MODEL PREDICTIVE CONTROL

Mini-Project- Rocket Thrust Control

ME-425

PROFESSOR: COLIN JONES

Authors:	*Scipper :*
Julia Dessons	282837
Louis Durand	288044
Christopher Stocker	266575

December 10, 2024

Contents

1 Introduction

This project has for objective to design a Model Predictive Control (MPC) controller for a rocket. First some linear MPCs regulators and trackers will be designed to control 4 independent linear subsystems of the rocket. An observer will also be added to better control the system when an offset such as a change in weight is added. Finally, a non-linear MPC will be designed and it's simulation results will be compared to those of the combined linear MPCs previously developed.

2 Linearization

2.1 Deliverable 2.1

Controlling a non-linear system is known to be a challenging task. Controlling linearized versions of a system can thus be a good alternative. The rocket was therefore linearized around a steady-state [xs, us] computed such that $\dot{x}$ = $f(x_s, u_s)$ = 0. Therefore the linear system can then be expressed as

$$\dot{x} = Ax(t) + Bu(t) \tag{1}$$

where x is a 12x1 vector, with x=[ω^T, ϕ^T, v^T, p^T] in which ω=[ω_x, ω_y, ω_z]^T are the angular velocities, ϕ=[α, β, γ]^T are the Euler angles, v=[v_x, v_y, v_z]^T are the velocities, and p=[x, y, z]^T are the positions. u is the input vector of the sytem, u=[d_1, d_2, P_{avg}, P_{diff}]^T, it's first two terms representing the deflection angles, and it's two last the average throttle and throttle difference between the two rocket's motors.

The resulting A, B, C and D matrices obtained after the linearization are the following:

A =

	wx	wy	wz	alpha	beta	gamma	vx	vy	vz	x	y	z
wx	0	0	0	0	0	0	0	0	0	0	0	0
wy	0	0	0	0	0	0	0	0	0	0	0	0
wz	0	0	0	0	0	0	0	0	0	0	0	0
alpha	1	0	0	0	0	0	0	0	0	0	0	0
beta	0	1	0	0	0	0	0	0	0	0	0	0
gamma	0	0	1	0	0	0	0	0	0	0	0	0
vx	0	0	0	0	9.81	0	0	0	0	0	0	0
vy	0	0	0	-9.81	0	0	0	0	0	0	0	0
vz	0	0	0	0	0	0	0	0	0	0	0	0
x	0	0	0	0	0	0	1	0	0	0	0	0
y	0	0	0	0	0	0	0	1	0	0	0	0
z	0	0	0	0	0	0	0	0	1	0	0	0

Figure 1: Matrix A

C =

	wx	wy	wz	alpha	beta	gamma	vx	vy	vz	x	y	z
wx	1	0	0	0	0	0	0	0	0	0	0	0
wy	0	1	0	0	0	0	0	0	0	0	0	0
wz	0	0	1	0	0	0	0	0	0	0	0	0
alpha	0	0	0	1	0	0	0	0	0	0	0	0
beta	0	0	0	0	1	0	0	0	0	0	0	0
gamma	0	0	0	0	0	1	0	0	0	0	0	0
vx	0	0	0	0	0	0	1	0	0	0	0	0
vy	0	0	0	0	0	0	0	1	0	0	0	0
vz	0	0	0	0	0	0	0	0	1	0	0	0
x	0	0	0	0	0	0	0	0	0	1	0	0
y	0	0	0	0	0	0	0	0	0	0	1	0
z	0	0	0	0	0	0	0	0	0	0	0	1

Figure 2: Matrix C

B =

	d1	d2	Pavg	Pdiff
wx	-55.68	0	0	0
wy	0	-55.68	0	0
wz	0	0	0	-0.104
alpha	0	0	0	0
beta	0	0	0	0
gamma	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

Figure 3: Matrix B

D =

	d1	d2	Pavg	Pdiff
wx	0	0	0	0
wy	0	0	0	0
wz	0	0	0	0
alpha	0	0	0	0
beta	0	0	0	0
gamma	0	0	0	0
vx	0	0	0	0
vy	0	0	0	0
vz	0	0	0	0
x	0	0	0	0
y	0	0	0	0
z	0	0	0	0

Figure 4: Matrix D

By studying these four matrices it can be deduced that this linearized system can be seperated in four independant systems. Indeed each parameter only interacts with one other parameter at most in each matrix. For example, from matrix B we get that d_1 interacts with w_x and from matrix A we get that w_x interacts with α which interacts with v_y which interacts with y which does not interact with anything else. Thus one system will have d_1 as input and y as output with w_x, α, v_y and y as states. From an intuitive mechanical perspective this separation is logical as it represents an acceleration in four independent frameworks.

3 Design MPC Controllers for Each Sub-System

After splitting the system in four linear sub-systems, MPC controllers are designed for each of them. First, said MPC regulators will be implemented. Second, these regulators will be modified into tracking controllers.

3.1 MPC Regulators

The objective of deliverable 3.1 was to design MPC regulators for each linear sub-systems. To design such a regulator we formulate the following optimization problem considering an infinite horizon:

$$V^*(x_0) = min \sum_{i=0}^{\infty} x_i^T Q x_i + u_i^T R u_i$$
$$s.t. \quad x_{i+1} = Ax_i + Bu_i$$

where x_i is the state and u_i the input of the system at time i and where Q is semi-positive definite and R is positive definite. An infinite horizon LQR cannot be solved thus, using an MPC, the problem can be truncated after a finite horizon N such that the new optimization problem to solve will be presented as follows:

$$V_N^*(x_0) = min \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i + x_N^T P x_N$$
$$s.t. \quad x_{i+1} = Ax_i + Bu_i$$
$$x_N \in X_f$$

Using an MPC however does not guarantee feasibility nor stability contrarily to the LQR form with infinite horizon. This is why a terminal cost $x_N^T P x_N$ and a terminal set $x_n \in X_f$ are added in this form, where X_f is the feasible set for which the MPC problem with an N horizon is feasible and for which P and X_f are chosen to simulate an infinite horizon. To guarantee the stability the following conditions have to be met:

1. The stage cost is a positive definite function
2. The terminal set is invariant under the local control law $\kappa_f(x)$ and all state and input constraints are satisfied in X_f
3. The terminal set is a continuous Lyupanov function in X_f

Four MPCs were thus designed. Several constraints were added to follow the rocket's limitations. Notably, δ_1 and δ_2 were constrained with $\pm$ 15° (0.26 rad), P_{avg} is in the range [50%, 80%], P_{diff} is

in the range $\pm$ 20% and α and β are inferior to $\pm$ 5° (0.0873 rad). Then, the system was linearized and trimmed around a point x_s such that, $\dot{x} \cong A(x - x_s) + B(u - u_s)$ while the MPC is designed for the system $\dot{x} = Ax + Bu$. Thus when the controller applies an input u*, the real input is u= u* + u_s. In this case, u_s is equal to zero for all inputs except for the steady state P_{avg} which is around 56.6%. Thus the constraint on P_{avg}* will be [-6.6%, 23.3%] so that the steady state P_{avg} + P_{avg}* resepct the P_{avg} constraints.

Once the constraints in place, it was time to choose values for parameters Q the state costs, R the input costs and H the horizon. Here, we deal with the trade-off of computation time vs performance with choosing H and the trade-off of conservative vs aggressive control with Q and R. Also, the choice of Q and R directly influence the terminal cost and constraint which may affect the feasibility of the problem.

For the simple task of regulation, the choice of Q and R are not so crucial. In fact, many values of Q and R allowed to converge to the trimming point within the eight seconds of simulation. Initially we chose rather generic values for Q and R such as Q = I and R = 10 for each subsystem MPC. Then as we moved on to parts 3.2 and 4.1 it became apparent that proper tuning should be made. Hence, we adopted the following tuning strategy. Starting from a conservative controller, we choose high input costs and low state costs. This gives rather poor performance. We then move progressively towards a more aggressive controller by reducing the input costs and increasing state costs. Such a controller will look to employ the maximum of its input to follow the trajectory given. Then, further improvements can be achieved by working with the state cost matrix Q and applying specific costs to each variable of the state vector. Indeed, when following a trajectory, only the position variables x,y,z,roll are given a tracking reference. It thus makes sense to apply the largest costs to these specific variables of the state vectors. This process lead us to the following tuning parameters.

$$Qx = Qy = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 15 \end{bmatrix} \qquad Qz = \begin{bmatrix} 1 & 0 \\ 0 & 15 \end{bmatrix} \qquad Qroll = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}$$

$$Rx = Ry = Rz = Rroll = 0.08$$

Finally, H was taken as 15 by gradually increasing it until the performance was no longer improved. These parameters were thus implemented throughout the whole code (parts 3, 4 and 5). These parameters are indeed also used for calculating the invariant sets below.

The terminal invariant sets were computed in matlab following the algorithm seen in class and in the labs. Thanks to the LQR linear control law $u = k \cdot x$, we can express the close loop system completely in function of x such that $A_{cl} = A + k \cdot B$. This allows to compute the pre-set of any set Ω as: $pre(\Omega) := \{x \mid Acl \cdot x \in \Omega\}$. Then, starting from the constrained set on states and inputs, we are able to calculate the maximum control invariant set C_∞ by iteratively calculating the pre-set and its intersection with the current set $pre(\Omega_i) \cap \Omega_i$. This iterative process stops once we the difference between the previous and current maximum invariant set is below a certain tolerance. All this is done before solving the optimization problem.

Below, the maximum control invariant sets are plotted for each of the state vectors x, y, z, γ. Note that the state constraints on α and β (due to the linear range around the trim point) are indeed active in the terminal sets of x and y.

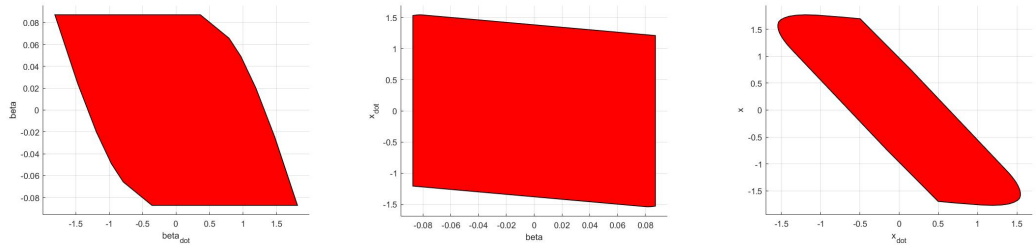


Figure 5: Projections of the 4 dimensional terminal set of x

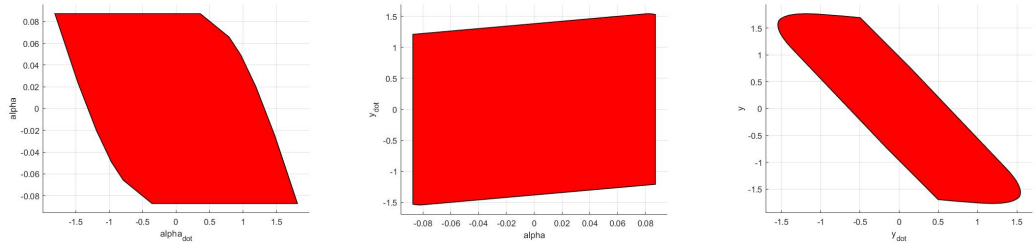


Figure 6: Projections of the 4 dimensional terminal set of y

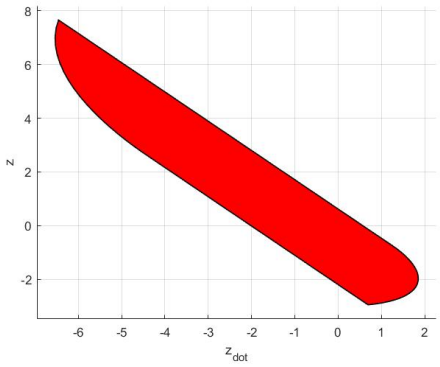


Figure 7: Projection of the 2 dimensional terminal set of z

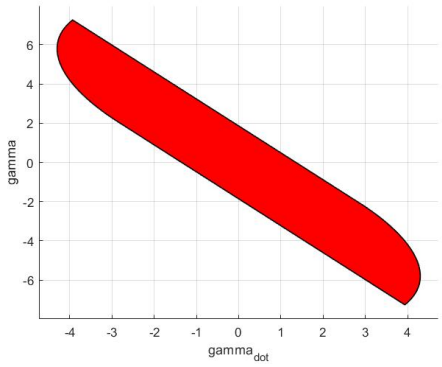
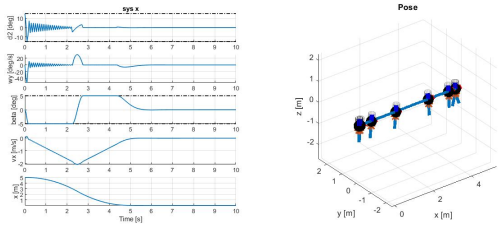
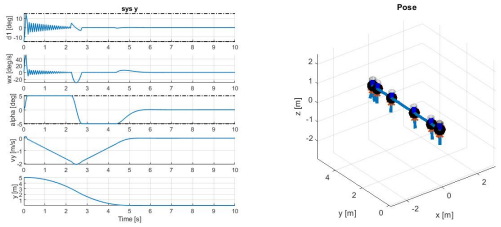


Figure 8: Projection of the 2 dimensional terminal set of gamma (roll)

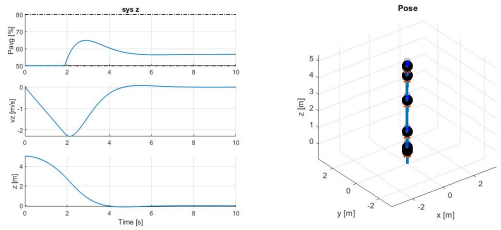
Once the control invariant sets calculated for each of the subsystems, the linear regulators were tested for an initial offset of 5 meters in the x, y and z direction as well as a 45 degree offset in roll. The rocket is able to converge quickly to 0 (well before 8 seconds) while respecting the constraints for each of these cases. However, it can be observed that the z and roll subsystems suffer slightly from overshooting due to high state costs Q and low input costs R which result in rather dynamic system responses. One possible way of eliminating overshoot can be to penalize large speeds.



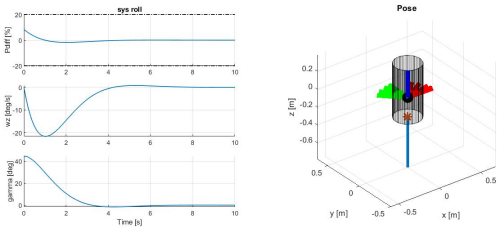
(a) x for regulating to 0



(b) y for regulating to 0



(c) z for regulating to 0



(d) roll for regulating to 0 degrees

Figure 9: Regulating trajectories for each of the linearized subsystems

3.2 Deliverable 3.2

Now we want to generalize our controller to be able to track a point in 3D space. To do so, we set a reference, which is the target point the rocket will aim for. First we define a steady state x_s with our reference output r as follows:

$$x_s = Ax_s + Bu_s$$

$$r = Cx_s$$

The steady state (x_s , u_s) will be computed using the following minimizing problem:

$$\min u_s^T R_s u_s$$

$$s.t. \quad \begin{pmatrix} I - A & -B \\ C & 0 \end{pmatrix} \begin{pmatrix} x_s \\ u_s \end{pmatrix} = \begin{pmatrix} 0 \\ r \end{pmatrix}$$

If no solution exists the closest reachable r will be computed as follows:

$$\min(Cx_s - r)^T Q_s (Cx_s - r)$$

$$s.t. \quad x_s = Ax_s + Bu_s$$

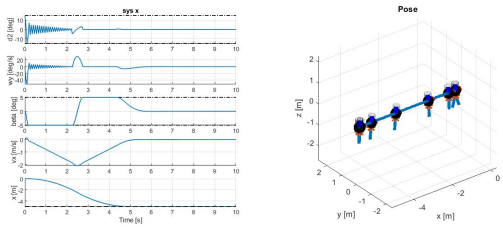
To obtain the delta formulation for the tracking, we compute the difference between the actual and desired state. The same is done with the input:

$$\Delta x = x - x_s$$

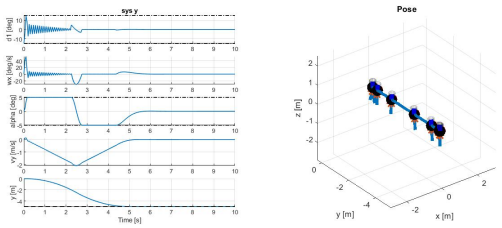
$$\Delta u = u - u_s$$

The four controllers are thus adapted from deliverable 3.1 to properly do reference tracking following the theory presented above. The tuning parameters are the same as in deliverable 3.1, because the general goal stays the same. Figure 10 shows the results of the simulation in terms of angles, thrust, linear velocity and position. As required, the simulation uses as reference -5m for x,y,z and 45° for the roll angle.

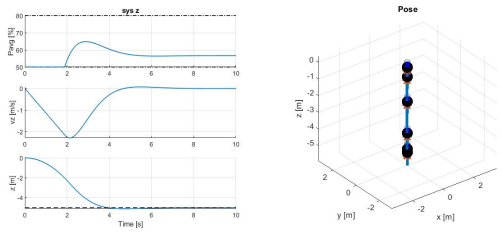
The reference objectives are all reached within eight seconds, all the while respecting input and state constraints. Again, it can be observed that both z and roll directions suffer from overshoot.



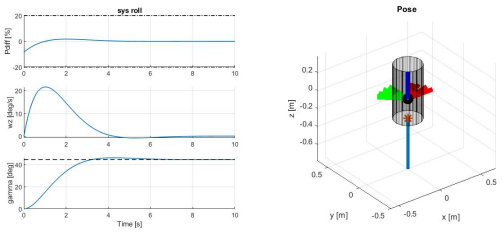
(a) x for tracking reference to -5



(b) y for tracking reference to -5



(c) z for tracking reference to -5



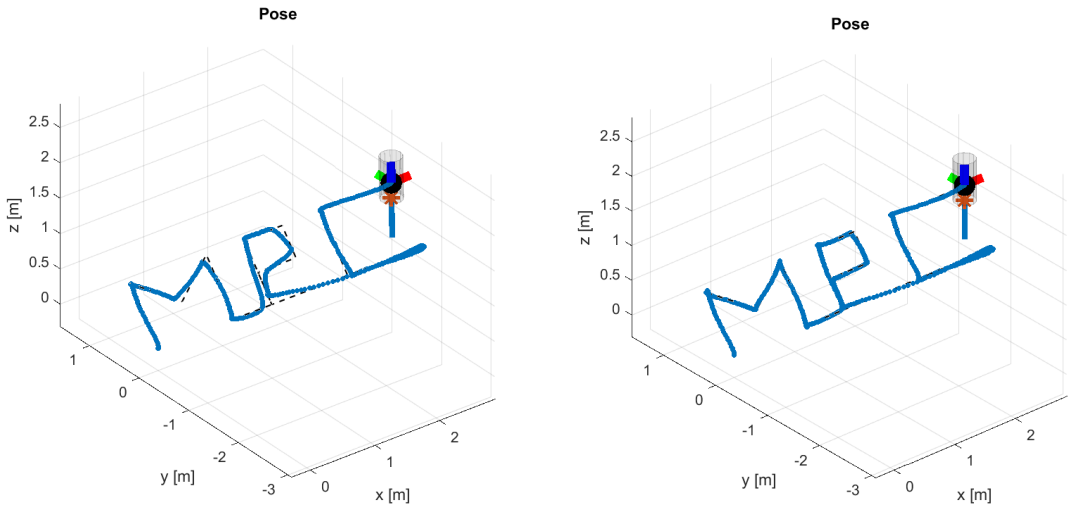
(d) roll for tracking reference to 45 degrees

Figure 10: Tracking trajectories for each of the linearized subsystems

4 Simulation with Nonlinear Rocket

4.1 Deliverable 4.1

In this section, a nonlinear simulation of the rocket is realized. The structure and parameters of the controllers are exactly the same as above. Following the tuning strategy explained earlier, we were able to obtain the results in figure 20. Note that the trajectory tracking performance improves with the increase of Tf, the simulation time. We believe that this is due to the fact that a higher Tf gives the rocket more time to complete the trajectory. Thus, the rocket has more time to reach each step reference and does can go slower.



(a) Tf=30 (b) Tf=50

Figure 11: Linear MPC controllers tracking with different Tf values.

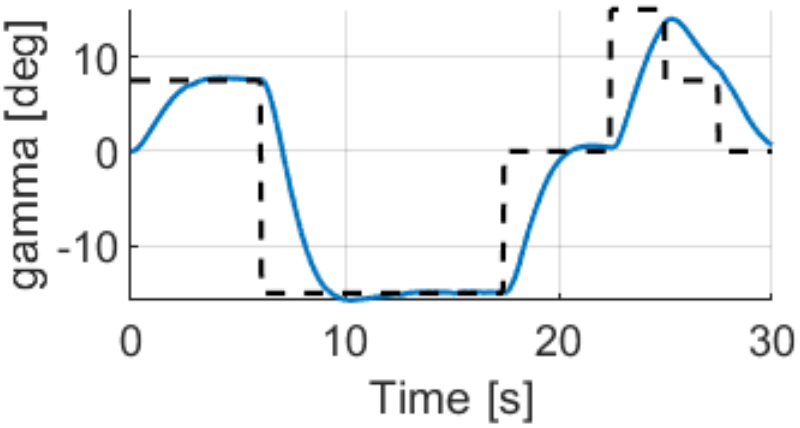


Figure 12: Simulation of the nonlinear rocket's roll angle tracking the given path with linear MPC controllers with Tf=30

5 Offset-Free Tracking

5.1 Deliverable 5.1

In this section the same linear system is controlled using the same structure and parameters for the MPC except for that of the z MPC controller. Indeed, the objective of this section is to introduce a disturbance d in the rocket's mass which will be compensated by extending the z-controller. The dynamics of the system in the z-direction is now :

$$x^+ = Ax + Bu + Bd$$

The objective function to minimize is now:

$$J^*(x) = \min \sum_{i=0}^{N-1} \left(x_i - x_{ref}\right)^T Q \left(x_i - x_{ref}\right) + u_i R u_i + \left(x_N - x_{ref}\right)^T Q \left(x_N - x_{ref}\right)$$

$$s.t. \quad x_{i+1} = Ax_i + Bu_i + d_i$$

An observer is therefore introduced to estimate the offset and the state of the system. Therefore, constraint satisfaction can no longer be ensured and the terminal set for the z-controller has to be dropped. A state and disturbance estimator L based on the augmented model is introduced:

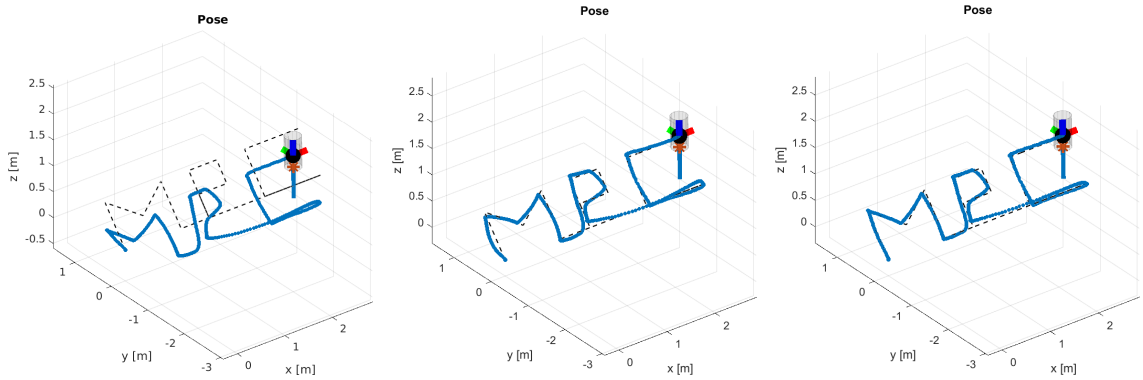
$$\bar{x}^+ = \bar{A}\bar{x} + \bar{B}u + L(\bar{C}\bar{x} - y)$$

in which $\bar{x}^+ = \begin{pmatrix} \hat{x} \\ \hat{d} \end{pmatrix}$, $\bar{A} = \begin{pmatrix} A & B \\ 0 & I \end{pmatrix}$, $\bar{B} = \begin{pmatrix} B \\ 0 \end{pmatrix}$, $\bar{C} = \begin{pmatrix} C & 0 \end{pmatrix}$.

The parameters of the controllers however stay the same as before. The observer L is computed using the Matlab "place" function in which the input is $\bar{A}$, $\bar{B}$ and the desired poles. The design strategy for the estimator L is to make it as aggressive as possible while keeping the system stable. The objective is for the estimator to be faster than the closed-loop controlled system. Therefore, its poles have to be smaller than the poles of the matrix A of the augmented z system. To select the said poles, we started from [0.6 0.7 0.8] and iteratively chose poles closer to 0. The chosen poles were those for which the simulation of the rocket would be as close as possible to the results found in Section 4. Finally, poles of [0.1 0.2 0.3] were chosen. This enables a dynamic rejection of the disturbance allowing the rocket to quickly adapt without causing loss of of feasibility or an unwanted overshoot of the estimation.

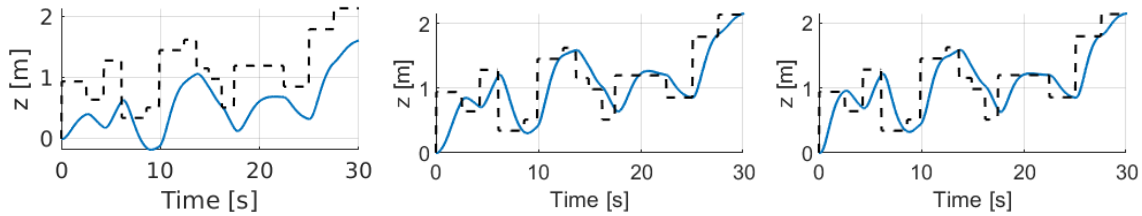
Using this design, we ran several simulations to evaluate the performance of disturbance rejection. Firstly, we change the mass of the rocket from 1.783 to 1.85. Such a disturbance causes the rocket

to under evaluate its altitude z and results in a bad tracking as can be seen in figures 13 (a) and 14 (a). When applying the observer, the disturbance can be estimated and thus rejected by implementing it in the system dynamics. This allows to compensate for the mass offset. We compare two cases, one with poles [0.7 0.8 0.9] whose proximity to the unit circle causes slow dynamics of the observer and the chosen poles of [0.1 0.2 0.3] which gives better rejection. As can be observed in graph (b) of figures 13 and 14, slow poles allow to reject the disturbance. Nonetheless, as shown by graph (c) of these figures, a much more accurate tracking is achieved with faster poles. One should note that setting too small poles can cause overshoot of the disturbance estimation and eventually lead to infeasibility.



(a) Without rejection (b) With rejection, slow poles (c) With rejection, fast poles

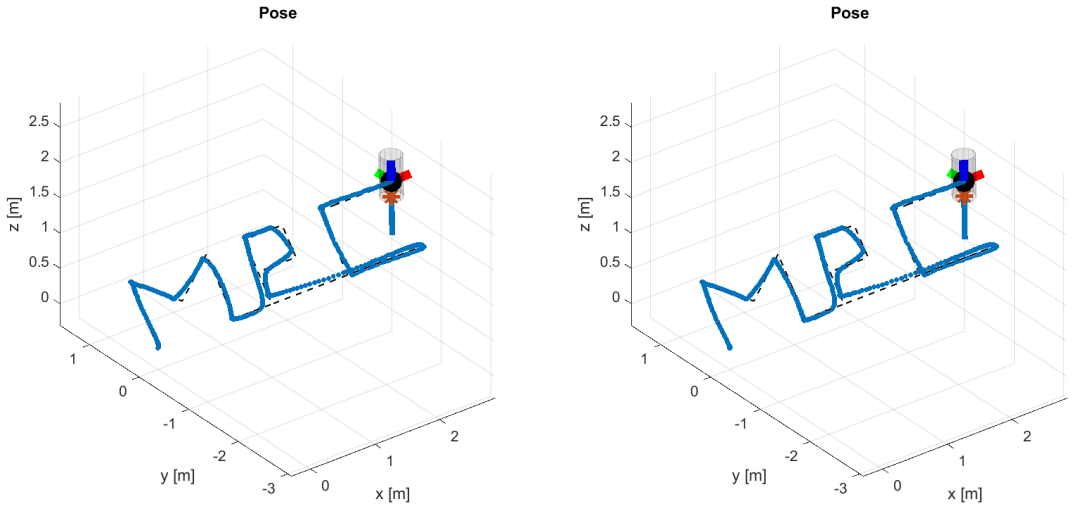
Figure 13: Full trajectory with rocket.mass = 1.85



(a) Without rejection (b) With rejection, slow poles (c) With rejection, fast poles

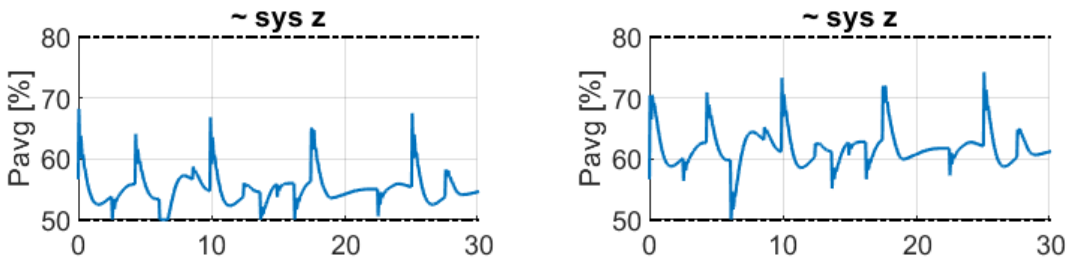
Figure 14: Z trajectory with rocket.mass = 1.85

As a further verification of the disturbance rejection, other mass were tested, notably ones lower than the original mass. As can be seen in figure 15, we obtain equally satisfying rejection and correct tracking of the trajectory. There is a difference, but it is barely noticeable. However, just to be sure, we plotted in figure 16 the average thrust over the simulation for both masses. Indeed, the simulation with lower mass has a lower average thrust compared to the simulation with higher mass. This confirms that there is a difference between the two simulations but that the implemented controller is able to adapt.



(a) rocket.mass = 1.65 (b) rocket.mass = 1.85

Figure 15: Full trajectory with with disturbance rejection for different mass



(a) rocket.mass = 1.65 (b) rocket.mass = 1.85

Figure 16: Z trajectory with with disturbance rejection for different mass

Finally, as previously shown, increasing the simulation time Tf from 30 to 50 allows to improve the tracking performance as can be seen in figure 17

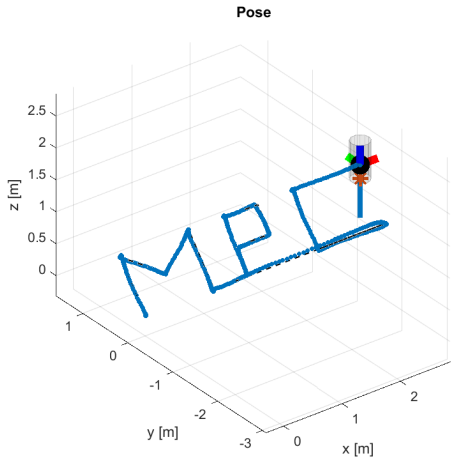


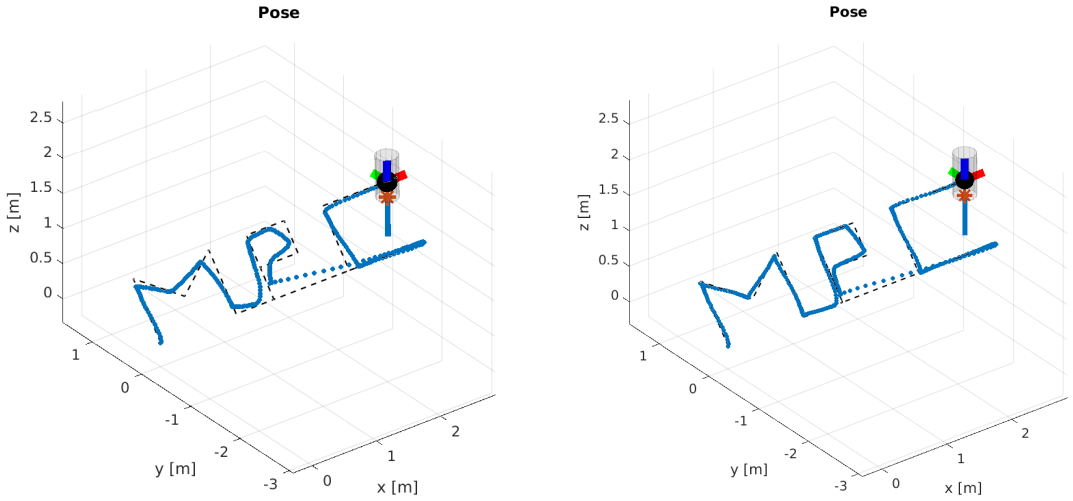
Figure 17: Disturbance rejection for Tf = 50

6 Nonlinear MPC

6.1 Deliverable 6.1

For this part of the project we developed a nonlinear MPC (NMPC) controller for the rocket using CASADI. The controller takes in the full state of the rocket as input, and the rocket is no longer decomposed into four sub-systems.

Our linear controllers work better than the nonlinear model particularly for the straight line trajectories. This may be due to some optimization in the tuning parameters, and the fact that we reduced the horizon for less costly calculations. We also found that our controller was struggling to reach the reference points in the given Tf=30s and when we increased the Tf=50s it was able to follow the path better as shown in Figure 18.



(a) Tf = 30 (b) Tf = 50

Figure 18: Tracking comparison with different Tf values

Typically NMPC controllers cover the whole nonlinear model and thus cover the whole state space eliminating the need to limit the system. However since we are working with Euler angles these are susceptible to a singularity known as gimbal lock. This phenomena occurs when the angle of the middle gimbal (pitch) approaches $\pm\frac{\pi}{2}$. For this reason we constrain this angle to be less than $|\beta| \leq 85°$ (1.48353 rad). In one instance maximum roll angle is kept to default $|\gamma_{\text{ref}}| = 15°$. This angle is also changed as commented on a second experiment with $|\gamma_{\text{ref}}| = 50°$. Naturally our physical constraints are kept for this model with P_{avg} and P_{diff} within valid range. We do not however put a constrain on δ_1 or δ_2 values as we did previously with linear controllers.

These boundary and physical conditions are made explicit in our code below.

```
% ---- boundary conditions and physical constraints ----
angle_limit = 1.4835;
Pavg_min =  50;     Pavg_max = 80;
Pdiff_min = -20;    Pdiff_max = 20;
opti.subject_to( -angle_limit < Xs(4,:) < angle_limit);
opti.subject_to( -angle_limit < Xs(5,:) < angle_limit);
opti.subject_to(Pavg_min <= Us(3,:) <= Pavg_max );
opti.subject_to(Pdiff_min <= Us(4,:) <= Pdiff_max );
```

Design Procedure

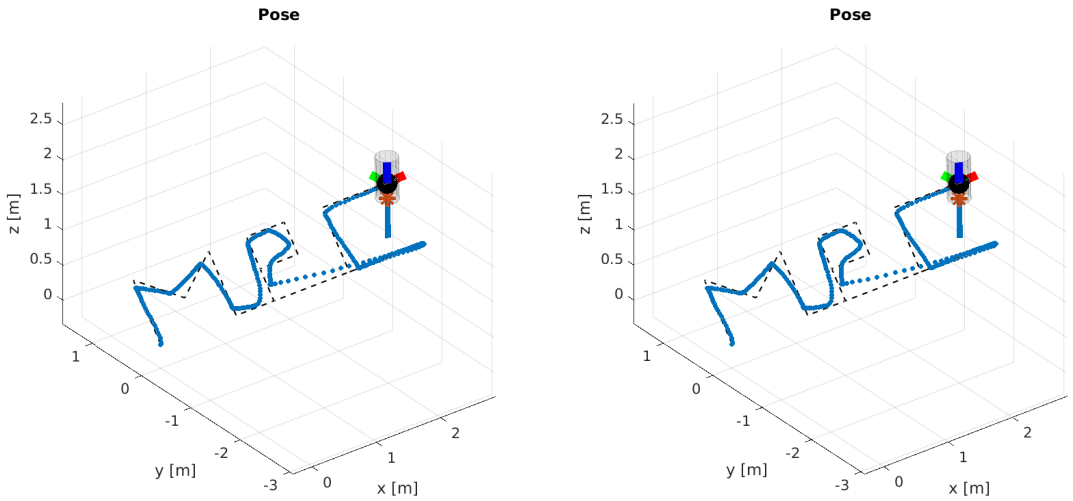
Firstly we defined our steady state trajectory and input variables (x_s, u_s). We used the Runge-Kutta 4 method to discretize our function as shown below.

```
% ---- steady-state variables ----
Xs = opti.variable(nx,1);
Us = opti.variable(nu, 1);
% ---- Runga-Kutta 4 Integration ----
k1 = rocket.f(Xs,         Us);
k2 = rocket.f(Xs+h/2*k1, Us);
k3 = rocket.f(Xs+h/2*k2, Us);
k4 = rocket.f(Xs+h*k3,    Us);
opti.subject_to(Xs == Xs + h/6*(k1+2*k2+2*k3+k4));
```

We used three gains as our tuning parameters: (*gain_x_ref*, *gain_speed*, *gain_cmd*). We used *gain_x_ref* to track the reference value, *gain_speed* to control the rocket's speed, and *gain_cmd* to minimize the input. We obtained the best results when we set a larger gain on tracking the reference, and a more moderate gain on the speed gain to avoid overshooting. Tuning these gains we found the best results appear with the following numbers.

gain_x_ref = 2000000, *gain_speed = 100*, *gain_cmd = 1*.

In our first test we used the default roll angle $|\gamma_{\text{ref}}| = 15°$, and then we tested our controller with $|\gamma_{\text{ref}}| = 50°$. The NMPC showed the almost the same result in both scenarios as shown in Figure 19.

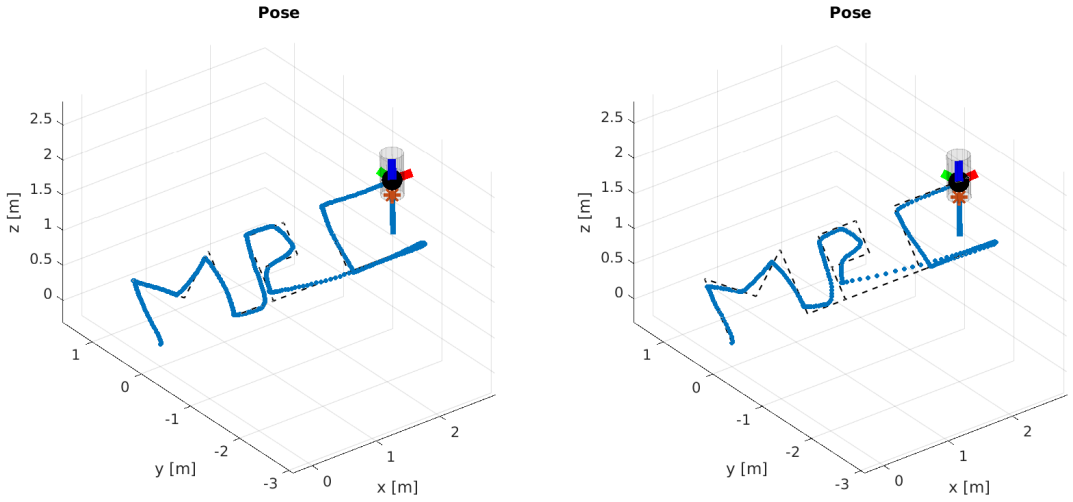


(a) Normal roll angle (b) Max roll angle

Figure 19: Comparison deliverable 6.1 with and without and default maximum roll angle

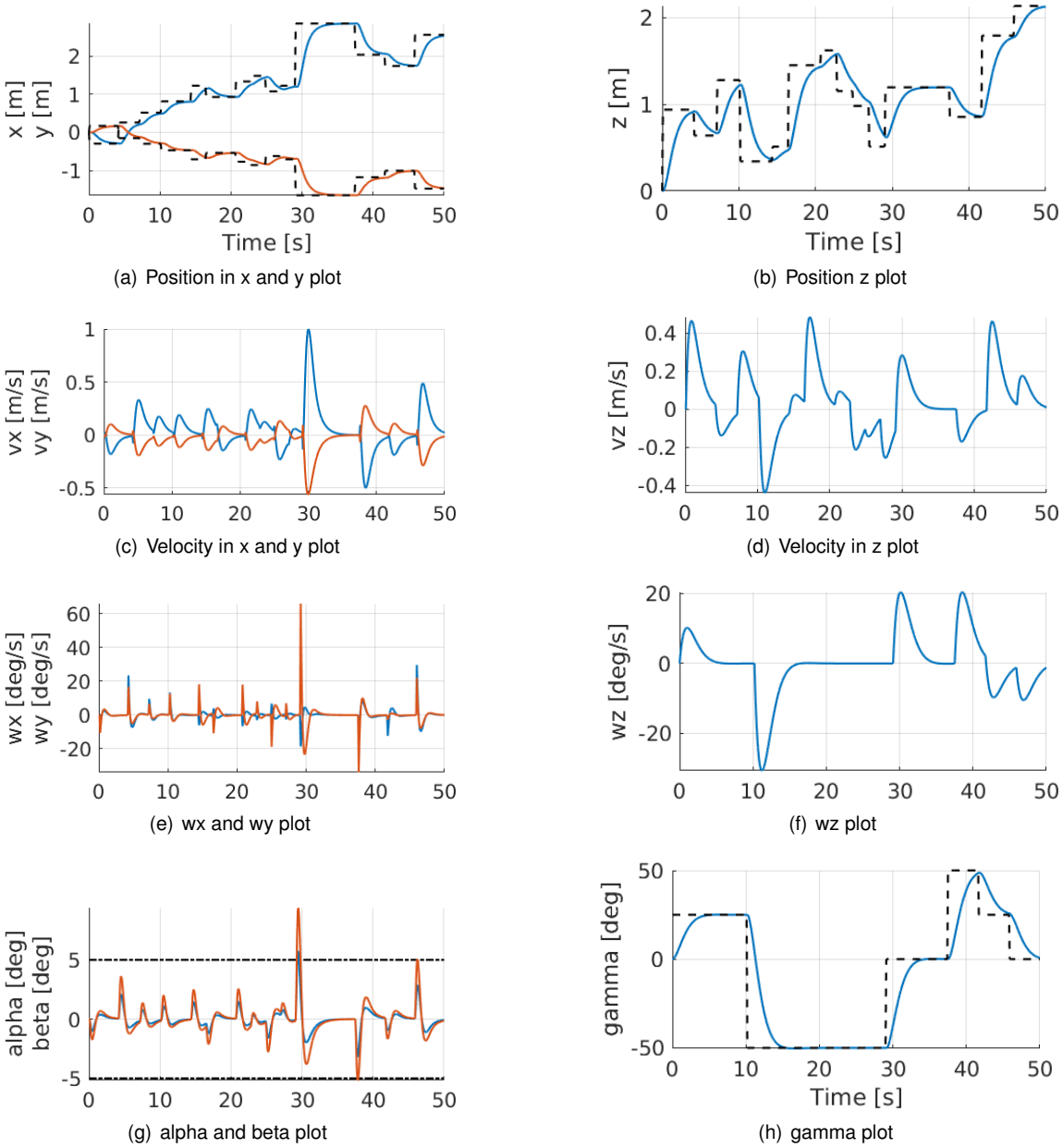
In a second experiment, we see how both the linear and nonlinear controllers perform for a maximum roll reference of $|\gamma_{\text{ref}}| = 50°$. We had to tune the Q and R values of our $MPC\_Control\_roll$ to (Q= 1*eye(nx); R = 100;) in order for the simulation to work.

The following plots show the final performance of our NMPC controller.



(a) Linear Controllers (b) Non-linear Controller

Figure 20: Comparison between linear and non-linear controller with $|\gamma_{\text{ref}}| = 50°$



(a) Position in x and y plot

(b) Position z plot

(c) Velocity in x and y plot

(d) Velocity in z plot

(e) wx and wy plot

(f) wz plot

(g) alpha and beta plot

(h) gamma plot

Figure 21: Plots of our different results.