POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Optimal trajectory tracking of a quadcopter along a race-track

Master's programme in Automation and Control Engineering
Constrained Numerical optimization for Estimation and Control

, Gabriele Ribolla[†]

Professor:

Co-advisors:

Academic year:
2021-2022

Abstract: This project's aim is to implement a strategy in order to track the time optimal trajectory through multiple waypoints with a quadrotor. The solution that will be proposed is based on the modeling of the quadrotor, the linearization of the model, the study of the optimization problem, and the implementation of a suitable optimal control strategy in order to track the desired trajectory. The mathematical model that has been used is quadrotor model obtained through a Newton approach. The control problem has been developed based on the model and the specifications.

[†]gabriele.ribolla@mail.polimi.it

Contents

1. Introduction

In this paper, we address the problem of flying a quadrotor through multiple waypoints in minimum time. This is often desired for delivery and inspection tasks, and, in the context of search and rescue and drone racing.In this project, we propose an MPC method that considers the full quadrotor dynamics and the real single-rotor thrust constraints to achieve near-time-optimal quadrotor flight. In contrast to standard trajectory tracking control methods, our MPC approach balances the maximization of the progress along a given 3D path and the minimization of the distance to it. The problem is addressed with a four steps approach:

- Modelling of the system
- Optimization problem abstraction and formulation
- Solution outline ad choice of the algorithm
- Testing (simulation and results analysis)

2. Tables of parameters and constraints

Symbol	Description	Value
m	mass	$0.85Kg$
J_x	x Inertial component	$0.0058319\frac{Kg}{m^2}$
J_y	y Inertial component	$0.0058319\frac{Kg}{m^2}$
J_z	z Inertial component	$0.0111886\frac{Kg}{m^2}$
$B = \frac{mg}{4}$	Base weight	$2.784N$
c_T	thrust coefficient	0.127
c_Q	drag coefficient	0.1
l	distance between centre and motor	$0.13m$

Table 1: Model parameters

Symbol	Description	Value
T_s	Sampling Time	$0.5s$
D	Circuit's total distance	$1.860370276320167e+03m$
n_P	number of points of discretized circuit	100
f_{min}	minimum thrust force	0
f_{max}	maximum thrust force	$2B$
ω_{min}	minimum angular velocity	-10
ω_{max}	maximum angular velocity	10
$\Delta v_{\theta min}$	minimum progress "acceleration"	$-\frac{D}{n_P T_s}$
$\Delta v_{\theta max}$	maximum progress "acceleration"	$\frac{D}{n_P T_s}$
Δf_{min}	minimum variation of thrust	$-\frac{2B}{T_s}$
Δf_{max}	maximum variation of thrust	$\frac{2B}{T_s}$
$v_{\theta min}$	minimum progress velocity	$-\frac{D}{n_P T_s^2}$
$v_{\theta max}$	maximum progress velocity	$\frac{D}{n_P T_s^2}$

Table 2: Constraints values

3. System modelling

3.1. Kinematics and Dynamics

Description of the quadrotor movement wrt a fixed reference frame called Earth, through three rotations along the roll,pitch, yaw axes. And for the angular velocities I am going to consider the transfer matrix. The position in the fixed reference frame will be called x_W, y_W, z_W, with the subscript W standing for World, while the position in the moving reference frame will be called x_B, y_B, z_B, with the subscript B standing for Body. Regarding the yaw, pitch and roll angles they will be called as ψ, θ, ϕ, and for the angular velocities in the body reference frame r, q, p.

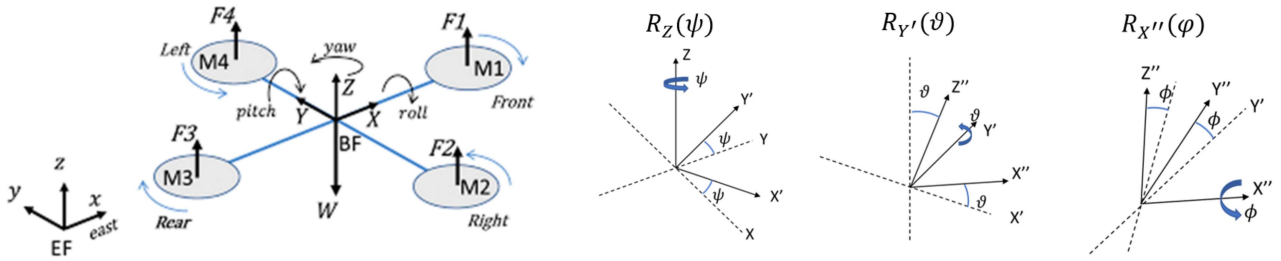

Figure 1: Reference systems

$$R_{BW} = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\theta - \sin\psi\cos\phi & \sin\psi\sin\phi + \cos\psi\cos\phi\cos\theta \\ \cos\theta\sin\theta & \cos\phi\cos\psi + \sin\psi\sin\phi\sin\theta & \sin\psi\sin\phi\sin\theta \\ -\sin\theta & -\cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{1}$$

$$T = \begin{bmatrix} \sin\theta & 0 & 1 \\ -\cos\theta\sin\phi & \cos\phi & 0 \\ \cos\phi\cos\theta & \sin\phi & 0 \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} \dot{x}_W \\ \dot{y}_W \\ \dot{z}_W \end{bmatrix} = R_{BW} \begin{bmatrix} \dot{x}_B \\ \dot{y}_B \\ \dot{z}_B \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = T^{-1} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{4}$$

$$\begin{cases} \dot{x}_W = \cos\psi\cos\theta\dot{x}_B + (\cos\psi\sin\theta\sin\theta - \sin\psi\cos\phi)\dot{y}_B + (\sin\psi\sin\phi + \cos\psi\cos\phi\cos\theta)\dot{z}_B \\ \dot{y}_W = \cos\theta\sin\theta\dot{x}_B + (\cos\phi\cos\psi + \sin\psi\sin\phi\sin\theta)\dot{y}_B + (\sin\theta\cos\phi\sin\psi - \cos\psi\sin\phi)\dot{z}_B \\ \dot{z}_W = -\sin\theta\dot{x}_B - \cos\theta\sin\phi\dot{y}_B + \cos\theta\cos\phi\dot{z}_B \\ \dot{\psi} = \dfrac{-\sin\phi}{\cos\phi}q + r \\ \dot{\theta} = \cos\phi q + \sin\phi r \\ \dot{\phi} = p + \sin\phi\tan\theta q - \cos\phi\tan\theta r \end{cases} \tag{5}$$

The dynamic equations are obtained computing the Newton-Euler formulas and satisfying the conservation of linear and angular momentum. We express the velocity of the quadcopter with v_{quad} as the velocity of the rigid body with respect to the inertial frame,the angular velocity as ω_{quad},the velocity of body with respect to the moving frame as v_B, and the rotational velocity in the moving frame as ω. The input space is composed by the vertical thrust T and the angular torques τ_ψ, τ_θ, τ_ϕ. And the gravitational acceleration is g.

$$m\dot{v}_{quad} = \sum F_i \tag{6}$$

$$m(\dot{v}_B + \omega \times v_B) = R_{BW}^T \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \tag{7}$$

$$\begin{cases} \ddot{x}_B = g\sin\theta + r\dot{y}_B - q\dot{z}_B \\ \ddot{y}_B = -g\sin\phi\cos theta - r\dot{y}_B + p\dot{z}_B \\ \ddot{z}_B = -g\cos theta\cos phi + \dfrac{T}{m} + q\dot{x}_B - p\dot{y}_B \end{cases} \tag{8}$$

$$J\dot{\omega}_{quad} = \sum \tau_i \tag{9}$$

$$J\ddot{\omega} + \omega \times J\omega = \begin{bmatrix} \tau_\psi \\ \tau_\theta \\ \tau_\phi \end{bmatrix} \tag{10}$$

$$\begin{cases} \dot{r} = \dfrac{\tau_\psi}{J_z} - \dfrac{J_y - J_x}{J_z}pq \\ \dot{q} = \dfrac{\tau_\theta}{J_y} - \dfrac{J_x - J_z}{J_y}pr \\ \ddot{p} = \dfrac{\tau_\phi}{J_x} - \dfrac{J_z - J_y}{J_x}qr \end{cases} \tag{11}$$

3.2. Input space and augmented dynamics

The input space given by T and τ_ψ, τ_θ, τ_ϕ is decomposed into single rotor thrusts f=[f_1,f_2,f_3,f_4] where f_i is the thrust at rotor i=1,2,3,4.

$$\begin{aligned} T &= \sum f_i \\ \tau_\psi &= c_\tau(-f_1 + f_2 - f_3 + f_4) \\ \tau_\theta &= (-f_1 + f_3) \\ \tau_\phi &= l(f_2 - f_4) \end{aligned} \tag{12}$$

with the quadrotor's arm length l and the rotor's torque constant c_τ.

3.3. Linearization around Hovering position

From the obtained non linear model we want to perform a linearization around the position of hovering with the euilibrium state coinciding with $T = f_1 + f_2 + f_3 + f_4 = mg$, to have in this way a linear model simpler and faster to simulate in different conditions. In fact we will use either the non linear model either the linearized one in the optimization algorithms in order to have different test cases.

So we have the non linear model:

$$\begin{cases} \dot{x}_W = \cos\psi\cos\theta\dot{x}_B + (\cos\psi\sin\theta\sin\theta - \sin\psi\cos\phi)\dot{y}_B + (\sin\psi\sin\phi + \cos\psi\cos\phi\cos\theta)\dot{z}_B \\ \dot{y}_W = \cos\theta\sin\theta\dot{x}_B + (\cos\phi\cos\psi + \sin\psi\sin\phi\sin\theta)\dot{y}_B + (\sin\theta\cos\phi\sin\psi - \cos\psi\sin\phi)\dot{z}_B \\ \dot{z}_W = -\sin\theta\dot{x}_B - \cos\theta\sin\phi\dot{y}_B + \cos\theta\cos\phi\dot{z}_B \\ \dot{\psi} = \dfrac{-\sin\phi}{\cos\phi}q + r \\ \dot{\theta} = \cos\phi q + \sin\phi r \\ \dot{\phi} = p + \sin\phi\tan\theta q - \cos\phi\tan\theta r \\ \ddot{x}_B = g\sin\theta + r\dot{y}_B - q\dot{z}_B \\ \ddot{y}_B = -g\sin\phi\cos theta - r\dot{y}_B + p\dot{z}_B \\ \ddot{z}_B = -g\cos theta\cos phi + \dfrac{T}{m} + q\dot{x}_B - p\dot{y}_B \\ \dot{r} = \dfrac{\tau_\psi}{J_z} - \dfrac{J_y - J_x}{J_z}pq \\ \dot{q} = \dfrac{\tau_\theta}{J_y} - \dfrac{J_x - J_z}{J_y}pr \\ \ddot{p} = \dfrac{\tau_\phi}{J_x} - \dfrac{J_z - J_y}{J_x}qr \end{cases} \tag{13}$$

and after the linearization we obtain:

$$\begin{cases} \dot{x}_W = \dot{x}_B \\ \dot{y}_W = \dot{y}_B \\ \dot{z}_W = \dot{z}_B \\ \dot{\psi} = r \\ \dot{\theta} = q \\ \dot{\phi} = p \\ \ddot{x}_B = g\theta \\ \ddot{y}_B = -g\phi \\ \ddot{z}_B = \dfrac{(f_1 + f_2 + f_3 + f_4)}{m} \\ \dot{r} = \dfrac{-f1 + f2 - f3 + f4}{J_z} \\ \dot{q} = \dfrac{-f1 + f3}{J_y} \\ \dot{p} = \dfrac{f2 - f4}{J_x} \end{cases} \tag{14}$$

4. Optimization problem formulation

The following approach tryes to achieve the tracking of a reference trajectory along a path. At first it has been parametrized the curve to track in order to keep into consideration the progress along the path, then it was augmented the dynamics of the system for the purpose. The second step was to implement a Finite Horizon Optimal Control Problem considering different conditions and last we tried to implement an MPC to track iteratively the trajectory along the circuit. The MPC implemented consider the higher-level task of minimizing the Euclidean distance to a three-dimensional path while maximizing the speed at which the path is traversed. The path choosen for our project is the race track of Las Vegas

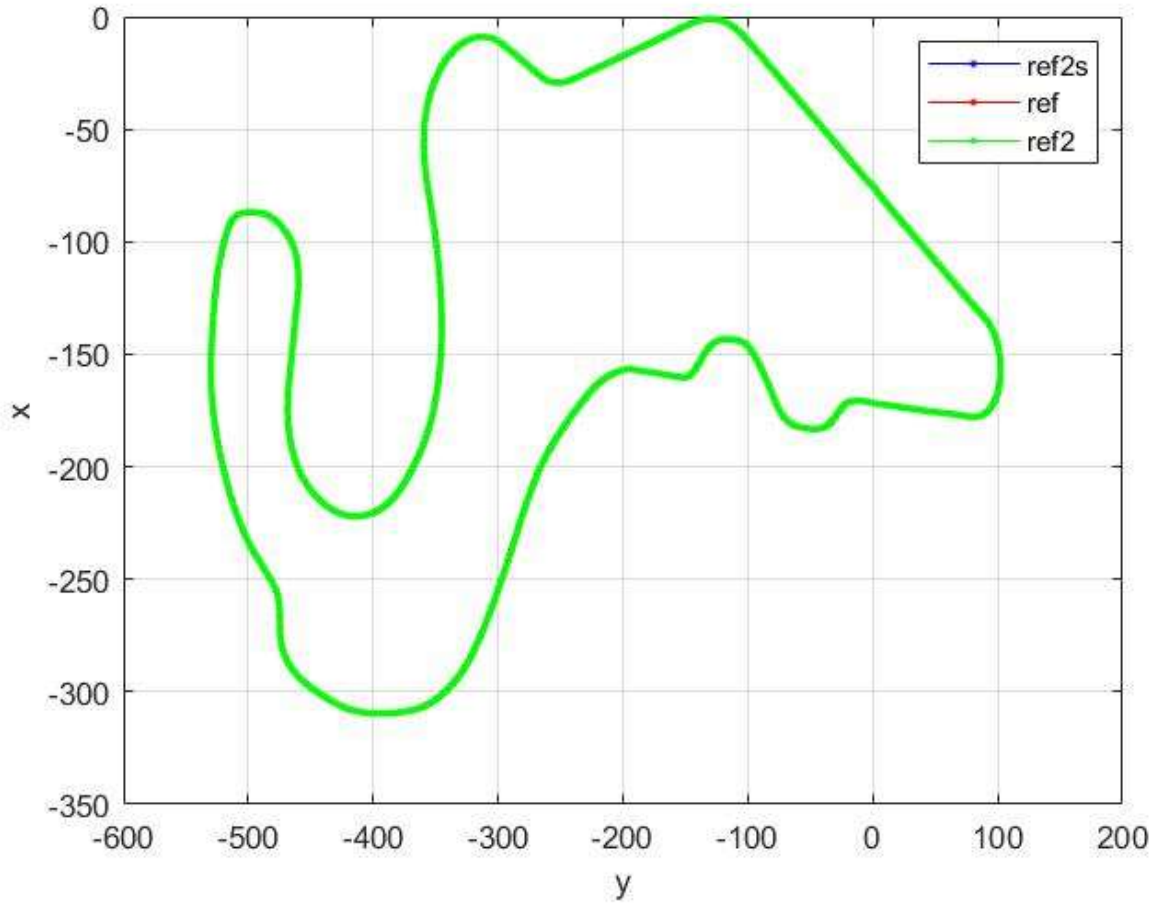

Figure 2: Reference race track

In order to achieve the goal of tracking the given path in the minimum time possible,, it was augmented the system in order to include the dynamics of the progress of the quadcopter with respect to the circuit. So in first place it was done a parametrization of the circuit.

4.1. Path parametrization

Let $p_d(t)$ be the 3D path to track, and it can be given in two forms: as a continuous time-dependent curve or as a sequence of sampled points in time. If it is given as a continuous curve, we use the bisection method from where the curve is sampled such that the arclength between samples is constant by numerically computing the integral and doing binary search until convergence. If it's given as a sequence of sampled points, we search for these equidistant segments by assuming linearity between samples.In both cases, for each point, we store the corresponding arc length, the position, and the

normalized velocity. his process results in a sequence of P of these points to which we fit 3rd order splines $\rho_i(\theta_k)$ i ∈ [0,...,P-1].

$$p_d(\theta_k) = \begin{cases} \rho_0(\theta_k)\text{with} & 0 \leq \theta_k \leq \theta_0 \\ \rho_1(\theta_k)\text{with} & \theta_1 \leq \theta_k \leq \theta_2 \\ \quad\ldots & \\ \rho_{P-1}(\theta_k)\text{with} & \theta_{P-1} \leq \theta_k \leq \theta_P \end{cases} \tag{15}$$

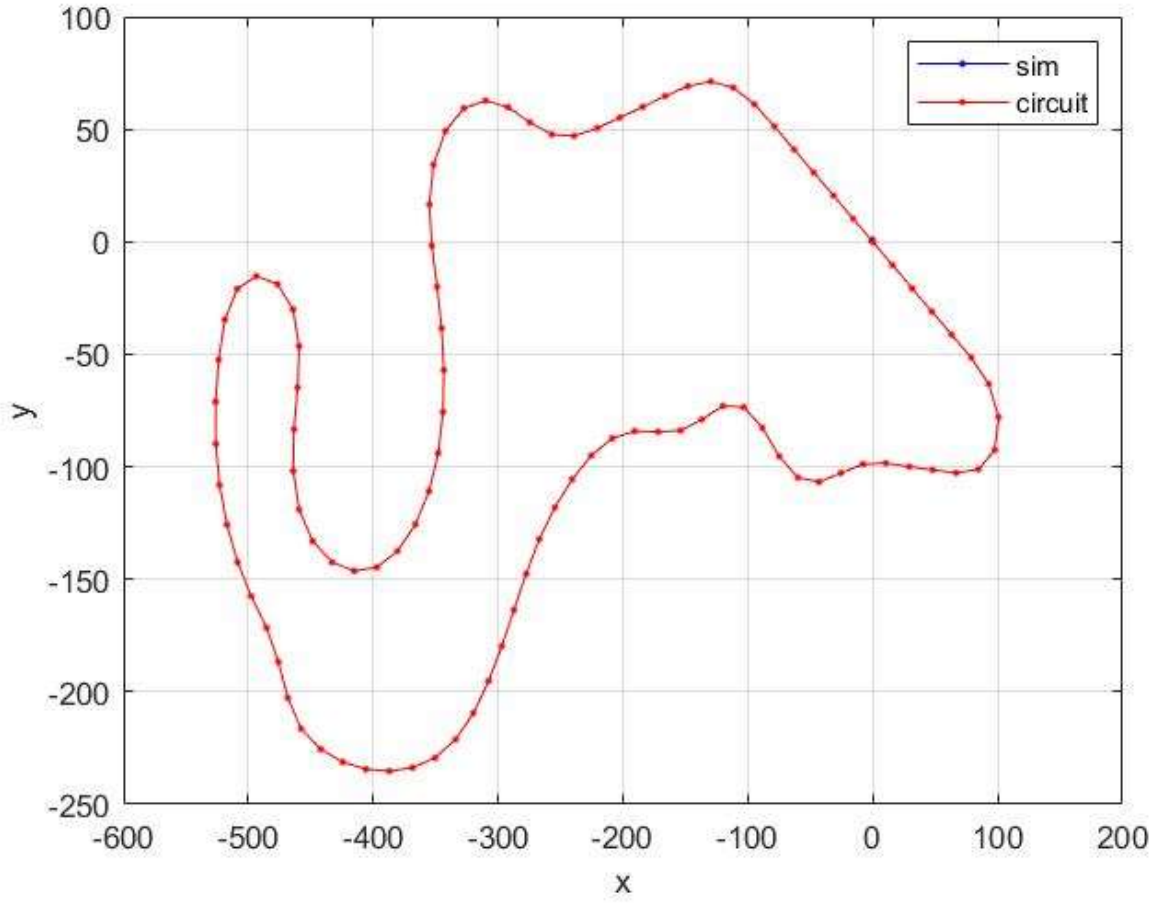

Figure 3: Reference race track

The complete theory for the technique used for the path parametrization is taken in [2]

4.2. Augmented dynamics

In order to augment the dynamics to include the progress into the states a direct way to achieve this is to add θ and v_θ, the velocity with θ varies, as additional states and Δv_θ, acceleration at which theta varies, as a virtual input. By defining the vector of inertial positions as p, the vector of euler angles as q, the vector of linear body velocities as v, the vector of angular body velocities as w, the vector of the rotor thrusts as f, the resulting augmented state and input spaces are shown in (16).

$$\begin{aligned} x &= \begin{bmatrix} p & q & v & w & f & \theta & v_\theta \end{bmatrix}^T \\ u &= \begin{bmatrix} \Delta f & \Delta v_\theta \end{bmatrix}^T \end{aligned} \tag{16}$$

where the dynamics of the augmented states is

$$\begin{aligned} f_{k+1} &= f_k + \Delta f_k \Delta t \\ \theta_{k+1} &= \theta_k + \Delta\theta_k \Delta t \\ v_{\theta,k+1} &= v_{\theta,k} + \Delta v_{\theta,k} \Delta t \end{aligned} \tag{17}$$

where Δv_{θ_k} is the acceleration at which the progress varies , and Δt is the sampling time.

4.2.1 State space system of augmented linear model

In order to simulate the system inside the optimization program we have written the model in state space notation considering the augmented dynamics. As outputs we consider the inertial position of the quacopter, that are the one that the algorithm will try to follow in terms of x_{ref} and y_{ref}.

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & g & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{m} & \frac{1}{m} & \frac{1}{m} & \frac{1}{m} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{\frac{c_t}{c_q}}{J_z} & \frac{\frac{c_t}{c_q}}{J_z} & -\frac{\frac{c_t}{c_q}}{J_z} & \frac{\frac{c_t}{c_q}}{J_z} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{l}{J_y} & 0 & -\frac{l}{J_y} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{l}{J_x} & 0 & \frac{l}{J_x} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} D = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{19}$$

4.3. Finite Horizon Optimal Control Problem

Different conditions have been tried to implement the Finite Horizon approach, in particular have been developed two different kinds of cost functions, a quadratic one and a non linear one. For what concerns the constraints, also in this case have been considered a single shooting approach, in which the variables optimized are only the control trajectory U, and a multiple shooting approach where the optimization variables are both the control trajectory U and the state trajectory $X = [z(1|t)^T, ..., z(N|t)^T]^T$, and the consistency of these trajectory with the system equations is enforced via constraints. Moreover the simulation of the dynamical system is performed on the linearized model .

4.3.1 Quadratic cost

The cost function choosen is a quadratic one in order to have a convex problem, in which is guaranteed to have a minimum in the optimization phase, and so a feasible solution. In this problem formulation we have the cost function composed by the first two terms which penalizes the difference between the coordinates desired and the actual ones, the cost on ω necessary for the stability of the solution since it forces the solver to keep low body rates whenever possible.

Single shooting with linear model

$$\begin{aligned}
\min_{U} \quad & \sum_{i=0}^{N} \|y_{ref} - y\|^2_{qy} + \|x_{ref} - x\|^2_{qx} + \|\omega_k\|^2_{q\omega} + \|\Delta v_{\theta_k}\|^2_{r\Delta v} + \|\Delta f_k\|^2_{r\Delta f} - \mu v_{\theta_k} \\
& z(i+1|t) = Az(i|t) + Bu(i|t), i = 0, ..., N-1 \\
& y(i|t) = Cz(i|t) + Du(i|t), i = 0, ..., N \\
\text{subj. to} \quad & \Delta v_{\theta min} \leq \Delta v_{\theta} \leq \Delta v_{\theta max} \\
& \Delta f_{min} \leq \Delta f \leq \Delta f_{max} \\
& z(0|t) = z_0
\end{aligned} \tag{20}$$

Multiple shooting with linear model

$$\begin{aligned}
\min_{U,X} \quad & \sum_{i=0}^{N} \|y_{ref} - y\|^2_{qy} + \|x_{ref} - x\|^2_{qx} + \|\omega_k\|^2_{q\omega} + \|\Delta v_{\theta_k}\|^2_{r\Delta v} + \|\Delta f_k\|^2_{r\Delta f} - \mu v_{\theta_k} \\
& z(i+1|t) = Az(i|t) + Bu(i|t), i = 0, ..., N-1 \\
& y(i|t) = Cz(i|t) + Du(i|t), i = 0, ..., N \\
& \omega_{min} \leq \omega \leq \omega_{max} \\
& f_{min} \leq f \leq f_{max} \\
\text{subj. to} \quad & 0 \leq v_{\theta} \leq v_{\theta max} \\
& \Delta v_{\theta min} \leq \Delta v_{\theta} \leq \Delta v_{\theta max} \\
& \Delta f_{min} \leq \Delta f \leq \Delta f_{max} \\
& z(0|t) = z_0
\end{aligned} \tag{21}$$

Constraints and costs on v_θ and f have been added such that the solver does not have complete freedom of choosing them arbitrarily. Lastly, the limits on v_θ ensure non-reversal of the reference path and set a maximum speed at which the path is traversed.[1]

4.3.2 Non linear cost

The non linear cost implies the definition of lag and contour error.[1]

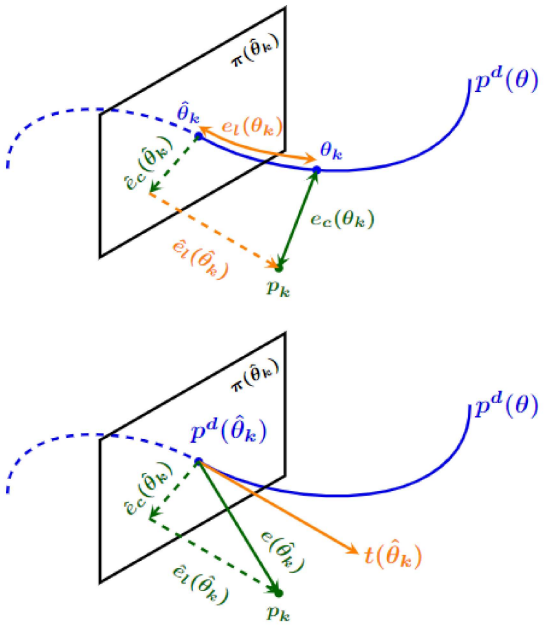

Figure 4: Top: definition of contour error and its approximation, shown in green, and lag error and its approximation, shown in orange pk is the current position of the platform at time k and $p_d(\theta_k)$ is the reference position at time k. Bottom: illustration of position error $e_c(\theta_k)$ and tangent vector $t(\theta_k)$.

$e_c(\theta_k)$ is the contour error at step k and is defined as the projection of p_k over $p_d(\theta_k)$

$$e_c(\theta_k) = \|p_k - p^d(\theta_k)\| \tag{22}$$

the contour error $e_c(\theta_k)$ is approximated by the norm of its linear projection $\hat{e}_c(\theta_k)$ onto the tangent plane $t(\theta_k)$. The arc length of the true projection θ_k and the arc length of the approximated projection $\hat{\theta}_k$ are linked by the lag error $e_l(\theta_k) = \|\theta_k - \hat{\theta}_k\|$. However, since the true projection θ_k is not known, the true lag error $e_l(\theta_k)$ is also approximated by the norm of $\hat{e}_l(\theta_k)$. Fig. 5 (top) depicts a graphical interpretation of $e_c(\theta_k)$,$e_l(\theta_k)$ and their approximations.
Let the position error at time k be $e(\theta_k) = \|p_k - p^d(\theta_k)\|$.Consider the tangent line of the desired path,$t(\theta_k) \in R^3$. $e(\theta_k)$ can be decomposed in a component that is projected onto $t(\theta_k)$,$e_l(\theta_k)$, and a component contained in $\pi(\theta_k)$,which is $e_c(\theta_k)$.

$$e_c(\theta_k) = \begin{bmatrix} 1 - t_x^2(\theta_k) & -t_x(\theta_k)t_y(\theta_k) & -t_x(\theta_k)t_z(\theta_k) \\ -t_x(\theta_k)t_y(\theta_k) & 1 - t_y^2(\theta_k) & -t_y(\theta_k)t_z(\theta_k) \\ -t_x(\theta_k)t_z(\theta_k) & -t_y(\theta_k)t_z(\theta_k) & 1 - t_z^2(\theta_k) \end{bmatrix} e(\theta_k) \tag{23}$$

$$e_l(\theta_k) = e(\theta_k) - e_c(\theta_k) \tag{24}$$

In our case, the objective is to follow a trajectory in terms of x and y, keeping a fixed height, so the Tangent matrix will take into account only the tangent terms of x and y. To evaluate this matrix we consider 2 different ways. The first considering a linearization of the derivative $t_x = \frac{(x_d(\theta_k) - x_k)}{\|(x_d(\theta_k) - x_k)\|_2}$ and similarity for the $t_y = \frac{(y_d(\theta_k) - y_k)}{\|(y_d(\theta_k) - y_k)\|_2}$.
The second instead using the derivative of the cubic spline used to interpolate the entire circuit $t_x = 3a(x_d(\theta_k))(\theta_k - \theta_k in(\theta_k)))^2 + 2b(x_d(\theta_k))(\theta_k - \theta_k in(\theta_k))) + c(x_d(\theta_k))$ and $t_y = 3a(y_d(\theta_k))(\theta_k - \theta_k in(\theta_k)))^2 + 2b(y_d(\theta_k))(\theta_k - \theta_k in(\theta_k))) + c(y_d(\theta_k))$. We tried both to have different ways to compute the cost function for the optimization problem.

Single shooting with linear model

$$
\begin{aligned}
\min_{U} \quad & \sum_{i=0}^{N} \|e_c(\theta_k)\|_{qc}^2 + \|e_l(\theta_k)\|_{ql}^2 + \|\omega_k\|_{q\omega}^2 + \|\Delta v_{\theta_k}\|_{r\Delta v}^2 + \|\Delta f_k\|_{r\Delta f}^2 - \mu v_{\theta_k} \\
& z(i+1|t) = Az(i|t) + Bu(i|t), i = 0, ..., N-1 \\
& y(i|t) = Cz(i|t) + Du(i|t), i = 0, ..., N \\
\text{subj. to} \quad & \Delta v_{\theta min} \leq \Delta v_{\theta} \leq \Delta v_{\theta max} \\
& \Delta f_{min} \leq \Delta f \leq \Delta f_{max} \\
& z(0|t) = z_0
\end{aligned}
\tag{25}
$$

Multiple shooting with linear model

$$
\begin{aligned}
\min_{U} \quad & \sum_{i=0}^{N} \|e_c(\theta_k)\|_{qc}^2 + \|e_l(\theta_k)\|_{ql}^2 + \|\omega_k\|_{q\omega}^2 + \|\Delta v_{\theta_k}\|_{r\Delta v}^2 + \|\Delta f_k\|_{r\Delta f}^2 - \mu v_{\theta_k} \\
& z(i+1|t) = Az(i|t) + Bu(i|t), i = 0, ..., N-1 \\
& y(i|t) = Cz(i|t) + Du(i|t), i = 0, ..., N \\
& \omega_{min} \leq \omega \leq \omega_{max} \\
& f_{min} \leq f \leq f_{max} \\
\text{subj. to} \quad & 0 \leq v_{\theta} \leq v_{\theta max} \\
& \Delta v_{\theta min} \leq \Delta v_{\theta} \leq \Delta v_{\theta max} \\
& \Delta f_{min} \leq \Delta f \leq \Delta f_{max} \\
& z(0|t) = z_0
\end{aligned}
\tag{26}
$$

5. Solution algorithm

The choice for the solution algorithm has fallen, given the varieties of conditions and costs that we have introduced, into the so called Sequential Quadratic Algorithm, that is used to solve the most general class of optimization problem, The Non Linear Constrained Programs. We have decided to use this algorithm also for the case in which we have a quadratic cost because we encountered some computational and plotting problem trying to solve it with the quadratic program algorithm. For the rest of cases we try to minimize a non linear cost function,plus we can have in some of the cases, the simulation that is computed over a non linear dynamical system. The main blocks of the algorithm are:

- Gradient computation
- Back Tracking line search algorithm
- A QP solver(Matlab's one)
- An overall SQP algorithm embedding the QP and the line search with merit function

For these main blocks we have used the codes given in the laboratory's sessions during the classes. At each iteration of the SQP algorithm, a QP is solved to obtain search directions p_k in the space of primal variables (i.e. the decision variable x) and of the Lagrange multipliers λ_k, μ_k respectively for equality and inequality constraints).

$$\begin{aligned} \min_{p_k} & \nabla_x f(x^k)^T p^k + \frac{1}{2} p^{kT} H^k p^k \\ \text{subj. to} \quad & \nabla_x g(x^k)^T p^k + g(x^k) = 0 \\ & \nabla_x h(x^k)^T p^k + h(x^k) \geq 0 \end{aligned} \tag{27}$$

To solve the QP has been used the solver provided by Matlab quadprog. Within this the Interior Point method was imposed in order to deal with the inequality constraints of the primal optimization problem. The QP also requires the definition of a suitable Hessian matrix H^k, which approximates the actual Hessian of the Lagrangian $\nabla_x^2 L(p_k, \lambda_k, \mu_k)$. For the estimation has been employed the BFGS update formula

- Lagrangian gradients $\nabla_x L(x^{k+1}, \lambda^{k+1}, \mu^{k+1})$, $\nabla_x L(x^k, \lambda^{k+1}, \mu^{k+1})$ then compute

$$\begin{aligned} y &= \nabla_x L(x^{k+1}, \lambda^{k+1}, \mu^{k+1}) - \nabla_x L(x^k, \lambda^{k+1}, \mu^{k+1}) \\ s &= x^{k+1} - x^k \end{aligned} \tag{28}$$

- Compute $H^{k+1} = H^k - \frac{H^k s s^T H^k}{s^T H^k s} + \frac{y y^T}{s^T y}$

After the QP has been solved,the update rule is performed:

$$\begin{aligned} x_{k+1} &= x_k + t_k p_k \\ \lambda_{k+1} &= \lambda_k + t_k \Delta\lambda_k \\ \mu_{k+1} &= \mu_k + t_k \Delta\mu_k \end{aligned} \tag{29}$$

Then the line search is carried out by using a merit function and its directional derivative, that can be both computed based on the local information (cost, constraints, and their gradients) available at each iteration.[3]
$\min_x T_1(x) = f(x) + \sum_i^p \sigma_i |g_i(x)| + \sum_i \tau_i \max(0, -h_i(x))$

6. Simulation results

6.1. FHOCP with quadratic cost

6.1.1 Simulation over the linearized dynamical model

Single shooting

In order to simulate this result, between the code given, It has to be selected the folder *FHOCP_quad_cost_single_shooting*.

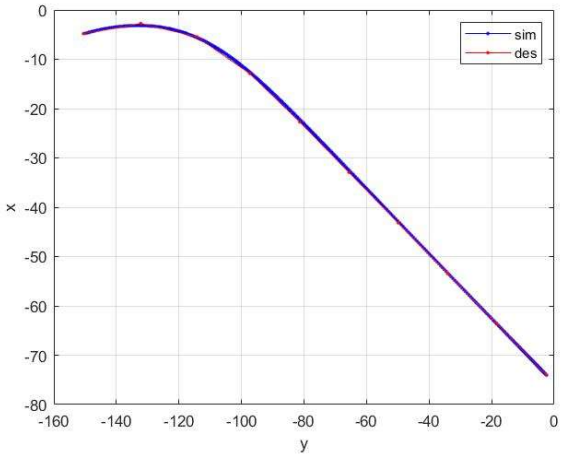

Figure 5: Simulated for 11 steps forward with sampling time equal to 0.5 seconds

Multiple shooting

In order to simulate this result, between the code given, It has to be selected the folder *FHOCP_quad_cost_multiple_shooting*

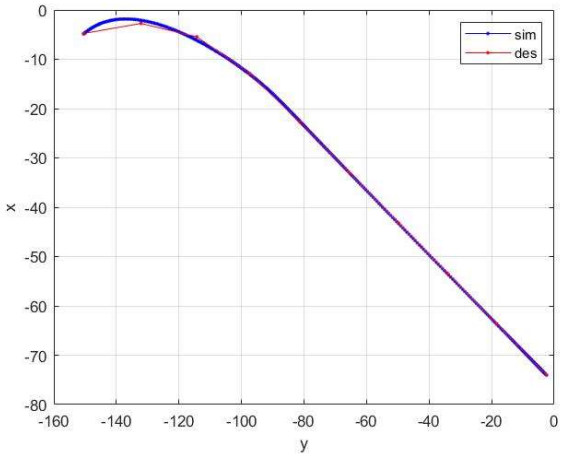

Figure 6: Simulated for 11 steps forward with sampling time equal to 0.5 seconds

6.2. FHOCP with non linear cost

The simulations taken with the non linear cost have been implemented with a smaller sampling time.

6.2.1 Simulation over the linearized dynamical model

Results considering the t_x, t_y computed using the first method

Single shooting

In order to simulate this result, between the code given, It has to be selected the folder *FHOCP_non_lincost_single_shooting*.

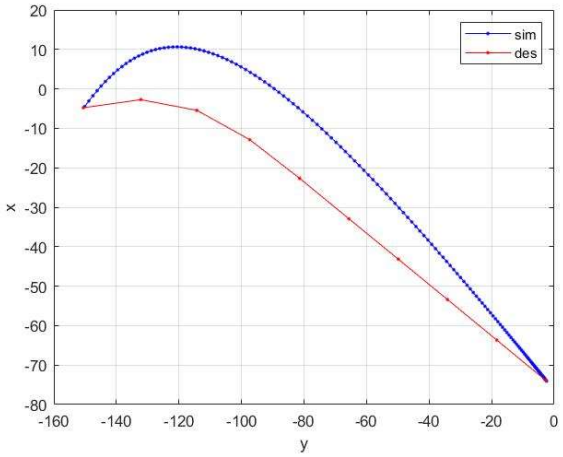

Figure 7: Simulated for 11 steps forward with sampling time equal to 0.2 seconds

Multiple shooting

In order to simulate this result, between the code given, It has to be selected the folder *FHOCP_non_lincost_multiple_shooting*.

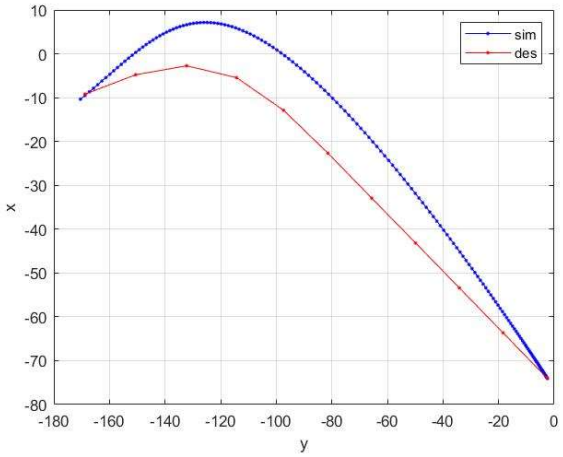

Figure 8: Simulated for 11 steps forward with sampling time equal to 0.2 seconds

Results considering the t_x, t_y computed using the second method For this method, it is introduced a dynamic cost $q_c(p_d(\theta_k))$ for the contour error that changes its value with respect to the distance of the simulated position to the position of desired points, called 'way-points'. The goal of it is to be used in a future development of the MPCC (model predictive control of the circuit's contour). It is defined in the following way:

- for the x component :

$$q_c(x_d(\theta_k)) = \sum_{j=0}^{M} \frac{e^{-\frac{1}{2}(x_d(\theta_k)-x_w^j(\theta_k)' \sum_x^- 1)(x_d(\theta_k)-x_w^j(\theta_k))}}{\sqrt{(2\pi)^3|\sum_x|}}$$

- for the y component :

$$q_c(y_d(\theta_k)) = \sum_{j=0}^{M} \frac{e^{-\frac{1}{2}(y_d(\theta_k)-y_w^j(\theta_k)' \sum_x^- 1)(y_d(\theta_k)-y_w^j(\theta_k))}}{\sqrt{(2\pi)^3|\sum_y|}}$$

where :

- $\sum_y$, is the covariance of the y of the way-points considered;
- $\sum_x$, is the covariance of the x of the way-points considered;
- $\sum_{j=0}^{M} p_w^j$ is the set of point $p_w^j(x_w^j, y_w^j)$

In the following simulation results the way-points considered are the Np points from the initial point of the circuit.

Single shooting

In order to simulate this result, between the code given, It has to be selected the folder *FHOCP_non_lincost_spline_single_shooting*.

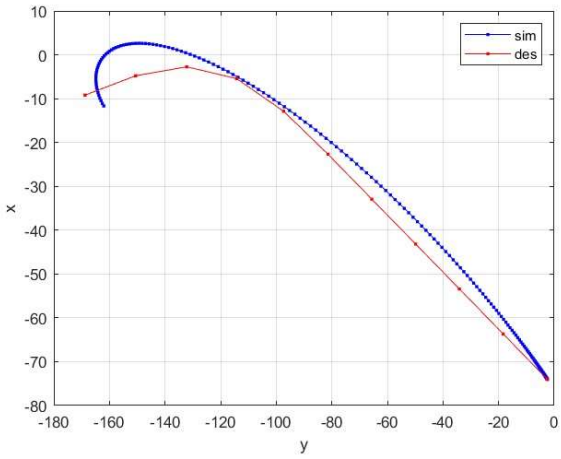

Figure 9: Simulated for 11 steps forward with sampling time equal to 0.2 seconds

Multiple shooting

In order to simulate this result, between the code given, It has to be selected the folder *FHOCP_non_lincost_spline_multiple_shooting*.

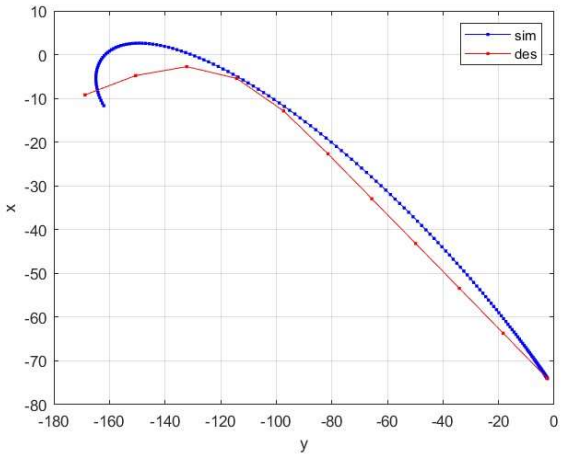

Figure 10: Simulated for 11 steps forward with sampling time equal to 0.2 seconds

6.3. MPC with non linear cost

6.3.1 Simulation over the linearized dynamical model

Results considering the tx, ty computed using the first method

Single shooting

In this simulation the $q_c(p_d(\theta_k))$ is not considered. In order to simulate this result, between the code given, It has to be selected the folder
MPC_non_lincost_1st_spline_multiple_shooting.

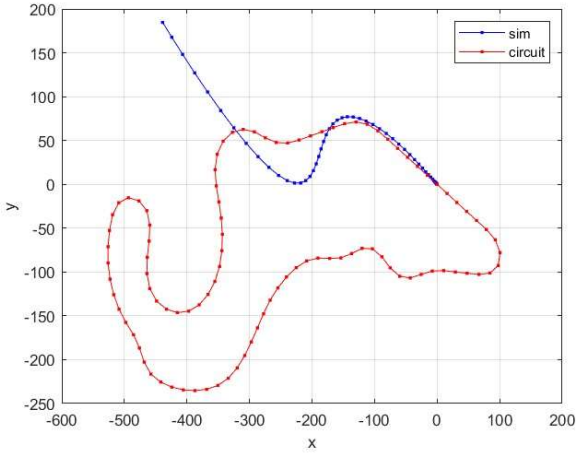

Figure 11: Simulated for 18 steps forward with sampling time equal to 0.3 seconds

7. Conclusions

The primal aim of this project work was to implement a non linear model predictive control for the tracking of the complete circuit. The objectives for the implementation of the finite horizon strategies have been completed, but we encountered several problems in the implementation of iterative procedure for the receding horizon strategy. We reached just a first result implementing an MPC with non linear cost that simulates a first part of the circuit. We tried also different approaches with the same order with which we have implemented the FHOCP strategies but at the current state we didn't manage to simulate these codes without problems. However we attach in the files also the codes for the other strategies that we wanted to develop for the MPC, with the hope that we can continue and improve the work in the future.

8. Bibliography

References

[1] Philipp Foehn Davide Scaramuzza Angel Romero, Sihao Sun. Model predictive contouring control fornear-time-optimal quadrotor flight. 21 June 2022.

[2] Kendall Atkinson Hongling Wang, Joseph Kearney. Arc-length parameterized spline curves for real-time simulation. 2002.

[3] Marco Lauricella Lorenzo Mario Fagiano. Laboratory session g - constrained continuous nonlinear programming via sqp: algorithm and application to finite horizon optimal control problem. September 12, 2021.