UNIVERSITÀ DI BOLOGNA

School of Engineering

Master Degree in Automation Engineering

Optimal Control

Course Project – Group 5
Optimal Control of a Quadrotor with Suspended Load

Professor: **Giuseppe Notarstefano**

Students: Friz Anna
Ruscelli Gabriele
Valeri Eleonora

Academic year 2023/2024

Abstract

This project addresses the optimal control of a planar quadrotor with a suspended load. Tasks involve dynamics discretization, trajectory generation through the Newton's method algorithm, trajectory tracking via Linear Quadratic Regulator (LQR) and Model Predictive Control (MPC). Finally, the quadrotor movement will be shown in an animation.

Contents

Introduction **6**

1 Task 0 - Problem setup **7**
1.1 State space and control inputs 7
1.2 Forward Euler discretization 8
1.3 Jacobians computation 9
1.4 *Dynamics* function 10

2 Task 1 - Trajectory generation (I) **11**
2.1 Equilibrium Computation 11
2.2 Reference trajectory definition 12
2.3 Linear Quadratic Regularization 14
2.3.1 Sketch of the algorithm 14
2.3.2 Costs definition 14
2.4 Results 15
2.4.1 Optimal trajectory 15
2.4.2 Suboptimal trajectories 18
2.4.3 Cost and Descent direction plot 21

3 Task 2 - Trajectory generation (II) **23**
3.1 Results 24
3.1.1 Optimal trajectory 28
3.1.2 Suboptimal trajectories 30

4 Task 3 - Trajectory tracking via LQR **34**
4.1 Q and R definition 34
4.2 Closed loop LQR 34
4.2.1 Plots 35

5 Task 4 - Trajectory tracking via MPC **39**
5.1 Model Predictive Control 39
5.1.1 The optimal control problem to be solved at each t . . 40
5.1.2 The Linear Quadratic Case 40
5.2 Tracking of (x_{opt}, u_{opt}) 41

5.3 Results 42
5.3.1 System trajectory and desired optimal trajectory . . . 42
5.3.2 Tracking error for different initial conditions 47

6 Task 5 - Animation **53**

Conclusions **57**

Introduction

In this project, we are required to design an optimal trajectory for a quadrotor with a suspended load.

The project consists of four main steps:

- Task 0 - Problem Setup: Discretization of the quadrotor dynamics and writing of the iscrete-time state-space equations.
- Task 1 - Trajectory Generation I: Design of an optimal trajectory to move from one equilibrium configuration to another, using a Newton's like algorithm.
- Task 2 - Trajectory Generation II: Generation of a desired (smooth) state−input curve and application of the trajectory generation task (Task 1) on this new desired trajectory.
- Task 3 - Trajectory Tracking via LQR: Linearization of the model around the optimal trajectory obtained in the previous point and definition of the optimal feedback controller to perform trajectory tracking through LQR algorithm.
- Task 4 - Trajectory tracking via MPC: Exploitation of an MPC algorithm to track the optimal trajectory (x_{opt}, u_{opt}) computed in Task 2 linearizing the vehicle dynamics about it.
- Task 5 - Animation: Producing a simple animation of the vehicle executing Task 3.

Task 0 - Problem setup

In task 0 we want to discretize the dynamics, write the discrete-time state-space equations and code the dynamics function.

1.1 State space and control inputs

The system can be modeled as a planar quadrotor with a downward pendulum attached at its center of mass, representing the load, as shown in Figure 1.1.

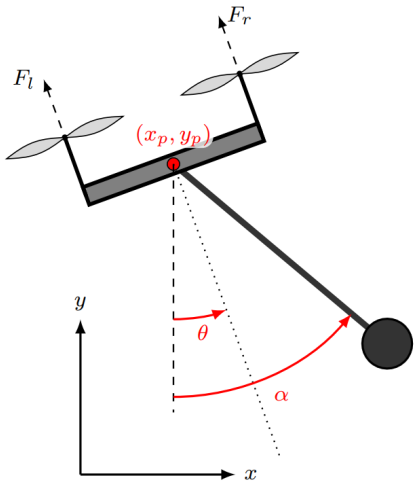

Figure 1.1: Model of a planar quadrotor with a suspended load

The state space consist in

$$x = [x_p, y_p, \alpha, \theta, v_x, v_y, \omega_\alpha, \omega_\theta]^T$$

where $(x_p, y_p) \in \mathbb{R}^2$ represent the position of the center of mass, θ the roll angle of the quadrotor, α the angle of the pendulum with respect to the inertial frame, (v_x, v_y) the velocity of the center of mass, ω_α and ω_θ angular rate of changes associated to α and θ, respectively.

The input is

$$u = [F_s, F_d]^T$$

where

$$\begin{cases} F_s = F_l + F_r \\ F_d = F_r - F_l \end{cases}$$

and (F_l, F_r) are the forces generated by the propellers.

The dynamical model is described by:

$$\begin{cases} (M+m)\dot{v}_x = mL\omega_\alpha^2 sin(\alpha) - F_s(sin(\theta) - \frac{m}{M}sin(\alpha-\theta)cos(\alpha)) \\ (M+m)\dot{v}_y = -mL\omega_\alpha^2 cos(\alpha) + F_s(cos(\theta) + \frac{m}{M}sin(\alpha-\theta)sin(\alpha)) - \frac{m}{M}g \\ ML\dot{\omega}_\alpha = -F_s sin(\alpha-\theta) \\ J\dot{\omega}_\theta = lF_d \end{cases}$$

Where, all the parameters of the quadrotor are:

M: mass of the drone	0.028 [kg]
m: mass of the load	0.04 [kg]
J: inertia of the drone	0.001 $[kgm^2]$
g: gravity	9.81 $[m/s^2]$
L: length of the pendulum	0.2 [m]
l: distance from center of mass to propeller	0.5 [m]

1.2 Forward Euler discretization

In order to write the discrete-time state-space equations, we consider the discrete time version of the drone dynamics, discretized via forward Euler:

$$x_{t+1} = x_t + \delta f_{CT}(x_t, u_t, t)$$

where $\delta > 0$ is a sufficiently small discretization step and $\delta f_{CT}(.,.,.)$ is the continuous-time dynamics. In particular, we choose $\delta = 1e^{-2}$.

We obtain the following discrete-time state-space equations:

$$\begin{cases} x_{1,t+1} = x_{1,t} + \delta x_{5,t} \\ x_{2,t+1} = x_{2,t} + \delta x_{6,t} \\ x_{3,t+1} = x_{3,t} + \delta x_{7,t} \\ x_{4,t+1} = x_{4,t} + \delta x_{8,t} \\ x_{5,t+1} = x_{5,t} + \frac{\delta}{m+M}[mLx_{7,t}^2 sin(x_{3,t}) - u_{1,t}(sin(x_{4,0}) + \\ \qquad\qquad -\frac{m}{M}sin(x_{3,t}-x_{4,t})cos(x_{3,t}))] \\ x_{6,t+1} = x_{6,t} + \frac{\delta}{m+M}[(-mLx_{7,t}^2 cos(x_{3,t}) + u_{1,t}(cos(x_{4,0}) + \\ \qquad\qquad +\frac{m}{M}sin(x_{3,t}-x_{4,t})sin(x_{3,t}))) - g] \\ x_{7,t+1} = x_{7,t} + \frac{\delta}{ML}[-u_{1,t}sin(x_{3,t}-x_{4,t})] \\ x_{8,t+1} = x_{8,t} + \frac{\delta}{J}lu_{2,t} \end{cases}$$

In which state and input vectors are:

$$\mathbf{x}_t = \begin{bmatrix} x_{p,t} \\ y_{p,t} \\ \alpha_t \\ \theta_t \\ v_{x,t} \\ v_{y,t} \\ \omega_{\alpha,t} \\ \omega_{\theta,t} \end{bmatrix} = \begin{bmatrix} x_{1,t} \\ x_{2,t} \\ x_{3,t} \\ x_{4,t} \\ x_{5,t} \\ x_{6,t} \\ x_{7,t} \\ x_{8,t} \end{bmatrix}$$

$$\mathbf{u}_t = \begin{bmatrix} F_{s,t} \\ F_{d,t} \end{bmatrix} = \begin{bmatrix} u_{1,t} \\ u_{2,t} \end{bmatrix}$$

1.3 Jacobians computation

For future purposes the gradient with respect to state and input are calculated.

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & a_{3,5} & a_{3,6} & a_{3,7} & 0 \\ 0 & 0 & 0 & 1 & a_{4,5} & a_{4,6} & a_{4,7} & 0 \\ \delta & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & \delta & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \delta & 0 & a_{7,5} & a_{7,6} & 1 & 0 \\ 0 & 0 & 0 & \delta & 0 & 0 & 0 & 1 \end{bmatrix}$$

with

$$a_{3,5} = \frac{\delta}{m+M}(mLx_{7,t}^2 cos(x_{3,t}) + u_{1,t}\frac{m}{M}cos(2x_{3,t} - x_{4,t}))$$

$$a_{4,5} = \frac{\delta u_{1,t}}{m+M}(cos(x_{4,t}) + \frac{m}{M}cos(x_{3,t} - x_{4,t})cos(x_{3,t}))$$

$$a_{7,5} = \frac{2\delta}{m+M}mLx_{7,t}sin(x_{3,t})$$

$$a_{3,6} = \frac{\delta u_{1,t} m}{m+M}(cos(x_{3,t} - x_{4,t})sin(x_{4,t}) + sin(x_{3,t} - x_{4,t})cos(x_{4,t}))$$

$$a_{4,6} = \frac{\delta}{m+M}(mLx_{7,t}^2 sin(x_{4,t}) - u_{1,t}sin(x_{4,t}) + \frac{m}{M}cos(x_{3,t} - x_{4,t})sin(x_{3,t}))$$

$$a_{7,6} = \frac{-2\delta}{m+M}(mLcos(x_{4,t})x_{7,t}))$$

$$a_{3,7} = \frac{\delta}{ML}(-u_{1,t}cos(x_{3,t} - x_{4,t}))$$

$$a_{4,7} = \frac{\delta}{ML}(u_{1,t}cos(x_{3,t} - x_{4,t}))$$

and

$$\mathbf{B}^T = \begin{bmatrix} 0 & 0 & 0 & 0 & b_{1,5} & b_{1,6} & b_{1,7} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta l/J \end{bmatrix}$$

with

$$b_{1,5} = -\delta(sin(x_{4,t}) - \frac{m}{M}sin(x_{3,t} - x_{4,t})cos(x_{3,t}))$$

$$b_{1,6} = \delta(cos(x_{4,t}) + \frac{m}{M}sin(x_{3,t} - x4, t)sin(x_{4,t}))$$

$$b_{1,7} = \frac{-\delta}{ML}sin(x_{3,t} - x_{4,t})$$

1.4 *Dynamics* function

We can now code a dynamics function that takes as inputs $x_t \in \mathbb{R}^8, u_t \in \mathbb{R}^2$ and returns as output the next state $x_{t+1} \in \mathbb{R}^8$ and the gradient of the dynamics $\mathbf{A} \in \mathbb{R}^{8x8}, \mathbf{B}^T \in \mathbb{R}^{2x8}$

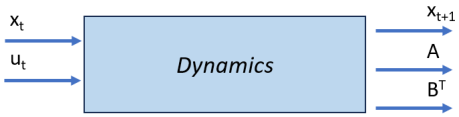

Figure 1.2: *Dynamics* function scheme block

Task 1 - Trajectory generation (I)

In task 1 we want to compute two equilibria for our system, define a reference curve between them and then compute the optimal transition to move from one equilibrium to another exploiting the Newton's−like algorithm for optimal control.

2.1 Equilibrium Computation

To compute the equilibria of the system, we set the derivatives of the system to zero:

$$\begin{cases} x_{5,t+1} - x_{5,t} = \dfrac{\delta}{m+M}\big[mLx_{7,t}^2 \sin(x_{3,t}) + \\ \qquad - u_{1,t}\big(\sin(x_{4,0}) - \dfrac{m}{M}\sin(x_{3,t} - x_{4,t})\cos(x_{3,t})\big)\big] = 0 \\ x_{6,t+1} - x_{6,t} = \dfrac{\delta}{m+M}\big[(-mLx_{7,t}^2 \cos(x_{3,t}) + \\ \qquad + u_{1,t}\big(\cos(x_{4,0}) + \dfrac{m}{M}\sin(x_{3,t} - x_{4,t})\sin(x_{3,t})\big)\big) - g\big] = 0 \\ x_{7,t+1} - x_{7,t} = \dfrac{\delta}{ML}[-u_{1,t}\sin(x_{3,t} - x_{4,t})] = 0 \\ x_{8,t+1} - x_{8,t} = \dfrac{\delta}{J} l u_{2,t} = 0 \end{cases}$$

Solving this system could lead to four possible configurations, but we will only consider the following case:

$$\begin{cases} x_{3,t} = 0 \\ x_{4,t} = x_{3,t} \\ u_{1,t} = (M+m)g \\ u_{2,t} = 0 \end{cases}$$

Note that this equilibrium point can represent either a stationary position or a constant velocity.

2.2 Reference trajectory definition

The planned path is made using the starting and ending equilibrium points. This path looks the same on both sides, with two long steady parts in between. Keep in mind that this reference trajectory is not a feasible trajectory for the system dynamics.

Here an example of reference trajectory.

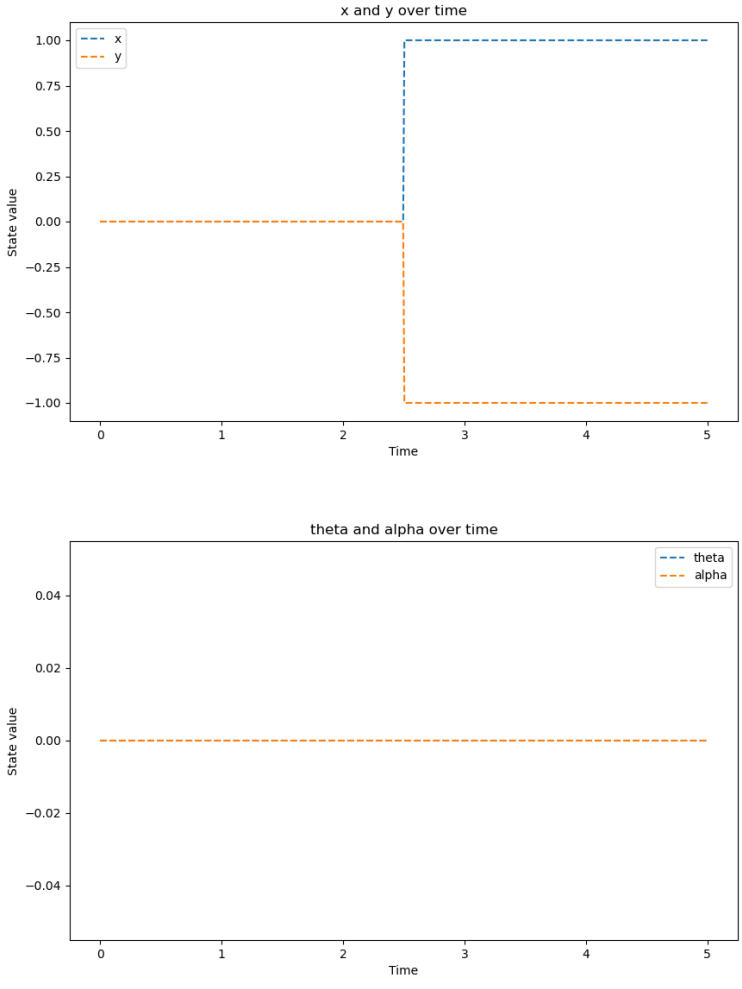

Figure 2.1: configuration reference over time.

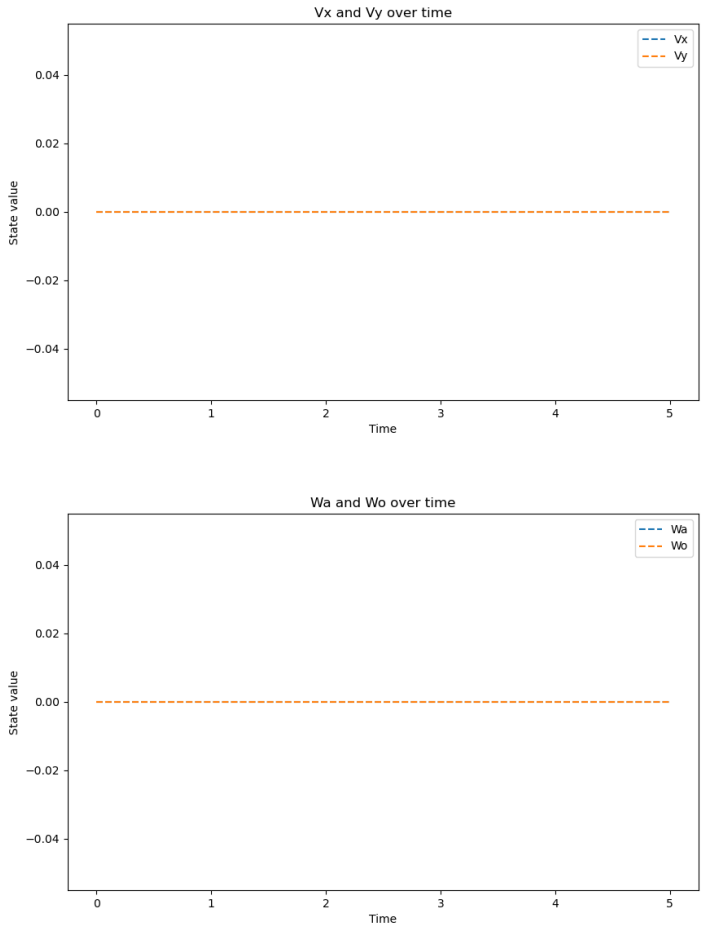

Figure 2.2: velocities reference over time.

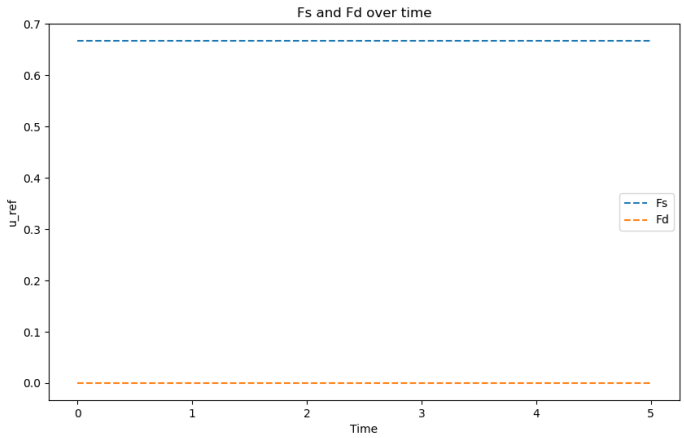

Figure 2.3: Input reference over time.

2.3 Linear Quadratic Regularization

To solve the optimal control problem we implemented Linear Quadratic Regularization (LQR).

2.3.1 Sketch of the algorithm

Initialization

Consider as initial guess trajectory the first equilibrium point (x_0, u_0). For $k = 0, 1, \ldots$:

Step 1: Compute Descent Direction

Linearize the system dynamics evaluating

$$\nabla_1 f_t(x_k^t, u_k^t), \nabla_2 f_t(x_k^t, u_k^t), \nabla_1 \ell_t(x_k^t, u_k^t), \nabla_2 \ell_t(x_k^t, u_k^t), \nabla \ell_T(x_k^T)$$

Compute the gradient of the reduced cost solving backwards the co-state equation, with $\lambda_k^T = \nabla \ell_T(x_k^T)$, and compute Q_k^t, R_k^t, S_k^t, and Q_k^T.

In order to define the LQR controller compute K_k^t, σ_k^t, for all $t = 0, \ldots, T-1$:

$$\begin{aligned}\min_{\Delta x, \Delta u} \sum_{t=0}^{T-1} & \left(\nabla_1 \ell_t(x_k^t, u_k^t)\Delta x_t + \nabla_2 \ell_t(x_k^t, u_k^t)\Delta u_t\right)^\top \begin{bmatrix}\Delta x_t \\ \Delta u_t\end{bmatrix} \\ & + \frac{1}{2}\begin{bmatrix}\Delta x_t \\ \Delta u_t\end{bmatrix}^\top \begin{bmatrix}Q_k^t & S_k^{t,\top} \\ S_k^t & R_k^t\end{bmatrix}\begin{bmatrix}\Delta x_t \\ \Delta u_t\end{bmatrix} \\ & + \nabla \ell_T(x_k^T)^\top \Delta x_T + \frac{1}{2}\Delta x_T^\top Q_k^T \Delta x_T\end{aligned}$$

Step 2: Compute new state-input trajectory

Implementing step-size selection rule, e.g., Armijo)
Forward integrate (closed-loop), for all t = 0, ..., T - 1, with $x_{k+1,0} = x_{\text{init}}$

$u_{k+1,t} = u_k^t + K_k^t(x_{k+1,t} - x_k^t)$

2.3.2 Costs definition

Two types of costs are used: in the case of position control, we give more weight to the position of the quadrotor, while in the case of velocity control, we give more weight to the cost of the quadrotor's velocity.

Costs for Position Control:

$$QQt = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$RRt = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$$

$$QQT = QQt$$

Costs for Velocity Control:

$$QQt_{\text{vel}} = \begin{bmatrix} 0.0001 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0001 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$RRt_{\text{vel}} = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix}$$

$$QQT_{\text{vel}} = QQt_{\text{vel}}$$

2.4 Results

In this section, we present the outcomes of the Linear Quadratic Regulator on the previous example of reference trajectory.

2.4.1 Optimal trajectory

In the following pages the optimal trajectory found by our algorithm is shown.

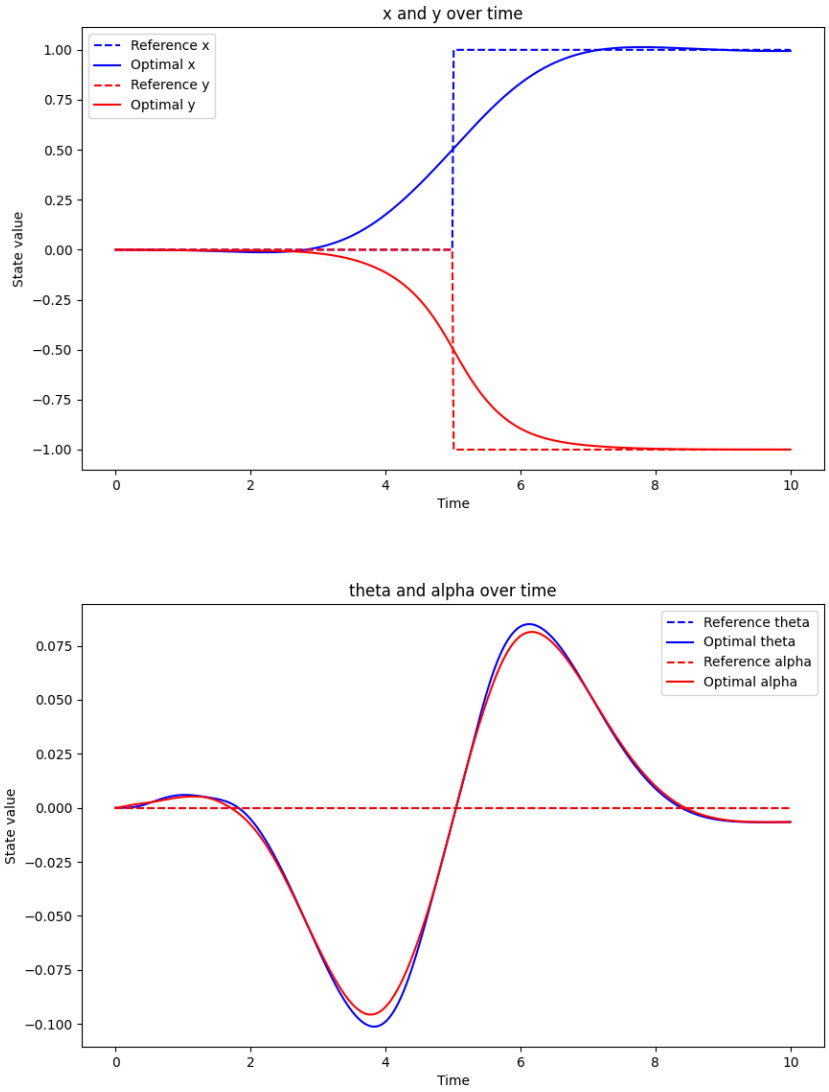

Figure 2.4: Optimal configuration over time.

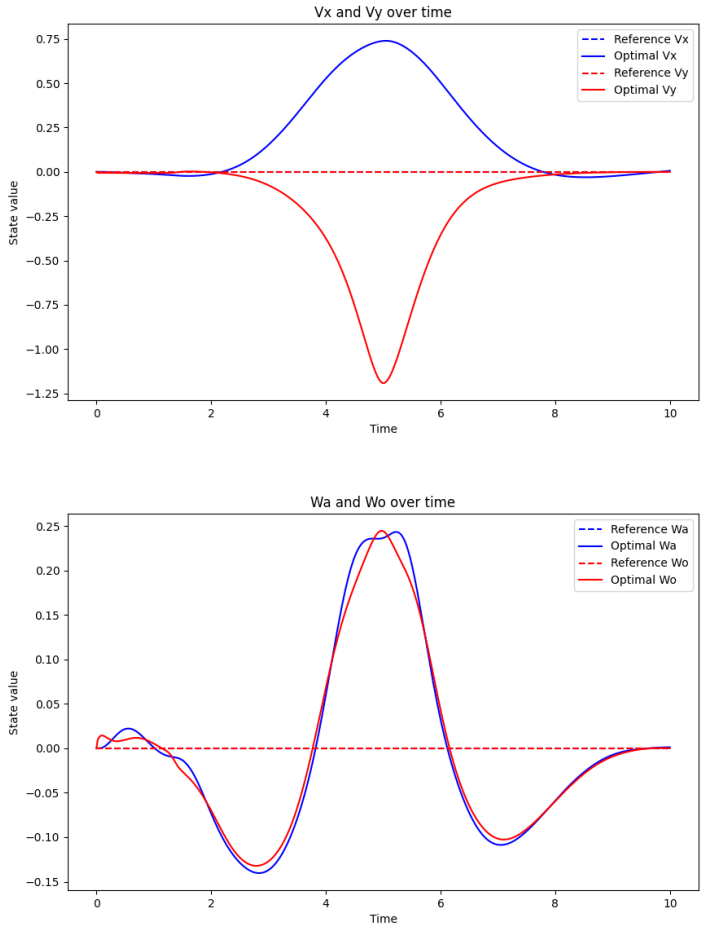

Figure 2.5: optimal velocities over time.

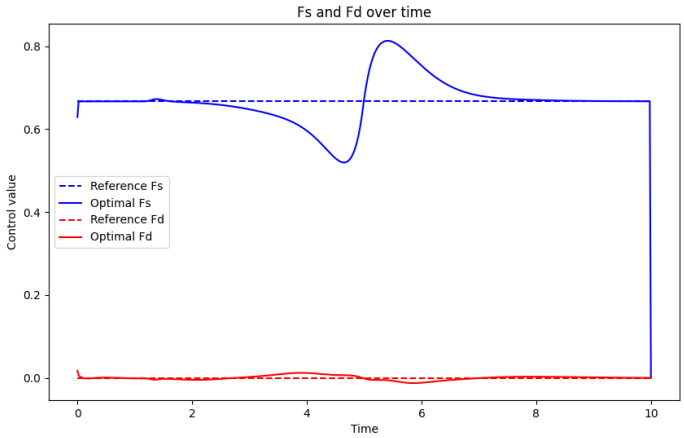

Figure 2.6: Optimal input over time.

2.4.2 Suboptimal trajectories

To illustrate the iterative evolution of the algorithm, we display only the x and y coordinates of the quadrotor over time and the armijos plot, providing an overview of the overall progression. Refer to the code for more details.

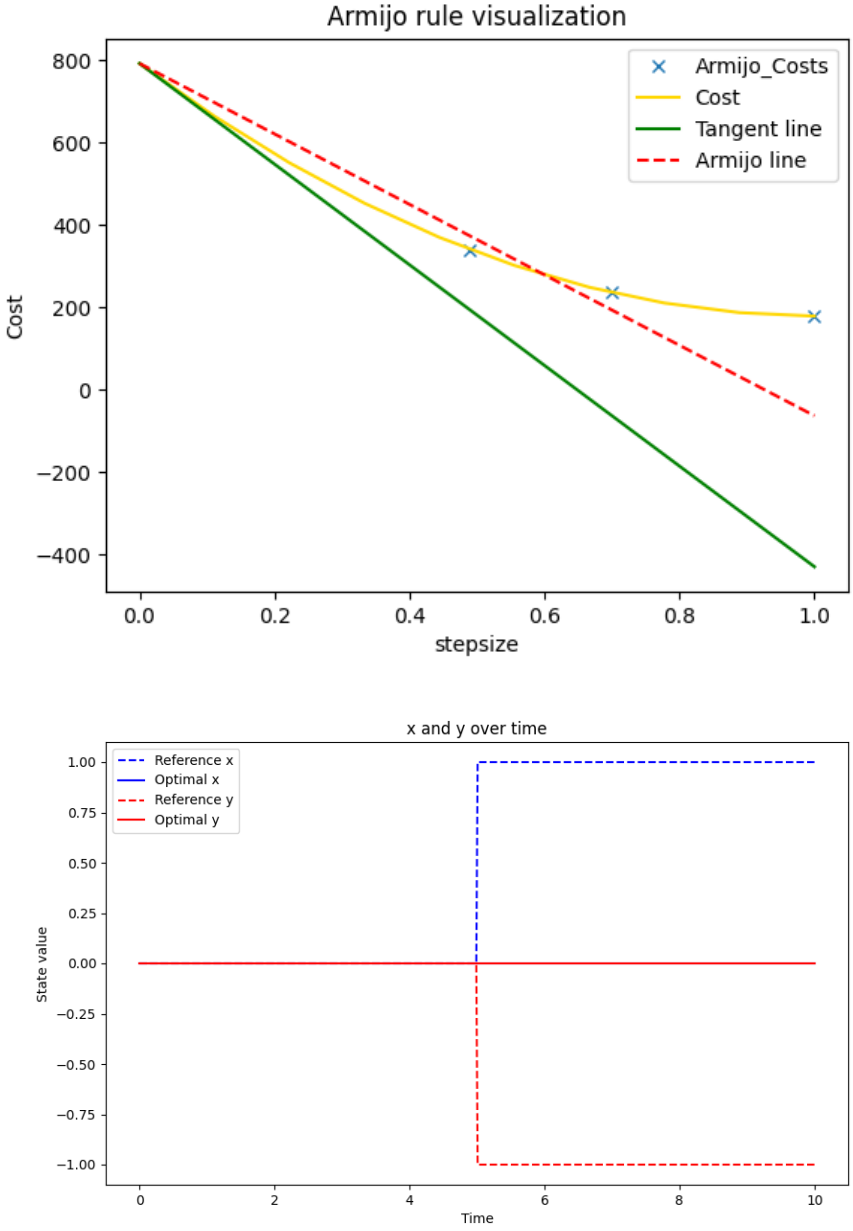

Figure 2.7: First iteration.

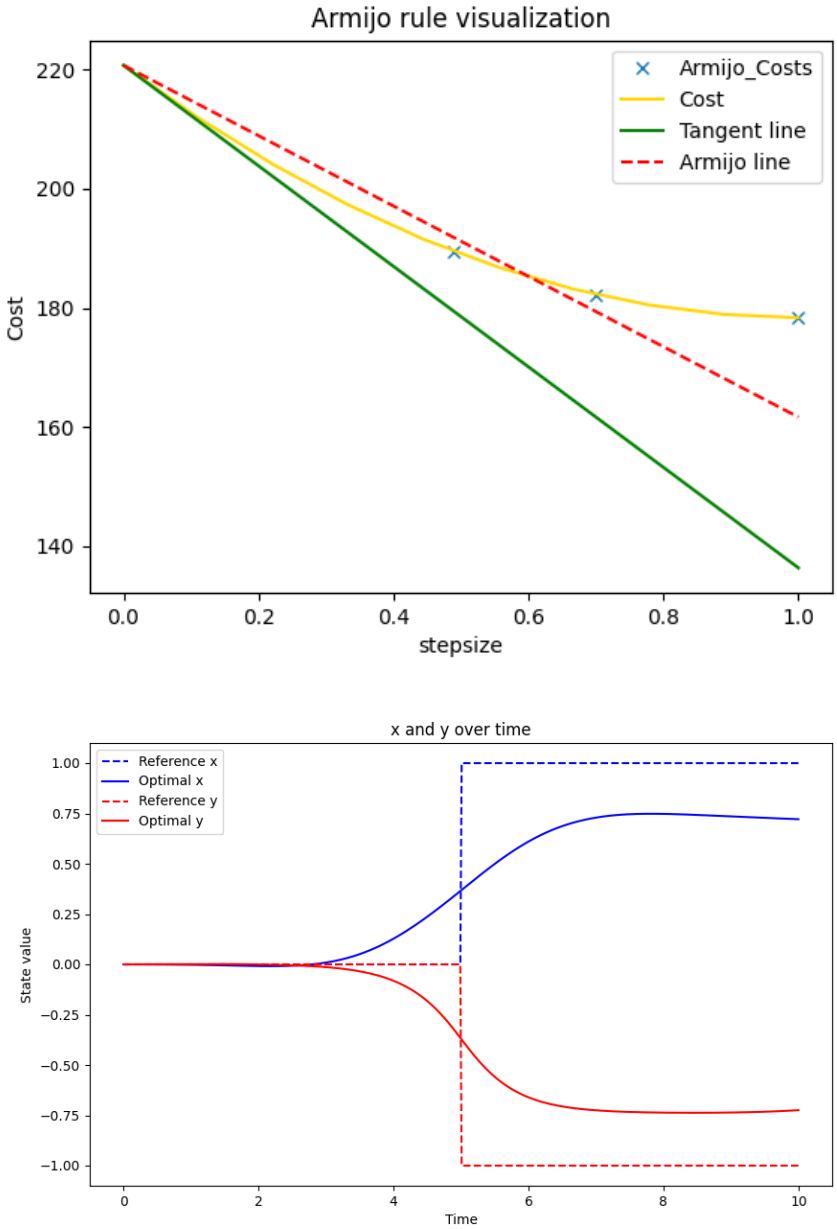

Figure 2.8: Third iteration.

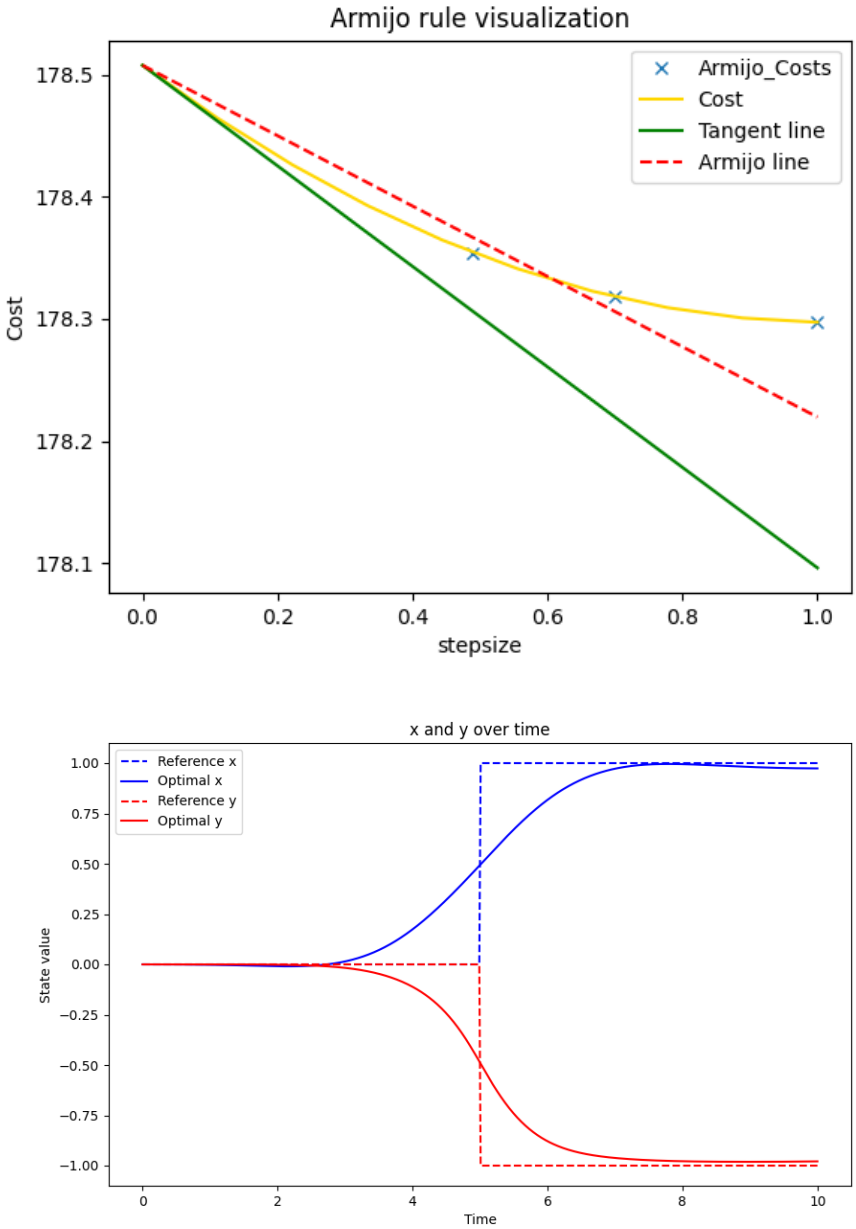

Figure 2.9: Seventh iteration.

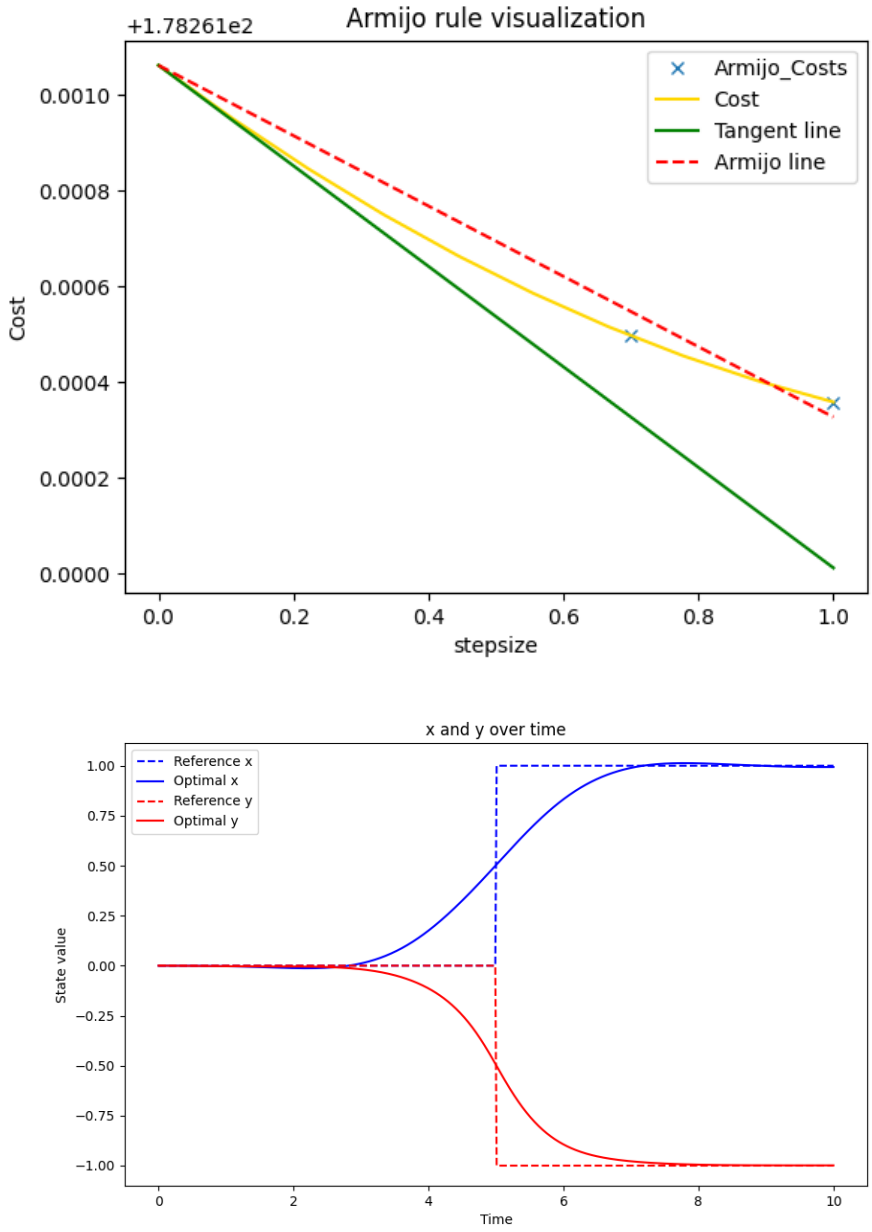

Figure 2.10: Final iteration.

2.4.3 Cost and Descent direction plot

Here the norm of the descent direction along iterations (in semi-logarithmic scale) and the cost along iterations (in semi-logarithmic scale) are shown:

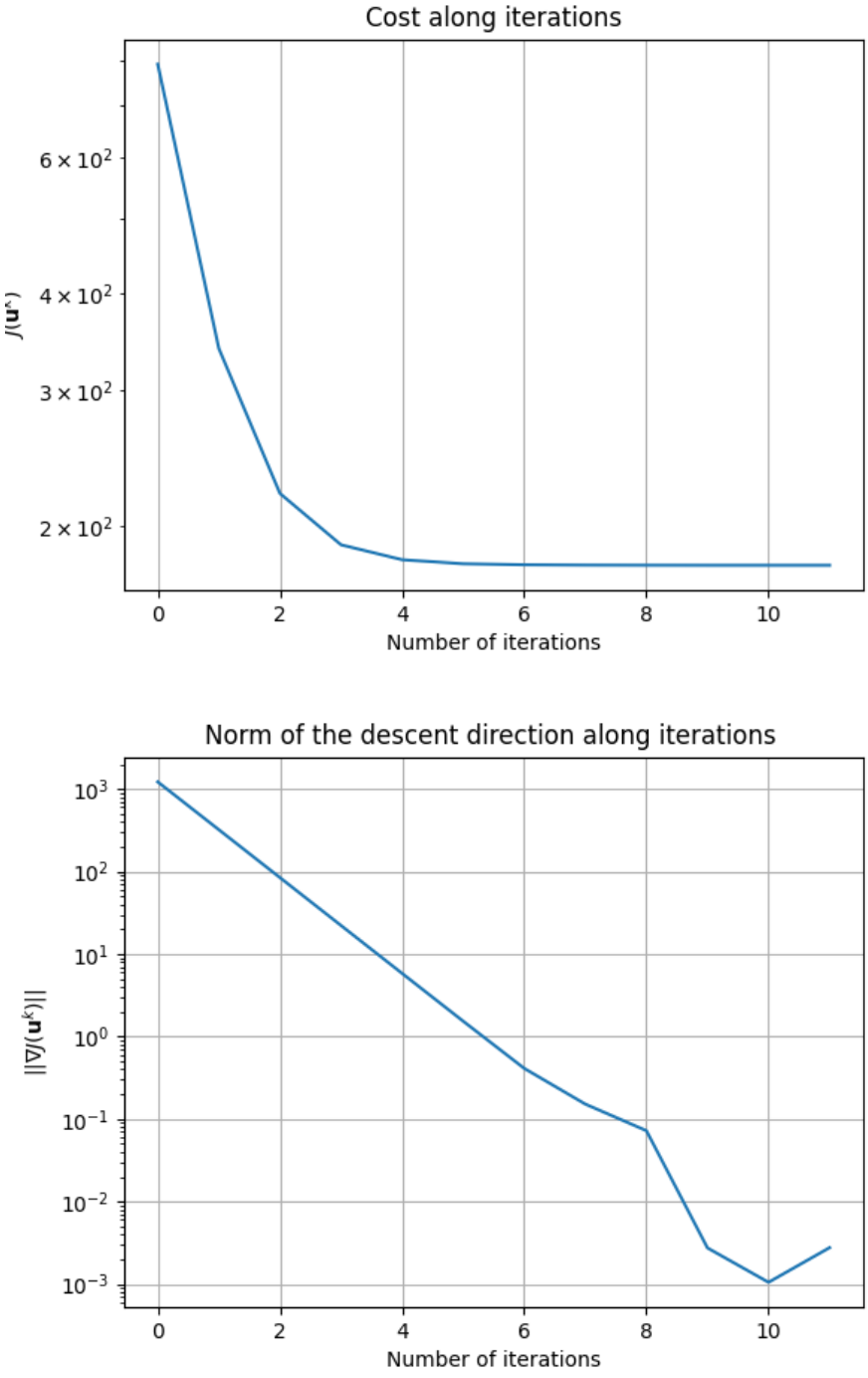

Figure 2.11: Cost and descent direction norm.

Task 2 - Trajectory generation (II)

In task 2 we want to generate a desired (smooth) state−input curve and perform the trajectory generation task of Task 1 on this new desired trajectory.

In order to generate a smooth state-input curve we opted for a sigmoid trajectory that starts on the first equilibrium state and finishes on the second one. The sigmoid function, often denoted as $\sigma(x)$, is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

The Figure below illustrates the behavior of the sigmoid function.

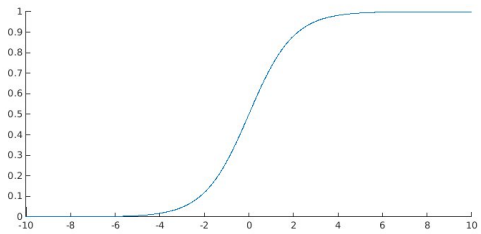

Figure 3.1: Behaviour of the sigmoid

The choice of reference trajectory plays a crucial role in the performance and convergence characteristics of the Linear Quadratic Regulator (LQR) algorithm, it trapresents a more feasible reference for the trajectory.

The continuity of a smooth trajectory can facilitate the convergence of the LQR algorithm, as the system can gradually adapt to reference variations.

Furthermore, a smooth trajectory can help minimize the control energy required by the system. If the trajectory is already "close" to the optimal solution, the LQR algorithm may converge more quickly than in a situation where the reference has sharp changes.

3.1 Results

In this section, we present the outcomes of the Linear Quadratic Regulator on a sigmoid reference trajectory.

We applied this method to two different cases. In the first case, the equilibria are two distinct positions in space, and in the second case they are two distinct constant linear velocities. In the second case the trajectory for the state relative to the position x and y is obtained with the integration of the speed reference.

An example of the two cases follows:

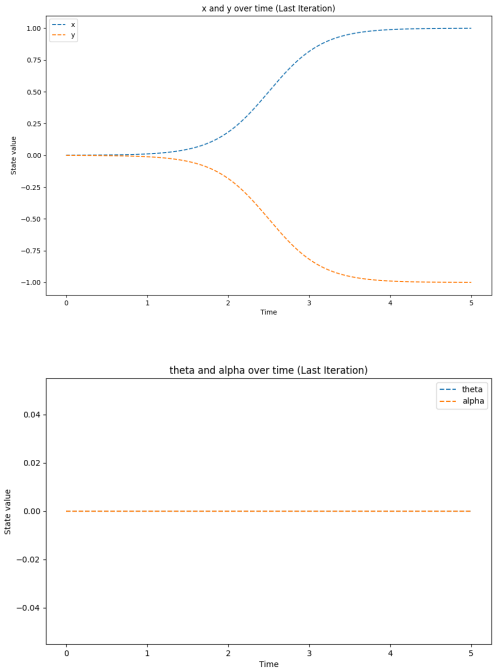

Figure 3.2: Case 1: configuration reference over time.

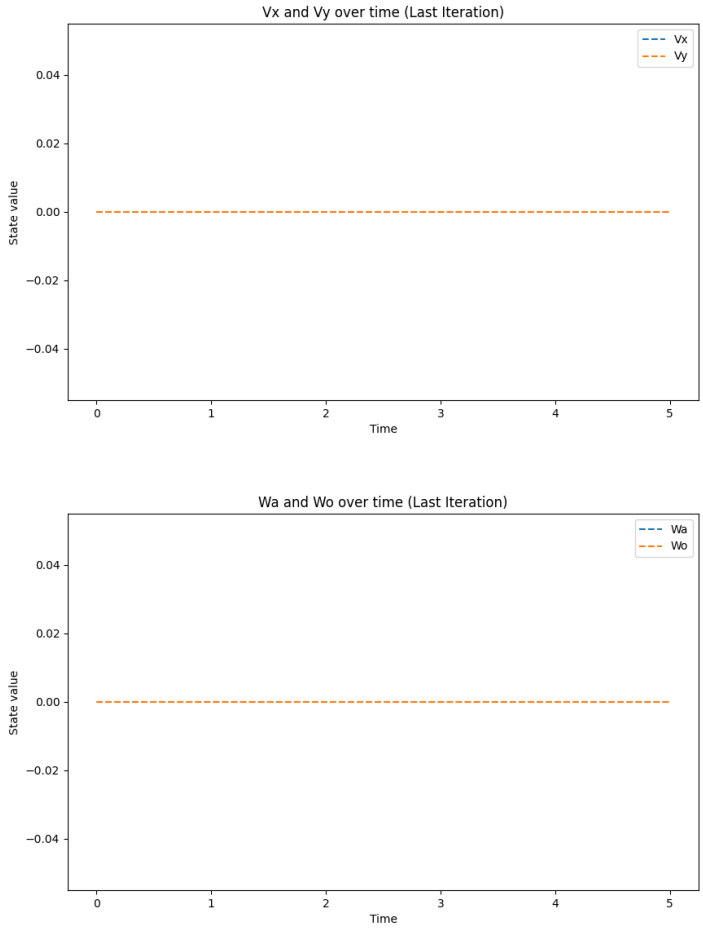

Figure 3.3: Case 1: velocities reference over time.

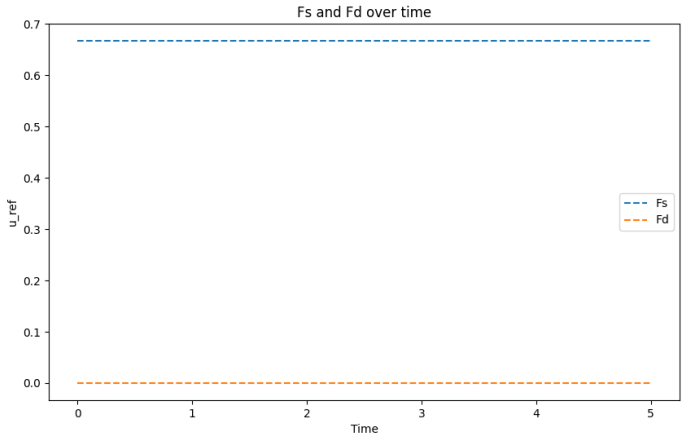

Figure 3.4: Case 1: Input reference over time.

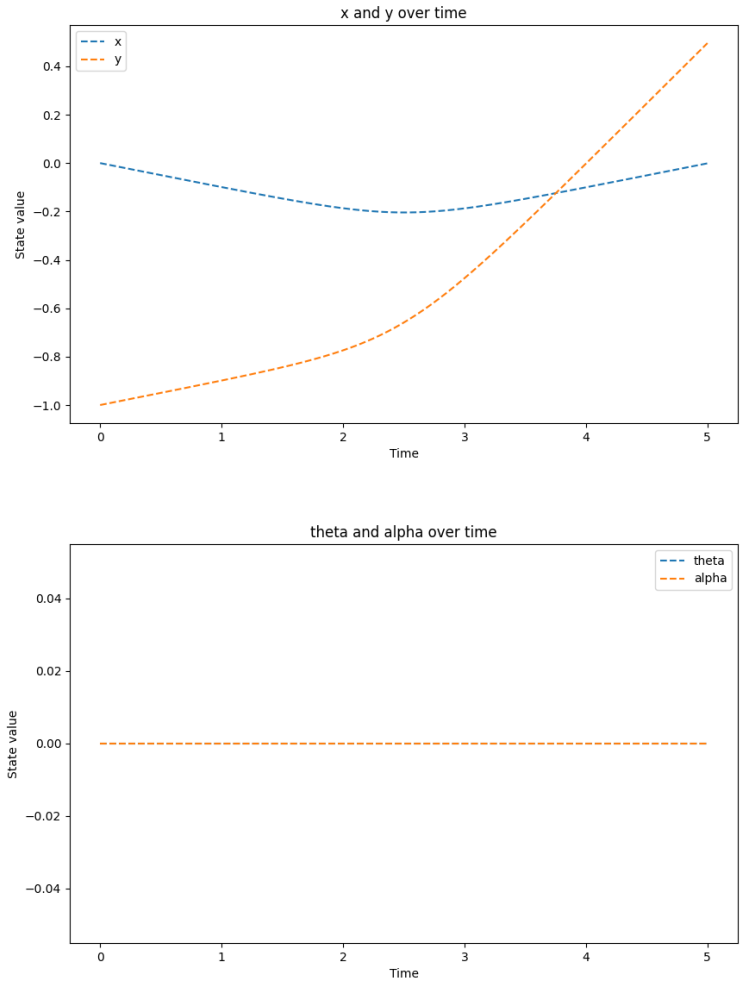

Figure 3.5: Case 2: configuration reference over time.

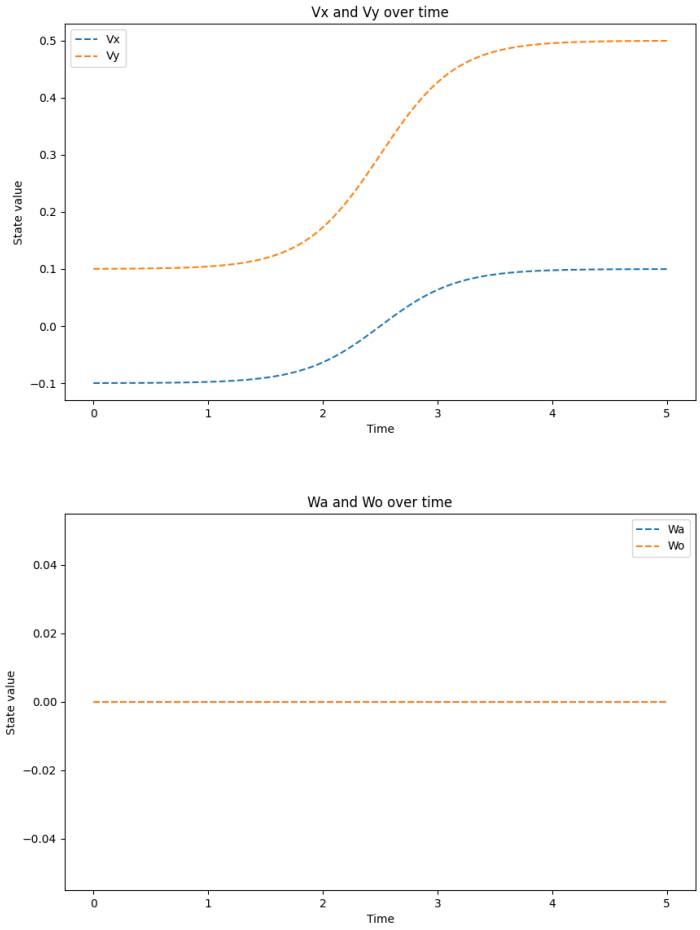

Figure 3.6: Case 2: velocities reference over time.

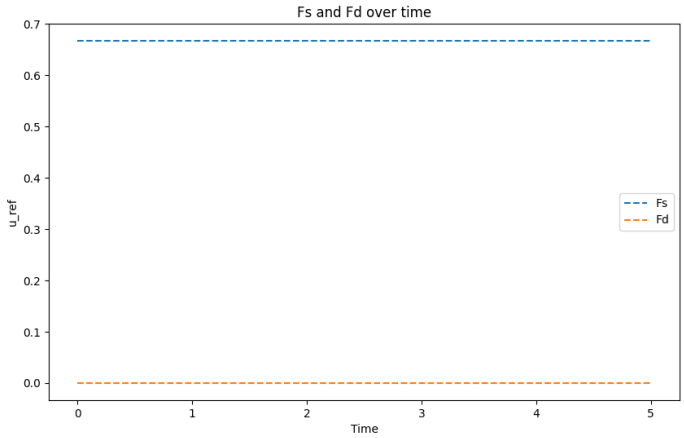

Figure 3.7: Case 2: Input reference over time.

3.1.1 Optimal trajectory

In this section, we present the outcomes of the Linear Quadratic Regulator on the reference trajectory provided by equilibria that are two distinct positions in space (Case 1).

In the following pages the optimal trajectory found by our algorithm is shown.

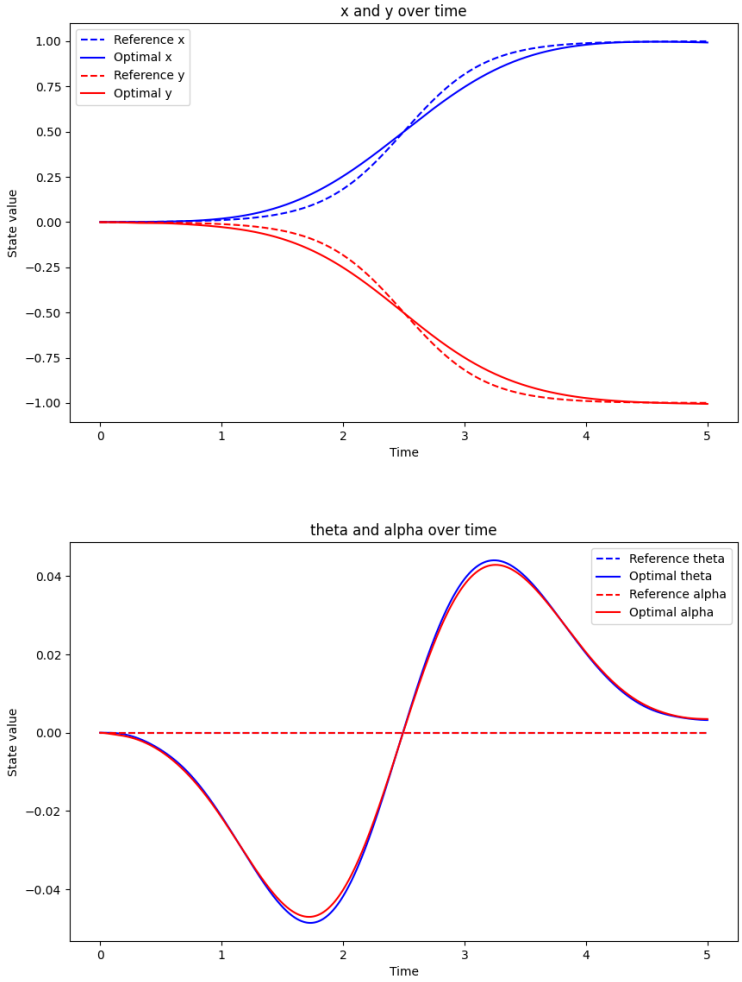

Figure 3.8: Optimal configuration over time.

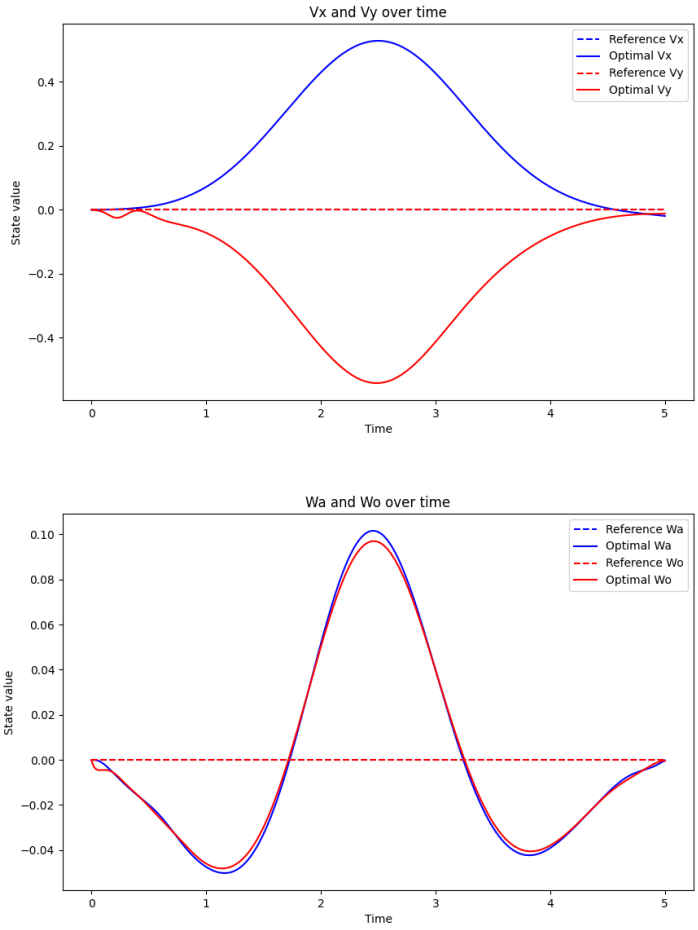

Figure 3.9: optimal velocities over time.

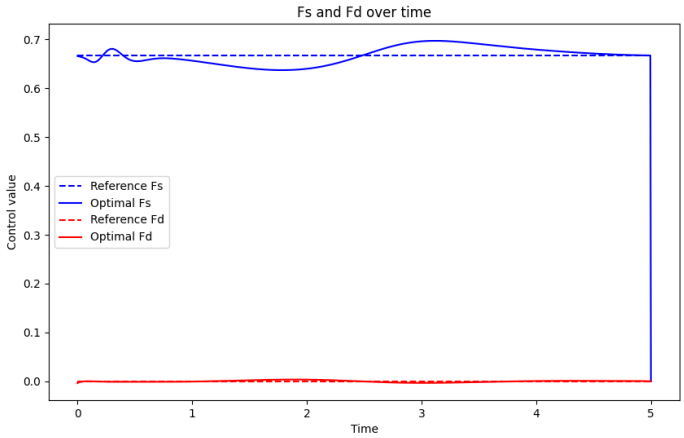

Figure 3.10: Optimal input over time.

3.1.2 Suboptimal trajectories

To illustrate the iterative evolution of the algorithm, we display only the x and y coordinates of the quadrotor over time and the armijos plot, providing an overview of the overall progression. Refer to the code for more details.

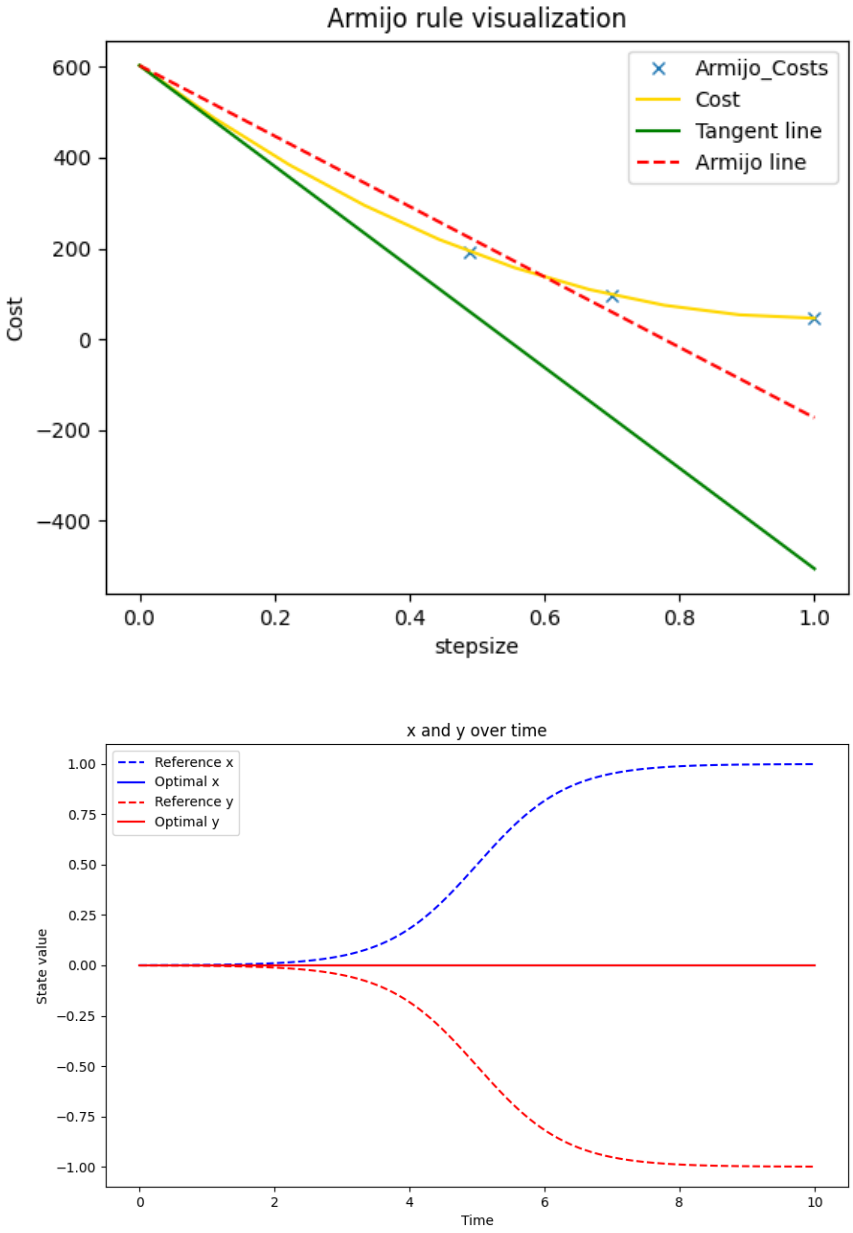

Figure 3.11: First iteration.

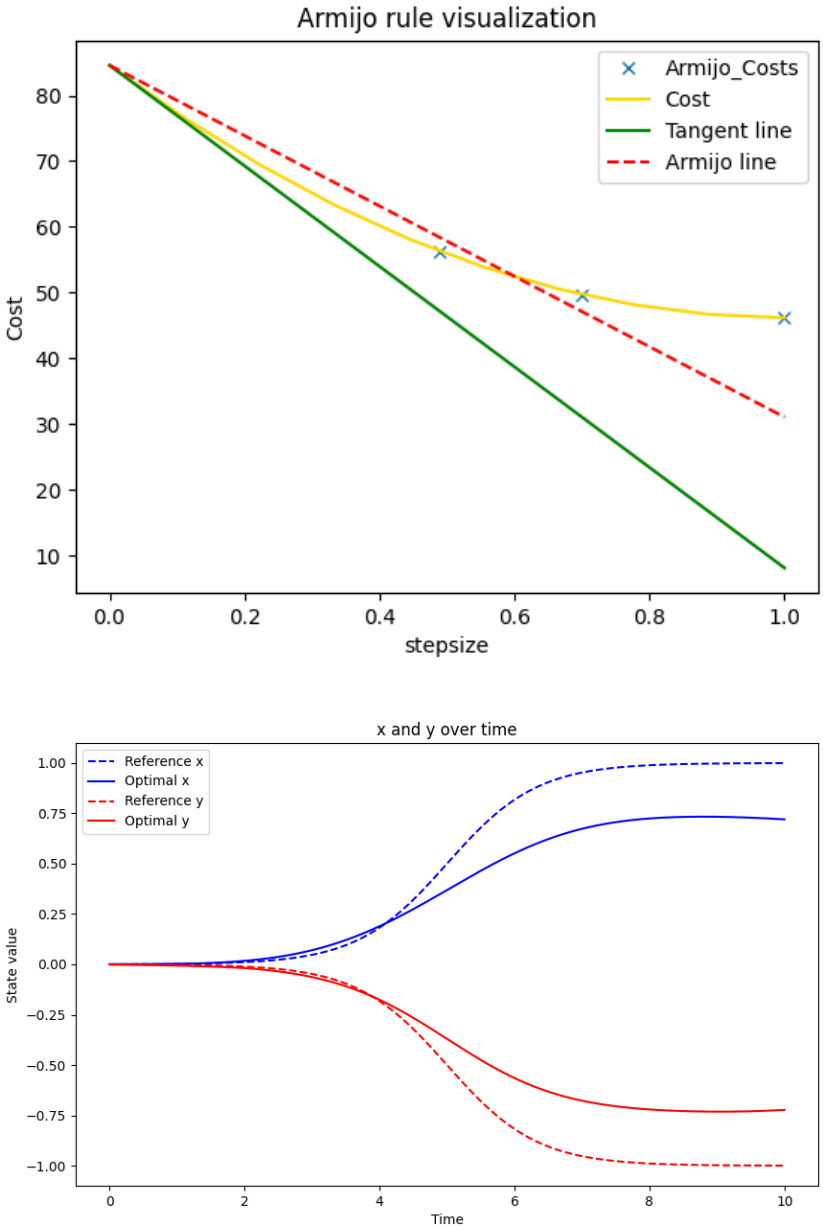

Figure 3.12: Third iteration.

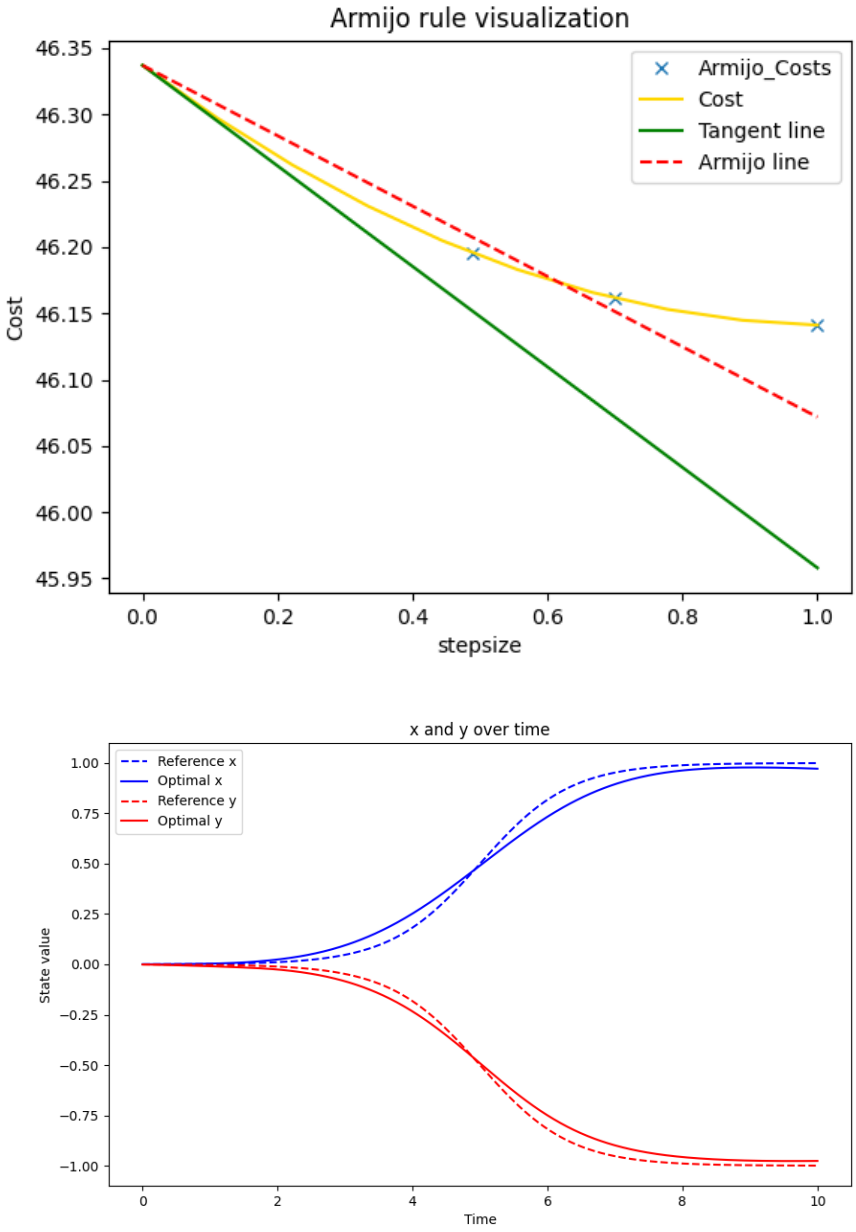

Figure 3.13: Seventh iteration.

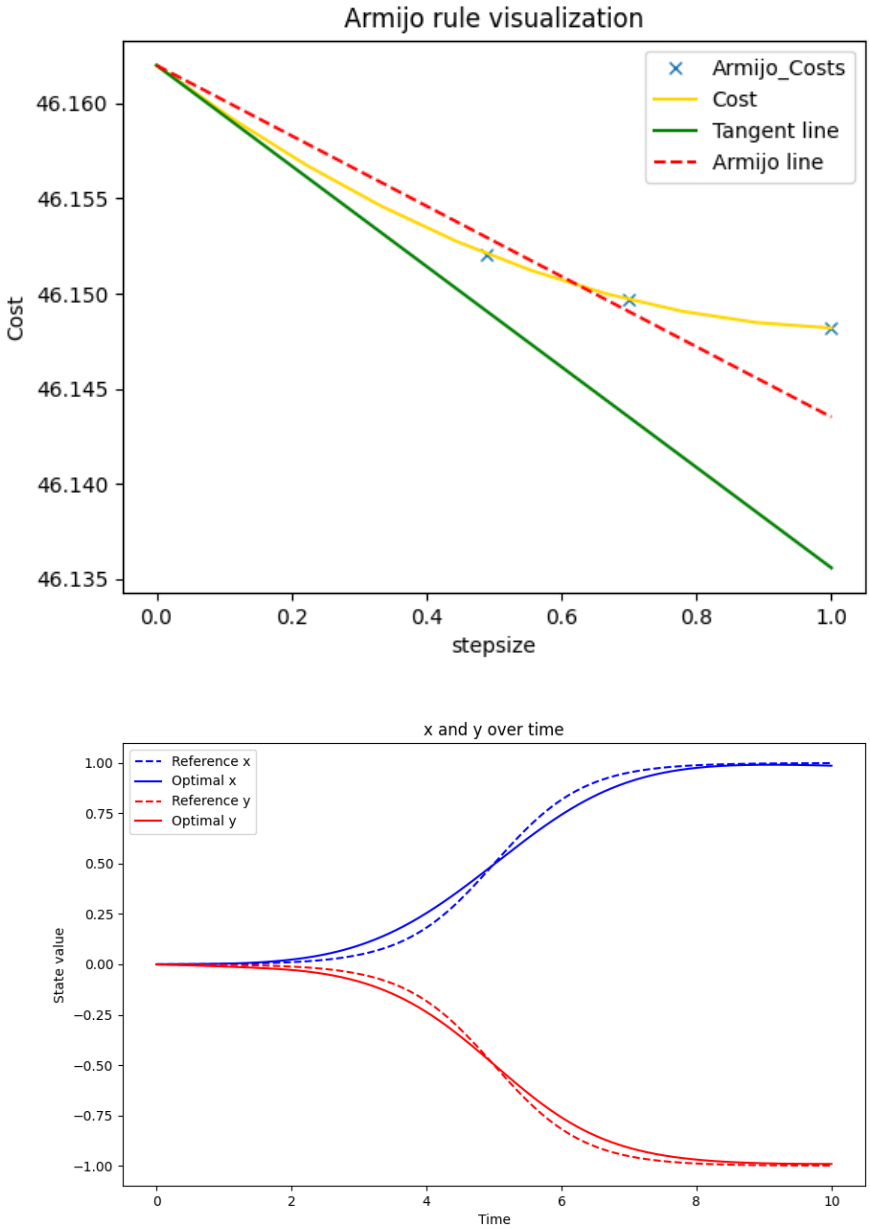

Figure 3.14: Final iteration.

On the code you will also find the outcomes of the Linear Quadratic Regulator on the reference trajectory provided by equilibria that are two distinct velocity in space (Case 2).

Task 3 - Trajectory tracking via LQR

In task 3 we want to take the vehicle dynamics linearized about the (optimal) trajectory (x_{opt}, u_{opt}) computed in Task 2 and then exploit the LQR algorithm to track this reference trajectory.

4.1 Q and R definition

Costs for closed loop Control:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$$

4.2 Closed loop LQR

Evaluate the closed loop gain

Linearize the system dynamics evaluating

$$\nabla_1 f_t(x_k^t, u_k^t), \nabla_2 f_t(x_k^t, u_k^t), \nabla_1 \ell_t(x_k^t, u_k^t), \nabla_2 \ell_t(x_k^t, u_k^t), \nabla \ell_T(x_k^T)$$

Compute the closed loop gain K_k^t, for all $t = 0, \ldots, T-1$:

$$\begin{aligned}\min_{\Delta x, \Delta u} \sum_{t=0}^{T-1} &\left(\nabla_1 \ell_t(x_k^t, u_k^t)\Delta x_t + \nabla_2 \ell_t(x_k^t, u_k^t)\Delta u_t\right)^\top \begin{bmatrix}\Delta x_t \\ \Delta u_t\end{bmatrix} \\ &+ \frac{1}{2}\begin{bmatrix}\Delta x_t \\ \Delta u_t\end{bmatrix}^\top \begin{bmatrix}Q_k^t & S_k^{t,\top} \\ S_k^t & R_k^t\end{bmatrix}\begin{bmatrix}\Delta x_t \\ \Delta u_t\end{bmatrix} \\ &+ \nabla \ell_T(x_k^T)^\top \Delta x_T + \frac{1}{2}\Delta x_T^\top Q_k^T \Delta x_T\end{aligned}$$

Step 2: Compute new state-input trajectory

Implementing step-size selection rule, e.g., Armijo)
Forward integrate (closed-loop), for all t = 0, ..., T - 1, with $x_{k+1,0} = x_{\text{init}}$

$u_{tracked,t} = u_k^t + K_k^t(x_{tracked,t} - x_{optimal,t})$
$x_{tracked,t+1} = f_t(x_{tracked,t}, u_{tracked,t})$

4.2.1 Plots

In order to evaluate the tracking performance of the controller, the initial position of the quadrotor is set to be different from that of the optimal trajectory. Examples of the tracking performance are shown in the following plots.

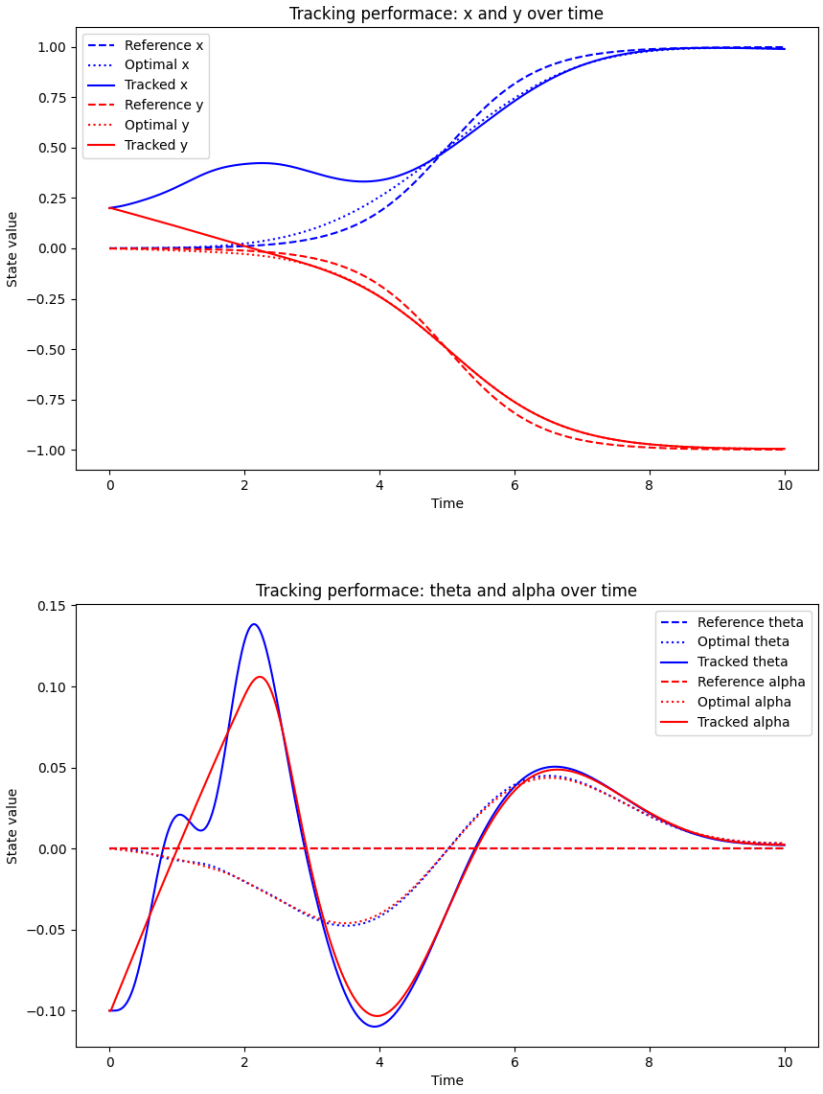

Figure 4.1: configuration tracking.

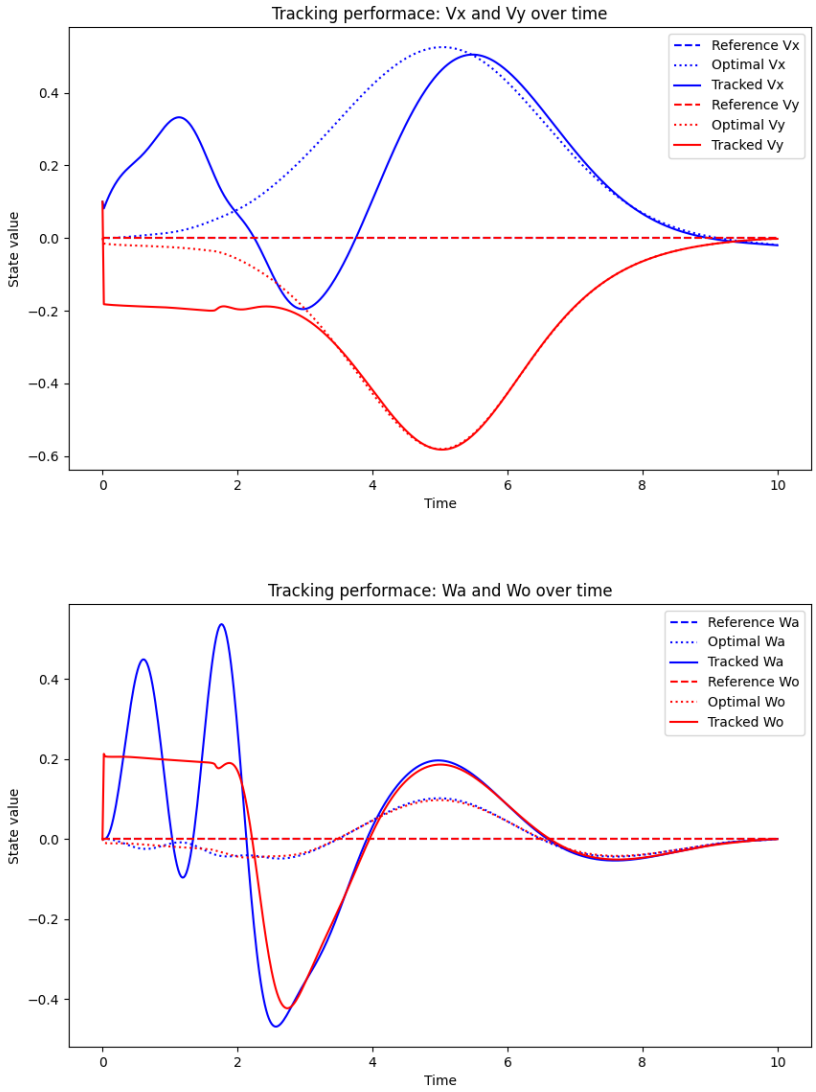

Figure 4.2: velocity tracking.

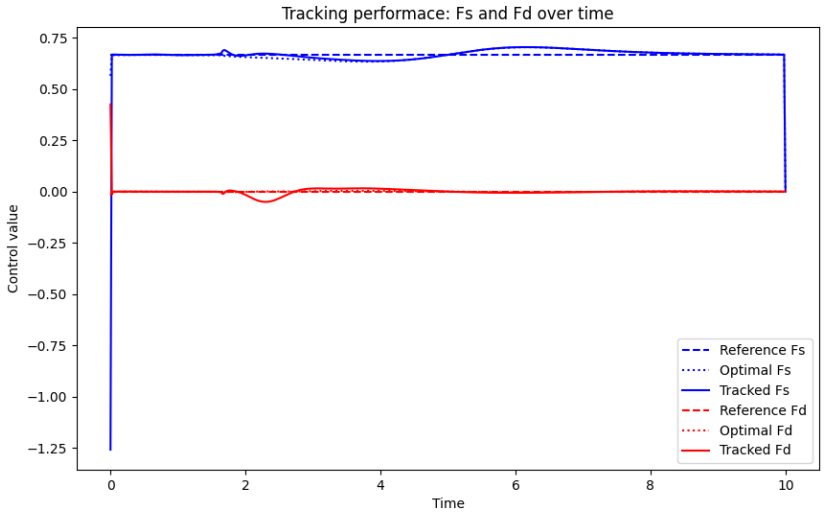

Figure 4.3: input tracking.

Task 4 - Trajectory tracking via MPC

In task 4 we want to take the vehicle dynamics linearized about the (optimal) trajectory (x_{opt}, u_{opt}) computed in Task 2 and then exploit an MPC algorithm to track this reference trajectory.

5.1 Model Predictive Control

The goal of Model predictive Control (MPC) is to control a system via a stabilizing controller that minimizes a certain cost function, enforces some constraint for all t and works online. This is implemented by solving an optimal control problem at each sampling time and applying the first optimal input:

For each t:

- Measure the current state x_t
- Compute the optimal trajectory $x^*_{t|t}, ..., x^*_{t+T|t}, u^*_{t|t}, ..., u^*_{t+T-1|t}$ with initial condition x_t
- Apply the first control input $u^*_{t|t}$
- Measure x_{t+1} and repeat

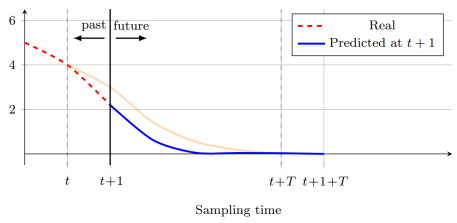

Figure 5.1: Model Predictive Control

5.1.1 The optimal control problem to be solved at each t

At each time instant we want to solve

$$\min_{\substack{x_t,\dots,x_{t+T} \\ u_t,\dots,u_{t+T-1}}} \sum_{\tau=t}^{t+T-1} l_\tau(x_\tau, u_\tau) + l_{t+T}(x_{t+T})$$

$$\begin{aligned} subject \quad to \qquad & x_{\tau+1} = f(x_\tau, u_\tau) \qquad \forall \tau = t, ..., t+T-1 \\ & x_\tau \in X, u_\tau \in U \qquad \forall \tau = t, ..., t+T \\ & x_t = x_t^{meas} \end{aligned}$$

where

- $x_{\tau+1} = f(x_\tau, u_\tau)$ is the so-called prediction model
- x_t^{meas} is the state (of the real system) measured at t

5.1.2 The Linear Quadratic Case

In our case, the dynamics is linear because we linearized it about the optimal trajectory (x_{opt}, u_{opt}) computed in task 2. Therefore we can consider the MPC case where the dynamics is linear and the cost quadratic:

At each time t with initial measured state x_t^{meas} we want to solve the LQ problem

$$\min_{\substack{x_t,\dots,x_{t+T} \\ u_t,\dots,u_{t+T-1}}} \sum_{\tau=t}^{t+T-1} x_\tau^T Q_\tau x_\tau + u_\tau^T R_\tau u_\tau + x_{t+T}^T Q_T x_{t+T}$$

$$\begin{aligned} subject \quad to \qquad & x_{\tau+1} = A_\tau x_\tau + B_\tau u_\tau \qquad \forall \tau = t, ..., t+T-1 \\ & x_t = x_t^{meas} \end{aligned}$$

where

- T is the prediction horizon
- $x_\tau \in \mathbb{R}^n$ and $u_\tau \in \mathbb{R}^m$
- $A_\tau \in \mathbb{R}^{nxn}$ and $B_\tau \in \mathbb{R}^{nxm}$ is the prediction model
- $Q_\tau \in \mathbb{R}^{nxn}$ and $Q_\tau = Q_\tau^T \geq 0 \forall \tau$
- $R_\tau \in \mathbb{R}^{mxm}$ and $R_\tau = R_\tau^T > 0 \forall \tau$
- $Q_T \in \mathbb{R}^{nxn}$ and $Q_T = Q_T^T \geq 0$

5.2 Tracking of (x_{opt}, u_{opt})

Since we want to track the optimal trajectory throught Model Predictive Control, the cost function that we are interested in minimizing is the one in which x_τ and u_τ are the errors between the predicted trajectory (x^{mpc}, u^{mpc}) and the optimal one (x^{opt}, u^{opt}):

$$x_\tau = x_\tau^{mpc} - x_\tau^{opt}$$
$$u_\tau = u_\tau^{mpc} - u_\tau^{opt}$$

Therefore, the LQ problem we are solving at each t results:

$$\min_{\substack{x_t,\dots,x_{t+T} \\ u_t,\dots,u_{t+T-1}}} \quad \sum_{\tau=t}^{t+T-1} (x_\tau^{mpc} - x_\tau^{opt})^T Q_\tau (x_\tau^{mpc} - x_\tau^{opt}) +$$
$$+ (u_\tau^{mpc} - u_\tau^{opt})^T R_\tau (u_\tau^{mpc} - u_\tau^{opt}) +$$
$$+ (x_{t+T}^{mpc} - x_{t+T}^{opt})^T Q_T (x_{t+T}^{mpc} - x_{t+T}^{opt})$$

$$subject \quad to \quad x_{\tau+1}^{mpc} - x_{\tau+1}^{opt} = A_\tau (x_\tau^{mpc} - x_\tau^{opt}) + B_\tau (u_\tau^{mpc} - u_\tau^{opt}) \qquad \forall \tau = t, \dots, t+T-1$$
$$x_t^{mpc} - x_t^{opt} = x_t^{meas} - x_t^{opt}$$

Our simulation horizon is of 500 samples, and we choose the MPC prediction horizon to be of 100 samples, which allows the system to exhibit satisfactory control performance, meet the desired objectives, and remain computationally feasible.

Regarding the costs, oure choiche is based on trial and error in order to guarantee some settling time and overshoot and it is the following:

$$QQ_\tau = \begin{bmatrix} 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}$$

$$RR_\tau = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$$

$$QQ_T = QQ_\tau$$

5.3 Results

To showcase the tracking performances, we consider a perturbed initial condition (different than $x_{opt,0}$):

$$\mathbf{x}_0 = \begin{bmatrix} x_{p,0} \\ y_{p,0} \\ \alpha_0 \\ \theta_0 \\ v_{x,0} \\ v_{y,0} \\ \omega_{\alpha,0} \\ \omega_{\theta,0} \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.4 \\ -0.1 \\ -0.1 \\ 0.1 \\ 0.1 \\ 0 \\ 0 \end{bmatrix}$$

5.3.1 System trajectory and desired optimal trajectory

Below, the plots of the MPC trajectory tracking the desired optimal trajectory (denoted as LQR) for the single states and inputs:

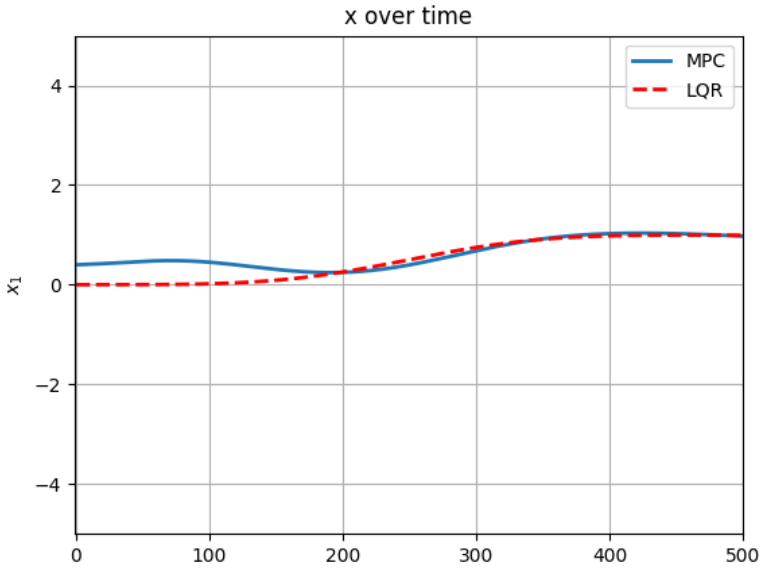

Figure 5.2: x_1

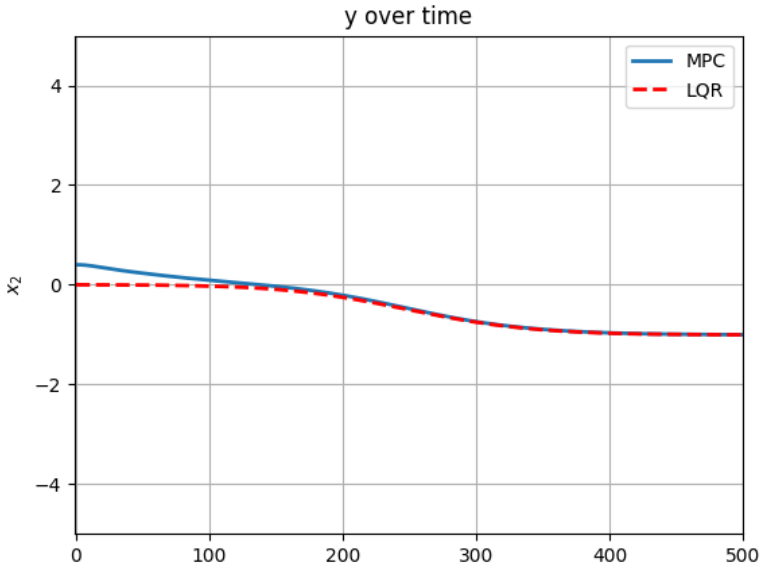

Figure 5.3: x_2

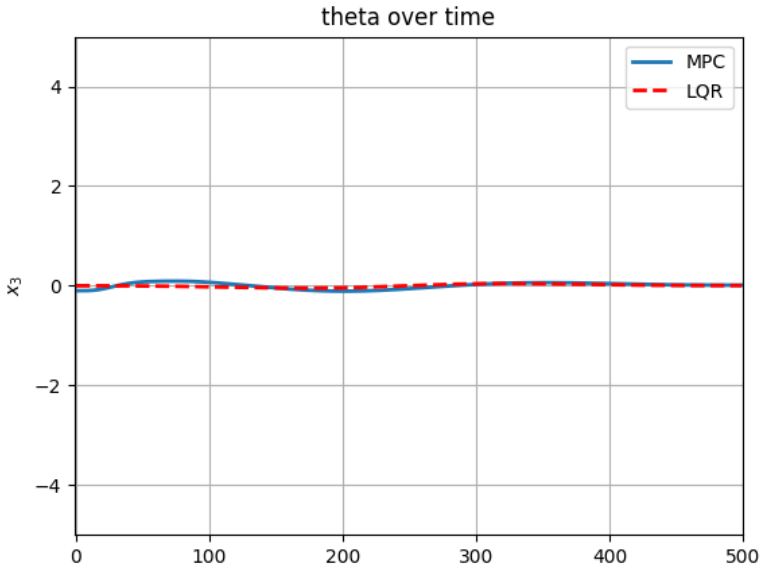

Figure 5.4: x_3

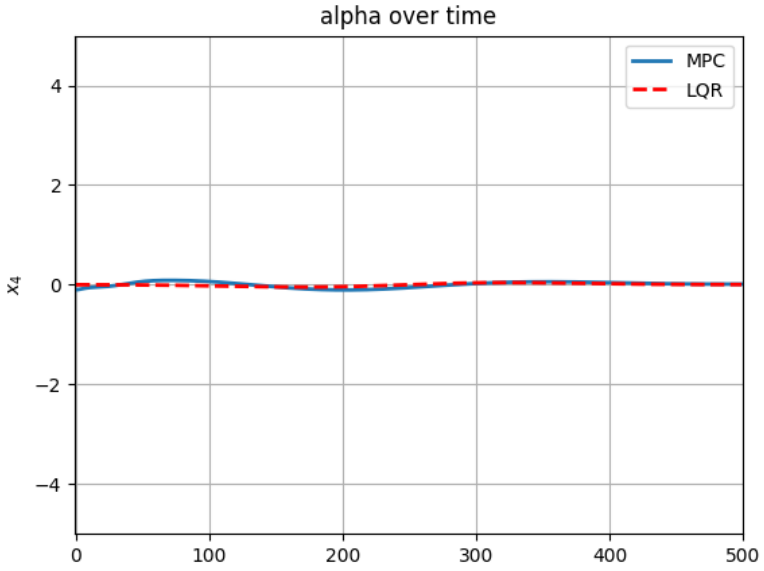

Figure 5.5: x_4

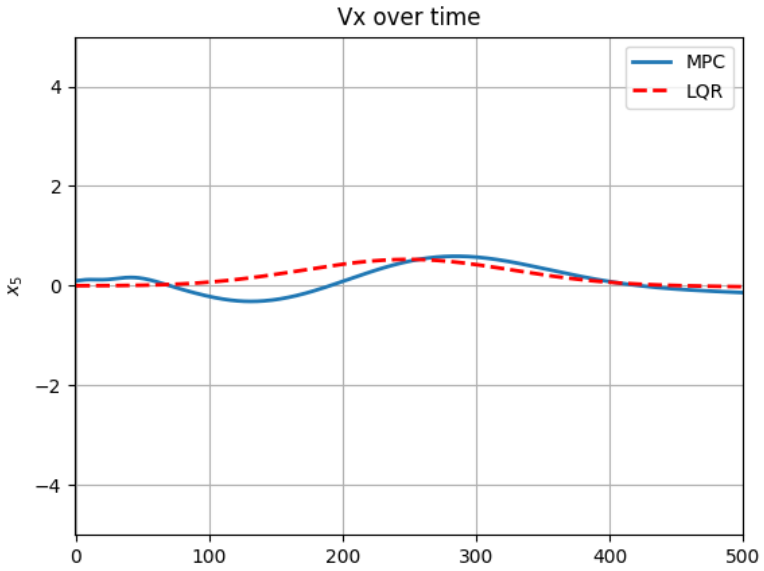

Figure 5.6: x_5

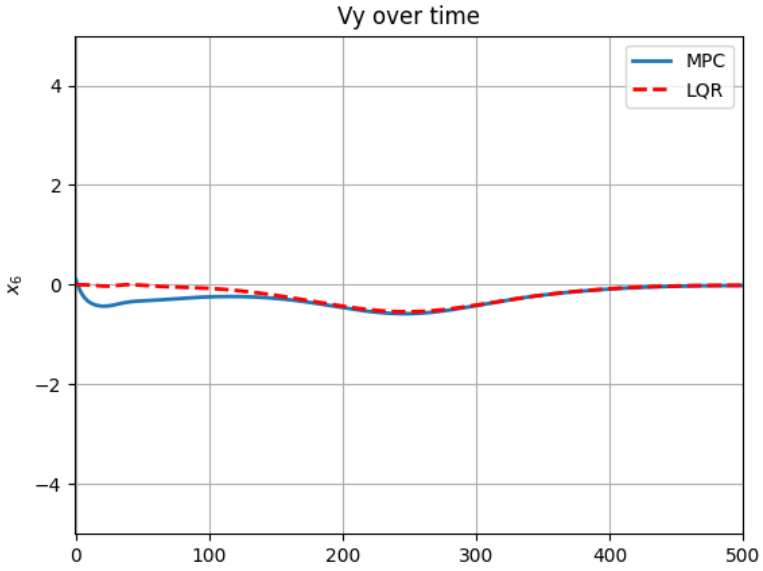

Figure 5.7: x_6

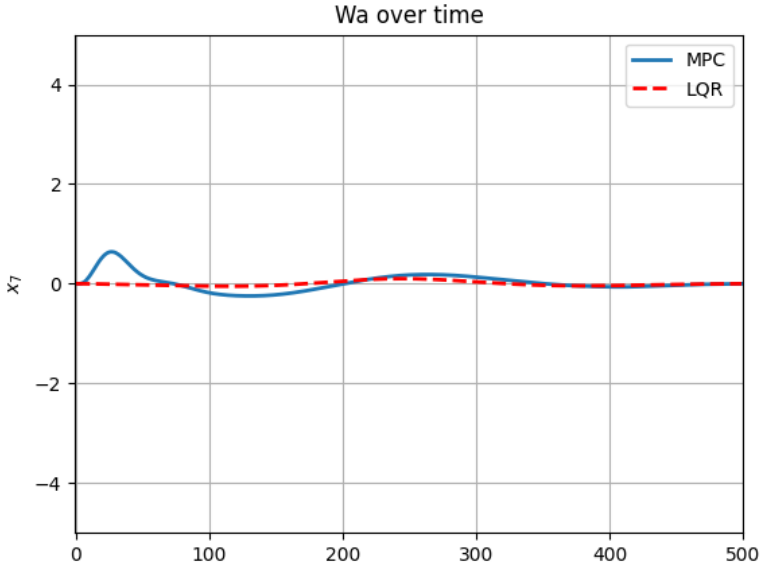

Figure 5.8: x_7

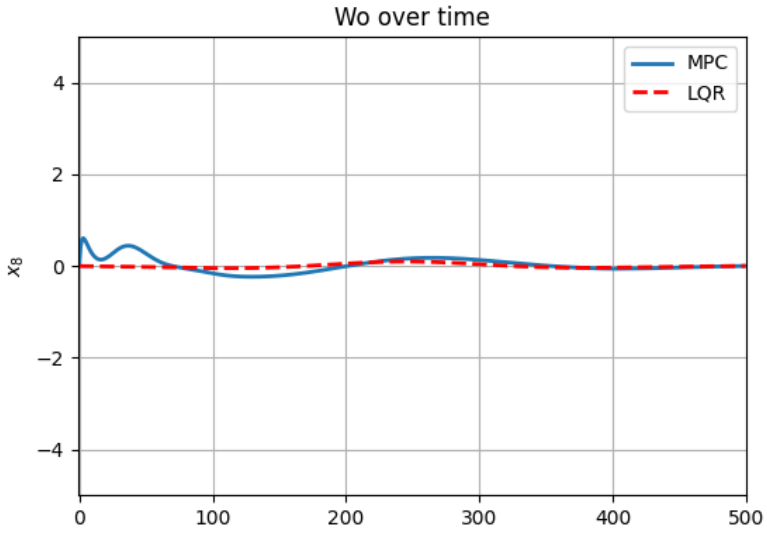

Figure 5.9: 8_1

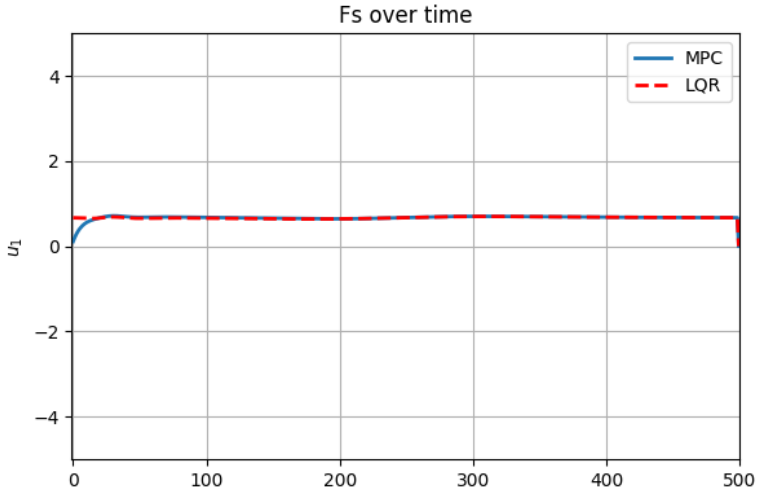

Figure 5.10: u_1

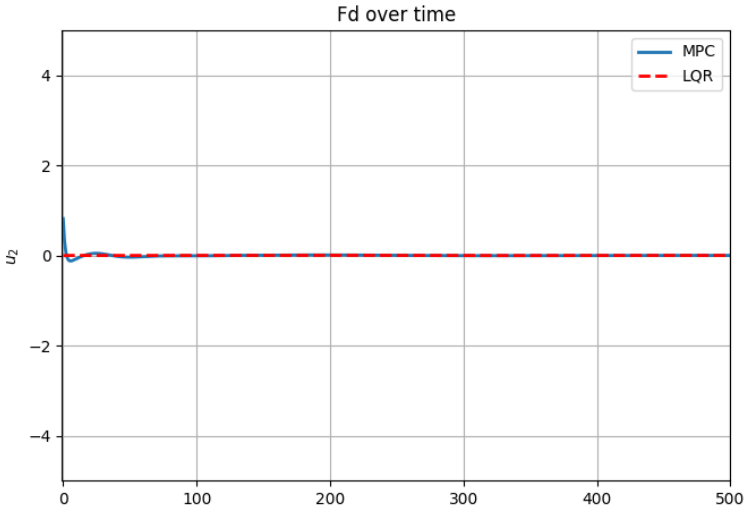

Figure 5.11: u_2

5.3.2 Tracking error for different initial conditions

Given the perturbed initial condition

$$x_0 = [0.4, 0.4, -0.1, -0.1, 0.1, 0.1, 0, 0]^T$$

the following are the plots of the tracking error for single states and inputs:

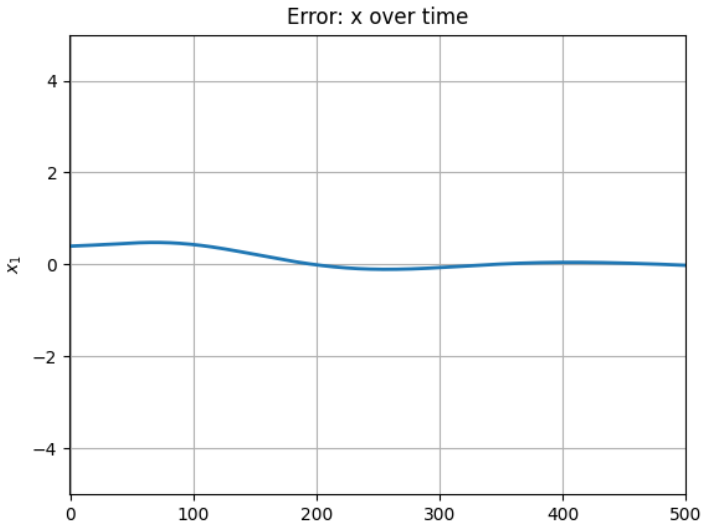

Figure 5.12: Tracking error for x_1

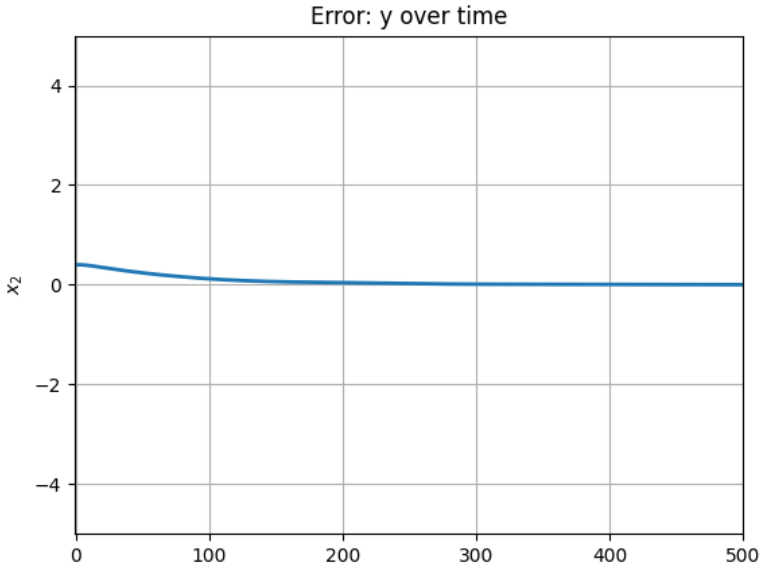

Figure 5.13: Tracking error for x_2

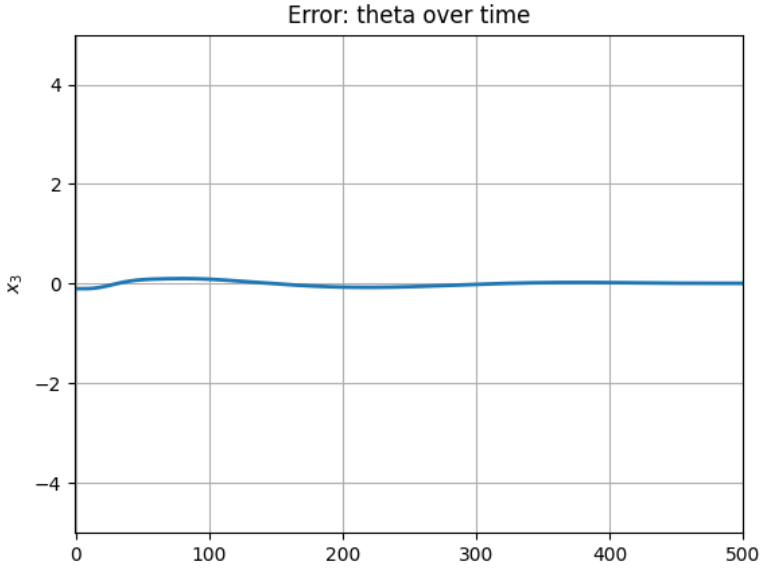

Figure 5.14: Tracking error for x_3

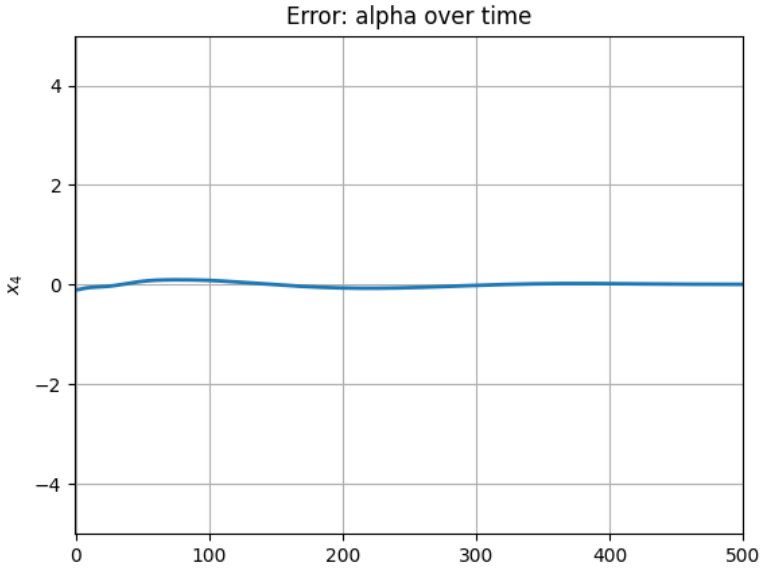

Figure 5.15: Tracking error for x_4

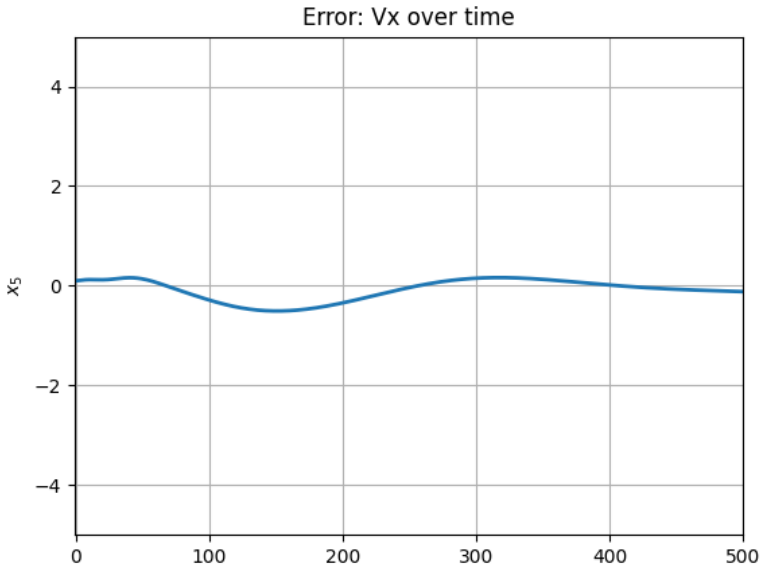

Figure 5.16: Tracking error for x_5

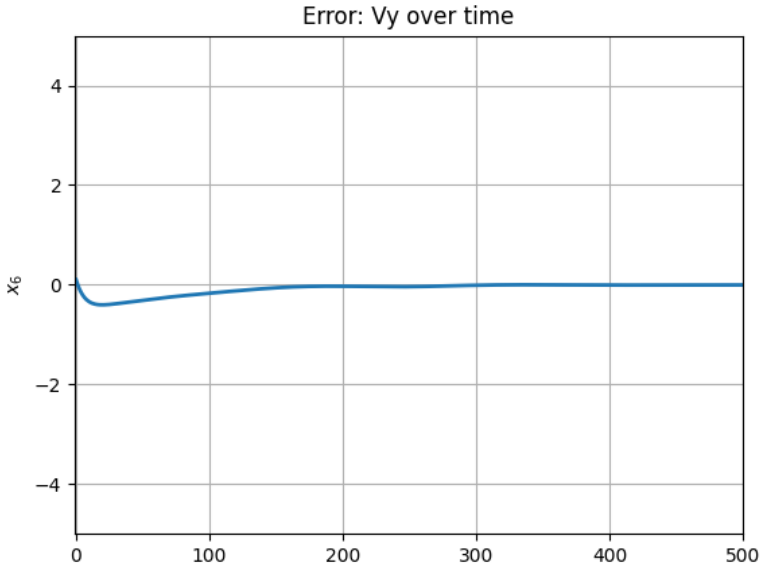

Figure 5.17: Tracking error for x_6

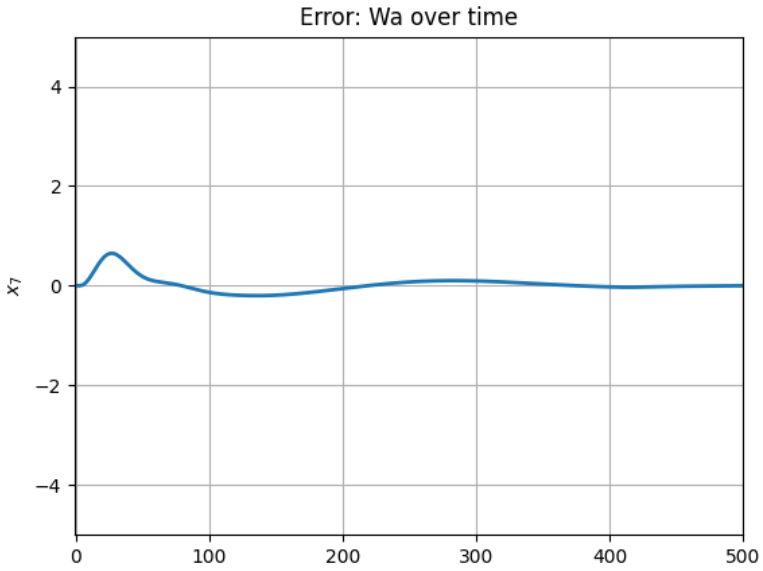

Figure 5.18: Tracking error for x_7

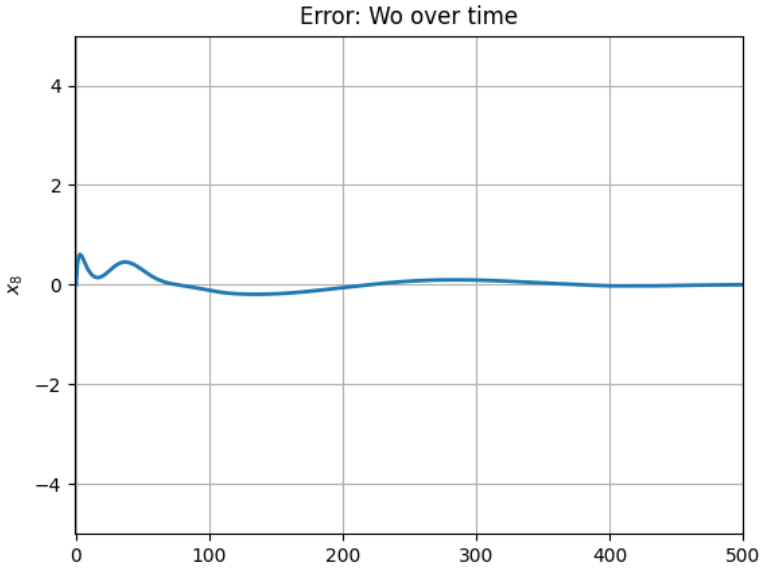

Figure 5.19: Tracking error for x_8

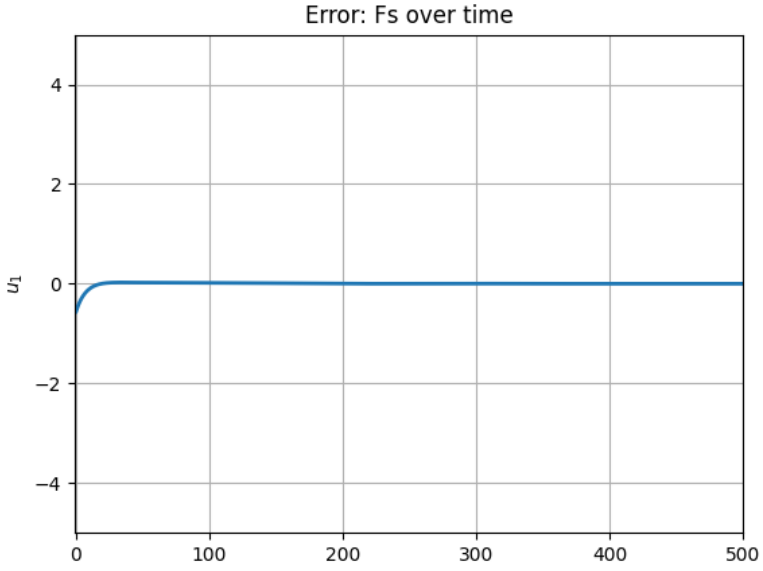

Figure 5.20: Tracking error for u_1

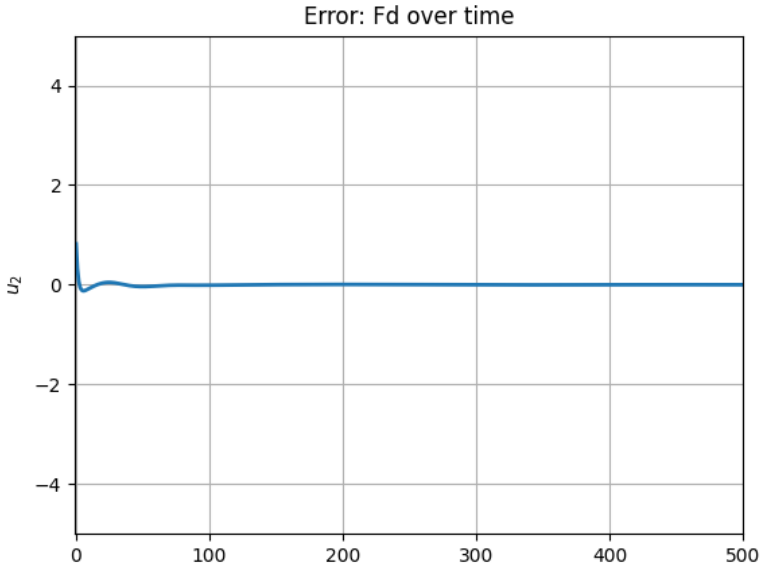

Figure 5.21: Tracking error for u_2

Task 5 - Animation

As required, we have implemented a simple animation, that starts automatically at the end of execution of Task 3.

Below, some frames of the animation showing the movement of the quadrotor along the computed optimal trajectory starting from a perturbed initial condition (see task 3):

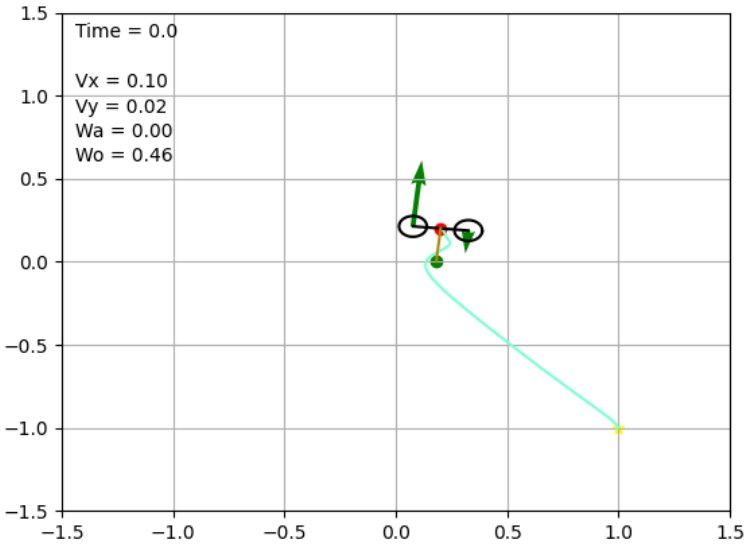

Figure 6.1: $t = 0s$

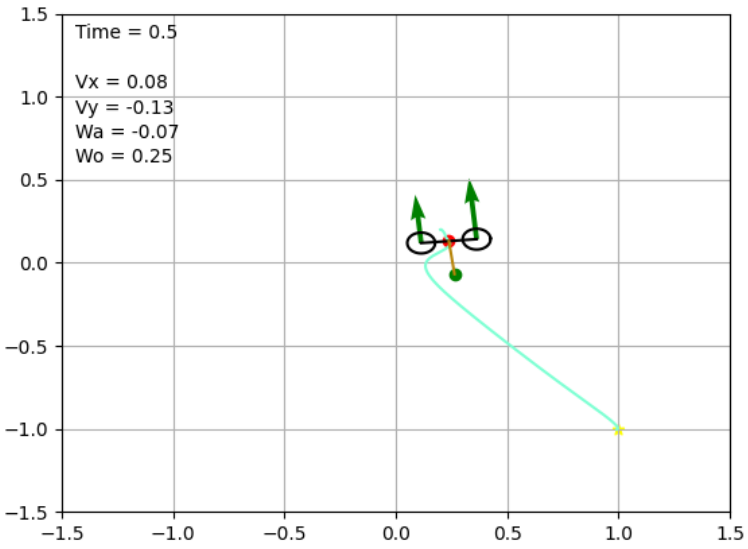

Figure 6.2: $t = 0.5s$

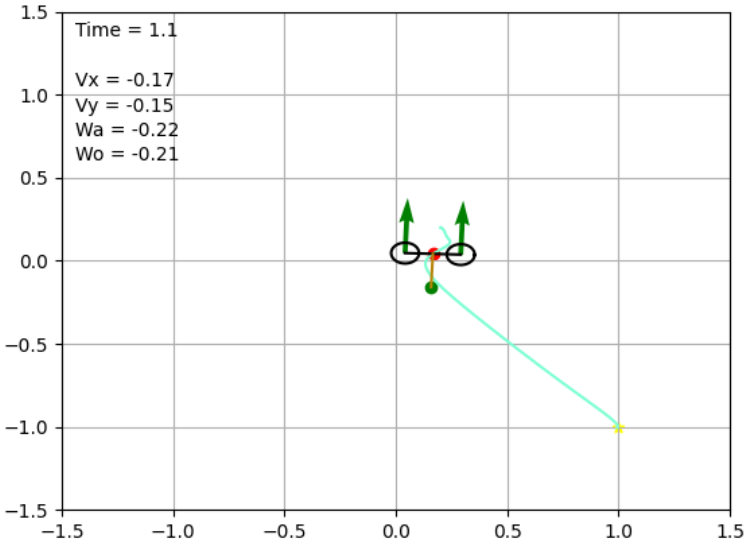

Figure 6.3: $t = 1.1s$

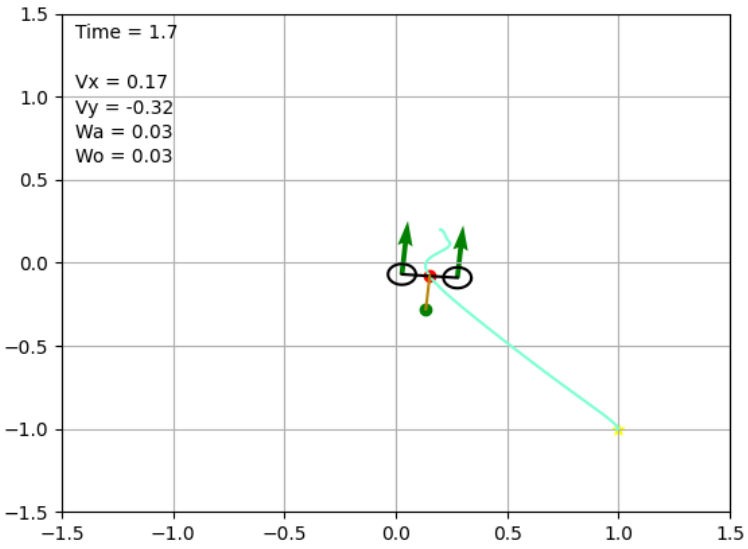

Figure 6.4: $t = 1.7s$

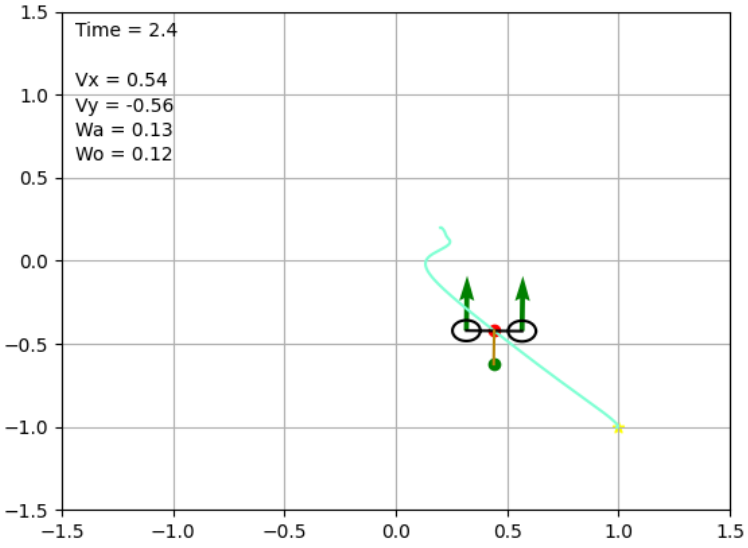

Figure 6.5: $t = 2.4s$

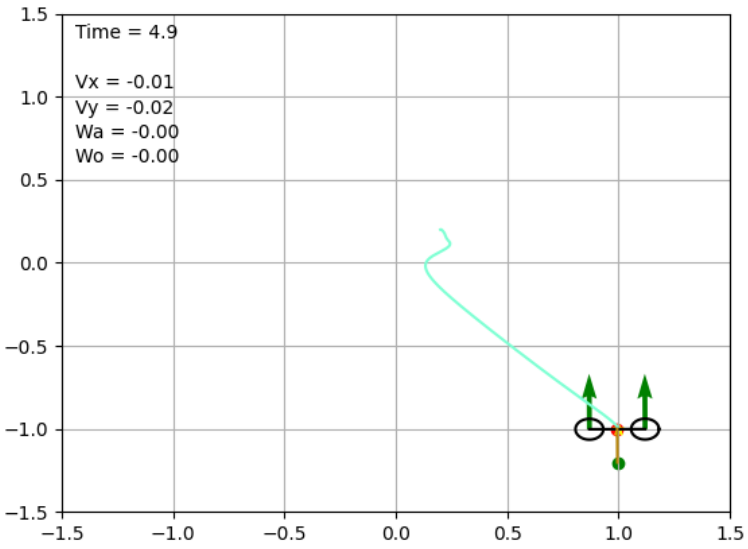

Figure 6.6: $t = 4.9s$

Conclusions

With the realization of this project we solved different tasks and were able to design an optimal control for a planar quadrotor with suspended load.

- In the first task, a Newton's like algorithm was implemented to design the optimal trajectoy allowing to follow a step reference to move from one equilibrium configuration to another. In order to do this, the Armijo algorithm for step size selection was exploited.
- In the second task we exploited again the Newton's like algorithm to design an optimal trajectory between two equilibria, but with a reference curve designed as a sigmoid function.
- In the third task we used the LQR method to follow the optimal trajectory designed in task 2. Furthermore, we proved that the controller is able to achieve the target starting with perturbed initial conditions.
- In the fourth task we applied the MPC algorithm to track the optimal trajectory designed in task 2 and we proved again that the controller is able to achieve the target starting with perturbed initial conditions.