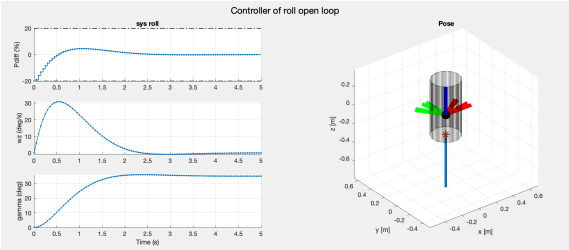

Figure 16: Roll Open-Loop Simulation in Tracking

We can see that the open-loop tracking performance obtained by each one of the controllers is very good : The given reference is tracked correctly starting at the origin and tracking a reference to -4 meters (for x, y and z) and to 35 degrees for roll. We can also see that the settling times are generally fast as the controllers always need less than 5 seconds to settle. We will now perform closed-loop tracking simulation to confirm these results and validate our tracking MPC controllers.

3.2.2 Closed-Loop Tracking Simulation

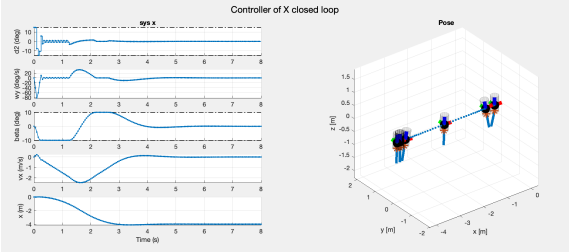

Figure 17: X Closed-Loop Simulation in Tracking

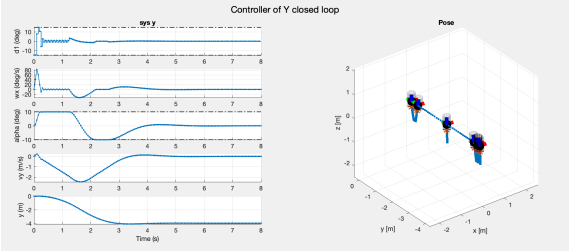

Figure 18: Y Closed-Loop Simulation in Tracking

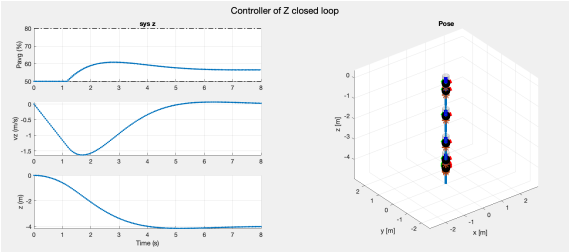

Figure 19: Z Closed-Loop Simulation in Tracking

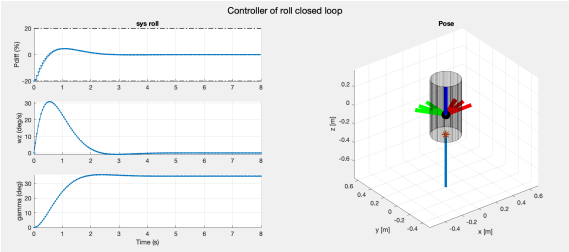

Figure 20: Roll Closed-Loop Simulation in Tracking

We can see that the closed-loop tracking simulation plots do in fact confirm the results that we have obtained through open-loop tracking simulation. This allows us to validate the design of our controllers.

4 Part 4

4.1 Deliverable 4.1 : Simulation with Nonlinear Rocket

We will now use our controllers to have the nonlinear rocket following a given path. To do so, the four sub-system controllers will in fact be merged into one full-system controller. The path that the rocket will follow will be the outline of the letters "TVC" meaning Thrust Vector Control.

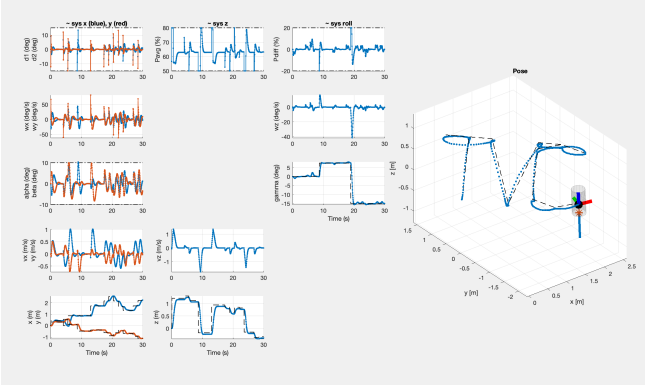

Figure 21: Merged linear MPC controller in nonlinear simulation

To perform this task, we have modified the tuning that we previously had for Part 3. In particular, the cost matrices Q and R were significantly modified through trial-and-error as the original cost matrices yielded performances that were far from ideal for the task at hand.

4.1.1 Choice of parameters

For the Horizon length N, we decided to set it equal to 5 seconds, which is the same value that we had before.

For the X controller, we have prioritized good tracking performance as before but we have also decided to make the controller action much smoother. To do so, we have increased R up to $R = 150$. As for Q, we have made the tracking on x much more agressive by increasing its weight. The trial-and-error process led us to set $Q = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix}$.

As for the Y controller, it is tuned in exactly the same way as the X controller.

For the Z controller, we want the controller to be even more agressive on the tracking for both v_z and z than it was before which leads us to increase the weights in Q. We also decided to significantly reduce R so that the inputs are less constrained : we set $R = 0.01$. so we focus on tuning Q and leave $R = 1$. Through trial-and-error, we obtain $Q = \begin{bmatrix} 10 & 0 \\ 0 & 200 \end{bmatrix}$.

Finally, for the roll controller, we would like to make it so that P_{diff} uses its full range (same as before) and we want to have even more agressive tracking on γ to more accurately track it. We have thus not modified R ($R = 0.01$) and we have set $Q = \begin{bmatrix} 1 & 0 \\ 0 & 600 \end{bmatrix}$.

This choice of parameters allows us to track the "TVC" outline much more accurately than with the previous parameters, although the tracking is still far from perfect because of the model mismatch between the linear controllers and the nonlinear rocket simulation.

5 Part 5

5.1 Deliverable 5.1 : Offset-Free Tracking

We now assume that the mass of the system is significantly different from what we have modeled, so we extend the z-controller to compensate. The dynamics of the Z system is now :

$$x^+ = Ax_i + Bu_i + Bd \tag{5}$$

The main difference with the previous dynamics is the introduction of d which is a constant and unknown disturbance which our Z controller will now have to reject. We will add an estimator L to the Z-controller to allow us to reject the disturbance. We can directly write the error dynamics of the augmented system :

$$\begin{bmatrix} x^+ - \hat{x} \\ d^+ - \hat{d} \end{bmatrix} = \left(\begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} + \begin{bmatrix} L_x \\ L_d \end{bmatrix} \begin{bmatrix} C & C_d \end{bmatrix} \right) \begin{bmatrix} x_k - \hat{x_k} \\ d_k - \hat{d_k} \end{bmatrix} \tag{6}$$

The estimator L is obtained by setting the poles at 0.1, 0.2 and 0.3 (poles in the unit-circle obtained through trial-and-error). The rest of the parameters are set in the same way as Part 4. Here are the results obtained with and without estimator by simulating for 8 seconds from $x_0 = (zeros(1, 9), 103)^T$ to $ref = (1.2, 0, 3, 0)^T$ with a 25 percent higher initial mass.

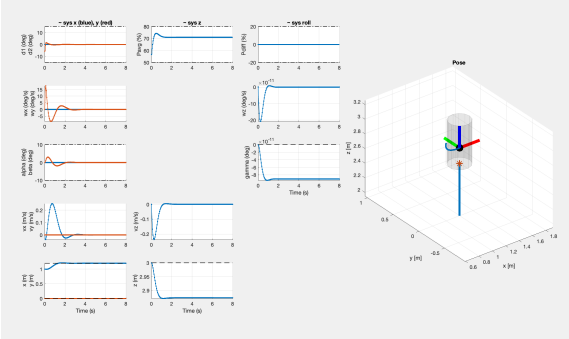

Figure 22: Tracking without Estimator

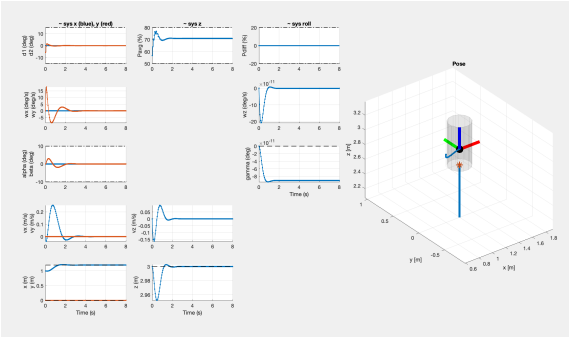

Figure 23: Offset-free Tracking with Estimator

As we can see from the plots, implementing an estimator allows the controller to accurately track the reference given on z, which is not tracked correctly without an estimator as a significant offset is introduced by the mass change and is not corrected.

5.2 Deliverable 5.2 : System with changing mass

In this section, we will simulate the system while taking into account the fact that its mass is decreasing at a constant rate due to fuel consumption. We will use the exact same controller as in Part 5.1. Here is a plot showing the impact of having a thrust-dependent mass decrease during flight for a 12 seconds simulation :

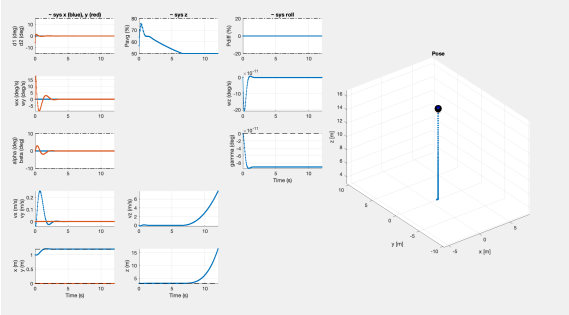

Figure 24: Tracking with fuel loss at constant rate

We can notice that there is a tracking error in height during the first few seconds despite the estimator, this is probably due to the fact that there is a certain lag in the estimator's response to the mass variation. Implementing a Kalman filter could help in improving the accuracy of state estimation and reduce tracking errors in this changing mass scenario. First, we can see that the rocket starts by agressively tracking the x and y references as its initial position is a little bit offset compared to the reference point which must be tracked. It will also correct the very small offset in height during that first phase. In a second phase, the rocket will simply maintain its position for a few seconds (The reference point is reached and the system stabilizes, this is the intended behaviour). Towards the end of the simulation, the rocket will unexpectedly raise its height significantly, which is in total contradiction with the intended behaviour. This is in fact due to the constraint which states that P_{avg} must always stay above 50 percent. The rocket is in a situation where, if it wants to maintain its position at the given reference point, P_{avg} has to drop below 50 percent which is not allowed by the constraints. As such, the only possibility to continue to satisfy the constraints is to increase the height which then means that the reference point is not tracked properly in z despite the estimator.

6 Part 6

6.1 Deliverable 6.1 : NMPC controller

In this section, we will design an NMPC controller which takes the full state as input and provides four input commands. The approach that we will use still relies on solving an optimization problem :

$$\begin{aligned} \min_{x_i \in \mathbb{R}^n, u_i \in \mathbb{R}^p} \quad & \sum_{i=0}^{N-1} l(x_i, u_i) + (x_N - x_{ref})^T Q (x_N - x_{ref}) \\ \text{s.t} \quad & x_{i+1} = f_{dis}(x_i, u_i) \\ & M u_i \leq m \\ & x_0 = X_f \end{aligned} \tag{7}$$

The cost function $l(x_i, u_i)$ now has the following expression :

$$l(x_i, u_i) = (x_i - x_{ref})^T Q (x_i - x_{ref}) + u_i^T R u_i \tag{8}$$

As for $f_{dis}(x_i, u_i)$, it is obtained through Runge-Kutta 4 discretization which has the following form :

$$\begin{aligned} k_1 &= f(x_i, u_i) \\ k_2 &= f(x_i + \frac{T_s}{2} k_1, u_i \\ k_3 &= f(x_i + \frac{T_s}{2} k_2, u_i \\ k_4 &= f(x_i + T_s k_3, u_i \\ f_{dis}(x_i, u_i) &= x_i + \frac{T_s}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \tag{9}$$

As for the constraints, they differ from the ones that we have seen before. The constraints on the inputs are the same as before but the constraints on the states are significantly relaxed as the controller can now cover the entire space. The only state constraint is on the angle β which must always stay between -75 and 75 degrees. All other state constraints that we had before are entirely dropped. As for the tuning of our controller, the horizon length has been left entirely unchanged while the matrices Q and R have to be radically changed (there is now only one Q and one R instead of four of each).

To achieve this tuning, we have used the same trial-and-error method that we used previously while still relying on the knowledge of the system that we built before, especially when it comes to how each parameter acts on the system and how they can have effects on each others.

For the matrix Q, we have $Q = diag([30; 30; 30; 10; 10; 10; 10; 10; 40; 75; 75; 100])$. The tracking weights on x, y and z are set particularly high as well as the weights on w_x, w_y and w_z. These high weights ensure good tracking performance when drawing the "TVC" outline. As for R, we have set it as $R = I_4 * 0.001$. The weights are very small as we do not want to penalize the inputs very much in this case.

6.1.1 Simulation for two different max roll angle values

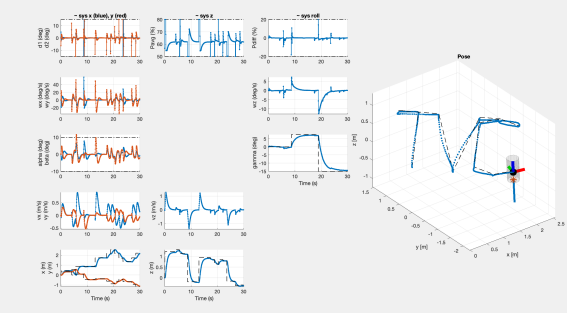

Figure 25: NMPC Controller Simulation for max roll = 15 degrees

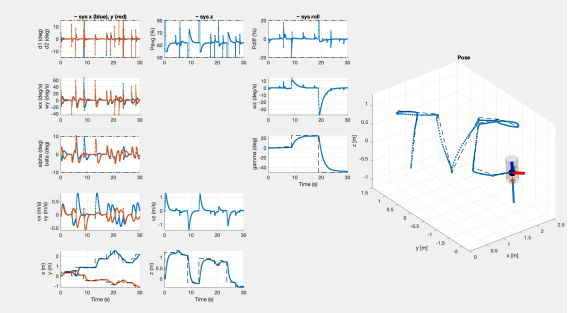

Figure 26: NMPC Controller Simulation for max roll = 50 degrees

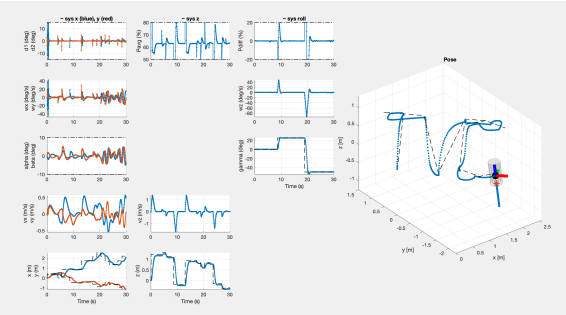

Figure 27: Merged linear MPC Controller Simulation for max roll = 50 degrees

We can see on the three previous figures that the tracking obtained with the NMPC controller is much more accurate than the one obtained with the merged linear controller. Indeed, the merged linear controller (which had to be slightly retuned for this section as the previous tuning led to infeasibility because of the much higher max roll angle reference) struggles a lot when following the outline, especially with the straight lines. This can be explained by many reasons. The first one is that the NMPC controller can cover the whole state space as it can directly handle nonlinearities, which makes it more efficient (more accurate predictions) and also allows it to track a given reference from a much wider set of initial positions. As we have seen previously, using the NMPC controller allows us to significantly relax the state constraints, while the linear MPC controller has to use additional constraints due to linearization. This makes the NMPC controller overall much more adaptable than the merged controller.

Though, one has to note that the NMPC controller is not perfect by any means. Indeed, its computational cost is much higher than the merged linear controller (The time needed for the program to run is approximately 2.5 to 3 times higher.) Its implementation is also much more complex and hard to properly implement. For relatively simple nonlinear systems where the nonlinearities are not too glaring and where the references to be tracked are not too far away from the initial position, it may be sufficient to just use a merged linear controller for the sake of simplicity.