Project report

ME-425 Model Predictive Control

January 22, 2021

Name: Brunoro Guilain **Sciper: 246898**

Name: Kravtsov Denis **Sciper: 282379**

Name: Renard Titouan **Sciper: 272257**

Contents

1 Introduction - System Dynamics

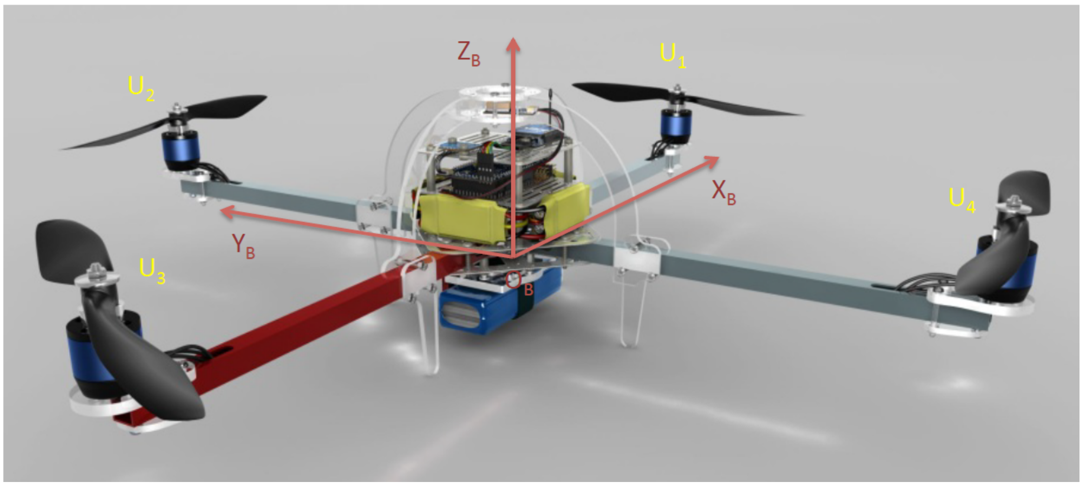

Figure 1: Quadcopter representation with axis and rotor labels

In this project we will develop MPC controllers for a simulated quadcopter UAV. We will first describe the dynamics of the system, which are modelled by a 12 states system.

$$\boldsymbol{x} = [\dot{\theta}, \theta, \dot{p}, p] \tag{1}$$

with p the center of mass position of the quadcopter in carthesian coordinates:

$$p = [x, y, z] \tag{2}$$

and θ its orientation:

$$\theta = [\alpha, \beta, \gamma] = [roll, pitch, yaw] \tag{3}$$

The input of the model is the thrusts $\boldsymbol{u}$ of the four rotors:

$$\boldsymbol{u} = [u_1, u_2, u_3, u_4]^T \tag{4}$$

Each rotor produces a force F_i and moment M_i according to:

$$F_i = k_F u_i, M_i = k_M u_i \tag{5}$$

with k_F and k_M the force and moment gain.

Hence the net body force F and body moments $(M_\alpha, M_\beta, M_\gamma)$ can then be expressed in terms of $\boldsymbol{u}$:

$$\begin{bmatrix} F \\ M_\alpha \\ M_\beta \\ M_\gamma \end{bmatrix} = \overbrace{\begin{bmatrix} k_F & k_F & k_F & k_F \\ 0 & k_F L & 0 & -k_F L \\ -k_F L & 0 & k_F L & 0 \\ k_M & k_M & k_M & k_M \end{bmatrix}}^{T} \boldsymbol{u} \tag{6}$$

with L the distance from the center of mass to the center of each rotors.

The acceleration of the center of mass is given by:

$$\ddot{O}_B = \begin{bmatrix} 0 \\ 0 \\ -m \cdot g \end{bmatrix} + F \boldsymbol{z}_B \tag{7}$$

With g the standard acceleration due to gravity and $\boldsymbol{z}_B$ the unit vector in the z direction of the body coordinate frame expressed in the world coordinates.

The angular dynamics are given by:

$$\dot{\omega} = I^{-1}(-\omega \times I\omega + \begin{bmatrix} M_\alpha \\ M_\beta \\ M_\gamma \end{bmatrix}) \tag{8}$$

with $\omega = \dot{\alpha}\boldsymbol{x}_B + \dot{\beta}\boldsymbol{y}_B + \dot{\gamma}\boldsymbol{z}_B$ the angular velocity of body coordinate frame with respect to the world coordinate frame, and I the inertia matrix of the quadrotor.

By combining 7 and 8 we get the dynamic equations of the system:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \tag{9}$$

From the equation 9 and numerical integration (we use the *ODE45* integrator from *Matlab*), we are able to simulate quadcopter dynamics for a given period of time.

2 Linearization and Diagonalization

2.1 Deliverable

From the system dynamics described by equation 9 we can derive a discrete linear approximation of the system dynamics around its steady state.

$$x^+ = A\boldsymbol{x} + B\boldsymbol{u} \tag{10}$$

Where x^+ represents the next x after one simulation step x, $A\boldsymbol{x}$ represents the open-loop dynamics and $B\boldsymbol{u}$ the influence of the inputs. Since the system's non-linearities are caused by the projection of the gravity vector on the quadcopter's reference frame the requirement for the linearization to give a good approximation is for the drone's pitch and yaw rotations to be relatively small.

As described in the equation 9 the matrix B describe the influence of the inputs on the state vector as a linear combination of rotor thrusts. As each motor has an effect on the force on the center of mass F and all three body moments (see equation 6), the system cannot be separated into independent subsystems when represented in that form.

To allow for the separation of our system into independent subsystems, we define a new input vector $\boldsymbol{v}$ through the change of variable $\boldsymbol{v} = T\boldsymbol{u}$ (T is defined in equation 6). Unlike $\boldsymbol{u}$, the new input vector $\boldsymbol{v}$ doesn't directly describe the thrust of the four rotors. Instead $\boldsymbol{v} = [F, M_\alpha, M_\beta, M_\gamma]^T$ describes the resulting forces and moments acting on the system which is now described by the following equation :

$$\dot{\boldsymbol{x}} = A \cdot \boldsymbol{x} + B \cdot T^{-1} \cdot \boldsymbol{v} \tag{11}$$

Forces and moments inputs allow for a more readable expression of the steady-state of our system since it is defined as $\dot{\boldsymbol{x}}_s = f(x_s, v_s) = 0$, the acceleration and velocity of the quadcopter must be equal to zero and the total forces and moments acting on the center of mass must be null. In our case, the only external force acting on the quadcopter to compensate to ensure steady state is the gravity, hence we have :

$$\begin{aligned} \boldsymbol{x_s} &= [0, 0, 0, 0, 0, \gamma, 0, 0, 0, x, y, z]^T \\ \boldsymbol{v_s} &= [m \cdot g, 0, 0, 0]^T \end{aligned} \tag{12}$$

The velocities, angular velocities and body orientations (except γ) must be equal to zero and the center of mass position $[x, y, z]$ could be placed anywhere as it isn't subject to any constraints and has no influence on system dynamics. Thus, the only steady-state input is a force compensating g.

The change of variable turns the matrix B into a new matrix $(B \cdot T^{-1})$ that clearly links each component of $\dot{\boldsymbol{x}}$ to a single force/moment. The matrix A remains unaffected by the change of variable as it is independent of inputs, nevertheless, it is useful to note that unsurprisingly, $\ddot{x}$ and $\ddot{y}$ are sensible to body orientation (β and α respectively) because the gravity vector is projected onto the vehicle's reference frame. We now have four independent linearized subsystems each sensible to a single force/moment of our input vector $\boldsymbol{v}$:

1. $[\ddot{\alpha}, \dot{\alpha}, \ddot{y}, \dot{y}] = f([\dot{\alpha}, \alpha, \dot{y}], M_\alpha) = A_\alpha \boldsymbol{x}_{\boldsymbol{\alpha}} + B_\alpha M_\alpha$
2. $[\ddot{\beta}, \dot{\beta}, \ddot{x}, \dot{x}] = f([\dot{\beta}, \beta, \dot{x}], M_\beta) = A_\beta \boldsymbol{x}_{\boldsymbol{\beta}} + B_\beta M_\beta$
3. $[\ddot{\gamma}, \dot{\gamma}] = f([\dot{\gamma}, \gamma], M_\gamma) = M_\gamma \boldsymbol{x}_{\boldsymbol{\gamma}} + B_\gamma v_\gamma$
4. $[\ddot{z}, \dot{z}] = f([\dot{z}], F) = A\boldsymbol{x} + BF$

And since $\boldsymbol{u} = T^{-1}\boldsymbol{v}$ we can use those subsystems to implement a control law.

3 Design MPC controllers for each Sub-System

For each of the 4 subsystems (x, y, z and yaw), we construct a controller by solving an optimization problem of the form:

$$\begin{aligned} J^*(x) = \min_{x,u} & \left[\sum_{i=0}^{N-1} I(x_i, u_i) + x_N^T Q_f x_N \right] \\ s.t.\ x_{i+1} &= Ax_i + Bu_i \\ Fx_i &\leq f \\ Mu_i &\leq m \\ x_N &\in X_f \end{aligned} \tag{13}$$

Where X_f is the terminal set, Q_f is the terminal cost (computed by an LQR controller) and F, f, M and m are the system (state and input) constraints. The state x_0 can be set arbitrarily. As for the horizon N, we chose it so that the receding horizon covers the settling time for step responses of $2m/45°$:

$$N = \frac{T_{settling}}{T_{sampling}} = 40 \tag{14}$$

Which we tend to think is a sensible heuristic.

3.1 Deliverable

We use the following quadratic cost function:

$$I(x, u) = x^T Q x + u^T R u \tag{15}$$

The parameters Q and R require tuning. Essentially the tuning of Q and R amounts to a trade-off between performance and the size of the terminal set. The smaller the terminal set is, the higher the likelihood to end up in an infeasible state is. However the task to track a path doesn't require the drone to have a particularly large feasible set. Therefore, we make the decision to design a controller that prioritizes following a path quickly and precisely. To do so, we assign higher weights to the velocity and position components in the Q matrix, while giving angle components a lower weight. As for the choice of the R matrix, in the absence of any specification given in the guidelines on input's cost importance, we chose it to be equal to one.

$$Q_{x,y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 50 \end{bmatrix} \quad Q_z = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix} \quad Q_{yaw} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad R = 1 \tag{16}$$

Additionally we define matrices F, f, M, m so that :

$$Fx \leq f \qquad and \qquad Mu \leq m \tag{17}$$

Because of the linearization, we need to ensure that the following state constraints are met :

$$F_{x,y} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} \qquad f_{x,y} = \begin{bmatrix} 0.035 \\ 0.035 \end{bmatrix} \tag{18}$$

We model the fact that our motors have limited thrust output by a constraint on the vector $\boldsymbol{u}$ where for each each component : $0 \leq u_i \leq 1.5$. After the change of variable from $\boldsymbol{u}$ to $\boldsymbol{v}$, and the separation into independent subsystems, this is reflected by the following input constraints on the inputs of each subsystem:

$$M_{x,y} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad m_{x,y} = \begin{bmatrix} 0.3 \\ 0.3 \end{bmatrix} \quad M_z = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad m_z = \begin{bmatrix} 0.3 \\ 0.2 \end{bmatrix} \quad M_{yaw} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad m_{yaw} = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix} \tag{19}$$

As for the yaw and z subsystems, since they do not require any state constraints, they do not require matrices F_z, F_{yaw}, f_z and f_{yaw} to be defined.

The terminal constraints enable us to compute terminal sets (figures 2 ,3 ,4, 5), 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).

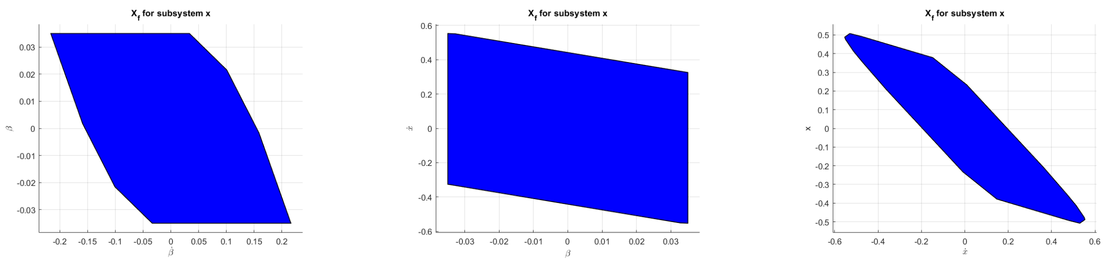

Figure 2: Terminal set for system $\boldsymbol{sys\_x}$

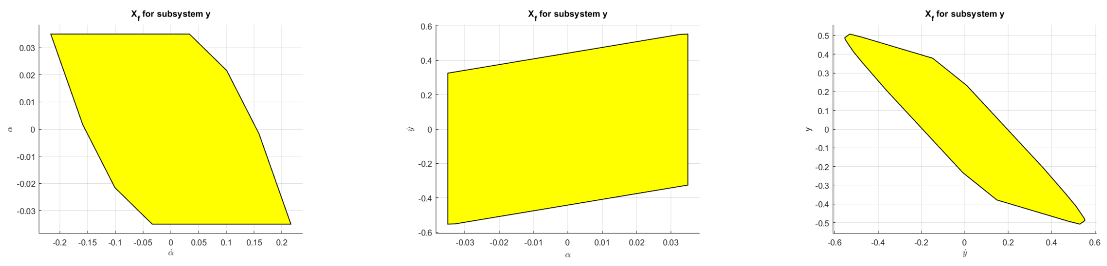

Figure 3: Terminal set for system $\boldsymbol{sys\_y}$

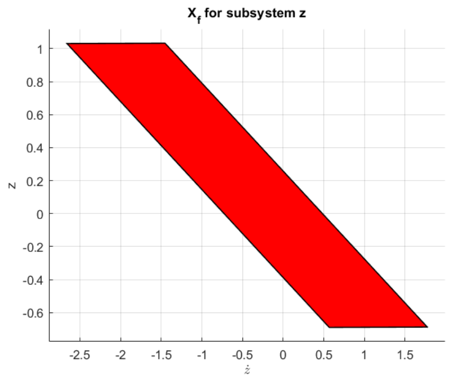

Figure 4: Terminal set for system ***sys_z***

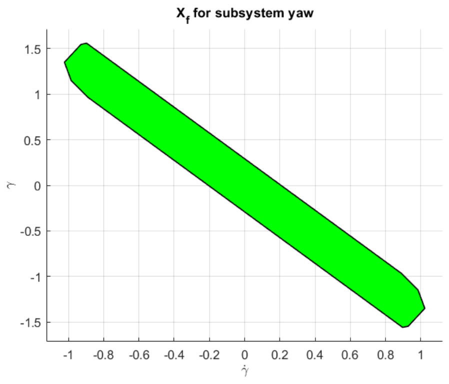

Figure 5: Terminal set for system ***sys_yaw***

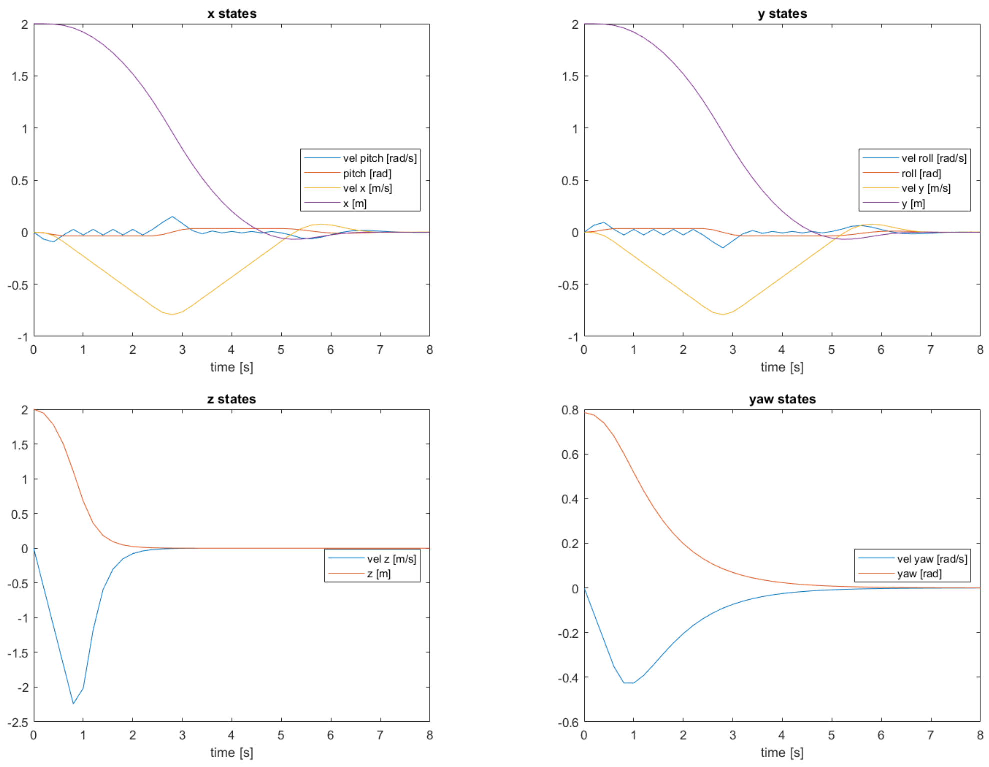

Figure 6: Subsystems evolution for regulation

Simulating the independent (x,y,z and yaw) subsystems's response to initial disturbances gives us the following results for the different subsystems, for a constant reference tracking and, with a starting point two meters away from the origin for x, y and z, and 45°for yaw: We observe that all our subsystems converge to zero in under 8 seconds, except for $\boldsymbol{z}$, which converges much faster due to the fact, that all 4 motors contribute to the movement in z direction. A slight overshoot in x and y directions is acceptable as the constraints are not violated and because of the fact that the overshoot in these directions is not critical in our application.

3.2 Deliverable

In this part, we introduce a constant reference to track. Thus, the adjustments that we implement mainly concern the delta formulation of the optimization problem stated above :

$$\begin{aligned} x_s &= Ax_s + Bu_s \\ r &= Cx_S \\ Fx_s &\leq f \\ Mu_s &\leq m \end{aligned} \tag{20}$$

Where r is the reference output. The delta formulation (difference between actual and target states) is given by : $\Delta x = x - x_s$ and $\Delta u = u - u_s$. We then insert these formulations into the set of equations 13. As for the system dynamics, they remain unchanged as constant reference doesn't affect system constraints. The following state evolution obtained when simulating tracking for the different subsystems independently is represented in the figure 7. We obtain the same performance as for regulation.

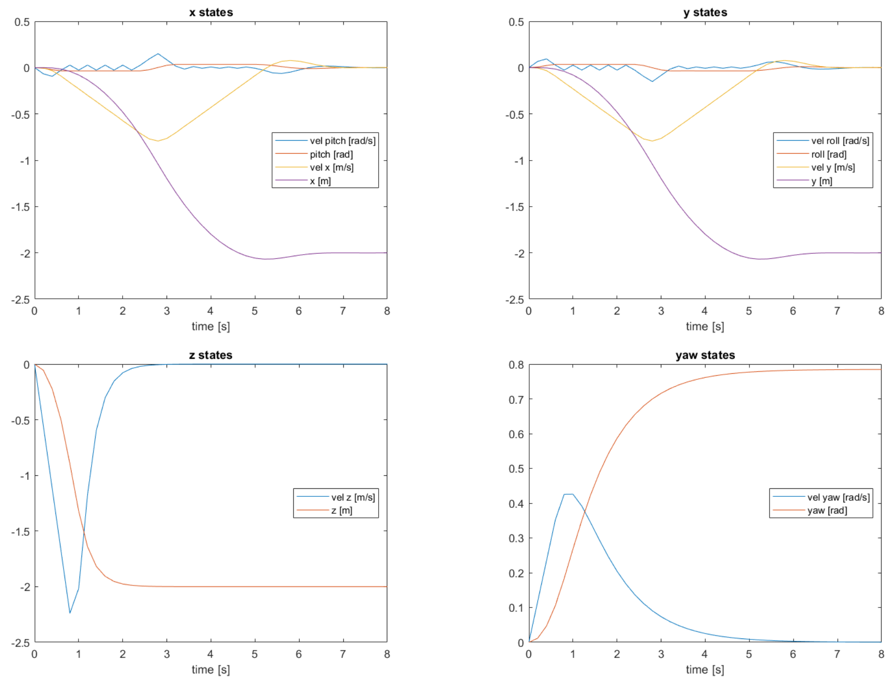

Figure 7: State evolution with constant tracking reference

4 Simulation with Nonlinear Quadcopter

4.1 Deliverable

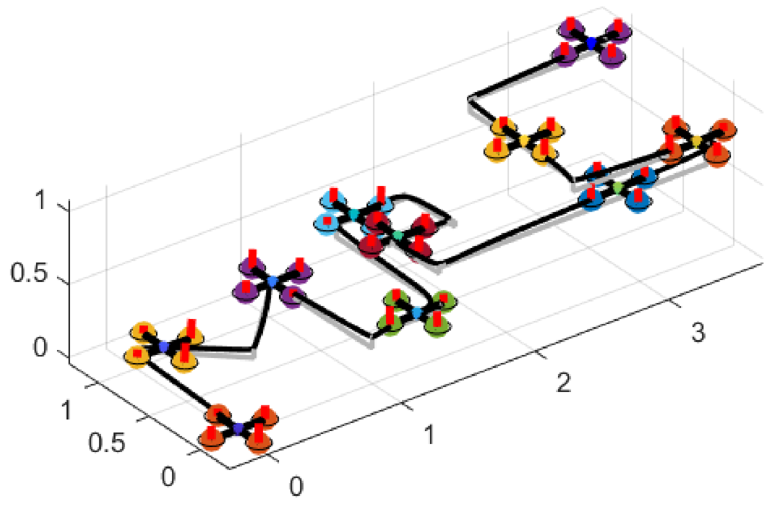

Figure 8: Tracking path, in grey line the reference trajectory, in black line the quadcopter trajectory

The results presented in the figure 8 show that the quadcopter is able to follow the required path. The tuned parameters seem to offer good performance as there is little error between the actual path and the desired one. By looking at the evolution of states in figure 9, we can observe that the state z follows nearly perfectly the reference, and that the states x and y have a rather small delay compared to their respective references.

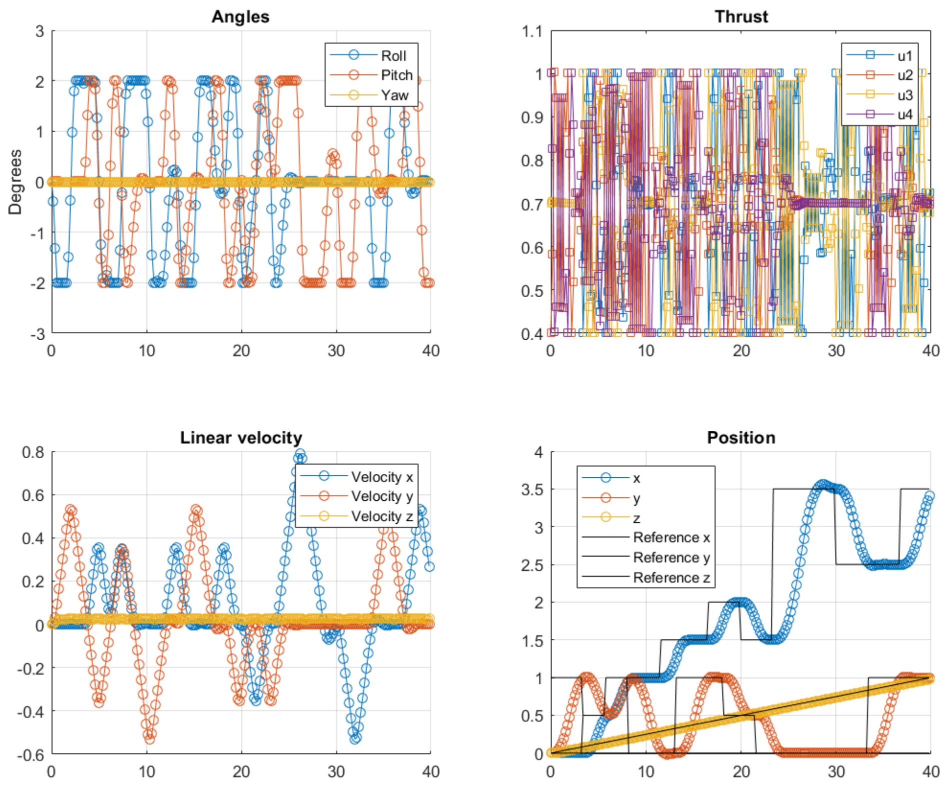

Figure 9: States evolution for path tracking

5 Offset-Free Tracking

5.1 Deliverable

In this part, the dynamics of the system in z-direction are changed to:

$$x^+ = Ax + Bu + Bd \tag{21}$$

with d an unknown constant disturbance simulating a mass variation for the quadcopter.
We are trying now to update the controller in order to reject this disturbance and track the set point references with no offset.

To achieve this we design a state and disturbance estimator that we will use to find the control inputs to remove the offset. The estimator is based on the augmented model:

$$\begin{bmatrix} \hat{x}_{k+1} \\ \hat{d}_{k+1} \end{bmatrix} = \begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ \hat{d}_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k + \begin{bmatrix} L_x \\ L_d \end{bmatrix} (C\hat{x}_k + C_d\hat{d}_k - y_k) \tag{22}$$

with $\hat{x}$ and $\hat{d}$ the estimates of the state and disturbance.

The error dynamics are then given by:

$$\begin{bmatrix} x_{k+1} - \hat{x}_{k+1} \\ d_{k+1} - \hat{d}_{k+1} \end{bmatrix} = \left(\begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} + \begin{bmatrix} L_x \\ L_d \end{bmatrix} \begin{bmatrix} C & C_d \end{bmatrix} \right) \begin{bmatrix} x_k - \hat{x}_k \\ d_k - \hat{d}_k \end{bmatrix} \tag{23}$$

With $\begin{bmatrix} A & B \\ 0 & I \end{bmatrix} = \bar{A}$, $\begin{bmatrix} L_x \\ L_d \end{bmatrix} = \bar{L}$ and $\begin{bmatrix} C & C_d \end{bmatrix} = \bar{C}$

The aim here is to choose $\bar{L}$ such that error dynamics are stable and converge to zero (i.e the estimator is stable). To do this we simply hand tune the pole placement of the observer loop system $\bar{A} + \bar{L}\bar{C}$. Given that $B_d = B$ and $C_d = 0$ we find the following pole values $[0, 0.05, 0.1]$ that ensure us fast dynamics, stability and error minimization.

The values of N and the Q, R matrix remain identical to the previous one found in section 3.1 because our objective stay the same.

Note that since we used here an observer to estimate the offset and the state of the system, we can no longer ensure constraint satisfaction and thus the terminal set is not computed.

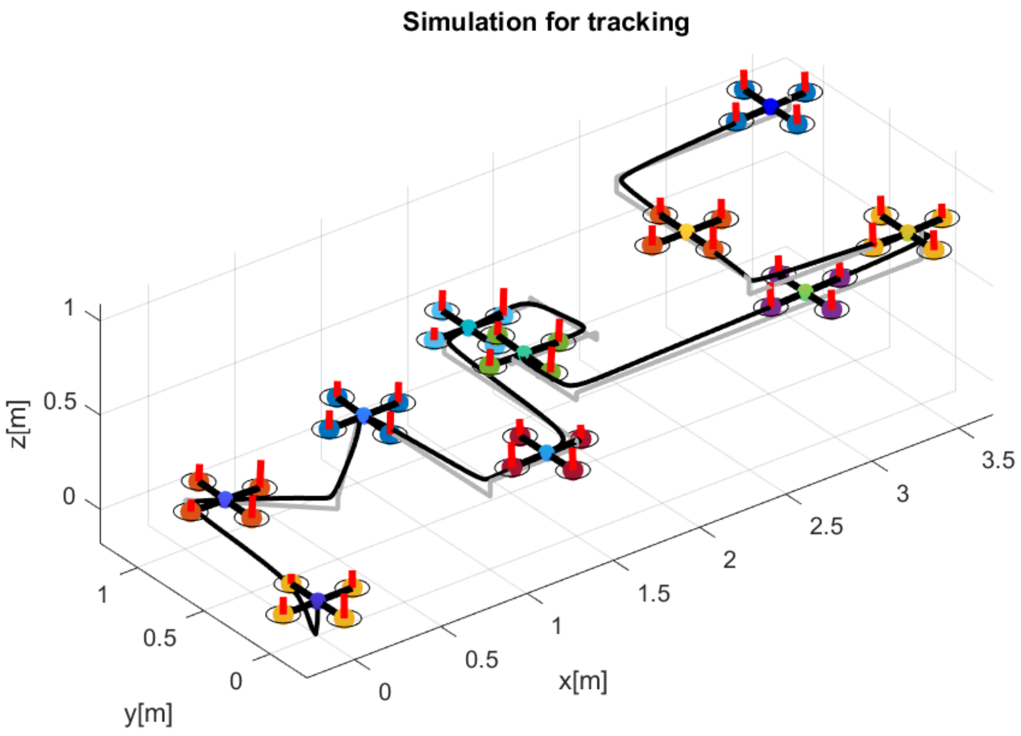

Figure 10: Tracking path with disturbance in z-direction, in grey line the reference trajectory, in black line the quadcopter trajectory

The results presented on figure 10 indicate that the quadcopter is able to follow the desired path even in presence of a bias $= -0.1$. It is useful to note that the design of the observer $\bar{L}$ will mainly influence the amplitude and rise-time of the initial drop in z direction that we can notice on the previous figure. Low values of the poles will tend to a shorter initial drop of the quadcopter.

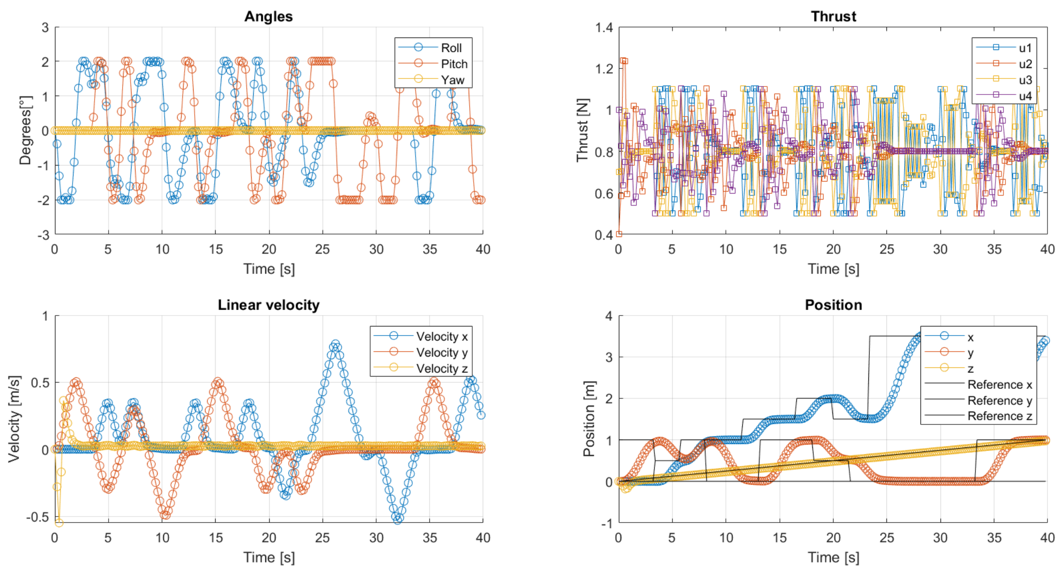

Figure 11: States evolution for the offset-free tracking path

6 Nonlinear MPC (Bonus)

6.1 Deliverable

In this part we design a completely new controller by dropping the linear approximation from part 2 and designing a non-linear MPC (NMPC) able to ensure good tracking performance on the same problem as the one posed in parts 3,4 and 5. As in part 5 we will drop the terminal set constraint.

We construct a tracking controller by solving the following non-linear optimization problem :

$$\begin{aligned} J^* = \min_{x,u} \left[\sum_{i=0}^{N-1} (x_i - x_{ref})^T \cdot Q \cdot (x_i - x_{ref}) + u_i^T R u_i \right] \\ s.t.\ x_{i+1} = F_{\text{discr}}(x_i, u_i) \\ M \cdot u_i \leq m \\ x_0 = X_0 \end{aligned} \tag{24}$$

We use the Runge-Kutta 4 method to integrate the system dynamics, for a given sampling period T_s it yields much more exact estimations than the trivial approach (Euler's method):

$$\begin{aligned} k_1 &= f(x,u)y \\ k_2 &= dotf(x + \frac{T_s}{2} * k_1, u) \\ k_3 &= f(x + \frac{T_s}{2} * k_2, u) \\ k_3 &= f(x + T_s * k_3, u) \\ F_{discr}(x,u) &= x + \frac{T_s}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \tag{25}$$

Because we rely on non-linear optimization our system doesn't require linearization constraints on pitch and roll to produce accurate predictions, this leaves us with no state constraints, the only constraints that remain are input constraints. The non-linear nature of the model also makes it impossible to separate the system into independent subsystems (as we did in part 2), this makes any change of variable pointless so we directly use the motor thrust vector $\boldsymbol{u}$ as our input vector. The input constraints $M \cdot u_i \leq m$ are therefore now directly expressed in term of motor thrusts (instead of as moments and forces as in deliverables 3.1, 3.2 and 5.1) :

$$\overbrace{\begin{bmatrix} 1 \\ -1 \end{bmatrix}}^{M} \cdot u_i \leq \overbrace{\begin{pmatrix} 1.5 \\ 0 \end{pmatrix}}^{m} \tag{26}$$

We choose a horizon larger than the settling time of our system : $N = 40$ steps, which corresponds a 8 second prediction (for the same reasons as in deliverable 3.1). This ensures good tracking performance and limited overshoot. In a non-simulated scenario the horizon length would be limited by the speed of the hardware available on the drone to run the controller but in this example we have no specification in that regards so there is no reason to limit ourselves. We use our cost matrices to tune the aggressivity of the tracking controller on the different tracked state variables.

$$J = \sum_{i=0}^{N-1} \begin{pmatrix} x - x_{ref} \\ y - y_{ref} \\ z - z_{ref} \\ \gamma - \gamma_{ref} \end{pmatrix}^T \overbrace{\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 30 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}^{Q} \begin{pmatrix} x - x_{ref} \\ y - y_{ref} \\ z - z_{ref} \\ \gamma - \gamma_{ref} \end{pmatrix} + \vec{u}^T \overbrace{4\boldsymbol{I}^T}^{R} \vec{u} \tag{27}$$

Note that the state vector $\boldsymbol{x}$ is actually contains 12 components, the equation 27 is a simplified representation where we removed all 0-cost states from $\boldsymbol{x}$ for readability. Our controller is tuned to be significantly more aggressive on the z variable than on any other, we made this decision after observing that our controller significantly under-performed on z-variable tracking compared to other variables. Our R matrix is simply $4 \cdot \boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix. It provides a way to control the "throttle aggressivity" of our controller, to reduce overshoot (note that if limiting overshoot was a critical requirement for our controller we could enforce it with constraints), and to smoothen the trajectory.

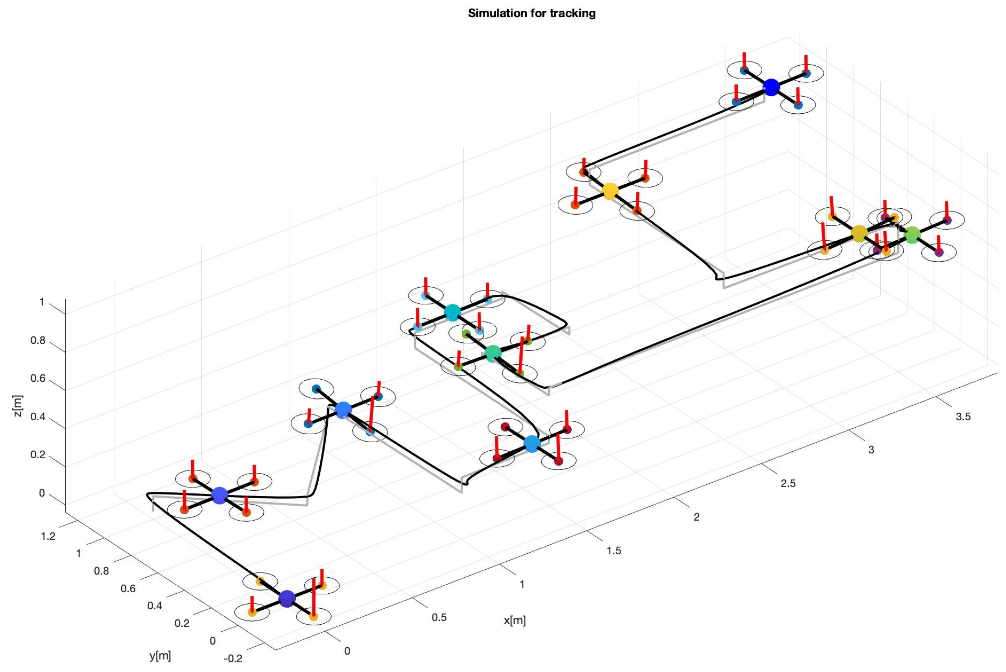

Figure 12: Tracking path using non-linear MPC

We observe that the non-linear controller attains good tracking performance with little overshoot and a settling time of around $7s$ when tracking a $2m$ step response in x,y, or z translation. The main difference in behavior between the non-linear controller and the linear controllers is the absence of angle constraints, this allows the non-linear controller to reach roll and pitch angles up $8°$ whereas the linear controller is constrained a max of $2°$ of roll or pitch because of the linearization. This allows for a faster settling time and for the drone to reach higher linear velocities.

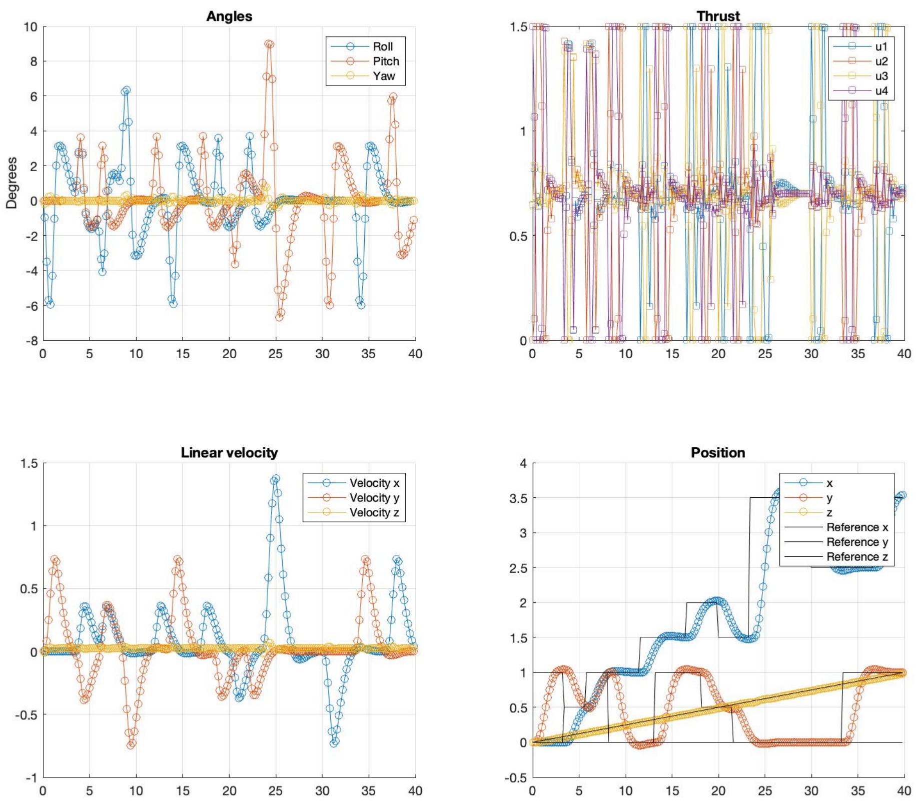

Figure 13: States evolution for the non-linear MPC tracking

Additionally the NMPC controller has a significantly larger feasible set, given enough time it is able to recover from practically any state. This makes it much more robust to external and unmodelled perturbations. We illustrate this capability with a rather extreme example. We make the controller recover from a starting point $\boldsymbol{x_0}$ where the quadcopter is totally inverted (180° pitch rotation) and we observe our controller is able to successfully recover and bring back the drone to a tracking point (whereas a linearized MPC would not even be able to compute a control law in that situation). The evolution of the attitude and position of the drone when recovering are respectively represented in figure 14 and figure 15. The drone does take a large time to recover (it falls more than $100m$ before properly recovering) but this is not a practical or realistic example but rather to illustrate the ability of the NMPC to compute a usable control law in essentially any possible state.

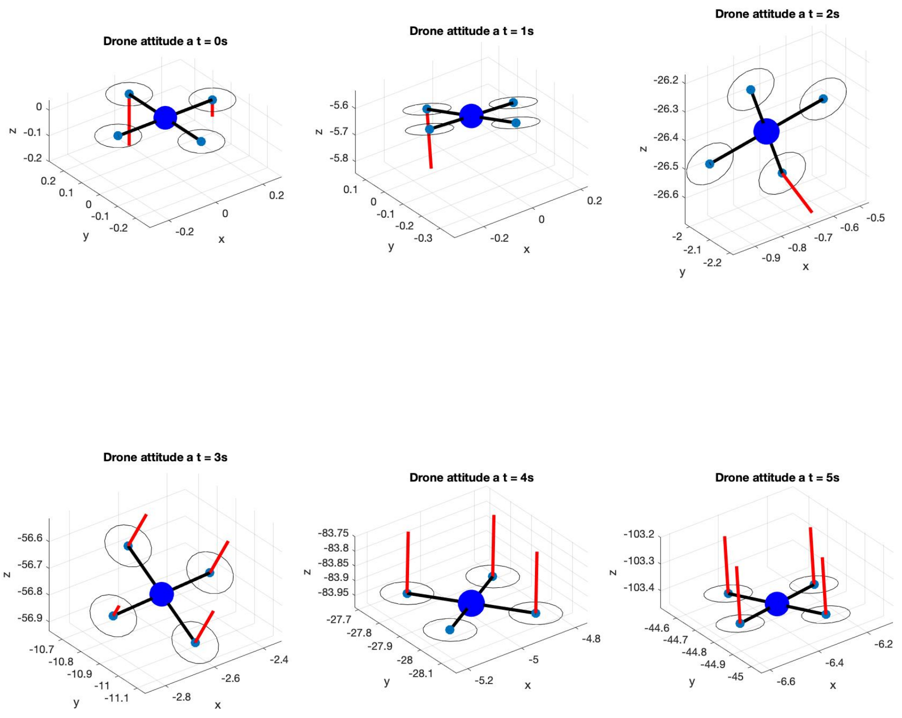

Figure 14: Evolution of the attitude of the drone when trying to recover from a 180° rotation on the pitch axis. The red segments represent the thrust value of each motor.

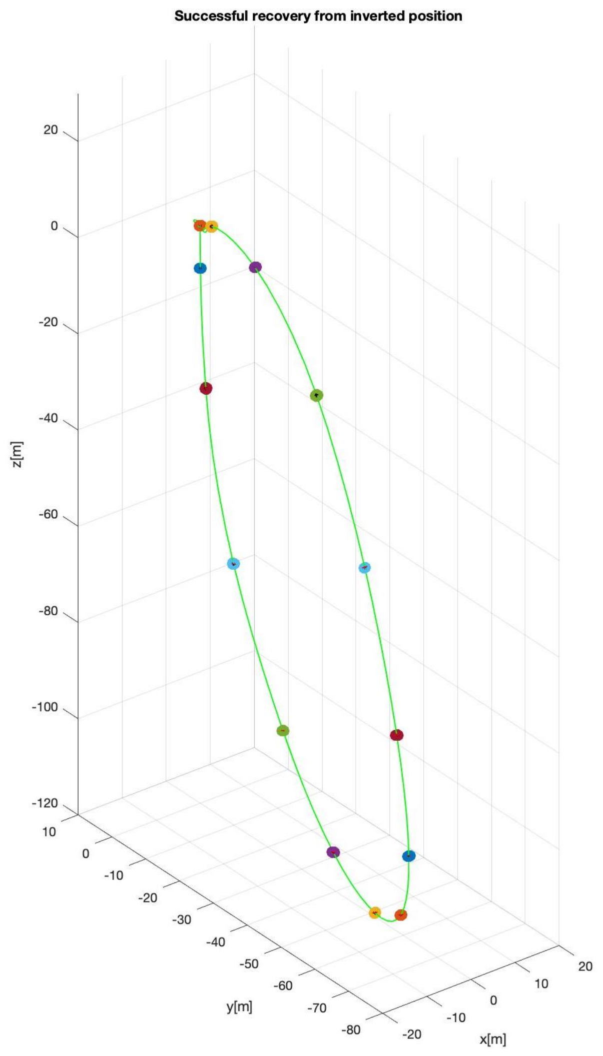

Figure 15: Evolution of the position of the drone when trying to recover from a 180° rotation on the pitch axis.