MODEL PREDICTIVE CONTROL

PROJECT

Quadropter

Authors:
Aymen Bahroun 238686
Bassel Belhadj 261313
Salim Ben Ghorbel 261316

January 6, 2019

Contents

1 Introduction

In this project, we will implement an MPC controller to fly a quadropter. In the first part, we will control a linearized version of the quadropter. We will design an MPC controller to regulate system state to the origin, then an MPC controller to track non-zero output set points, then an offset-free tracking MPC.
In the second part, we will implement a Nonlinear MPC and compare them.

2 Linearization and diagonalization

In this section, in order to control a linearized version of the quadcopter, we linearize our system and obtain a continuous-time state-space linear model

$$\begin{aligned}\dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u} \\ \mathbf{y} &= \mathbf{x}\end{aligned} \tag{1}$$

where $\mathbf{x}$ is our 12-state vector such that $\mathbf{x} = [\dot{\theta} \quad \theta \quad \dot{p} \quad p]$, with $\theta = [\alpha \quad \beta \quad \gamma]$ the roll, pitch, and yaw of the quadcopter, and $p = [x \quad y \quad z]$ the position of the center of mass of the quadcopter.
The input of the model $\mathbf{u} = [u_1 \quad u_2 \quad u_3 \quad u_4]^T$ represents the four input thrusts of the rotors.

From the dynamics of the system, we can relate the net body force F and body moments(M_α, M_β, M_γ) to the input through equation 2

$$\mathbf{v} = \begin{bmatrix} F \\ M_\alpha \\ M_\beta \\ M_\gamma \end{bmatrix} = \begin{bmatrix} k_F & k_F & k_F & k_F \\ 0 & k_F L & 0 & -k_F L \\ -k_F L & 0 & k_F L & 0 \\ k_M & -k_M & k_M & -k_M \end{bmatrix} \mathbf{u} = T\mathbf{u} \tag{2}$$

we use this transformation to do change of variables and have $\mathbf{v}$ as our new input

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} = A\mathbf{x} + BT^{-1}\mathbf{v} \tag{3}$$

2.1 Deliverable 2.1

The resulting A,B,C and D matrices from the transformation above applied to our system show that it breaks it into four independent sub-systems, this means that:
Different subsets of the states depend just on other same subset of states with the linearised system. This occurs only when each of the corresponding values of the matrix A are null. This is only possible if the corresponding elements of the linearised system matrix are zero A is just the Jacobian matrix of the nonlinear system, evaluated at the linearisation point (x_s, u_s):

$$J_x = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \left[\frac{\partial f_i}{\partial x_j}\right]_{i,j} \tag{4}$$

Computing J_x yielded to non-zero values for (1,1), (1,2), (2,3), (3,1), (5,2), (6,3), (7,5), (8,4), (8,5), (9,4), (9,5), (10,7), (11,8), (12,9) elements of the matrix, with zero value for all other elements.

Studying the Jacobian matrix, we can observe that an element can be zero in the case where it is identically equal to zero or when evaluating the element at (x_s, u_s), gives zero. The first case, the element still equal to zero independently of the steady state where we evaluate the matrix. But for the second case, we can't assure that this happens for other equilibrium states values.

This is basically due to the transformation $\mathbf{v} = T\mathbf{u}$, converting the multiple subsystems vector u to a matrix v, which does not have the same property. This transformation is not related to the Jacobian matrix and can be made at any equilibrium point.

Given that this transfromation is independant from the jacobian matrix; for other equilibrium points, we may not have the same conditions setting the Jacobian to zero, and then decoupling the system into multiple non-interacting systems.

3 Design MPC controllers for each sub-system

In this section we design a recursively feasible and stablizing MPC controller for the discrete-time models of the systems previously produced, obtained by discretization of the continuous-time models with a sampling period $T_s = 0.2$ seconds, that stabilizes the system to the origin(set all the states to zero).
In addition we must ensure constraints on the angles that the quadcopter could take to validate our approximation :

$$|\alpha| \leq 2^\circ = 0.035rad$$
$$|\beta| \leq 2^\circ = 0.035rad$$

Since also our quadcopter inputs have constraints $0 \leq \mathbf{u} \leq 1.5$ which give constraints on $\mathbf{v}$ defined by :

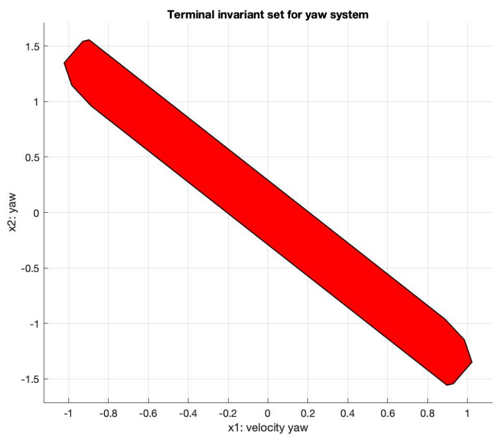
$$\begin{aligned} -0.3 \leq M_\alpha \leq 0.3 \\ -0.3 \leq M_\beta \leq 0.3 \\ -0.2 \leq F \leq 0.3 \\ -0.2 \leq M_\gamma \leq 0.2 \end{aligned}$$

3.1 Deliverable 3.1

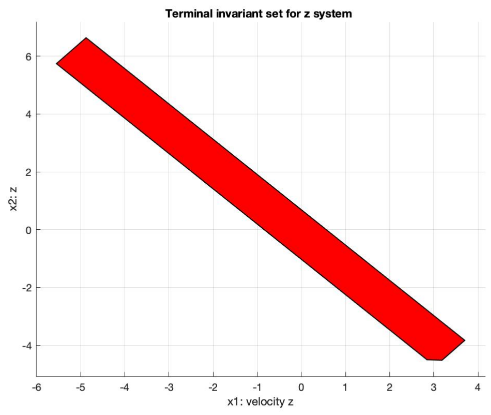
To hold the recursive feasability i.e. guarantee feasability at every state along the closed-loop trajectory for all feasible initial states, and ensure stability, we designed our MPC controllers by introducing terminal cost and constraints these values are chosen to simulate an infinite horizon. The terminal constraint provides a sufficient condition for constraint satisfaction, we use the following cost function :

$$J = \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i + x_N^T Q f x_N \tag{5}$$

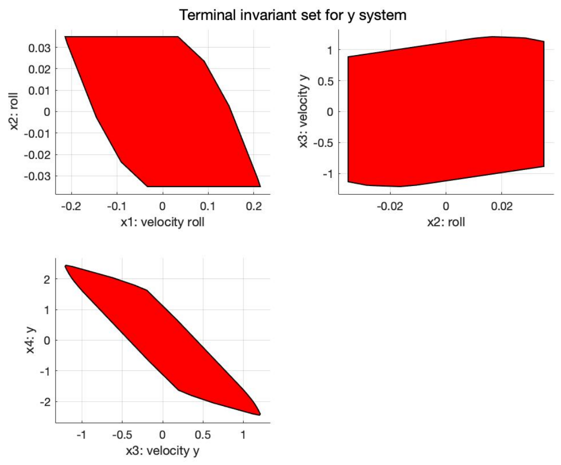
And $x_N \in \chi_f$, with χ_f being the terminal constraint and $x_N^T Q f x_N$ from equation 5 being the terminal cost (Infinite horizon cost starting from x_N).
The parameters Q, R, and N were tuned such that they minimize the settling time while ensuring at the same time a low R to avoid aggressive control action. You can refer to the file tuneLQR.m where Q and R gains were tuned such that we have optimal settling times with minimal R gains were set. The selection criterium was to chose the Q,R weights that minize the settling time of the closed-loop system. Then, the MPC controllers were tested with the selected weights and very similar settling times were found.
As for the horizon, a range of values were tested. For N<9, the controller was not able to stabilize the system as it reached unfeasible states. We chose N = 15 as a safe choice of the horizon length. In fact, we tested lower values of N in the following deliverables and the control action was not performant enough to track trajectories.
The final choice for all the controllers was $N = 15$ for the horizon, and $Q = I_n$, and $R = 7$ for x,y controllers and $Q = I_n$, and $R = 1$ for z,yaw controllers.



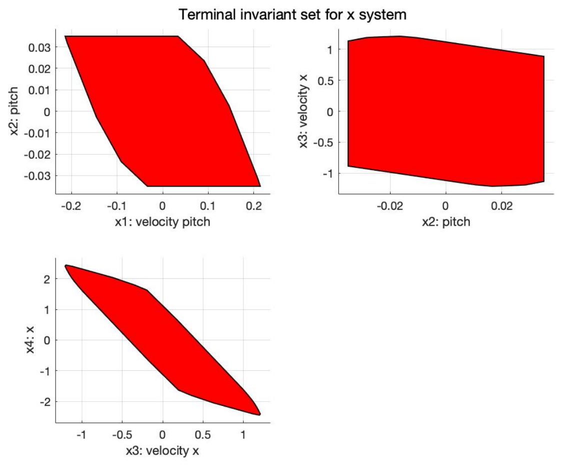
(a) Terminal set projection for yaw system



(b) Terminal set projection for z system



(c) Terminal set projections for y system



(d) Terminal set projections for x system

Figure 1: Terminal invariant sets projections for each dimension for each sub-system

We can observe from the projections above, that for every case, the origin is included in the terminal sets, as well as the values +/-2 for x,y,z and +/- 45 degrees for yaw. This means that the region of attraction, created by input and output constraints as well as the Q,R weights include enough range.

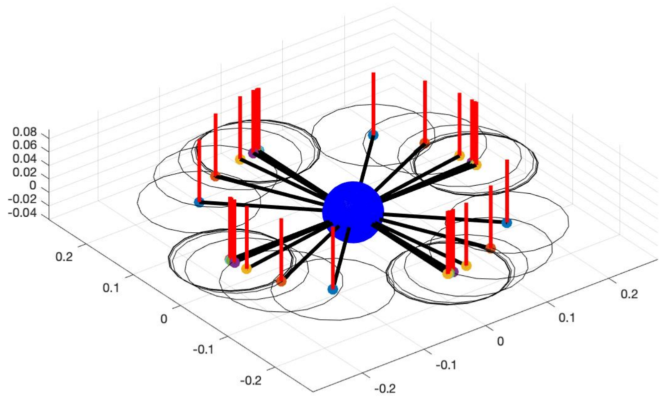


Figure 2: Quad starting stationary at 45 degress yaw

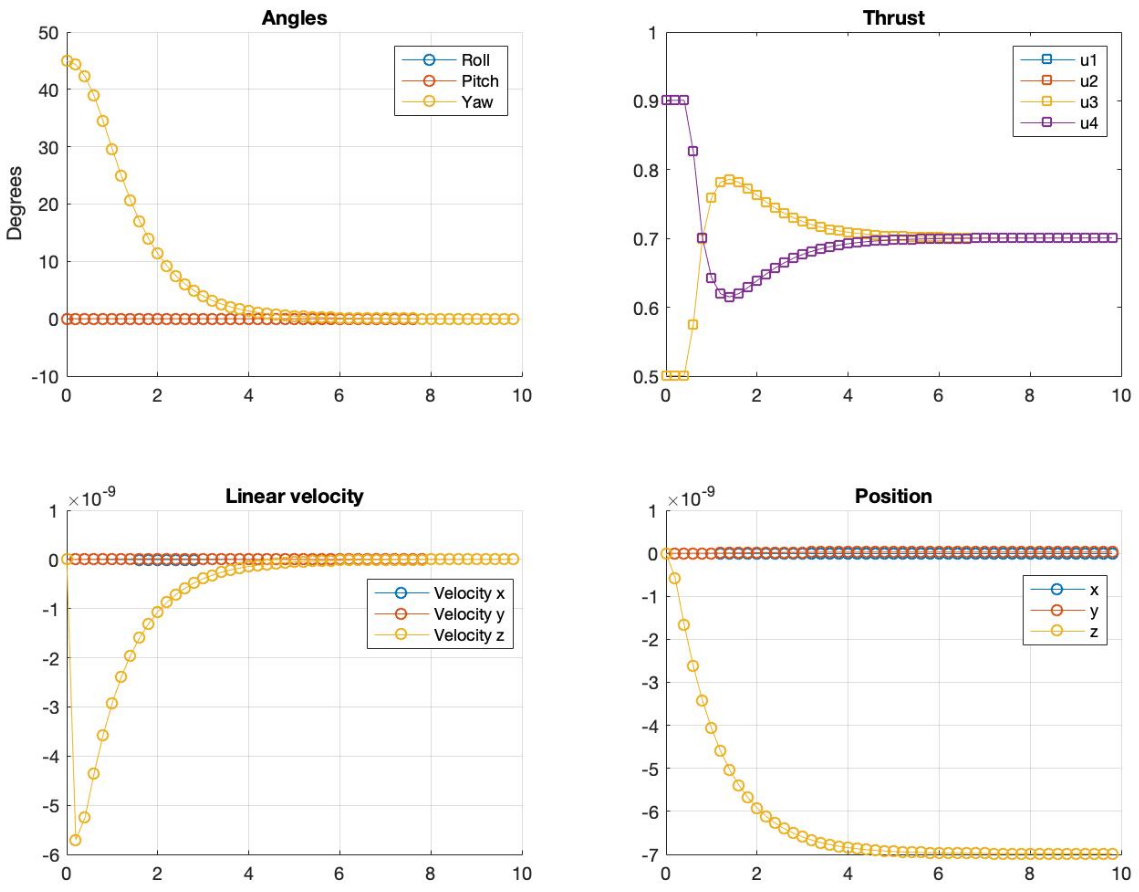


Figure 3: Evolution of thrusts inputs and quad states over time

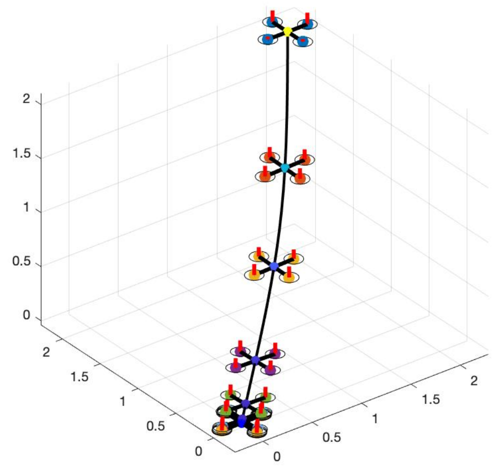


Figure 4: Quad starting stationary at 2 meters from origin (x,y,z)

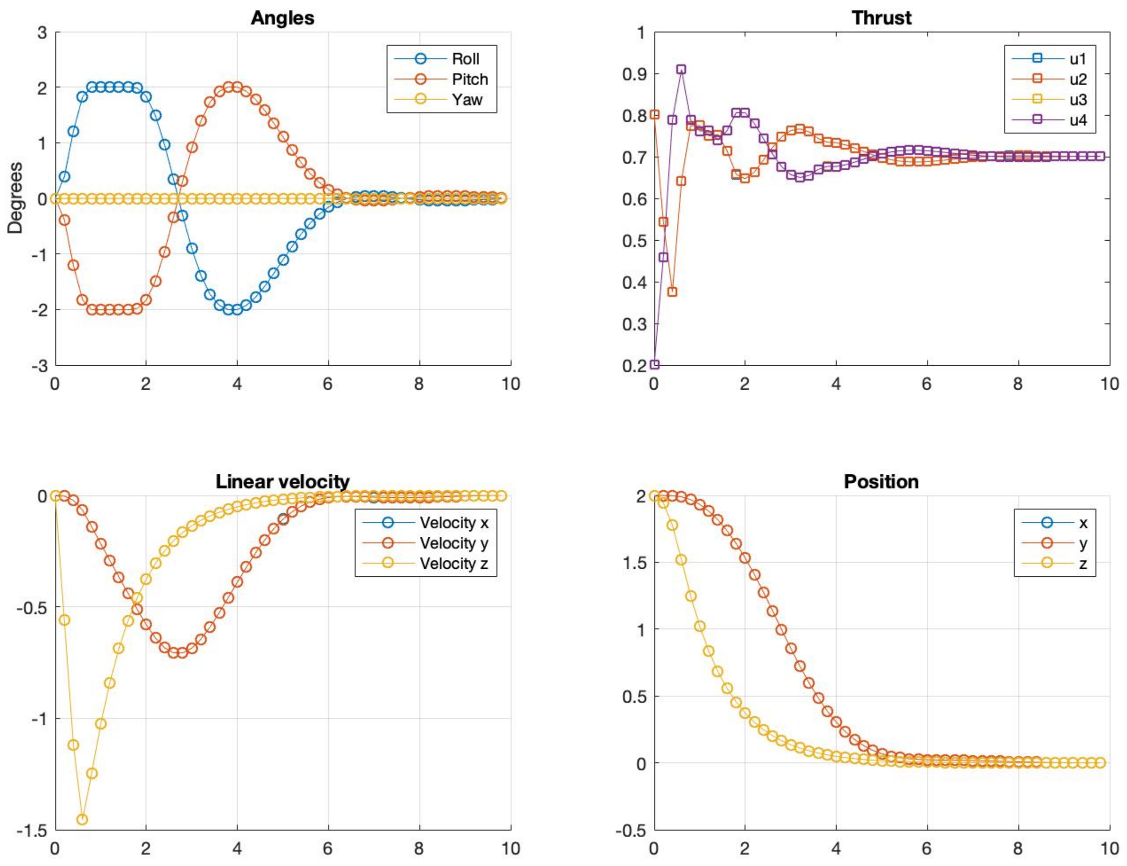


Figure 5: Evolution of thrusts inputs and quad states over time

We can clearly see from Figures 2,4 that the MPC stabilizes the system to the origin from either at 45 degrees yaw or at 2 meters(x, y, z) from the origin, while ensuring the constraints on the thrust inputs and the constraints on the maximum angles that the quad can take (Figures 3,5). The settling time after which the system is stabilized for the first case is 3.8 seconds and the second case around 5 seconds. We can see, from Figure 5 that the z system is stabilized before the y or x systems, this is due to the dynamics of the z system that are such that the coordinated inputs from the four motors makes it faster to reach steady-state.

3.2 Deliverable 3.2

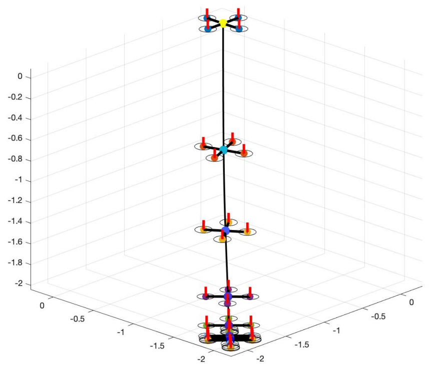
In this section, we design our controllers such that they track a reference path(constant references), while maintaining recursive feasability. We get rid of the terminal set since it reduces the feasible set and potentially adds large number of extra constraints, but here our feasible set will not be invariant. we design a target selector that returns a target steady-state (x_s, u_s) corresponding to the reference r, this ensures that there exists an input that keeps the system at target. The cost function is as follows :

$$\begin{aligned} & (Cx_s - r)^T(Cx_s - r) \\ s.t. \quad & x_s = Ax_s + Bu_s \\ & Mu_s \leq m \\ & Fx_s \leq f \end{aligned} \tag{6}$$

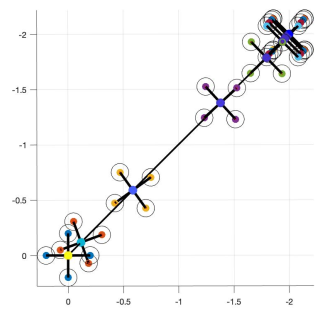
Then we apply the regulation problem in delta formulation by minimizing this cost function :

$$\begin{aligned} & J = \sum_{i=0}^{N-1} \Delta x_i^T Q \Delta x_i + \Delta u_i^T R \Delta u_i \\ s.t. \quad & \Delta x_i = x_i - x_s \\ & \Delta u_i = u_i - u_s \end{aligned} \tag{7}$$

This gives us the optimal input to our system to track the reference path.
We show plots of our system starting at the origin and tracking a reference path to 45 degrees yaw and -2 meters from origin (for x, y, and z) in Figure 8. The constraint satisfaction is achieved and shown in Figure 7 for the thrusts inputs and the maximum angles that could be handled by the quadcopter.



(a) view 1



(b) view 2

Figure 6: Plots of the quad tracking the reference path

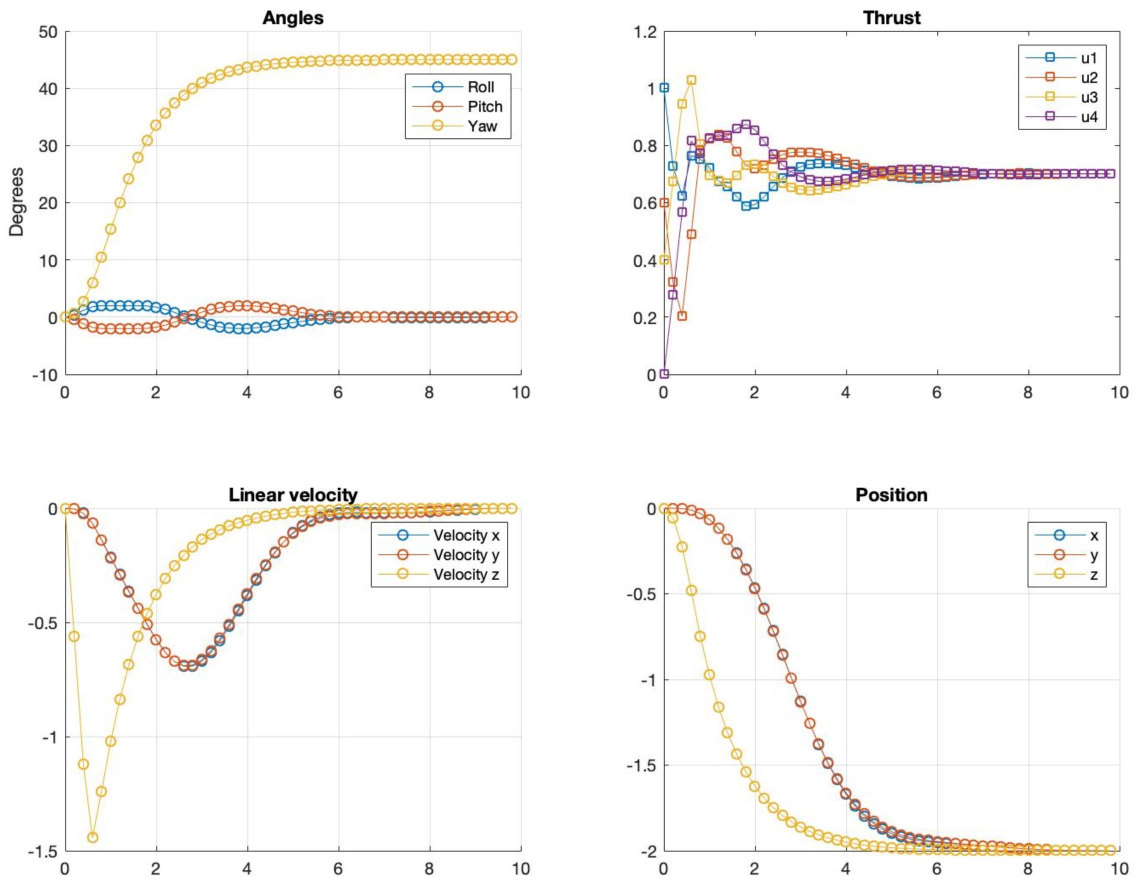
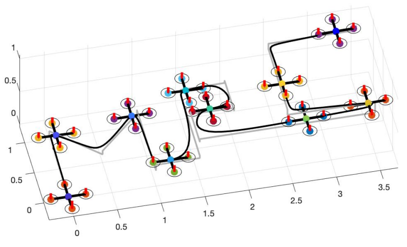


Figure 7: Evolution of thrusts inputs and quad states over time

4 Simulation with Nonlinear Quadcopter

4.1 Deliverable 4.1

In this section we simulate the controllers to track a given reference path. Input and state constraints are satisfied. System reaches target references in a reasonable time.


(a) view 1

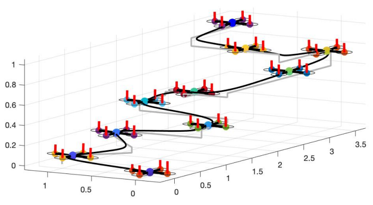

(b) view 2

Figure 8: Plots of the quadcopter successfully tracking the reference path

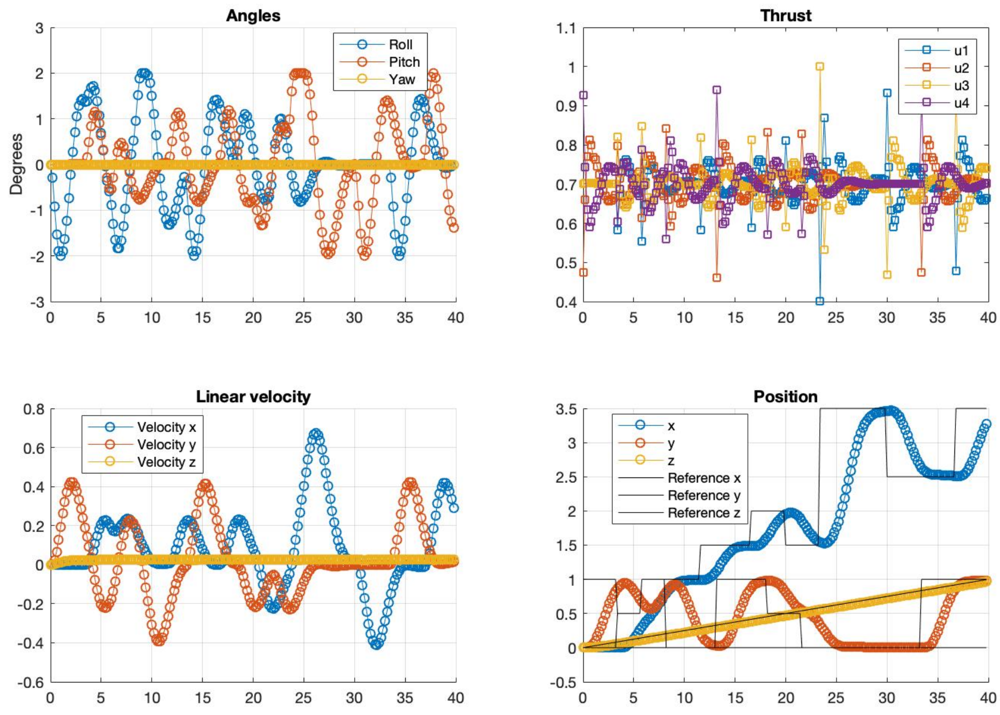


Figure 9: Evolution of thrusts inputs and quad states over time

5 Offset-free tracking

In this section, we assume that the mass of the quadcopter changes, and our goal is to design an offset- free MPC controller that rejects the disturbance for the z system.

the new dynamics of the z system are now :

$$x^+ = Ax + Bu + Bd \tag{8}$$

Where d is a constant unknown disturbance.

5.1 Deliverable 5.1

The procedure consisted of the design of a state and disturbance estimator based on the augmented model :

$$\begin{bmatrix} \hat{x}_{k+1} \\ \hat{d}_{k+1} \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ \hat{d}_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k + L(\begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ \hat{d}_k \end{bmatrix} - y_k) \tag{9}$$

where $\hat{x}$ and $\hat{d}$ are estimates of the state and disturbance. For the offset free tracking the system steady-state has to take into account for the disturbance on state evolution, the new condition at steady-state is :

$$\begin{aligned} x_s &= Ax_s + Bu_s + Bd_s \\ y_s &= Cx_s = r \end{aligned} \tag{10}$$

This gives us the steady-state, we then solve the MPC problem for tracking using the disturbance estimate $\hat{d}$, we minimize the folllowing objective function :

$$\begin{aligned} J = &\sum_{i=0}^{N-1} \Delta x_i^T Q \Delta x_i + \Delta u_i^T R \Delta u_i \\ s.t. \quad & \Delta x_i = x_i - x_s \\ & \Delta u_i = u_i - u_s \\ & x_0 = \hat{x} \\ & d_i = \hat{d} \\ & x_{i+1} = Ax_i + Bu_i + d_i \\ & Mu_i \leq m \end{aligned} \tag{11}$$

The design criterion for estimator pole placement is that they should be such that estimator dynamics are faster than that of the plant , i.e. of the matrix A of the augmented z system.
We calculated the poles of the augmented z system and found that they are all equal to 1.
It is also recommended that the poles of the estimator should ensure that estimator dynamics are faster than the closed-loop controlled system.
These values are clear in a PID controller or a state-space controller, but in the case of an MPC controller, it is not straightforward to obtain them.
For this reason, our choice of the estimator poles to be placed with the gain L(which are 0.3, 0.2 and 0.1) are such that they are smaller than the open-loop augmented z system and offer an tradeoff between fast disturbance rejection compared to the system dynamics of the MPC controller and low estimator gains.
A series of the same simulation were run to test the performance of each set of estimator gains. We show here plots of the quadcopter achieving offset-free tracking with a BIAS = -0.1, we can clearly see that the system rejects the disturbance and tracks set point references with zero offset with a maximum initial overshoot lower than -0.3.

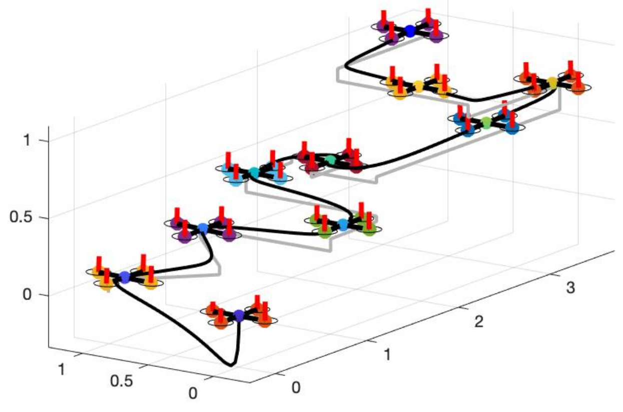


Figure 10: Plot of quadcopter achieving offset-free tracking

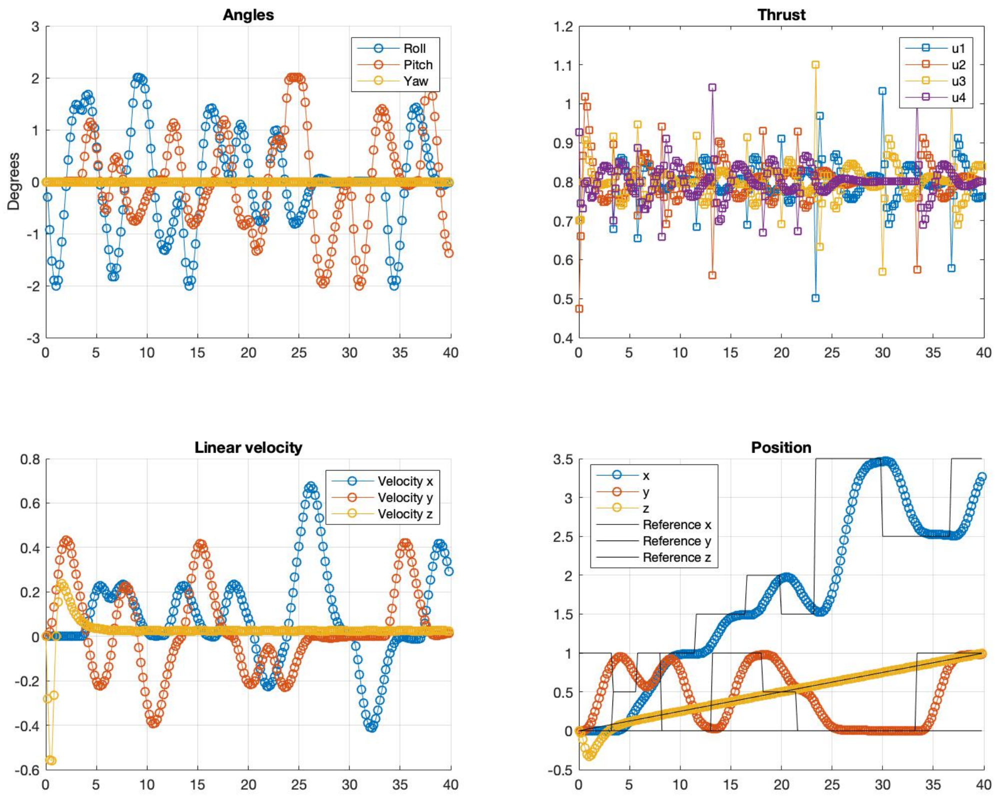


Figure 11: Evolution of thrusts inputs and quad states over time

6 Nonlinear MPC

6.1 Deliverable 6.1

In this section, we will implement the nonlinear MPC control, which has the same pricipale as the linear one. The changes consists on the problem definition; due the nonlinear aspect of the system and the controller:

Here, the cost function is defined by the integral of the entire states values among the operation time.
The constraints of the problem are defined as beyond:

Since, the system is no longer linearized, we don't use the discretisation methods applied for the linear system. The model is defined with a nonlinear continuous equation, we used, hence, the Runge-Kutta approximation for an order of 4.

In order to improve control performance, a steady-state target calculator was included in the setup of the optimizer. Due to the way the architecture of the controller files is presented, the steady-state target calculation was included inside the total cost function. Separate costs were then assigned to the steady-state calculation, the reference tracking of the states and the reference tracking of the system input.

Multiple trials were run with different combinations of costs in order to optimize performance. The optimal costs we found are 1000 for the steady-state calculation (Wss), 1 for minimizing the difference between x and xs (Wxs) and 1 for minimizing the difference between u and us (Wus) for each step. Below are some of the combinations tested (No observable changes were introduced when increasing Wss from 1000 to 5000. Therefore we decided to stay with the lower value of Wss):

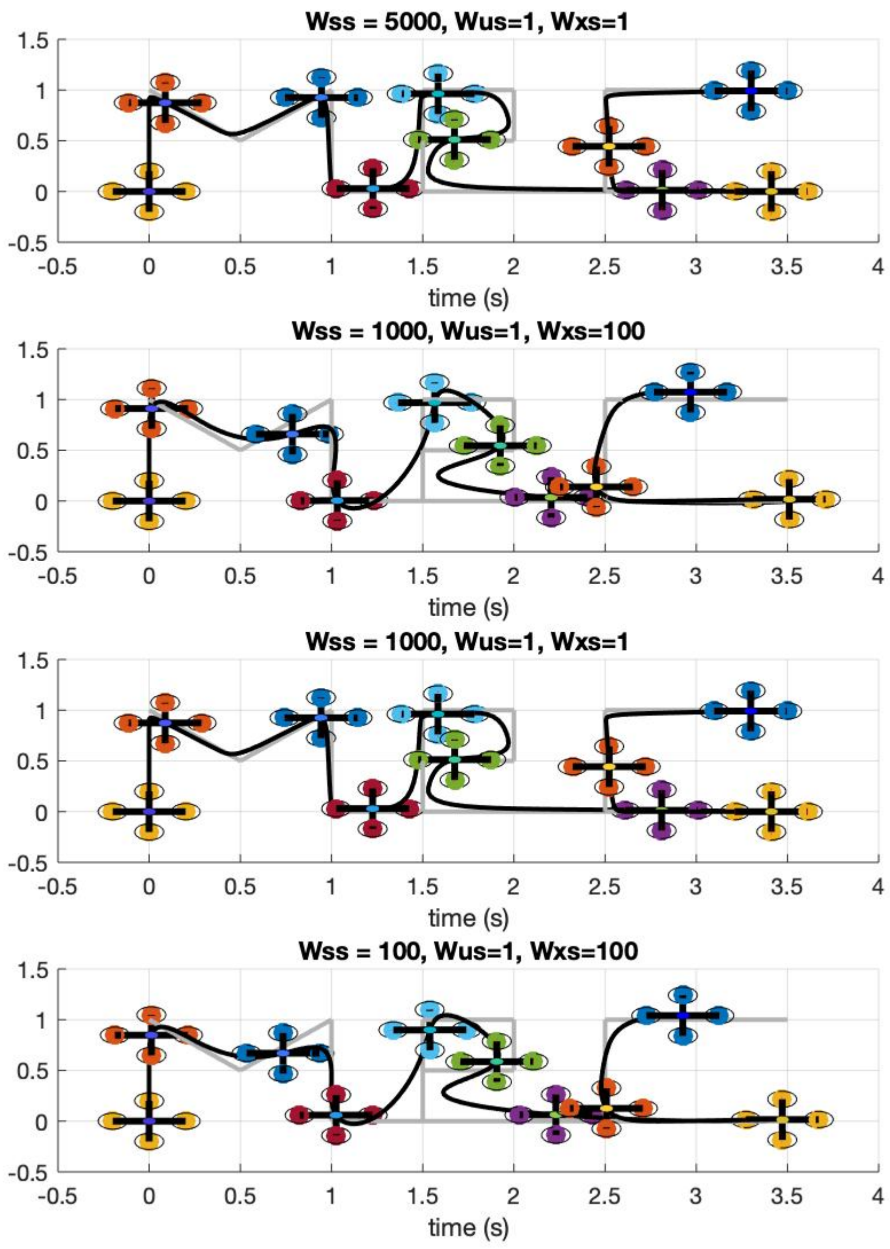


Figure 12: Performance comparison of some costs used to tune the non-linear controller

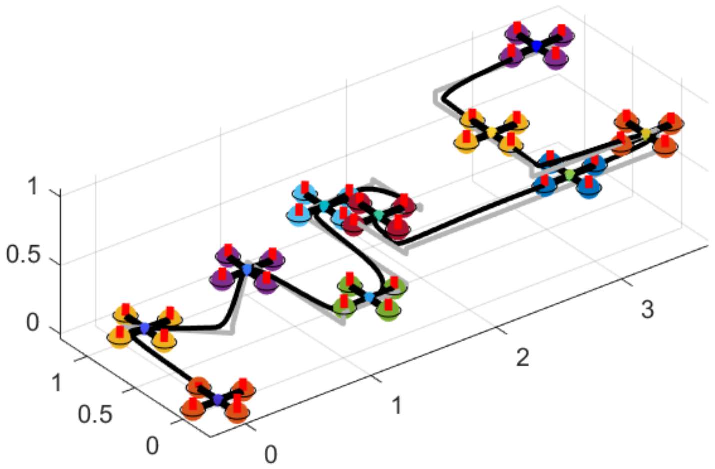


Figure 13: Plot of quadcopter

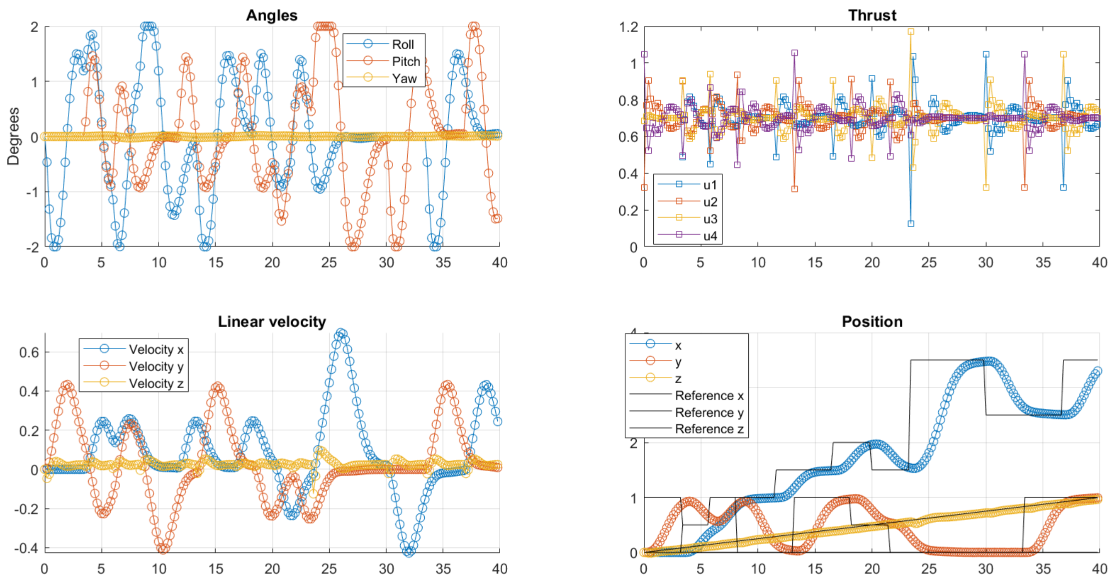


Figure 14: Evolution of thrusts inputs and quad states over time

Why is the nonlinear controller working better than the linear one ?

With the nonlinear model, we avoid problems occuring with the linear problem. In fact, sharp deviations of the system from its operating trajectory can not be supported by the linear controller because of its possible nonlinearities, While, the nonlinear controller, use an accurate model remaining available although the system deviations from the operating trajectory. Hence, it performs better, at the price of more computations.

Need to be mentioned that for the nonlinear system, the problem is no longer convex, because of non-linearities, system may find a local suboptimum, and be non-optimal then. While the linear one always came up with an optimal solution.

7 Conclusion

In this project we were able to control the movement of a quadcopter by implementing a series of MPC control schemes. The first controller managed to stabilize the system to its zero-state, when starting from a non-zero initial state. Then, we augmented the controller such that it was able to perform steady-state calculation - given a target state - and to track said reference. In a further step, a state estimator was added to the controller in order to estimate system states as well as input disturbance. The controller was thus capable of offset-free reference tracking. Along the way, we had to tune a series of gains and weights in an effort to optimize control performance.

As a bonus part, a non-linear MPC controller was implemented, performing reference tracking with steady-state calculations. An overall cost function was devised and the weights of the different errors to be minimized were tuned in order to obtain optimal control operation.

Finally, the different properties of linear and non-linear controllers were compared and key aspects of each one were presented.