Model Predictive Control
Automatic Control Laboratory

Development of an MPC controller to fly a quadcopter

Teacher :

Colin Jones

Student - Group 46:

Omari Lamyae - 316803
Lefebure Nicolas - 258458
Du Pasquier Grégoire - 293360

January 12, 2020

Contents

Introduction

In the scope of the ME-425 course, we developed a controller for a quadcopter using a Model Predictive Control (MPC) approach. Different optimization strategies were established depending on the goal being achieved by the quadcopter. In sections 1 and 2, the system's inputs are described along with the state variables and how this representation can be linearized and decoupled. In section 3, the design of the MPC controllers is shown for the four sub-systems in both regulation and tracking. Section 4 shows a simulation of the controllers when the non-linear quadcopter tracks the given "MPC" path. In section 5, a disturbance was added to the system and the controller was expanded to achieve Offset-Free Tracking. Finally, in section 6, an attempt at designing a nonlinear MPC for the quadcopter is presented.

1 System description

The quadcopter dynamics is derived from a 12-state representation, given by the vector:

$$\mathbf{x} = \begin{bmatrix} \dot{\alpha} & \dot{\beta} & \dot{\gamma} & \alpha & \beta & \gamma & \dot{x} & \dot{y} & \dot{z} & x & y & z \end{bmatrix}^T$$

where x, y and z are the Cartesian coordinates of the center of mass of the quadcopter, while α, β and γ are respectively its roll, pitch and yaw. The inputs of the system are the thrusts from each of the four rotors:

$$\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^T$$

For a better representation of the input, a transformation T can be applied to directly express the input as the vertical thrust and the three moments acting on the quadcopter:

$$\mathbf{v} = \begin{bmatrix} F \\ M_\alpha \\ M_\beta \\ M_\gamma \end{bmatrix} = \begin{bmatrix} k_F & k_F & k_F & k_F \\ 0 & k_F L & 0 & -k_F L \\ -k_F L & 0 & k_F L & 0 \\ k_M & -k_M & k_M & -k_M \end{bmatrix} \mathbf{u} = T\mathbf{u}$$

where F is the thrust in the vertical direction, and M_α, M_β and M_γ are respectively the moments of roll, pitch, and yaw. L is the distance from the center of mass to the center of the rotors, assuming a constant distance for all four rotors. k_F and k_M are some proportionality constants between the inputs and the forces.

With the states and inputs conveniently defined, the equations of motion can be derived as:

$$\ddot{O}_B = \begin{bmatrix} 0 \\ 0 \\ -m \cdot g \end{bmatrix} + F\mathbf{z}_B \tag{1}$$

$$\dot{\omega} = \mathcal{I}^{-1} \left(-\omega \times \mathcal{I}\omega + \begin{bmatrix} M_\alpha \\ M_\beta \\ M_\gamma \end{bmatrix} \right) \tag{2}$$

Here, $\ddot{O}_B$ is the origin attached to the center of mass of the quadcopter. $\mathbf{z}_B$ is the unit vector pointing in the z-direction of the body coordinate frame expressed in the world coordinates. m and g are respectively the mass of the robot and the acceleration due to gravity. The vector ω is the angular velocity of the body coordinate frame with respect to the world coordinate frame: $\omega = \dot{\alpha}\mathbf{x}_B + \dot{\beta}\mathbf{y}_B + \dot{\gamma}\mathbf{z}_B$. $\mathcal{I}$ is the inertia matrix of the quadcopter, given by:

$$\mathcal{I} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix}$$

From the dynamics equations, the system can be written in a state-space representation:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \tag{3}$$

2 Linearization and Diagonalization

In the first part of this project, a linearized version of the quadcopter is being controlled. The linearization is done using the so-called "small-signal linearization", where the Jacobian of the original system is used to approximate the dynamics around a nominal point, typically the equilibrium or in the case the quadcopter in a flat and stationary position. The linearized dynamics becomes:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} = A\mathbf{x} + BT^{-1}\mathbf{v} \tag{4}$$

By looking at the resulting matrices in *Deliverable_2_1.m*, it can be seen that the system breaks down into four independent/non-interacting systems, each dependent on only one input force/moment.

For instance, looking at the state variables z and $\dot{z}$, they are only related to the input F. Indeed, line 9 (state variable $\dot{z}$) of matrix A is only made of zero and line 9 of matrix B has only a non-zero component in column 1 (input F). The 12^{th} line (state variable z) has only a non-zero component in column 9 (state variable $\dot{z}$) and line 1 of matrix B only contains zeros. Thus, these three variables (z, $\dot{z}$ and F) do not interact with the other variables but only with themselves. The same reasoning goes for the other sub-systems.

Each sub-system can also be identified by computing the controllability matrix for every input variable. For example, by looking at the result in *Deliverable_2_1.m*, the controllability matrix for the roll (M_α) has four independent columns, meaning that the input M_α can control four state variables, as described above. The independent lines correspond to index: 1, 4, 8 and 11 which corresponds respectively to state variables $\dot{\alpha}$, α, $\dot{y}$ and y. The same reasoning can be made for the other three inputs F, M_β and M_γ.

The resulting four sub-systems are: roll moment input M_α to position y, pitch moment input M_β to position x, total thrust input F to height z and yaw moment input M_γ to yaw angle γ.

If the linearization is extended around any equilibrium point, then the steady-state solution is given by the following expression:

$$\mathbf{x}_s = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \bar{\gamma} & 0 & 0 & 0 & \bar{x} & \bar{y} & \bar{z} \end{bmatrix}^T$$
$$\mathbf{u}_s = \bar{u} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T$$

To make sure that the decoupling of the system is still valid for any other steady-state solution, the linearization of the system must be independent from the steady-state solution. By looking at equation 1 and 2 we see that the state variables x, y, z and γ do not appear in the equations, only their derivatives do. This means that these variables will not appear in the state-space representation in equation 3. Thus, the matrices A and B will not be dependant on $\bar{x}$, $\bar{y}$, $\bar{z}$ and $\bar{\gamma}$:

$$\mathbf{A} = \left.\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\bar{\mathbf{x}} \\ \mathbf{u}=\bar{\mathbf{u}}}} \neq A(\bar{x}, \bar{y}, \bar{z}, \bar{\gamma}) \tag{5}$$

$$\mathbf{B} = \left.\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\right|_{\mathbf{x}=\bar{\mathbf{x}}} \neq B(\bar{x}, \bar{y}, \bar{z}, \bar{\gamma}) \tag{6}$$

3 Design MPC Controllers for each sub-system

Now that the dynamics of every sub-system is set up, their respective MPC controllers can be designed. First, a regulator is built in section 3.1, then reference tracking is added to the controller in section 3.2.

3.1 MPC Regulators

A regulator is a controller that drives the system's state to a steady-state solution. In the scope of this project, the steady-state solution that is being attempted is $\bar{\mathbf{x}} = \mathbf{0}$. In other words the controller regulates the system to the origin.

3.1.1 Design procedure

Since an infinite-horizon prediction cannot be considered when solving an optimization problem, the MPC problem was set up like the general finite-horizon optimal control problem with a terminal set and terminal constraint.

$$V_N^*(x_0) = \min \sum_{i=0}^{N-1} I(x_i, u_i) + V_f(x_N) \tag{7}$$

$$\begin{aligned} \text{s.t. } & x_{i+1} = f(x_i, u_i) \\ & (x_i, u_i) \in \mathbb{X}, \mathbb{U} \end{aligned}$$

Adding a terminal constraint provides a sufficient condition for stability, as it ensures that the system will be within a stabilizing set after N steps. This stabilizing set approximates the "tail" of the constraints, which is necessary due to the truncation of the horizon. It was chosen as the maximum invariant set $\mathcal{X}_f$ of the system, and was computed using an LQR controller, as described in Section 3.1.3. The LQR controller is then used as the terminal controller $\kappa_f(x)$, since it ensures constraint satisfaction and invariance in the terminal set $\mathcal{X}_f$.

The stage cost $I(x_i, u_i)$ was defined as the quadratic cost function below, with $Q \succ 0$ and $R \succ 0$.

$$I(x_i, u_i) = x_i^T Q x_i + u_i^T R u_i \tag{8}$$

It ensures that the cost function is a positive definite function, which is a requirement for stability.

And the terminal cost as:

$$V_f(x_N) = x_N^T Q x_N \tag{9}$$

This final cost serves as an approximation of the "tail" of the cost for $N > i$, and is also a Lyapunov function in the terminal set $\mathcal{X}_f$.

The state and input constraints were simply taken from the project handout and put into the proper linear form for each of the four independent systems. For instance, the system defining the x-location of the drone has as input the pitch moment M_β and the four states $\dot{x}, x, \dot{\beta}, \beta$. But since these are constrained by: $-0.3 \leq M_\beta \leq 0.3$ and $|\beta| \leq 2°$, they must be rewritten in the form:

$$\begin{aligned} \mathbb{X} &= \{x \mid Fx \leq f\} \\ \mathbb{U} &= \{u \mid Mu \leq m\} \end{aligned}$$

Where:

$$M = [1 \quad -1]^T$$
$$m = [0.3 \quad 0.3]^T$$
$$F = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}$$
$$f = [0.035 \quad 0.035]^T$$

The same procedure was applied to the other three systems, but for the sake of redundancy they have been omitted here.

Putting all of the above together we have:

$$u^*(x) := \text{argmin} \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i + x_N^T Q x_N \tag{10}$$

$$\begin{aligned} \text{s.t.} \quad x_0 &= x \\ x_{i+1} &= Ax_i + Bu_i \\ (x_i, u_i) &\in \mathbb{X}, \mathbb{U} \\ x_N &\in \mathcal{X}_f \end{aligned}$$

The theorem for stability and feasibility for MPC presented on page 38 of Lecture 5 states that if the problem satisfies the three standing assumptions listed:

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 $\mathcal{X}_f$
3. Terminal cost is a continuous Lyapunov function in the terminal set $\mathcal{X}_f$

Then, the closed-loop system under the MPC control law $u^*(x)$ is stable and the system $x_+ = Ax + Bu^*(x)$ is invariant in the feasible set $\mathbb{X}_N$. Since all three points have been shown above, the system is stable and feasible and it can be implemented for regulation tasks.

3.1.2 Tuning parameters

The optimization problem established previously involves the use of hyper-parameters that have to be tune by hand. The performance criteria function defined can be seen as a cost function that characterises the outlay of the system up to a particular horizon N. The parameters Q and R can be used as design parameters to give different weights to the state variables and the control signals. The larger a weight is, the more penalty is given to that specific parameter. The matrix Q represents the importance associated to a discrepancy in the states of the system, i.e. if $\mathbf{x}$ is not close to its expected value, while R represents to the amount of energy $\mathbf{u}$ that needs to be spent to move the system to its next state. The prediction horizon N is how far ahead the model computes a prediction.

In this project, the parameters Q, R, and N are set according to the given performance requirements. For example, the settling time of every sub-system should be around eight seconds when starting at a stationary position two meters from the origin (for x, y and z), or stationary at 45 degrees for yaw. Our objective is to tune the hyper-parameters to ensure a settling time of 8 seconds, while using as little energy as possible and finding an accurate prediction horizon.

We started by finding a horizon for each subsystem capable of stabilizing the system. A few more horizon steps were added so that the solution is not subject to destabilization when changing the

hyper-parameters. The matrices Q and R were then set to the identity as a starting point. However, Q is not really the identity matrix, since we chose to penalize only the output states and not their rate of change (i.e. penalize x rather than $\dot{x}$). Therefore, all entries of Q are zero except for the last one which is set to 1 ($Q(n,n) = 1$). At this time, we noticed that every sub-system already achieved a settling time inferior to eight seconds due to an unnecessary surplus of energy being used by the controller. To reduce this excess, the weights in R were increased for each subsystems to bring the settling time closer to eight seconds. Figure 1 shows the difference between the initial input and the optimized input. To determine the energy savings the total work made by the robot was calculated for each control sequence and is shown in Table 1. Finally, the prediction horizon was adapted to be as small as possible (+ a few more steps just to be safe). The table 1 summarizes all of the optimized hyper-parameters for each system.

system	x/M_β	y/M_α	z/F	γ/M_γ
settling time [s]	7.4193	7.4193	7.3330	7.5680
Work [J]	0.0503	0.0503	0.0361	0.0251
Q(n,n)	1	1	1	1
R	50	50	400	20
N	15	15	10	10

Table 1: Optimized hyper-parameters for the control sequence

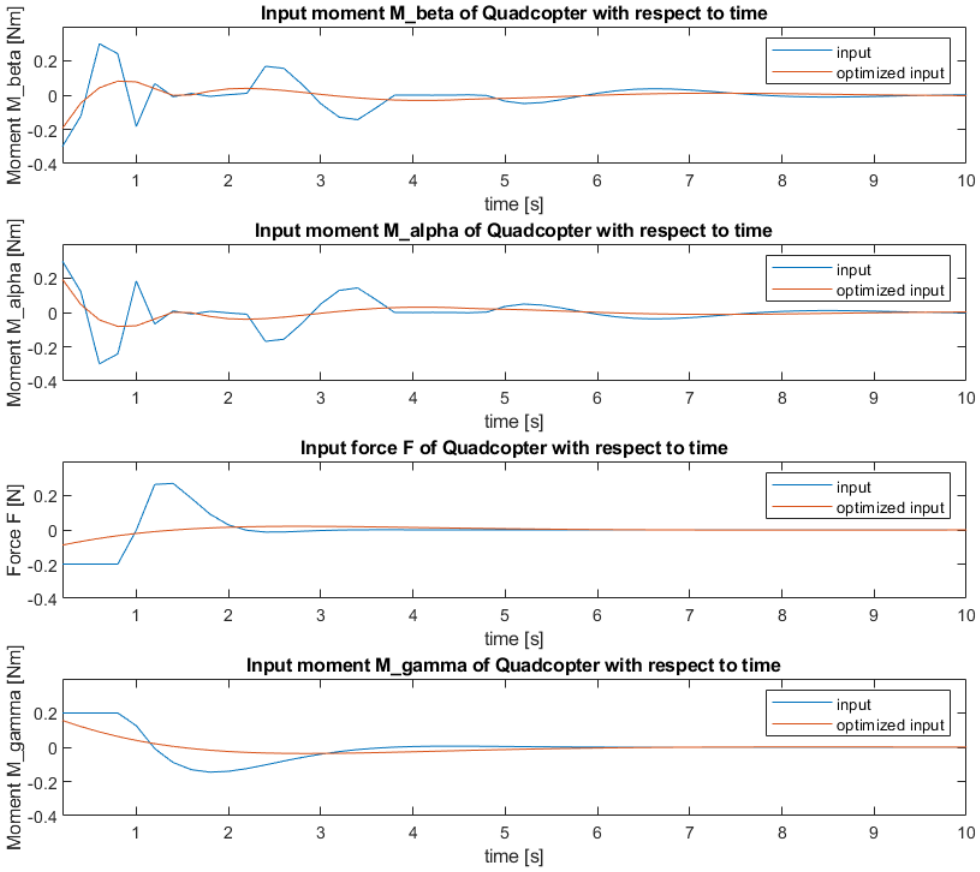

Figure 1: Plot of the initial inputs and the optimized inputs for the Regulator

3.1.3 Terminal invariant set

As shown in Section 3.1.1, the terminal set is computed using an LQR controller. This choice was motivated by the simplicity of such controller and the fact that it provides a stabilizing control input in its feasible region. Once the controller gain $\kappa(x)$ is computed, it is used to find the maximum invariant set of the system. This is done by setting the initial set as the feasibility region (given by the constraints of the system), and computing the preset of that region given the LQR controller. The intersection of this new preset with the original feasible region defines a new set, which can again be used to compute its preset and intersection. This process is repeated until a set remains equal to its preset, which infers the property of invariance. In fact, this is the maximum invariant set and it was computed for each of the four systems, as shown on Figures 2 to 5.

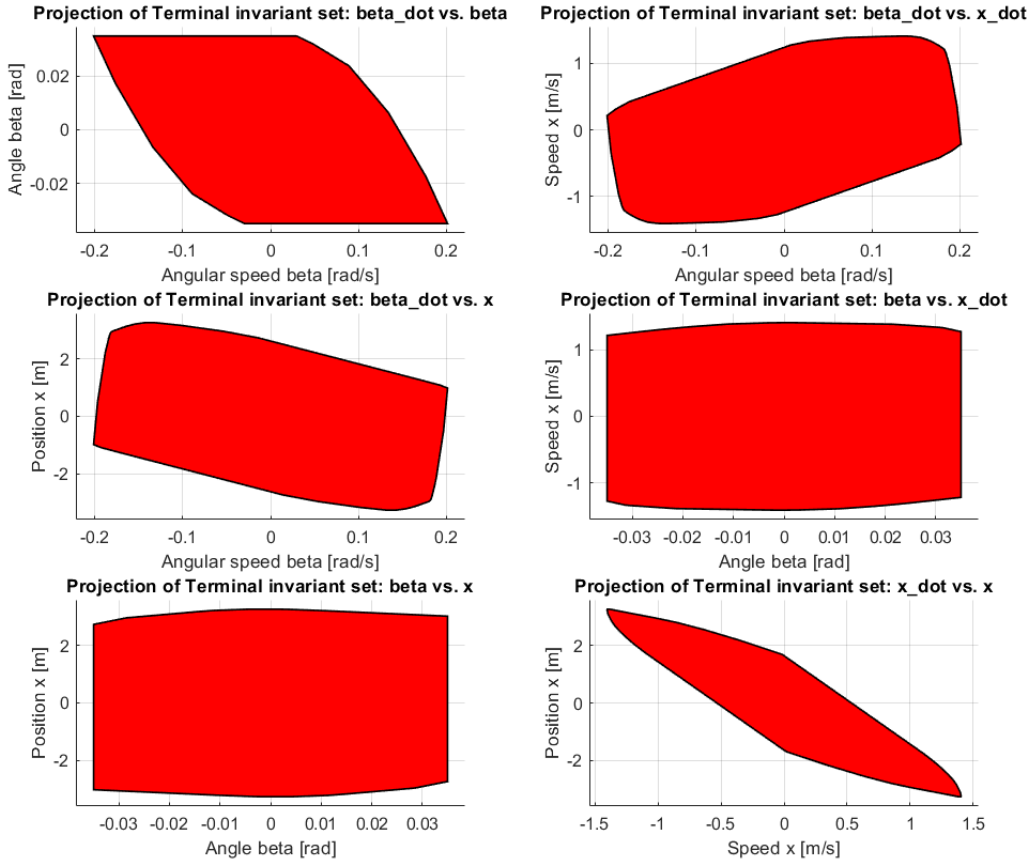

Figure 2: Terminal invariant set for system x/M_β

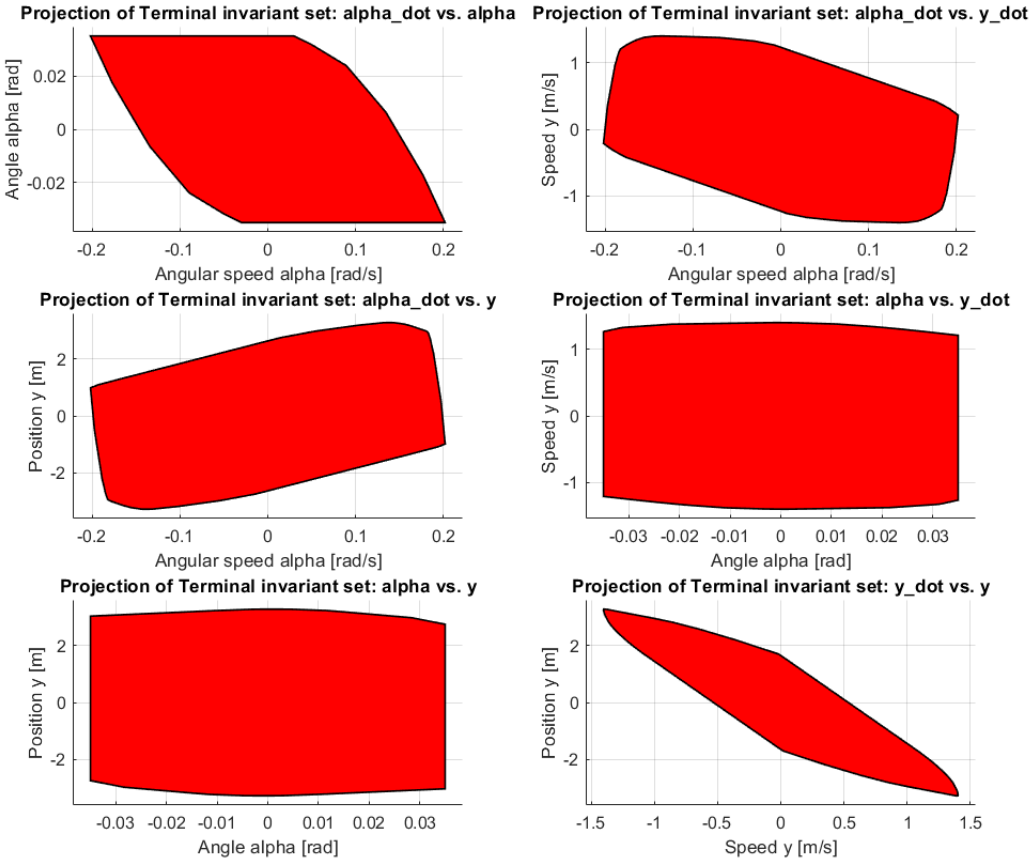

Figure 3: Terminal invariant set for system y/M_α

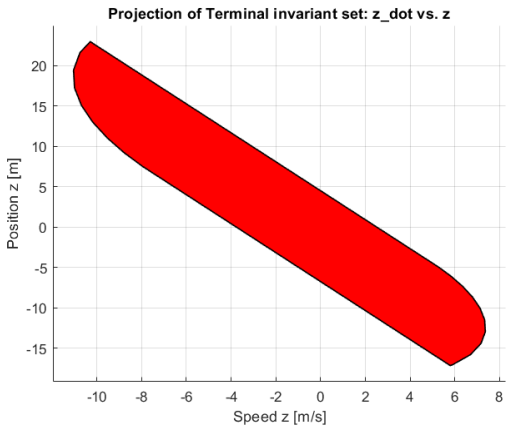

Figure 4: Terminal invariant set for system z/F

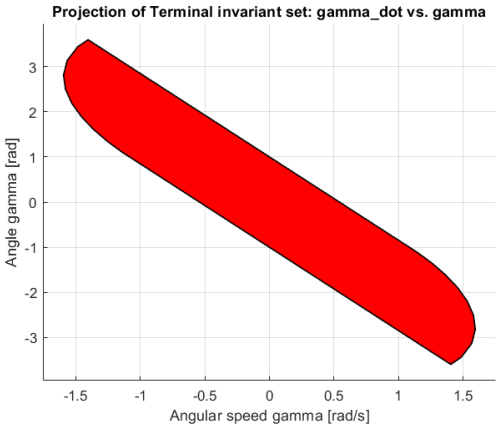

Figure 5: Terminal invariant set for system γ/M_γ

3.1.4 Results

Figure 6 shows the response of the four states based on the control input sequence that was optimized previously. Their respective settling times are shown in Table 1 and are all under eight seconds. Furthermore, figures 7 and 8 show the constraint variables. As they have been optimized to be as small as possible, the inputs are far from the constraints at all times. However, the states variables graze the constraints, which is normal since the controller was purposefully designed this way.

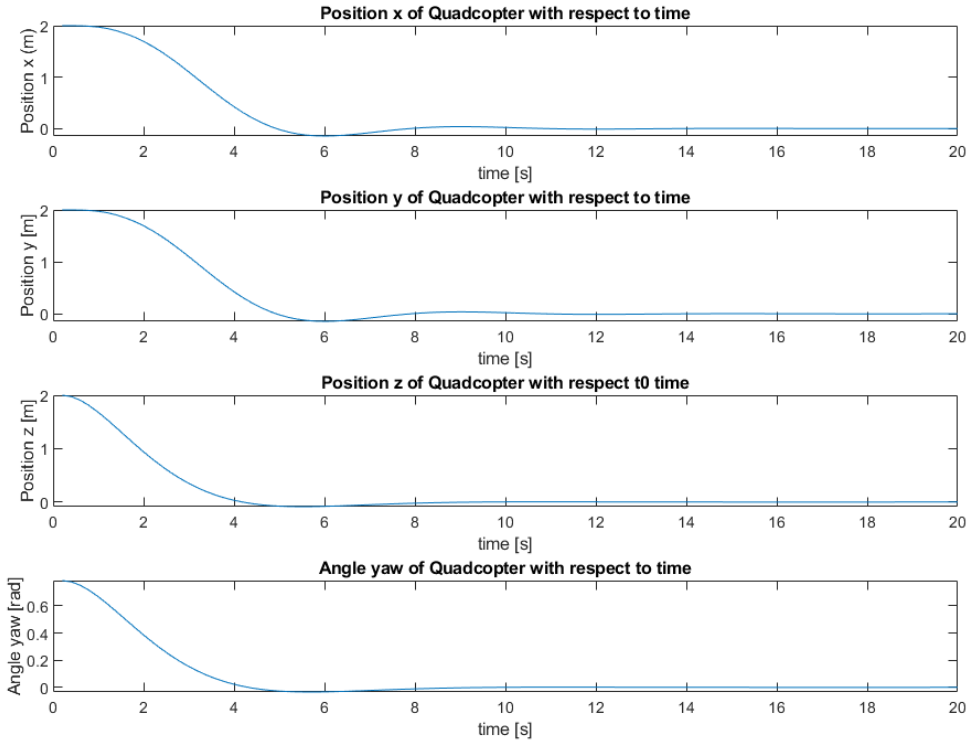

Figure 6: Regulation plot for each dimension starting stationary at two meters from the origin (for x, y and z) or stationary at 45 degrees for yaw

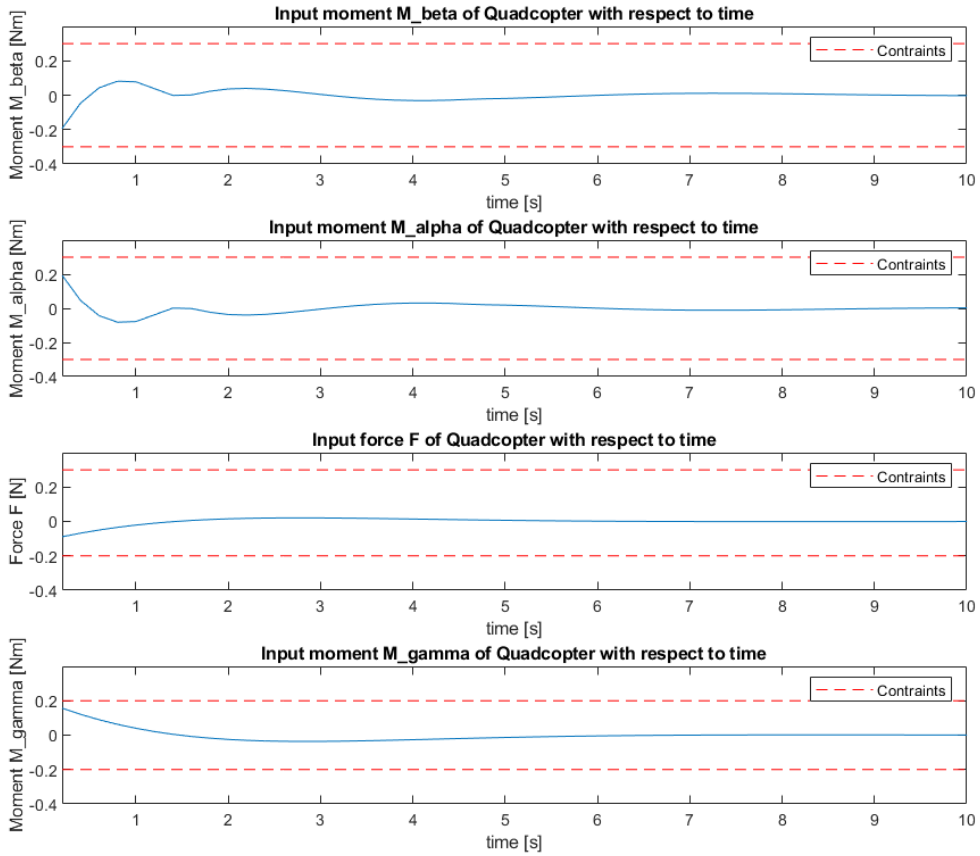

Figure 7: Regulation plot of the input for each dimension and constraints verification

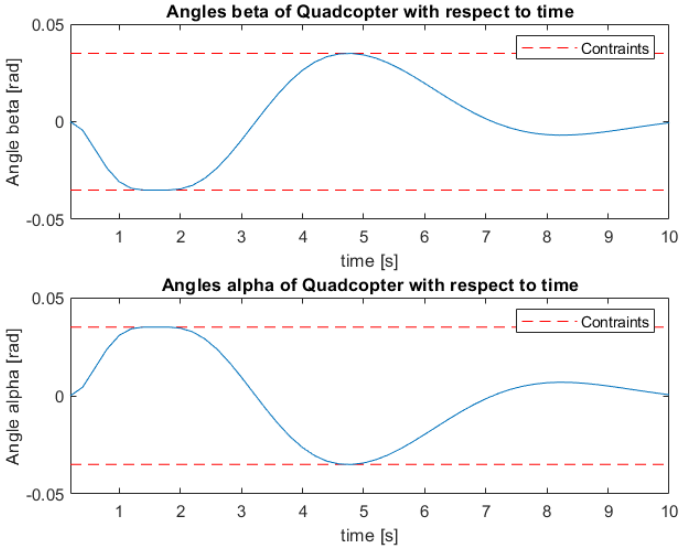

Figure 8: Regulation plot of the states for each dimension and constraints verification

3.2 MPC Tracking Controllers

In this section, the MPC controllers are modified to track a constant reference other than the origin while maintaining recursive feasibility.

3.2.1 Design procedure

When tracking a constant reference, the system should not converge to the origin but rather to some steady-state (x_s, u_s), called the target state. At this steady-state, The reference is achieved if $y_s = Cx_s = ref$, where ref represents the reference to be followed. A reasonable choice of steady-state is the minimal norm input, which can be formulated as a convex problem:

$$\begin{aligned} \min_{x_s,u_s} \quad & {u_s}^T R_s u_s \\ \text{s.t} \quad & \begin{bmatrix} I-A & -B \\ C & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} 0 \\ r \end{bmatrix} \\ & (x_s, u_s) \in \mathbb{X}, \mathbb{U} \end{aligned}$$

But the feasibility of the target problem cannot be guaranteed, so instead of using the actual set point in the controller a better way is to compute a reachable set point that is closest to ref (the closest admissible steady-state):

$$\begin{aligned} \min_{x_s,u_s} \quad & (Cx_s - r)^T Q_s (Cx_s - r) \\ \text{s.t} \quad & x_s = Ax_s + Bu_s \\ & (x_s, u_s) \in \mathbb{X}, \mathbb{U} \end{aligned}$$

Now that we have the target state, a simple approach to implement the reference tracking problem is to consider it as a regulation problem with a coordinate shift:

$$\begin{aligned} \min \quad & \sum_{i=0}^{N-1} (x_i - x_s)^T Q (x_i - x_s) + (u_i - u_s)^T R (u_i - u_s) + V_f((x_n - x_s)) \\ \text{s.t} \quad & x_0 = x \\ & x_{i+1} = Ax_i + Bu_i \\ & (x_i, u_i) \in \mathbb{X}, \mathbb{U} \\ & x_N \in \mathbb{X}_f \end{aligned}$$

Here, Δx and Δu represent the deviation variables, i.e $\Delta x = x - x_s$, $\Delta u = u - u_s$. The constraints are the same as in the previous problem. Note that the terminal constraint is on x_N only and not on $x_N - x_s$, this can be done here since there are no constrains on the position of the quadcopter and all of the non-position states are zero at steady-state. A shifted version of the invariant set is therefore still invariant. This also makes physical sense, since the quadcopter could be hovering anywhere in space without affecting the requirements for invariance.

3.2.2 Tuning parameters

The method followed for tuning the parameters is the same as in section 3.1.2, except that for MPC Tracking controllers there is no constraint on the settling time. Our aim this time was to minimize the settling time without worrying about the work done by the robot. The goal was to achieve the most rapid and accurate tracking. Figure 9 shows the initial (identity matrix for Q and R) and optimized control sequence. One can see that the input reaches the limit of its constraints to offer a faster dynamic of the quadcopter. Table 2 presents the final hyper-parameters of the system. As a general point of view, we see that the matrices Q have large diagonal components which induce a low settling time thanks to highly solicited inputs (the work done by each tracking controllers is more than twice the work done by the regulators).

system	x/M_β	y/M_α	z/F	γ/M_γ
settling time [s]	4.8040	4.8040	1.5848	2.2687
Work [J]	0.23	0.23	0.4472	0.1488
Q	$\begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 50 \end{bmatrix}$	$\begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 50 \end{bmatrix}$	$\begin{bmatrix} 20 & 0 \\ 0 & 800 \end{bmatrix}$	$\begin{bmatrix} 20 & 0 \\ 0 & 190 \end{bmatrix}$
R	10	10	1	1
N	50	50	10	10

Table 2: Optimized hyper-parameters for the control sequence

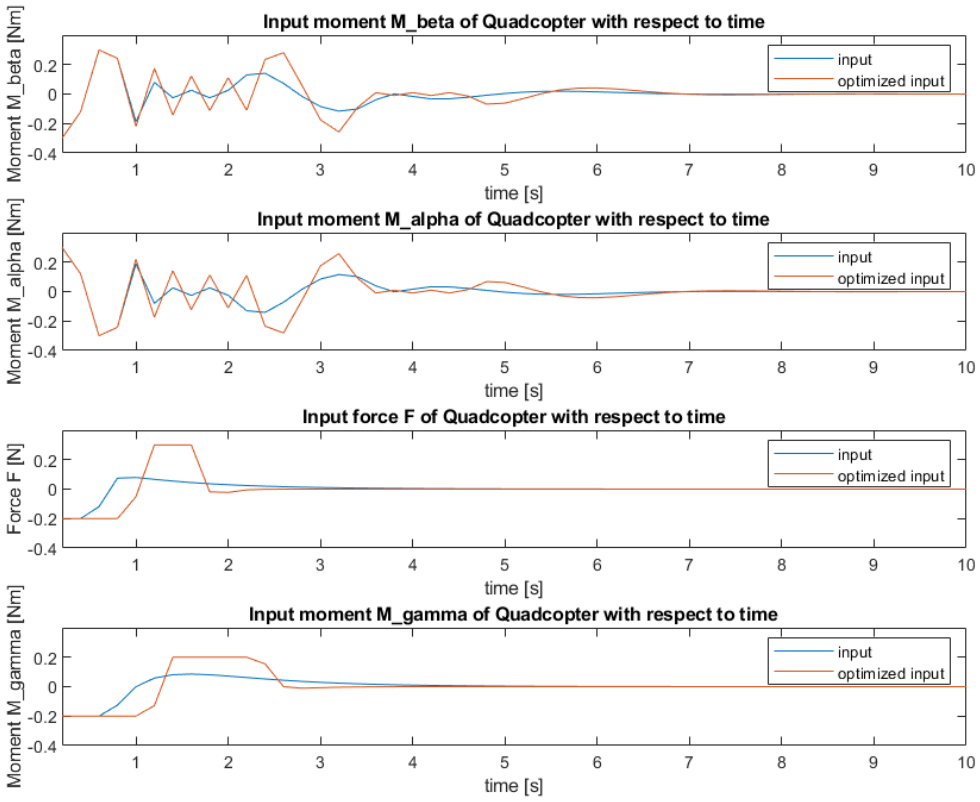

Figure 9: Plot of the initial inputs and the optimized inputs for the Tracking controller

3.2.3 Results

Figure 10 shows the response of the four states based on the control input sequence that was optimized previously. In this simulation, $x = -2\text{m}$, $y = -2\text{m}$, $z = -2\text{m}$, $\gamma = \frac{\pi}{4}$ are selected as the reference points. The quadcopter reaches the desired position in a minimum time while taking all the constraints into account.

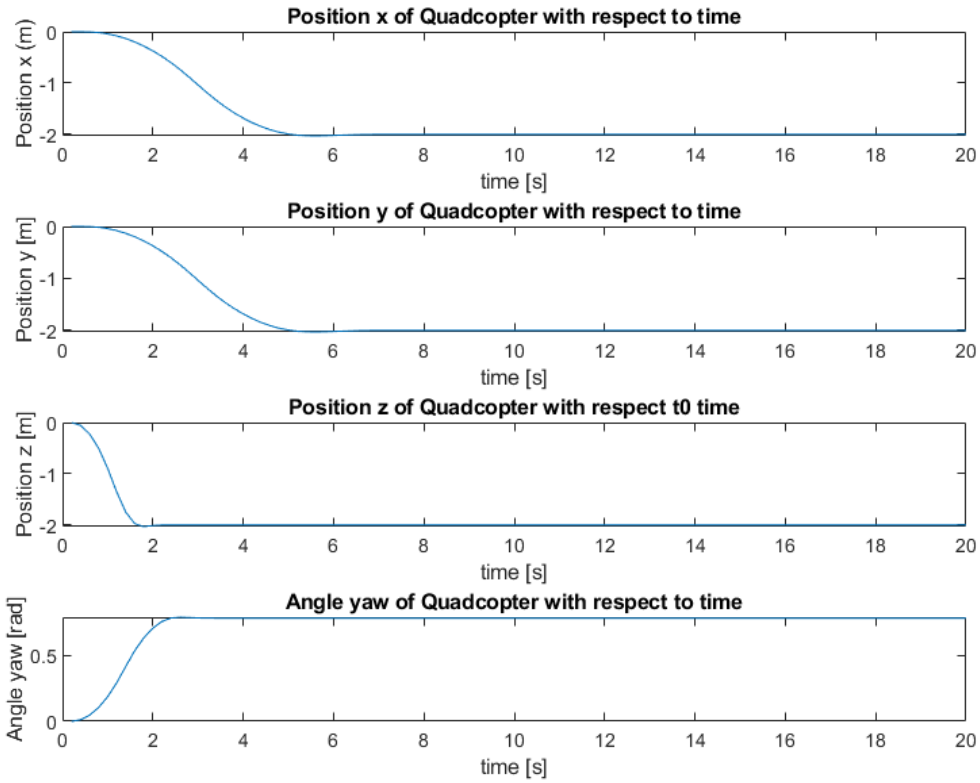

Figure 10: Tracking controller plot for each dimension starting stationary at the origin and going to negative two meters for x, y and z, and $\frac{\pi}{4}$ for yaw.

4 Simulation with Nonlinear Quadcopter

Figure 11 shows a simulation of the full nonlinear system with the four tracking controllers that were designed in section 3.2, while Figure 12 shows the states evolution along this simulation. We can see that the optimization done in Section 3.2.2 yields reasonably good performances, as all the states are tracked accurately (with small discrepancy). Another interesting observation is that the inputs do not stay saturated for a long time, they rather briefly reach their limit and come back down. This is generally for controllers that do not utilize the system to its full potential, i.e. the actuators could be used to a more aggressive extent. However, in this case the constraints on the pitch and roll angles drastically limit the value of the control inputs that can be applied, consequently producing results shown below.

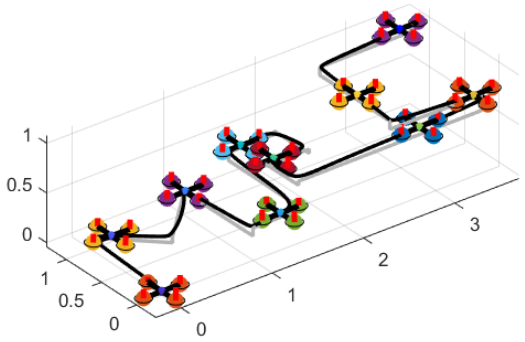

Figure 11: Tracking Controller successfully following "MPC" path

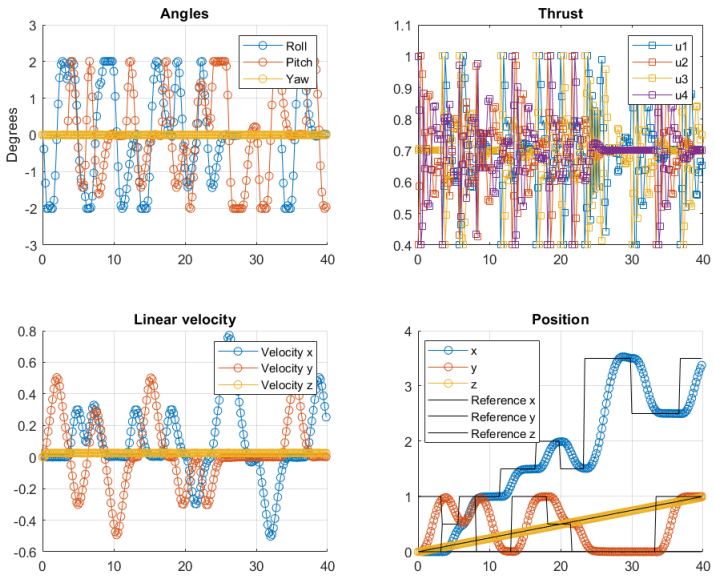

Figure 12: States evolution of the Tracking MPC along the reference path

5 Offset-Free Tracking

For the last controller, the system is perturbed by a bias in the negative z direction. This could physically represent a change in mass of the quadcopter, for example by adding cameras or a larger battery pack, or modelling error. Hence, the MPC controller on the vertical positioning z related to the thrust F must be adapted.

5.1 Design procedure

When adding a disturbance to the system, the dynamics becomes:

$$x^+ = Ax + Bu + Bd \tag{11}$$

where d is the unknown disturbance. First, a disturbance model was introduced assuming integral disturbance dynamics. To compensate this disturbance, the system was extended to include a new state $\hat{d}_k$, providing us with the disturbance estimator $\hat{d}_{k+1}$. The observer is written as follows:

$$\begin{bmatrix} \hat{x}_{k+1} \\ \hat{d}_{k+1} \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ \hat{d}_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k + \begin{bmatrix} L_x \\ L_d \end{bmatrix} (C\hat{x}_k - y_k) \tag{12}$$

The vector $L = \begin{bmatrix} L_x & L_d \end{bmatrix}^T$ has to be tuned to ensure that the estimator is stable. The poles placed by the vector L should be faster than the dynamics of the system but slow enough to not intensify noise. To tune these poles we started with poles close to 1 and progressively decreased them until they did not significantly improve the estimation anymore. The resulting poles are $P = \begin{bmatrix} 0.2 & 0.3 & 0.1 \end{bmatrix}^T$ and the resulting Kalman filter has the gains $P = \begin{bmatrix} -8.29 & -2.4 & -0.9 \end{bmatrix}^T$.

Also, the system's steady-state was modified to account for the effect of the disturbance. It is now given by the following optimization problem:

$$\begin{aligned} \min_{x_s, u_s} \quad & {u_s}^T R_s u_s \\ \text{s.t} \quad & \begin{bmatrix} I - A & -B \\ C & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} B \cdot \hat{d} \\ r \end{bmatrix} \\ & (x_s, u_s) \in \mathbb{X}, \mathbb{U} \end{aligned}$$

The convex optimization problem to compute the control sequence is similar to the previous sections, with the exception that it takes the disturbance estimate into account.

$$\min \sum_{i=0}^{N-1} (x_i - x_s)^T Q (x_i - x_s) + (u_i - u_s)^T R (u_i - u_s) + V_f ((x_n - x_s)) \tag{13}$$

$$\begin{aligned} s.t. \quad & x_0 = \hat{x} \\ & d_i = \hat{d} \\ & x_{i+1} = Ax_i + Bu_i + d_i \\ & (x_i, u_i) \in \mathbb{X}, \mathbb{U} \\ & (x_N - xs) \in \mathbb{X}_f \end{aligned}$$

The tuning parameters Q, R and N were chosen identical to the tracking controller.

5.2 Results

The new dynamics were simulated over a path composed of two consecutive steps with different amplitudes. The result is shown on Figure 13. For both steps, the controller successfully achieves offset-free tracking.

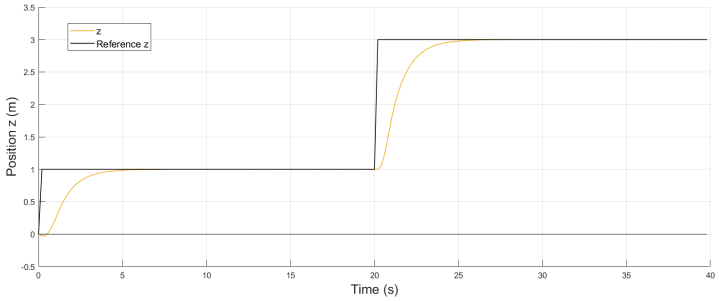

Figure 13: Plot showing that the z-controller achieves offset-free tracking

The quadcopter was also simulated when tracking the "MPC path", the results are shown on figure 14. Looking at the path there is a height drop at the beginning of the simulation, which is due to the offset-free controller. Since the disturbance estimation is initially unknown (zero), the controller does not apply any compensation yet and the quadcopter drops because of the vertical bias that was introduced. After the first iteration, the controller is able to estimate this disturbance and correct the system's trajectory, and after just a few seconds, the quadcopter is back on the reference path on which it remains until the end.

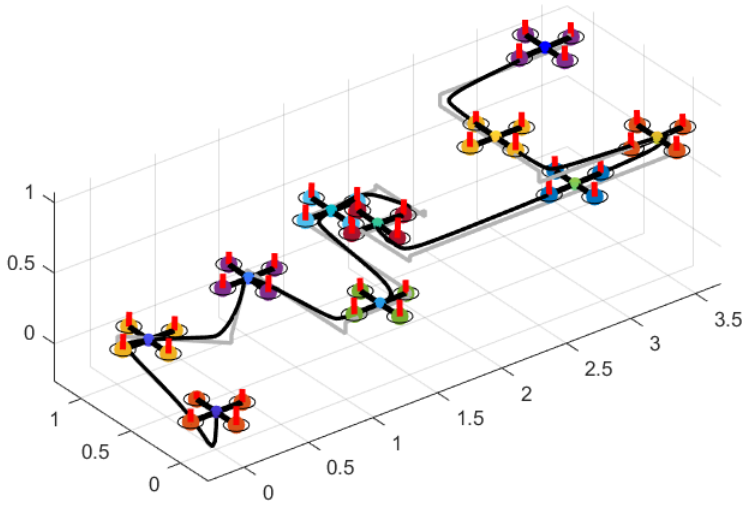

Figure 14: Plot showing that the z-controller achieving offset-free MPC tracking

6 Nonlinear MPC

In this section, the design of a non-linear MPC is carried out. The main difference with the linear MPC presented above is that the nonlinear dynamics of the quadcopter is now considered and we can therefore not separate the system into four independent sub-systems any more. The non-linear dynamics have already been presented in Section 1, especially Equations 1, 2, and 3, which requires the use of the full-state representation:

$$\mathbf{x} = \begin{bmatrix} \dot{\alpha} & \dot{\beta} & \dot{\gamma} & \alpha & \beta & \gamma & \dot{x} & \dot{y} & \dot{z} & x & y & z \end{bmatrix}^T$$

6.1 Design

A similar approach to the linear MPC was used here with the use of the deviation variables $\Delta x = x - x_s$, $\Delta u = u - u_s$. In this case, x_s and x_s were taken from the steady-state values while hovering given by the function`quad.trim()`. From our general understanding of the quadcopter's dynamics we know that during perfect hovering all 12 states are zero except for the Cartesian position (x, y, z) and the yaw angle (γ), which can be any arbitrary value with no effect on the hovering dynamics. Moreover, the four non-zero states just mentioned (x, y, z, γ) are set by the given reference path. Therefore, the steady-state vector was designed as follows: $x_s = [0\ 0\ 0\ 0\ 0\ \gamma_{ref}\ 0\ 0\ 0\ x_{ref}\ y_{ref}\ z_{ref}]^T$ and is updated with the new reference values from the reference path at each sampling time. Similarly, the steady-state input has the form: $u_s = [u_{1,ref}\ u_{2,ref}\ u_{3,ref}\ u_{4,ref}]^T$.

Now that the steady-state target is defined, the objective function can be addressed based on the same formulation as presented in Section 3.2.1. The problem was adapted to the non-linear case by removing the terminal constraint due to the complexity of calculating an invariant set, and by defining the dynamics using a non-linear notation.

$$\begin{aligned} \min \sum_{i=0}^{N-1} & (x_i - x_s)^T Q (x_i - x_s) + (u_i - u_s)^T R (u_i - u_s) \\ \text{s.t} \quad & x_0 = x \\ & x_{i+1} = f(x, u) \\ & (u_i) \in \mathbb{U} \end{aligned}$$

Because of the linear approximation in the previous MPC, constraints were placed on the states of the system ($\alpha and \beta$) to ensure that the approximation was valid. These are no longer needed and can be dropped. Hence, the only remaining constraint is on the input: $0 \leq u_i \leq 1.5$, ensuring that the control values do not exceed the physical limit of the rotors.

The matrices Q and R were set by observing the trajectory of the quadcopter and adjusting the the individual weights accordingly. The matrix Q is diagonal and has zeros everywhere except for the four states x, y, z, and γ since those correspond to the reference path that we wish to track. The matrix R is simply the identity matrix scaled by a constant to provide an equal importance to the control input on all four rotors. The final weights used are:

$$\begin{aligned} Q &= \text{diag}([0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 100\ 100\ 10000]) \\ R &= \text{diag}([3\ 3\ 3\ 3]) \end{aligned}$$

While the last weight in Q may seem very high at first, it was chosen as such for a reason. Since the controller now considers the full-state representation of the system, the various states remain coupled, which means that a certain input may affect several states simultaneously. While this behaviour is useful to achieve faster convergence to the reference, it can also deteriorate the performance of some states. In our case, when the quadcopter wants to change its location in the x-y-plane, it must tilt

and therefore will loose vertical thrust if the control input is not compensated for appropriately. Our simulations have shown that a high weight on the state z leads to better tracking performances.

With the objective defined, we proceeded to implementing the system's dynamics ensuring $x_{i+1} = f(x, u)$. To do so, the system was discretized using the fourth order Runge-Kutta method at a sampling period of $h = 1/5$ (taken from Part 3). This method was chosen over Euler because it provides a 4^{th} order approximation, as opposed to 1^{st} order for Euler, which should yield more accurate results at the cost of increased computational effort.

6.2 Results and Analysis

Simulating the quadcopter over the "MPC" reference path leads to the following results.

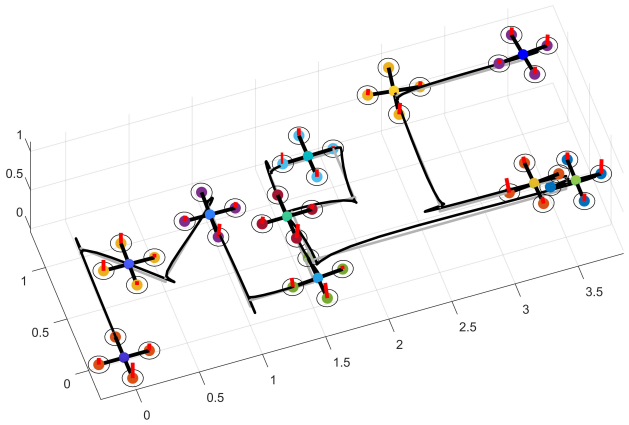

Figure 15: "MPC" reference path tracked by the non-linear MPC controller

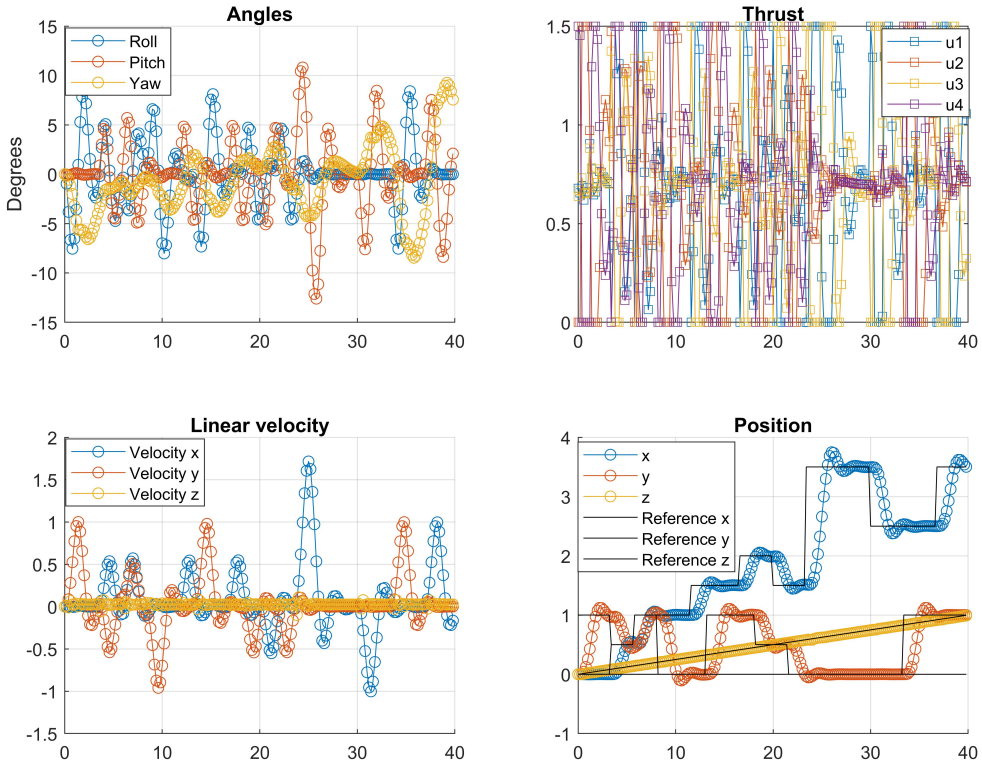

Figure 16: States evolution of the non-linear MPC along the reference path

The performance gains with the non-linear MPC lies in the ability to consider the coupled dynamics as a whole, and not be limited within a neighbourhood of the linearizing operating point. This allows the quadcopter to achieve greater pitch and roll angles, hence improving the physical responsiveness of the drone. This is shown on Figure 16 where the pitch angle reaches values of up to 10°, while previously being constrained to only 2° in the linear case. The choice of Q and R was such to achieve the best tracking performances possible without too much regard on the control input. This can be seen in the thrust plot on Figure 16, where the input is often saturated so to steer the system to the reference as fast as possible.

Overall, the non-linear MPC seem like a more accurate and performance-oriented solution, since its properties can be tailored to a broader range of systems. However the states cannot be decoupled any more, which adds complexity to the model, but they can now evolve more freely allowing for faster dynamics, as seen with the pitch angle. However, since invariant sets are much more challenging to compute they are often dropped, as we did here, which may introduce local minima in the optimization problem rather than guaranteeing the global minimum like in the linear case.

Additionally, for the sake of curiosity the NMPC was simulated again using the Euler discretization instead of RK4. With the same Q, R, and N as before we obtain much smoother results, shown on Figure 17, but this dynamics is likely less accurate since it does not take higher order terms into account.

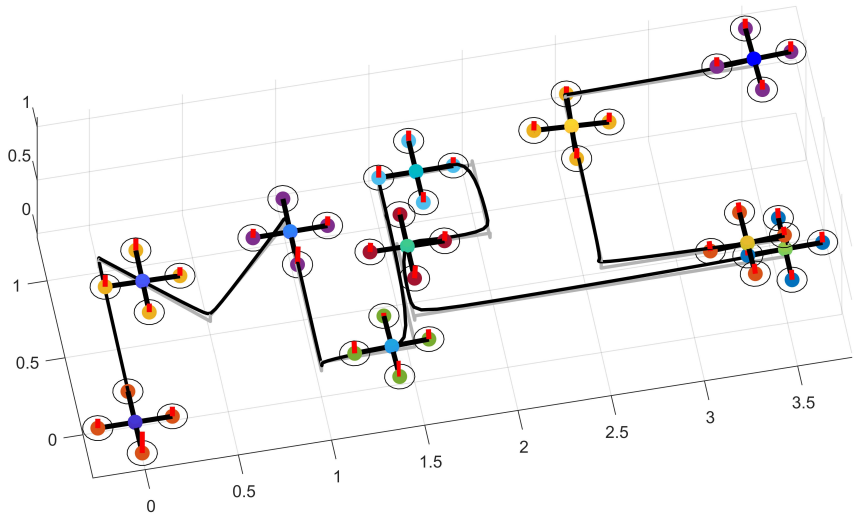

Figure 17: "MPC" reference path tracked by the non-linear MPC controller with Euler discretization