EPFL

ME-425 Model predictive control

Mini-project : MPC controller for a rocket prototype

Group 18

Tom MERY	**297217**
Ramy CHARFEDDINE	**295758**

Contents

List of Figures

Introduction

In the aerospace technology field, and many other engineering fields of study more broadly, the control of a system's trajectory is essential to achieve complex objectives. Also known as Thrust Vector Control (TVC), this is carried out by independently controlling the thrust direction and magnitude of each of the engines for a rocket for instance. To simulate this type of rocket's control in a simplified way, this project deals with the study of a small-scale prototype where the system is replaced by a high-performance drone racing propellers. A non-exhaustive explanation of the working principle of such system is given later in the report.

The under-constraints control of this drone is achieved through the development of Model Predictive Control (MPC) methods for different situations and different control objectives. This report tackles the design of MPC regulators and MPC tracking controllers for the linearized system. As well as, the simulation of the nonlinear rocket behaviour subject to the controllers previously designed. The offset-free tracking method is also addressed. Finally, an approach to non-linear MPC is given.

1 System Dynamics

The system's dynamics is determined according to two reference frames. The first one is the body frame (subscript b) with the origin O_b attached to the center of mass of the rocket (fig.1). The second one is the world frame (subscript w) which is a fixed, inertial frame.

The non-linear model is derived according to 12 states represented by the state vector :

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} \boldsymbol{\omega}^T & \boldsymbol{\varphi}^T & \mathbf{v}^T & \mathbf{p}^T \end{bmatrix}^T \\ [\mathbf{x}] &= \begin{bmatrix} \text{rad/s} & \text{rad} & \text{m/s} & \text{m} \end{bmatrix}^T \end{aligned} \tag{1}$$

where $\boldsymbol{\omega} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$ are the angular velocities about the body axes. The Euler angles $\boldsymbol{\varphi} = \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix}^T$ represent the attitude of the body frame with respect to the world frame. The velocity and position, $\mathbf{v} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T$ and $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T$, are expressed in the world frame with $\mathbf{v} = \dot{\mathbf{p}}$.

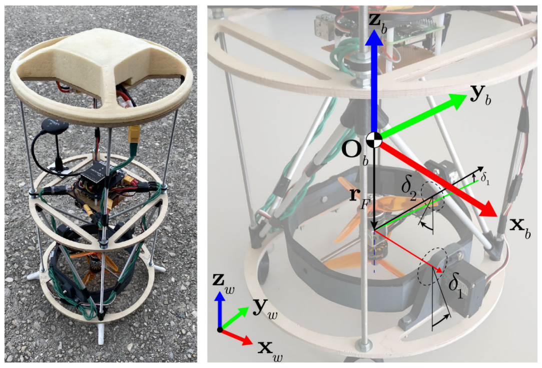


Figure 1: System's representation

The control vector is denoted by $\mathbf{u}$ and given in eq.2. The prototype is controlled through the command servo angles δ_1 and δ_2, as well as the speed of the bottom and top propellers, P_1 and P_2 respectively. The propellers counter-rotate such that their torques cancel each other out in stationary flight. Being mounted on gimbals, the propellers can be tilted by the two servo motors. For a sake of convenience, one will consider the propellers' average command speed as an input $P_{avg} = \frac{P_1+P_2}{2}$ as well as the command speed difference $P_{diff} = P_2 - P_1$, which gives the following input vector :

$$\mathbf{u} = \begin{bmatrix} \delta_1 & \delta_2 & P_{avg} & P_{diff} \end{bmatrix}^T, \quad [\mathbf{u}] = \begin{bmatrix} \text{rad} & \text{rad} & \% & \% \end{bmatrix}^T \tag{2}$$

The non-linear model is derived considering that the system is only subject to gravity mg, thrust F_b, and total torque M_b including the torque due to the thrust vector and the torque M_Δ caused by the speed difference between the two propellers. By deriving all the dynamic equations given and explained in the description of the project [1], the non-linear model can be represented as :

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{\omega} \\ \dot{\varphi} \\ \dot{p} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} J^{-1}\left(M_b - \omega \times (J\omega)\right) \\ R(\varphi) \cdot \omega \\ v \\ \frac{T_{wb} \cdot F_b}{m} - g \end{bmatrix} \tag{3}$$

with $R(\varphi) = \frac{1}{\cos\beta} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma\cos\beta & \cos\gamma\cos\beta & 0 \\ -\cos\gamma\sin\beta & \sin\gamma\sin\beta & \cos\beta \end{bmatrix}$ and T_{wb} being the direction cosine matrix from body to world frame. Both the thrust F_b and the torque M_Δ vectors are determined by the gimbal angles such that :

$$\frac{F_b}{\|F_b\|} = \frac{M_\Delta}{\|M_\Delta\|} = \begin{bmatrix} \sin\delta_2 \\ -\sin\delta_1\cos\delta_2 \\ \cos\delta_1\cos\delta_2 \end{bmatrix} \tag{4}$$

2 Linearization

Once the system's non-linear model is derived, and because a linear model is easier to understand and often necessary for most control system design methods, a linearization of the system is performed. Linearization is used to give important information about how the system behaves in the neighborhood of equilibrium points. Typically, one can learn whether the point is stable or unstable, as well as something about how the system approaches (or moves away from) the equilibrium point. By using the trim Matlab function, a steady-state state and input pair (x_s, u_s) is found and is equivalent to finding the state and input pair for which :

$$f(x_s, u_s) = 0 \tag{5}$$

Linearization around this equilibrium point is then performed, corresponding to the following system's transformation

$$\begin{cases} \dot{x}(t) = f[x(t), u(t), t] \\ y(t) = g[x(t), u(t), t] \end{cases} \quad \rightarrow \quad \begin{cases} \dot{\tilde{x}}(t) = A(x(t) - x_s(t)) + B(u(t) - u_s(t)) \\ \tilde{y}(t) = C(x(t) - x_s(t)) + D(u(t) - u_s(t)) \end{cases} \tag{6}$$

with,

$$\begin{aligned} A &= \left.\frac{\partial f(x,u)}{\partial x}\right|_{x_s,u_s} & B &= \left.\frac{\partial f(x,u)}{\partial u}\right|_{x_s,u_s} \\ C &= \left.\frac{\partial g(x,u)}{\partial x}\right|_{x_s,u_s} & D &= \left.\frac{\partial g(x,u)}{\partial u}\right|_{x_s,u_s} \end{aligned} \tag{7}$$

2.1 Deliverable 2.1

By performing the linearization step, the nominal point to linearize around is given as

$$\begin{aligned} \mathbf{x_s} &= \left[\begin{array}{cccc} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{array}\right]^T \\ \mathbf{u_s} &= \left[\begin{array}{cccc} 0 & 0 & 56.67 & 0 \end{array}\right]^T \end{aligned} \tag{8}$$

The matrices A, B, C, D are then given using the linearize Matlab function and the system can be decomposed into the following four independent continuous time models

1. *sys_x*: Thrust vector angle δ_2 to position x

$$\begin{aligned} \begin{pmatrix} \dot{\omega_y} \\ \dot{\beta} \\ \dot{v_x} \\ \dot{x} \end{pmatrix} &= \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 9.81 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \omega_y \\ \beta \\ v_x \\ x \end{pmatrix} + \begin{pmatrix} -55.68 \\ 0 \\ 9.81 \\ 0 \end{pmatrix} \delta_2 \\ \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \omega_y \\ \beta \\ v_x \\ x \end{pmatrix} \end{aligned} \tag{9}$$

2. *sys_y*: Thrust vector angle δ_1 to position y

$$\begin{aligned}
\begin{pmatrix} \dot{\omega_x} \\ \dot{\alpha} \\ \dot{v_y} \\ \dot{y} \end{pmatrix} &= \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -9.81 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \omega_x \\ \alpha \\ v_y \\ y \end{pmatrix} + \begin{pmatrix} -55.68 \\ 0 \\ -9.81 \\ 0 \end{pmatrix} \delta_1 \\
\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \omega_x \\ \alpha \\ v_y \\ y \end{pmatrix}
\end{aligned} \tag{10}$$

3. *sys_z*: Average throttle P_{avg} to height z

$$\begin{aligned}
\begin{pmatrix} \dot{v_z} \\ \dot{z} \end{pmatrix} &= \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} v_z \\ z \end{pmatrix} + \begin{pmatrix} 0.1731 \\ 0 \end{pmatrix} P_{avg} \\
\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v_z \\ z \end{pmatrix}
\end{aligned} \tag{11}$$

4. *sys_roll*: Differential throttle P_{diff} to roll angle γ

$$\begin{aligned}
\begin{pmatrix} \dot{\omega_z} \\ \dot{\gamma} \end{pmatrix} &= \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \omega_z \\ \gamma \end{pmatrix} + \begin{pmatrix} -0.104 \\ 0 \end{pmatrix} P_{diff} \\
\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \omega_z \\ \gamma \end{pmatrix}
\end{aligned} \tag{12}$$

The decomposition of the main system into 4 independent sub-systems can be explained in an intuitive physical way by looking at the effect of each of the inputs $(\delta_1, \delta_2, P_{avg}, P_{diff})$ on the rocket's trajectory and it's orientation.

1. *sys_x*: First, let's consider the case where only δ_2 is imposed. In this case, one knows that the deflection angle δ_2 of servo 2 corresponds to a rotation of the rocket around axis y_b. That's why the angular velocity ω_y about the y body axis is sufficient. Besides, this actually corresponds to a rotation of β of the body frame with respect to the world frame. Having the rocket rotated and in the plan x-z means that under the activation of the propellers, the system will move in the x-z plan. Therefore, by projection, one only needs the x-coordinate to determine the position of the rocket. The velocity along the x-axis is then deduced from the x-position. Thus, under the effect of δ_2, the system can be fully represented by the state parameters ω_y, β, v_x, x.

2. *sys_y*: Similarly, the effect of the deflection angle of servo 1, δ_1 corresponds to a rotation about the body axis x_b. Thus, if one keeps the same reasoning as before, the rocket will move along the y-z plane and therefore the system can be fully represented by ω_x, α, v_y, y.

3. *sys_z*: If only the propellers are activated and there is no throttle difference between the motors and no tilting by servo 1 and 2 neither, then the rocket remains on its z_b axis. Since no rotation occurs around the body frame nor around the world frame then the system is only determined by the speed and position along z body axis.

4. *sys_roll*: Let's consider now that only a throttle difference between the motors exists. Then, the rockets stays on the z body-axis since no tilting occurs under zero servo deflection angles. However, in this case, the rocket will rotate around it's z-body axis and consequently, the body frame will also rotate with respect to the world frame. Thus, in this case, the system is described by ω_z and γ.

3 Design MPC Controllers for Each Sub-System

Once the system is linearized around the nominal point, decomposed and discretized, the objective is to design a recursively feasible, stabilizing MPC controller for each of the subsystems. Two control cases are considered in this part of the report. First, the design of a MPC regulator to ensure the converges of each system to the origin. Then, the design of a MPC tracking controller where in addition to feasibility, the controller has to ensure the stabilization to a point of reference. In general case, the design of a MPC controller is equivalent to finding the optimal input sequence that will minimize a cost function (eq.13) while ensuring states and inputs' constraints, stabilization and maximization of the feasible set.

$$\begin{aligned} V_N^\star(x_0) = \min_{x,u} & \sum_{i=0}^{N-1} I(x_i, u_i) + V_f(x_N) \\ \text{s.t. } & x_{i+1} = f(x_i, u_i) \\ & (x_i, u_i) \in \mathbb{X}, \mathbb{U} \\ & x_N \in \mathcal{X}_f \end{aligned} \tag{13}$$

In eq.13, $\mathbb{X}, \mathbb{U}$ are the state and input constraints set respectively. $I(x, u)$ corresponds to the cost of being in state x and applying input u. N is the horizon length. One knows that the feasibility and stability of systems is ensured for an infinite horizon optimal control. However, such controller being impossible to design as it is computationally intractable, MPC as finite horizon optimal control is used as an alternative solution. Thus, to simulate the infinite horizon behaviour, a terminal cost $V_f(x_N)$ and a terminal constraint set $\mathcal{X}_f$ are introduced to explicitly ensure feasibility and stability. In this particular application, as it is not a real time MPC implementation, the time horizon is mainly chosen in order to keep the computation time under control. Computation time can be reduced by limiting the time horizon of the prediction. But limiting the time horizon results in sub optimal control and infeasibility can occur if the terminal set is dropped. Thus a trade-off has to be made. Multiple simulations have been carried out with increasing values of H and N consequently. The horizon length has then been chosen as the smallest value ensuring the respect of the required settling time (8s) for all states variables. With a sampling time $T_s = \frac{1}{20}$s, the horizon length in seconds is set to $\mathbf{H} = \mathbf{3s}$ by experimenting. Giving $N = \frac{H}{T_s} = 60$. This value of the horizon length is considered as a fixed value for all following controllers.

3.1 Deliverable 3.1

Let's first handle the case where the rocket has to head to the origin. To do so, given linear system's dynamics and linear input and state's constraints, a linear MPC with quadratic cost controller is implemented. The optimization problem is expressed as

$$\begin{aligned} J^\star(x) = \min_{x,u} & \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i + x_N^T Q_f x_N \\ \text{s.t. } & x_{i+1} = A x_i + B u_i \\ & (x_i, u_i) \in \mathbb{X} = \{x | Fx \leq f\}, \mathbb{U} = \{u | Mu \leq m\} \\ & x_N \in \mathcal{X}_f \end{aligned} \tag{14}$$

As explained earlier, this approach simulates the infinite horizon problem separated into two sub-problems. The first right part corresponds to a finite horizon optimal control taking into account the constraints. The second part of the cost function is equivalent to an unconstrained LQR starting from state x_N and approximating the infinite horizon cost. The terminal penalty weight Q_f is the solution of the discrete-time algebraic Ricatti equation and it's a parameter determining the terminal invariant set. Its computation is explained a bit later. The separation into sub-problems is at the core of the design procedure to ensure recursive constraint satisfaction.

The penalty weights Q and R are the main tuning parameters. The value assigned to each of them ends up in a trade-off between performance and the size of the terminal set. The smaller is the terminal set, the higher is the likelihood to end up in an infeasible state. Therefore, to ensure a maximization of the feasible set, the penalty of the states and inputs is chosen to be the smallest possible while ensuring the stabilization under the settling time requirement. The weights Q and R are first set to the Identity matrix with the right dimensions. Then, depending on the behaviour of the system, higher weights are assigned in the Q matrix to the states that need more penalty or in other words more control. Typically, for the systems sys_z and sys_{roll}, we give more importance to achieve the origin than the speed at which we achieve it, so more weight is given to z and γ. As for the choice of the R matrix, in the absence of any specification given in the guidelines on input's cost importance, we keep it set to identity for all subsubsystems.

$$Q_{x,y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Q_{z,roll} = \begin{bmatrix} 1 & 0 \\ 0 & 100 \end{bmatrix} \quad R_{x,y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_{z,roll} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{15}$$

Given these weight matrices, the MPC controllers are designed using the Matlab optimization toolbox YALMIP. To do so, input and state constraints matrices need to be determined for each subsystem such that

$$Fx \leq f \qquad Mu \leq m \tag{16}$$

Because of the approximate linearization that is done, constraints on the maximum angles that the rocket can take need to be satisfied

$$\begin{aligned} |\alpha| &\leq 5^\circ = 0.0873\text{rad} \\ |\beta| &\leq 5^\circ = 0.0873\text{rad} \end{aligned} \tag{17}$$

which ends up giving

$$F_{x,y} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} \quad f_{x,y} = \begin{bmatrix} 0.0873 \\ 0.0873 \end{bmatrix} \tag{18}$$

As their is no state constraints for the z and roll subsystems, the matrices F_z, F_{roll}, f_z and f_{roll} don't have to be explicitly defined. On the other hand the input constraints coming from a mechanical and engineering perspective are summed up as

$$\begin{aligned} |\delta_1| &\leq 15^\circ = 0.26\text{rad} \\ |\delta_2| &\leq 15^\circ = 0.26\text{rad} \\ 50\% \leq P_{\text{avg}} &\leq 80\% \\ -20\% \leq P_{\text{diff}} &\leq 20\% \end{aligned} \tag{19}$$

The separation into independent subsystems, is reflected by the following input constraints on the inputs of each subsystem

$$\begin{gathered} M_{x,y} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad m_{x,y} = \begin{bmatrix} 0.26 \\ 0.26 \end{bmatrix} \quad M_z = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad m_z = \begin{bmatrix} 80 - P_{avg_s} \\ -50 + P_{avg_s} \end{bmatrix} \\ M_{roll} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad m_{roll} = \begin{bmatrix} 20 \\ 20 \end{bmatrix} \end{gathered} \tag{20}$$

On the other hand, the terminal set and terminal weight Q_f are determined using MPT3. By using this method, one doesn't have to rewrite all the algorithm to find the terminal set by iterating the space of constraints over the closed loop system $A_{cl} = (A + BK)$ (where K is the feedback matrix of the LQR controller with cost Q and R) as it is automatically done by the toolbox. The invariant set for each of the subsystems is given in fig.2 to 5.

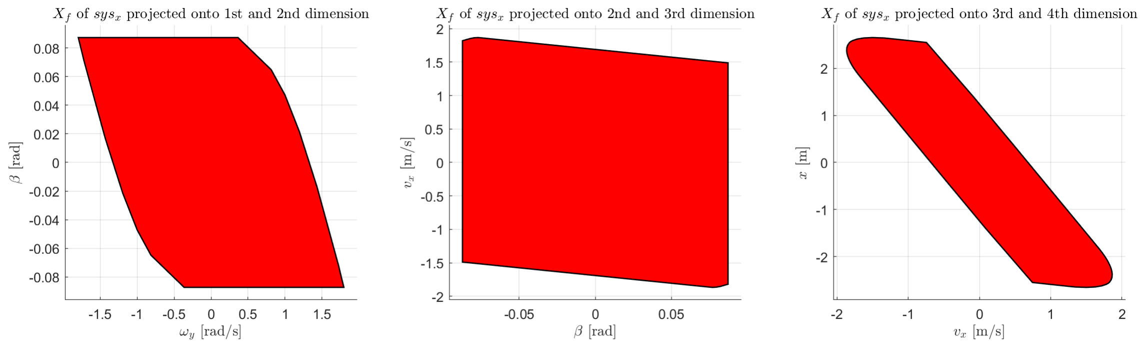


Figure 2: Terminal invariant set for sys_x

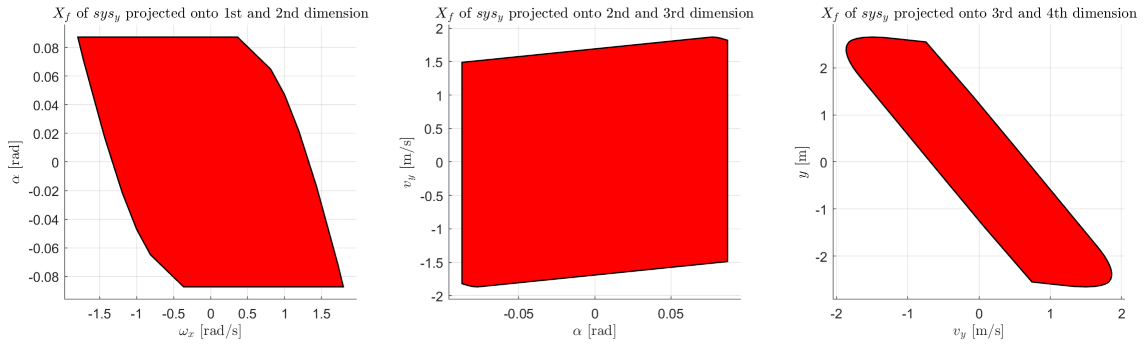


Figure 3: Terminal invariant set for sys_y

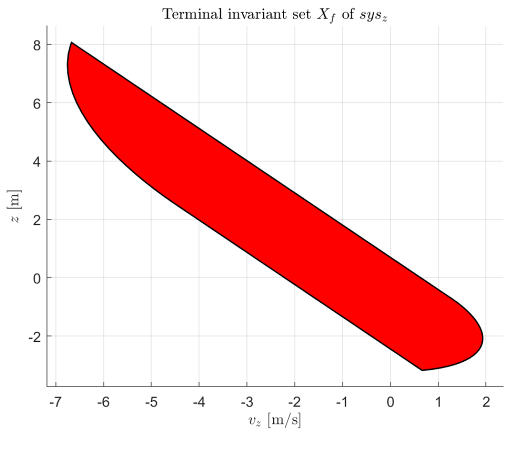


Figure 4: Terminal invariant set fro sys_z

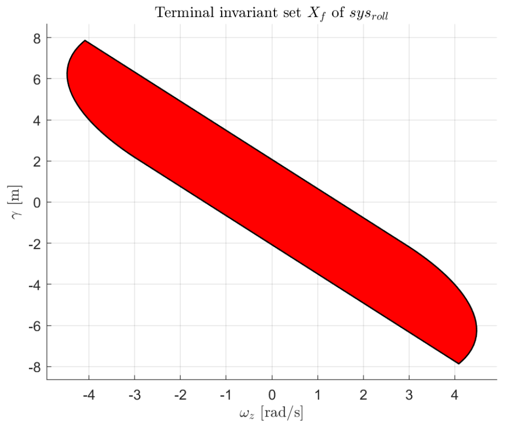


Figure 5: Terminal invariant set for sys_roll

Simulating the independent subsystems's response for 15 seconds gives the following results (fig.6-9). That's for a stabilization to the origin, with a starting point 5 meters away from the origin for x, y and z, and 45° for roll. One can notice that all 4 subsystems converge to 0 under 8 seconds.

One can however notice a slight overshoot on the z and directions which are not critical for this application.

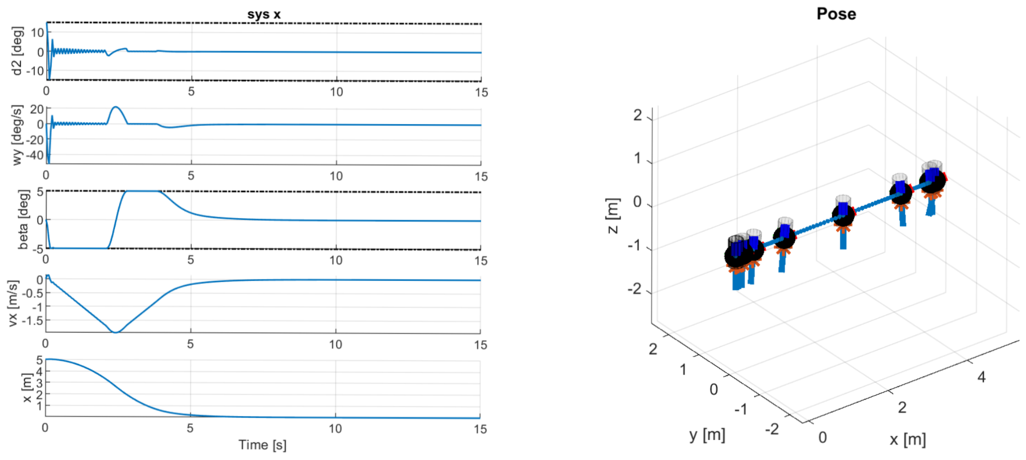


Figure 6: Simulation of discrete-time linear model sys_x in closed-loop starting at $x_0 = 5m$

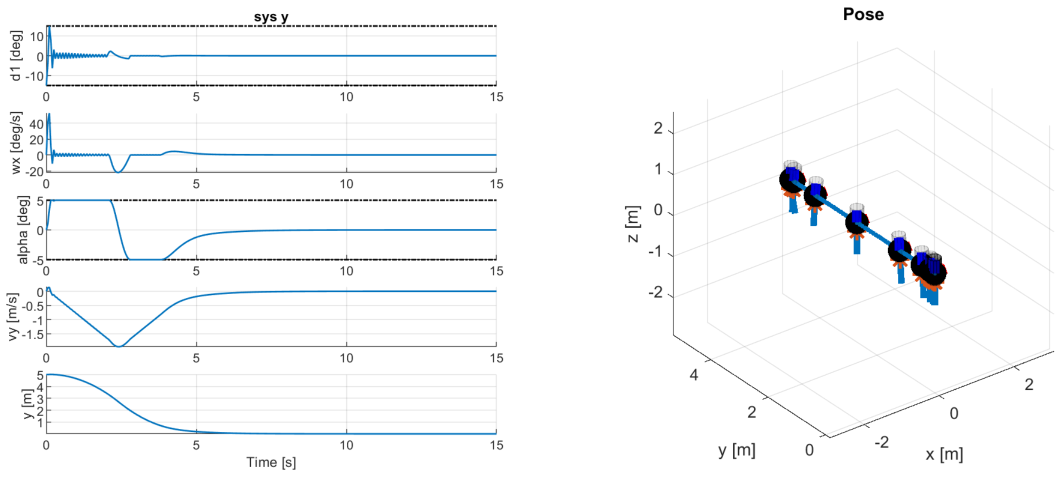


Figure 7: Simulation of discrete-time linear model sys_y in closed-loop starting at $y_0 = 5m$

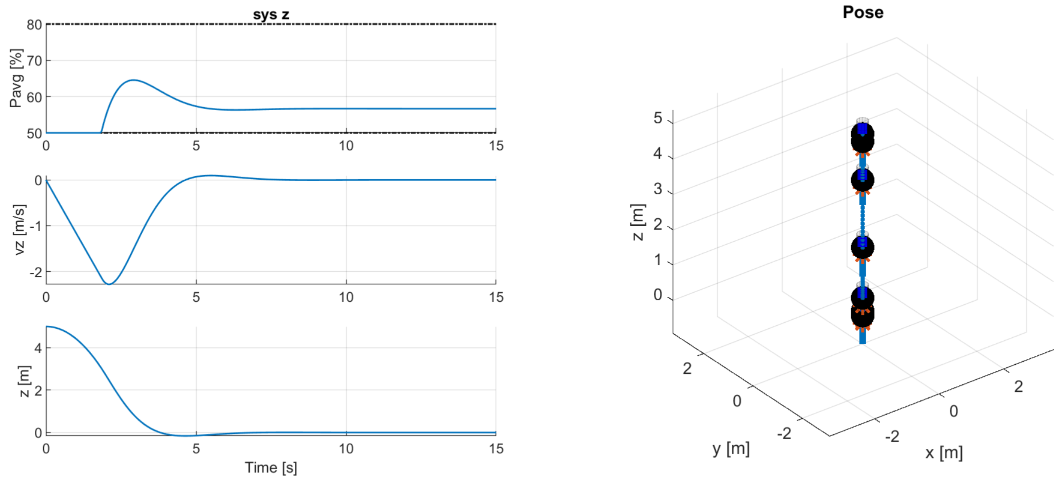


Figure 8: Simulation of discrete-time linear model sys_z in closed-loop starting at $z_0 = 5m$

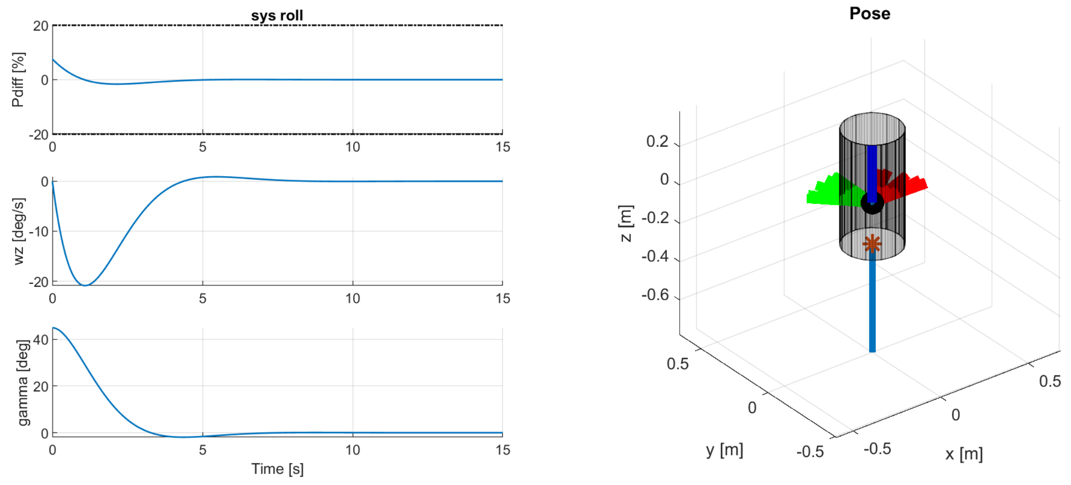


Figure 9: Simulation of discrete-time linear model sys_{roll} in closed-loop starting at $\gamma_0 = 45°$

3.2 Deliverable 3.2

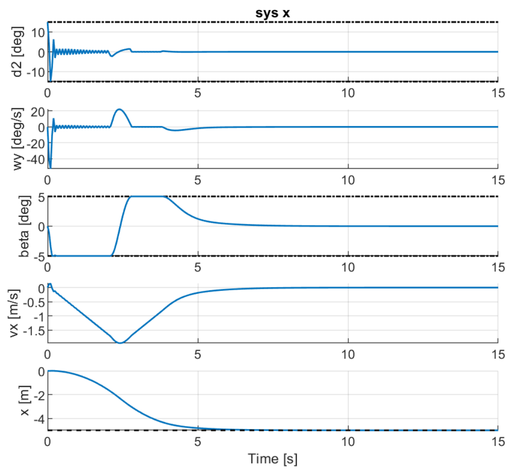
Let's now handle the case where each system is tracking a reference. For simplicity the terminal set is dropped from now and through all the following. The tracking problem can actually be seen as the design of a MPC regulator with a coordinate transformation. The main adjustment that has to be implemented concerns the delta formulation. Thus, one seeks a control input corresponding to the steady state leading to the reference target .

$$\begin{aligned} x_s &= Ax_s + Bu_s \\ r &= Cx_S \\ Fx_s &\leq f \\ Mu_s &\leq m \end{aligned} \tag{21}$$

Where r is the reference. The delta formulation states that $\Delta x = x - x_s$ and $\Delta u = u - u_s$. By inserting these in eq.14, the optimisation problem is expressed as

$$\begin{aligned} J^\star(x) = \min_{x,u} & \sum_{i=0}^{N-1} \Delta x_i^T Q \Delta x_i + \Delta u_i^T R \Delta u_i + \Delta x_N^T Q_f x_N \\ \text{s.t. } & \Delta x_0 = \Delta x \\ & \Delta x_{i+1} = A\Delta x_i + B\Delta u_i \\ & H_x \Delta x_i \leq k_x - H_x x_s \\ & H_u \Delta u_i \leq k_u - H_u u_s \end{aligned} \tag{22}$$

Here the objective is first to find the optimal sequence of $\Delta u^\star$ then deduce from it the input applied to the system as $u_0^\star = \Delta u^\star + u_s$. The implementation of the MPC controller is the same as previously using YALMIP and without any change in the tuning parameters. The state evolution obtained when simulating tracking for the different subsystems is given in fig.10 to fig.13.



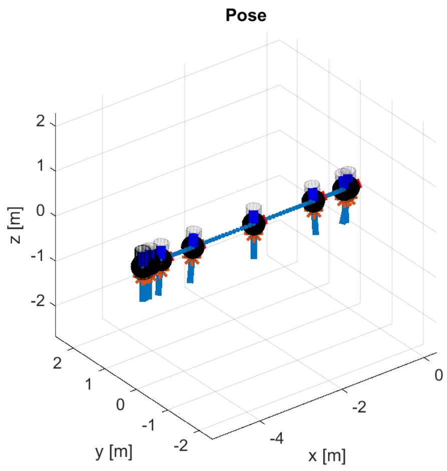


Figure 10: Simulation of discrete-time linear model sys_x in closed-loop tracking a reference at $x_{ref} = -5m$

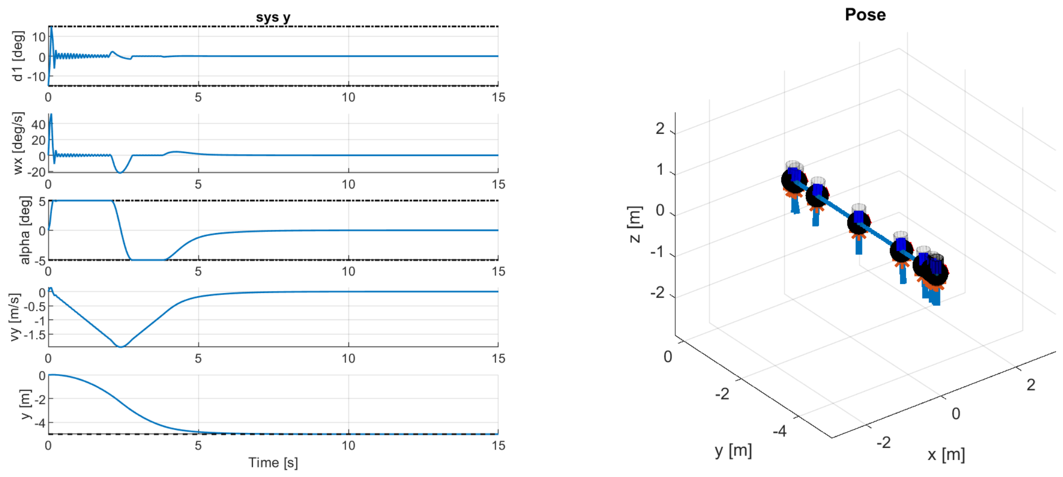


Figure 11: Simulation of discrete-time linear model sys_y in closed-loop tracking a reference at $y_{ref} = -5m$

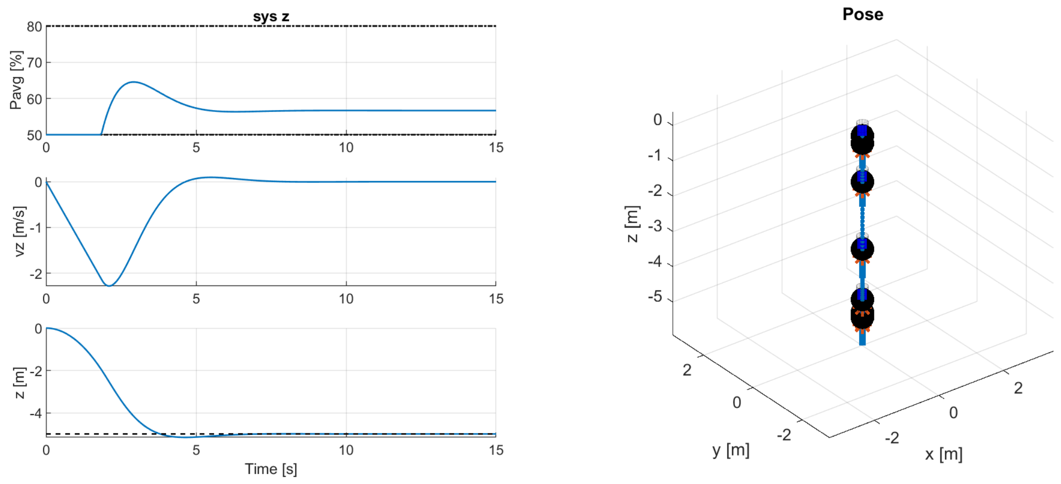


Figure 12: Simulation of discrete-time linear model sys_z in closed-loop tracking a reference at $z_{ref} = -5m$

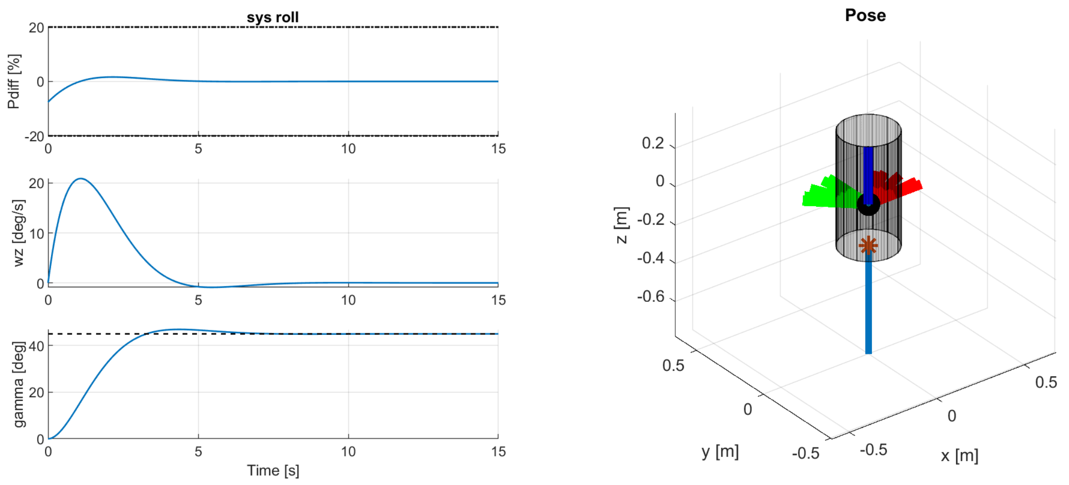


Figure 13: Simulation of discrete-time linear model *sys_roll* in closed-loop tracking a reference at $\gamma_{ref} = 45°$

4 Simulation with Nonlinear Rocket

Now that a controller has been implemented for each of the subsystems, it can be interesting to see how by merging the four of them, one can make the non linear rocket track a given path from the origin as initial state.

4.1 Deliverable 4.1

Running the linear controllers in the nonlinear simulation results in a model mismatch for the predictive controllers. Besides, with the previously chosen weight matrices, infeasibility of the linear controllers and bad tracking performance was encountered. To avoid angle constraint violations (α, β), the weights on the angular velocities (ω_x, ω_y) are increased. Moreover, more weight is given to the states x, y, z, γ to ensure reference tracking. This leads to the following matrices where R remains unchanged for all subsystems.

$$Q_{x,y} = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix} \quad Q_{z,roll} = \begin{bmatrix} 1 & 0 \\ 0 & 250 \end{bmatrix} \tag{23}$$

The simulation results in fig.14 show that the tuning parameters seem to offer good performance as the rocket is able to follow the desired path with only a small error between the real and the ideal path. One can also notice on the states graphs evolution, that the reference's tracking is achieved with a small delay due to non-real time control. This doesn't seem to affect much the tracking of the desired path on the animation so these results can be considered as acceptable.

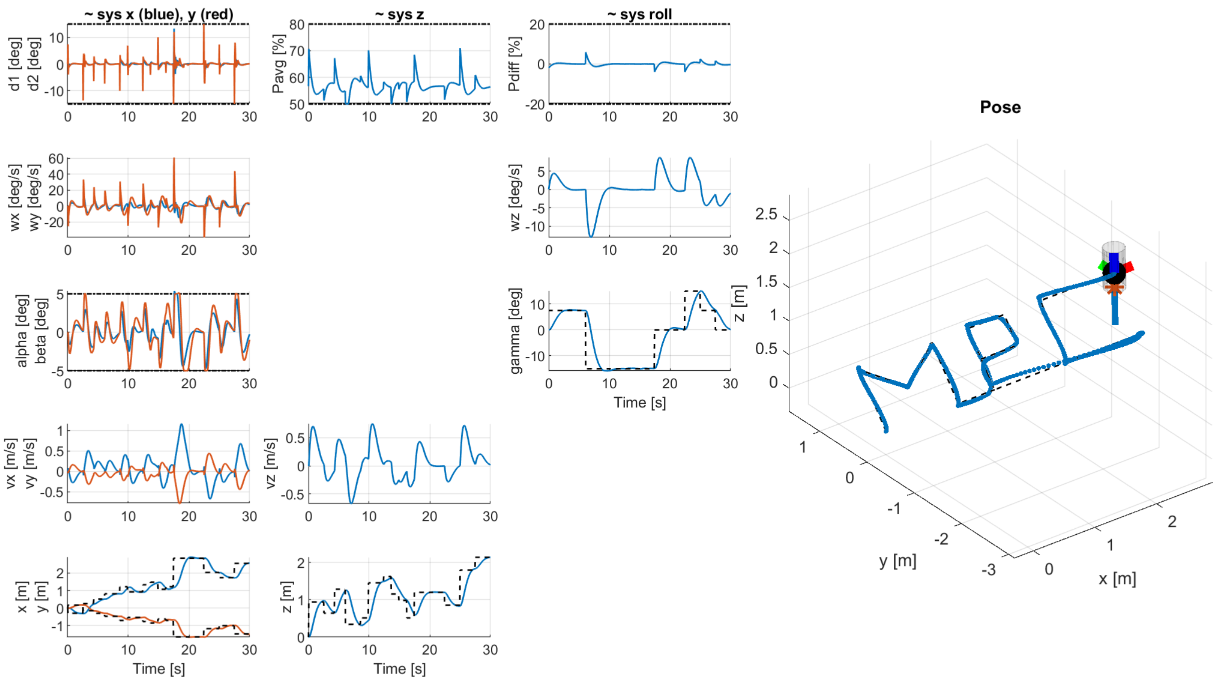


Figure 14: Simulation of the full nonlinear system tracking a given reference path with the four MPC controllers

5 Offset-Free Tracking

In this part of the project, the mass of the rocket is changed and this has an influence on the dynamics of the system in the z-direction which is now given as :

$$\mathbf{x}^{+} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{B}d \tag{24}$$

with d a constant unknown disturbance. The objective is therefore to design a MPC controller using what has been done in previous sections and finding the control inputs that use a disturbance estimate to remove the offset and ensure offset-free tracking of a reference.

5.1 Deliverable 5.1

One first designs a state and disturbance estimator based on the augmented model (eq.25).

$$\begin{bmatrix} \hat{x}_{k+1} \\ \hat{d}_{k+1} \end{bmatrix} = \begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ \hat{d}_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k + \begin{bmatrix} L_x \\ L_d \end{bmatrix} \left(C\hat{x}_k + C_d\hat{d}_k - y_k \right) \tag{25}$$

$\hat{x}$ and $\hat{d}$ are the state and disturbance estimates respectively. The error dynamics is given as

$$\begin{bmatrix} x_{k+1} - \hat{x}_{k+1} \\ d_{k+1} - \hat{d}_{k+1} \end{bmatrix} = \left(\begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} + \begin{bmatrix} L_x \\ L_d \end{bmatrix} \begin{bmatrix} C & C_d \end{bmatrix} \right) \begin{bmatrix} x_k - \hat{x}_k \\ d_k - \hat{d}_k \end{bmatrix}$$
$$\delta(k+1) = (\bar{A} + \bar{L}\bar{C})\delta(k) \tag{26}$$

To design such Luenberger Observer, one should find the observer gain $\bar{L}$ such that the error dynamics converges to zero. This is achieved by choosing the suitable observer eigenvalues ensuring fast dynamics, stability and error minimization using the duality principle between a controller and an observer. Thus, the observer eigenvalues can be chosen as the same as the controller eigenvalues given by solving :

$$det(\lambda I - A - BK) = 0 \tag{27}$$

However, in practice, the observer's dynamics should be faster than the state feedback controller's dynamics to compensate for the estimation error on the controller performance. The set of poles to improve the observation performance is therefore usually chosen as being 20% of the poles of the controller. Thus, the pole placement of the observer loop system $\bar{A} + \bar{L}\bar{C}$ being a tuning parameter, it is chosen as $[0.4, 0.5, 0.6]$. This choice is one possible solution giving good simulation results.

On the other hand, the tuning parameters N, Q and R remain the same as in previous section as there is no change of the objective. Indeed, the goal is still to have the rocket following a reference path which means that one needs to modify the system steady-state and target to take into account the effect of disturbance when designing the MPC controller. Besides, note that since an observer to estimate the offset and the state of the system is used, constraint satisfaction can no longer be ensured and the terminal set is hence not added.

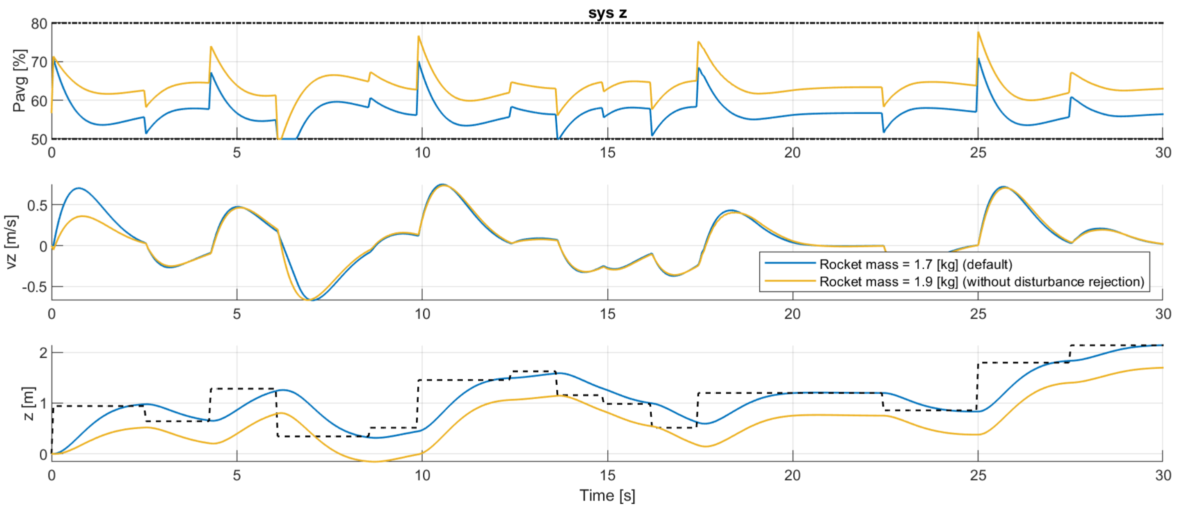


Figure 15: Simulation of the system with the original MPC controller

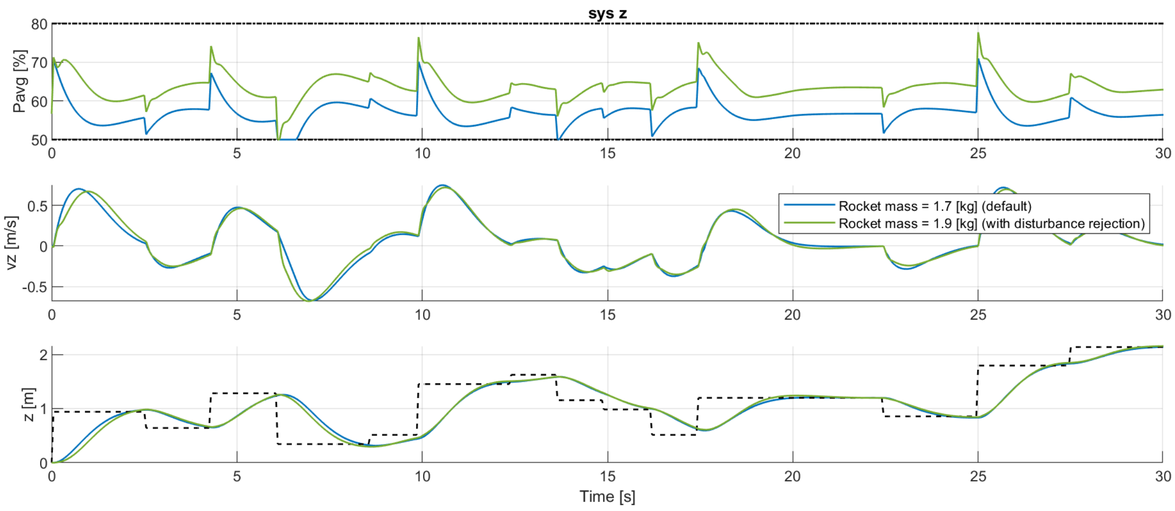


Figure 16: Simulation of the system with MPC controller acting based on the state estimates provided by the observer

The simulation results show first the impact of changing the mass on the original controller (i.e without disturbance rejection) (fig.15), then the effect of the new controller achieving off-set free tracking when disturbance rejection is added (fig.16). One can notice that in the first case the z state is always shifted by a constant disturbance ($\sim 0.2m$) from the reference path, whereas this error is barely noticeable when using a controller rejecting the disturbance. However, note that, in both cases, when the rocket gets heavier, it makes sense to have larger values of P_{avg} as the propellers will have to turn faster and thus give more thrust to compensate for the extra mass.

6 Nonlinear MPC

In this part of the project, the objective is to go back to the non-linear model of the system (eq.3) and implement a nonlinear MPC controller for the path tracking problem. By doing so, one will then be able to compare this controller to the previously designed linear one and determine how is the linearization process impacting the performance of the controller.

6.1 Deliverable 6.1

First of all, when designing a NMPC controller, one should pay attention to the the computation time which can grow rapidly leading to uncontrolled situation. Consequently, the time horizon is reduced to $H = 1\text{s}$ equivalent to $N = \frac{1}{0.1} = 10$. Designing a NMPC controller relies on the same theory as for linear MPC. The tracking controller is thus designed by solving the following non linear optimization problem,

$$
\begin{aligned}
J^{\star}(x) = \min_{x,u} & \sum_{i=0}^{N-1} (x_i - x_{ref})^T Q (x_i - x_{ref}) + u_i^T R u_i + (x_N - x_{ref})^T Q_f (x_N - x_{ref}) \\
\text{s.t. } & x_0 = X_0 \\
& x_{i+1} = F_{discr}(x_i, u_i) \\
& (x_i, u_i) \in \mathbb{X} = \{x | Fx \leq f\}, \mathbb{U} = \{u | Mu \leq m\}
\end{aligned} \tag{28}
$$

The only remaining state constraint is $|\beta| \leq 85°$ to safe numerical values and all input constraints are unchanged (eq.19)

Let's first explain what F_{discr} corresponds to. Having a continuous time non-linear model, one should discretize it in order to design a NMPC controller for it. The two most popular methods to integrate such non-linear systems are the Euler method and the Runge-Kutta 4 (RK4). This one being an approximation of order 4, it's gives more accurate discretization and is therefore performed in this case for a sampling period $T_s = h = 0.1$s. The basic principle is to find the next state x_{n+1} given the current one x_n and a weighted average of 4 increments (eq.29).

$$
\begin{aligned}
K_1 &= f(x_n, u_n) \\
K_2 &= f\left(x_n + \tfrac{h}{2} \cdot K_1, u_n\right) \\
K_3 &= f\left(x_n + \tfrac{h}{2} \cdot K_2, u_n\right) \\
K_4 &= f\left(x_n + h \cdot K_3, u_n\right) \\
F_{\text{discr}}(x_n, u_n) = x_{n+1} &= x_n + \tfrac{h}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right)
\end{aligned} \tag{29}
$$

Concerning the stage cost, having a tracking problem to solve, it takes into account the tracking residual corresponding to the difference between the predicted x and the desired trajectory states x_{ref}. As for the linear tracking case, the stage cost penalizes the squared tracking residual as well as the control input. Besides, note that since the objective is to control the rocket's position $[x, y, z]$ and angle γ, the reference x_{ref} only applies to these states and is considered as 0 for all remaining 8 states. As for linear MPC, the design of the NMPC controller relies on the tuning of the penalty weight matrices Q and R. Q is diagonal sparse matrix.

$$
Q = \text{diag}\left[5/10, 5/10, 1/250, 1/10, 1/10, 250/250, 1/10, 1/10, 1/250, 10/10, 10/10, 250/250\right] \tag{30}
$$

$$
R = \begin{bmatrix} 1/10 & 0 & 0 & 0 \\ 0 & 1/10 & 0 & 0 \\ 0 & 0 & 1/250 & 0 \\ 0 & 0 & 0 & 1/250 \end{bmatrix} \tag{31}
$$

The tuning weights are purely based on simulation experiments and are only one possible choice among several others. However, the main idea behind this choice is to take the initial weights set for the linear MPC design and scale them. In other words, in the linear case the 4 subsystems were independent and therefore giving more importance to one of the states was only impacting the behaviour of the states part of the same subsystems. Having now a unique weight matrix penalizing the 12 states at the same time, each state will have an influence on all the others. That's why to keep the same behaviour as the linear case, which gives a descent solution for the tracking problem, the weights for each state is the weight set in the linear case scaled by the maximum weight among all states in each subsystem. To keep a certain consistency, the same scale is applied to the R matrix. However, remember that in the NMPC design we are not dealing with 4 subsystems but indeed a unique 12 states and 4 inputs' system.

Finally, since the time horizon is reduced, the control prediction operates on a reduced period of time and the controller "sees" in a smaller time in the future. Consequently, the error between the prediction and reality will increase negatively influencing the performance of the controller and the system's stability. To compensate for this shortage of time horizon, the design of a terminal cost is helpful. To design it, one first needs to linearize the system as it has been done in the beginning of the project and use the same approach as for the linear MPC case by designing a LQR controller. The weight matrices are this time the above Q and R matrices (eq.30, eq.31). This is how Q_f is computed. Simulation results are represented in fig.17 to fig.19.

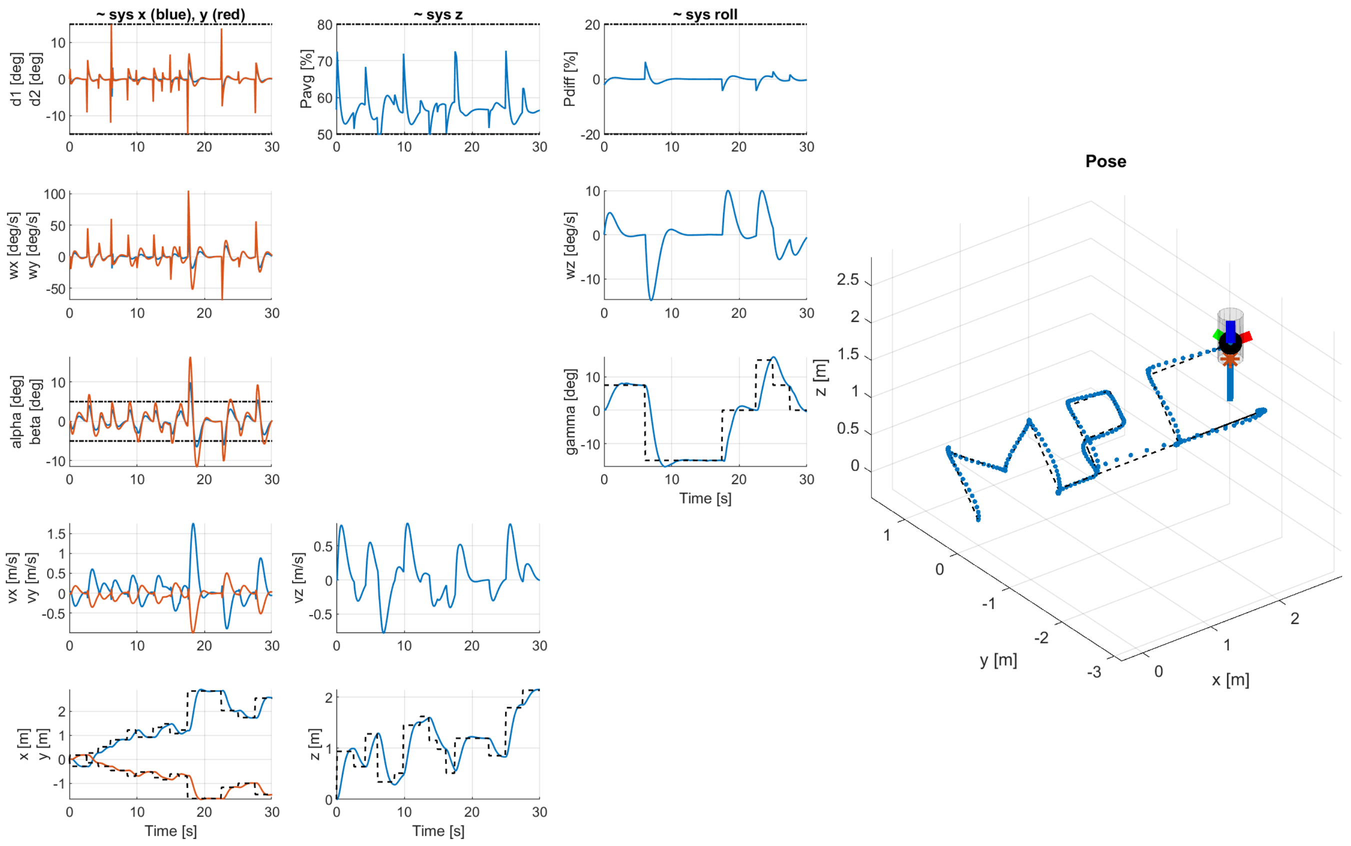


Figure 17: NMPC in nonlinear simulation under $\gamma_{max} = 15°$

One can notice that the NMPC controller provides a descent solution to the reference tracking problem. The results are quite similar to the linear MPC as expected since the tuning parameters are chosen on the same principle. Notice as well the delay between the reference path and the real behaviour of states which is due to the settling time and the computation time of the controller.

When changing the maximum roll reference to $|\gamma_{ref}| = 15°$, the NMPC controller continues to give a proper solution to the path tracking problem. However, when using the linear MPC controller on the non linear system, the feasibility and stability of the system wasn't ensured and an increase of the angular velocity weights from 5 to 50 is needed to avoid angle constraint violations only (as we want to compare the performances of the MPC with the NMPC, the other weights of the MPC are kept). Regarding fig.18 and fig.19 it is clear that the NMPC performs much better than the MPC under this more extreme reference request.

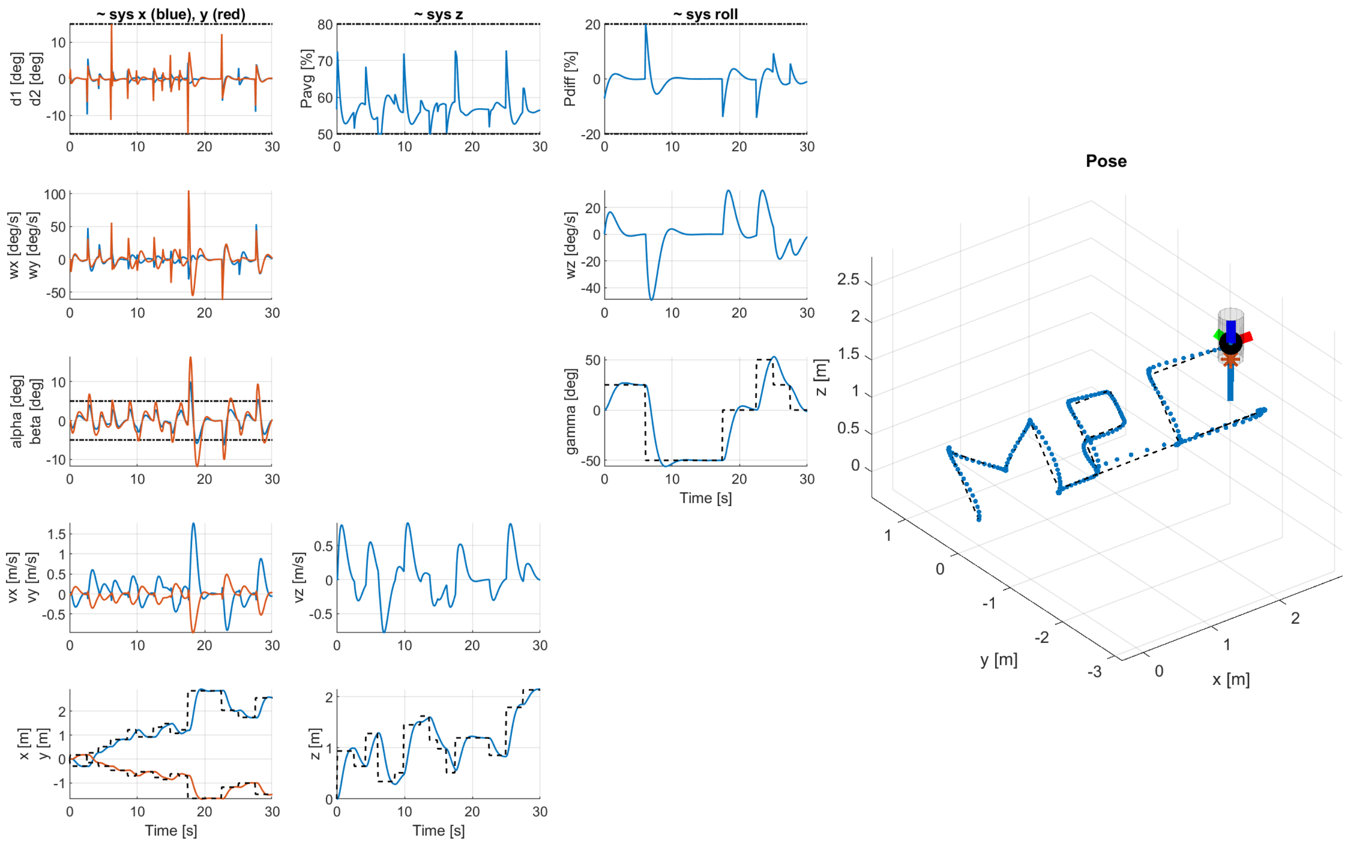


Figure 18: NMPC in nonlinear simulation under $\gamma_{max} = 50°$

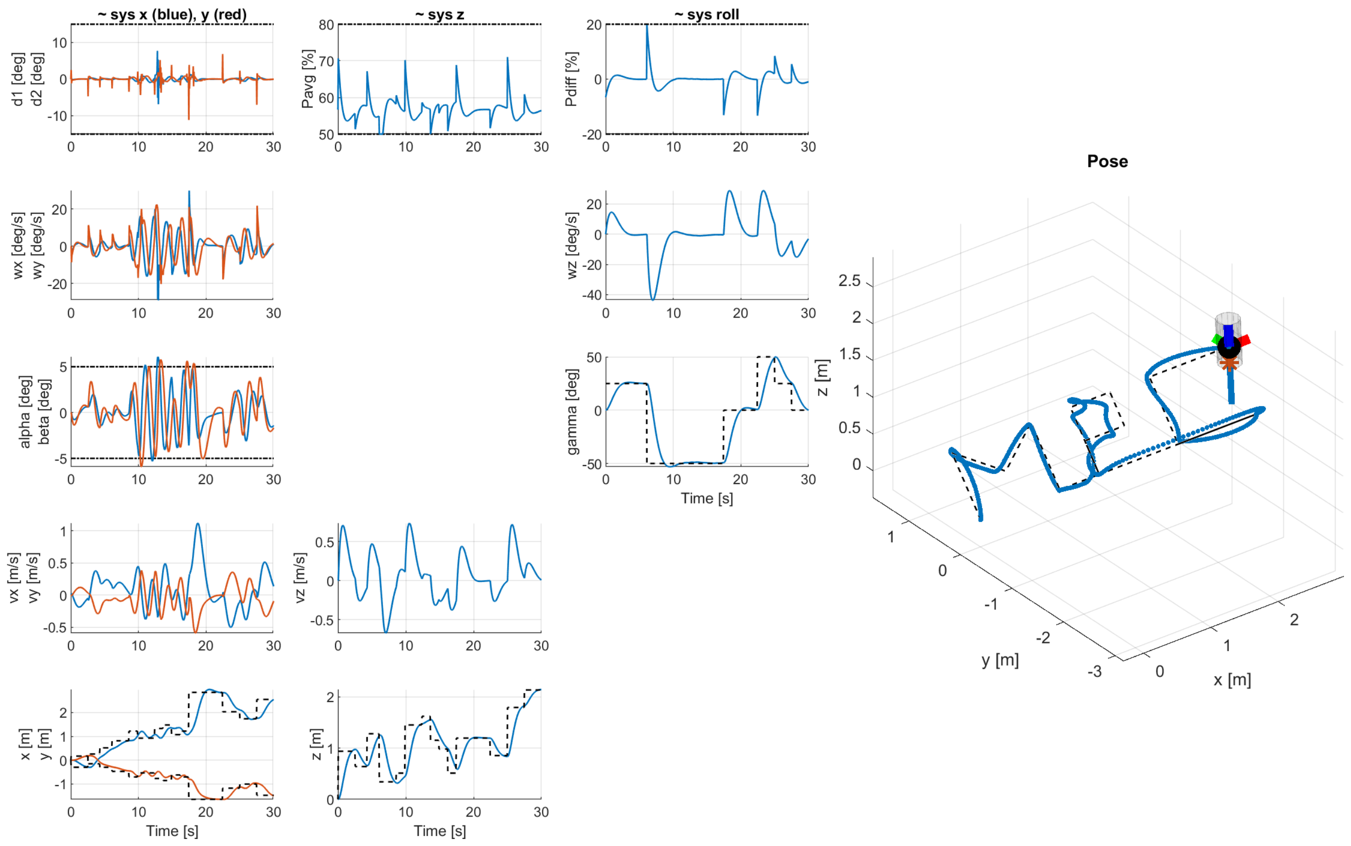


Figure 19: Merged lin. MPC in nonlinear simulation under $\gamma_{max} = 50°$

Conclusion

This report presents an approach to Thrust Vector Control for a rocket prototype through the design of MPC controllers to achieve different control objectives. Starting with the linearization of the system, a linear MPC controller is designed to make the rocket stabilize to the the origin while satisfying input and state constraints. The reference tracking problem is also tackled. Once through the design of a MPC controller for the linearized system including disturbance rejection or not. And once through the implementation of a non-linear MPC controller directly operating on the non-linear system's model. Overall, the design of these controllers is based on the tuning of parameters according to the simulation results obtained for the different cases. Therefore, one should be aware that the proposed solutions are only possible options among several others.

References

[1] Colin Jones. *Model Predictive Control: Mini-Project.* EPFL, 2021.