UNIVERSITÀ DI BOLOGNA

School of Engineering

Master Degree in Computer Engineering

Optimal Control

Optimal Control of a Vehicle

Professor: **Giuseppe Notarstefano**
Tutor: **Lorenzo Sforni**

Students:

Lisa Innocenti Uccini

Davide Di Molfetta

Giovanni Praino

Academic year 2023/2024

Abstract

This report documents the development of a project aimed at the optimal control of a simplified model of an autonomous vehicle. This project has facilitated a thorough exploration and consolidation of the primary algorithms employed to identify optimal trajectories for the system. In particular, the approaches considered are the Newton's method, trajectory generation and tracking based on Linear Quadratic Regulator (LQR), and Model Predictive Control (MPC).

Contents

Introduction

This project deals with the design and implementation of an optimal control law for a simple bicycle model with static load transfer. The project is divided into the following task:

- **Task 0 - Problem setup**: discretization of the vehicle dynamics, writing of the discrete-time state-space equations and implementation of the dynamics function;
- **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 Task 2 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.

Chapter 1

Task 0 - Problem setup

1.1 Model's dynamics

A fundamental task in high-performances autonomous driving is represented by the generation of optimal trajectories considering the whole vehicle dynamics.
This project deals with the design and implementation of an optimal control law for a simple bicycle model with static load transfer, schematically represented in Figure 1.1.

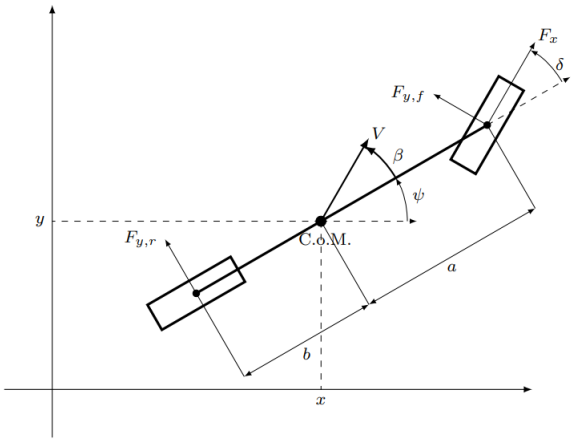

Figure 1.1: Bicycle model

The dynamic model is:

$$\begin{aligned}
\dot{x} &= V cos\beta cos\psi - V sin\beta sin\psi \\
\dot{y} &= V cos\beta sin\psi + V sin\beta cos\psi \\
m\dot{V} &= F_{y,r}\, sin\beta + F_x cos(\beta - \delta) + F_{y,f}\, sin(\beta - \delta) \\
\dot{\beta} &= \frac{1}{mV}(F_{y,r}\, cos\beta + F_{y,f}\, cos(\beta - \delta) - F_x sin(\beta - \delta) - \dot{\psi}) \\
I_z\ddot{\psi} &= (F_x sin\delta + F_{y,f}\, cos\delta)a - F_{y,r}\, b.
\end{aligned}$$

where:

- (x, y, ψ) are the cartesian coordinates of the center of mass of the vehicle and its yaw in a global reference frame;
- V is the speed modulus in body fixed reference frame;

- β is the angle between the speed and the body fixed reference frame;
- $\dot{\psi}$ is the yaw rate;
- δ is the steering angle of the vehicle;
- F_x is the force applied by the front wheel.

The lateral forces $F_{y,f}$,$F_{y,r}$ can be found as

$$F_{y,f} = \mu F_{z,f}\, \beta_f$$
$$F_{y,r} = \mu F_{z,f}\, \beta_r$$

where β_f, β_r are the front and real sideslip angles,

$$\beta_f = \delta - \frac{V sin(\beta) + a\dot{\psi}}{V cos(\beta)}$$
$$\beta_r = -\frac{V sin(\beta) - b\dot{\psi}}{V cos(\beta)}$$

The vertical forces on the front and rear wheel can be found as:

$$F_{z,f} = \frac{mgb}{a+b}$$
$$F_{z,r} = \frac{mga}{a+b}$$

The mechanical parameters of the car are available in Table 1.1.

	Parameters:	
m	1480	$[Kg]$
I_z	1950	$[Kgm^2]$
a	1.421	$[m]$
b	1.029	$[m]$
μ	1	$[nodim]$
g	9.81	$[\frac{m}{s^2}]$

Table 1.1: Mechanical parameters of the vehicle

1.2 State space and control inputs

Considering the following state space $x = [x, y, \psi, V, \beta, \dot{\psi}]^T$ and the control input $u = [\delta, F_x]^T$ it's possible to rewrite the dynamics in state-space model:

$$\dot{x}_1 = x_4 cos x_5 cos x_3 - x_4 sin x_5 sin x_3$$
$$\dot{x}_2 = x_4 cos x_5 sin x_3 + x_4 sin x_5 cos x_3$$
$$\dot{x}_3 = x_6$$
$$\dot{x}_4 = \frac{F_{y,r} sin x_5 + u_2 cos(x_5 - u_1) + F_{y,f} sin(x_5 - u_1)}{m}$$
$$\dot{x}_5 = \frac{1}{m_{x4}}(F_{y,r} cos x_5 + F_{y,f} cos(x_5 - u_1) - u_2 sin(x_5 - u_1) - x_6)$$
$$\dot{x}_6 = \frac{(u_2 sin u_1 + F_{y,f} cos u_1)a - F_{y,r} b}{I_z}$$

To represent the dynamics in the state space, we decide to add a state x_3 to avoid deriving $\dot{x}_5$ at a later time step. Just to handle the derivation of states in a more appropriate way.

1.3 Discretization

The discrete-time state-space equations were defined using the Forward Euler discretization method, since the algorithm works in discrete time:

$$
\begin{aligned}
x_{1,t+1} &= x_{1,t} + d(x_{4,t}cosx_{5,t}cosx_{3,t} - x_{4,t}sinx_{5,t}sinx_{3,t}) \\
x_{2,t+1} &= x_{2,t} + d(x_{4,t}cosx_{5,t}sinx_{3,t} + x_{4,t}sinx_{5,t}cosx_{3,t}) \\
x_{3,t+1} &= x_6 \\
x_{4,t+1} &= x_{4,t} + d\frac{F_{y,r,t}sinx_{5,t} + u_{2,t}cos(x_{5,t} - u_{1,t}) + F_{y,f,t}sin(x_{5,t} - u_{1,t})}{m} \\
x_{5,t+1} &= x_{5,t} + d\left(\frac{1}{mx_{4,t}}\left(F_{y,r,t}\cos x_{5,t} + F_{y,f,t}\cos(x_{5,t} - u_{1,t}) - u_{2,t}\sin(x_{5,t} - u_{1,t}) - x_{6,t}\right)\right) \\
x_{6,t+1} &= x_{6,t} + d\frac{(u_{2,t}sinu_{1,t} + F_{y,f,t}cosu_{1,t})a - F_{y,r,t}b}{I_z}
\end{aligned}
$$

Where d is a time interval, usually set to 10^{-3}.

1.4 Dynamics function

The dynamics function takes as input:

- x : state of the system at time t
- u : given input for the system

It mathematically describes how the state variables of the system evolve over time, producing an output that represents the state of the system at the next instant, namely t+1.

The function `dynamics()` is located within the file `dynamics.py`, which, in addition to simulating the dynamics and returning the state at the next time instant, also returns the Jacobians with respect to the states and inputs. The latter are computed for future purposes.

Chapter 2

Task 1 - Trajectory generation (I)

Compute two equilibria for your system and define a reference curve between the two. 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, the derivatives of the system have been set to zero:

$$
\begin{gathered}
\mathrm{x}_{1,t+1} - x_{1,t} - d(x_{4,t}cosx_{5,t}cosx_{3,t} - x_{4,t}sinx_{5,t}sinx_{3,t}) = 0 \\
\mathrm{x}_{2,t+1} - x_{2,t} - d(x_{4,t}cosx_{5,t}sinx_{3,t} + x_{4,t}sinx_{5,t}cosx_{3,t}) = 0 \\
\mathrm{x}_{3,t+1} - x_6 = 0 \\
\mathrm{x}_{4,t+1} - x_{4,t} - d\frac{F_{y,r,t}sinx_{5,t}+u_{2,t}cos(x_{5,t}-u_{1,t})+F_{y,f,t}sin(x_{5,t}-u_{1,t})}{m} = 0 \\
\mathrm{x}_{5,t+1} - x_{5,t} - d\left(\frac{1}{mx_{4,t}}\left(F_{y,r,t}\cos x_{5,t} + F_{y,f,t}\cos(x_{5,t}-u_{1,t}) - u_{2,t}\sin(x_{5,t}-u_{1,t}) - x_{6,t}\right)\right) = 0 \\
\mathrm{x}_{6,t+1} - x_{6,t} - d\frac{(u_{2,t}sinu_{1,t}+F_{y,f,t}cosu_{1,t})a-F_{y,r,t}b}{I_z} = 0
\end{gathered}
$$

When solving this system it was deemed appropriate to focus on one of the possible configurations:

$$
\begin{gathered}
\mathrm{x}_1 = 0 \\
\mathrm{x}_2 = 0 \\
\mathrm{x}_3 = 0 \\
\mathrm{x}_4 = \widetilde{x_4} \\
\mathrm{x}_5 = 0 \\
\mathrm{x}_6 = 0 \\
\mathrm{u}_1 = 0 \\
\mathrm{u}_2 = 0
\end{gathered}
$$

$x_4 = \widetilde{x_4}$ denotes a constant value.
Therefore the following two equilibria were chosen:

$$
\begin{gathered}
\mathrm{x}_{1,e} = [0, 0, 0, 10, 0, 0]^T \\
\mathrm{x}_{2,e} = [0, 0, 0, 20, 0, 0]^T \\
\mathrm{u}_{1,e} = u_{2,e} = [0, 0]^T
\end{gathered}
$$

2.2 Reference curve definition

The reference curve was chosen in such a way as to pass from the equilibrium $(x_{1,e}, u_{1,e})$ to the equilibrium $(x_{2,e}, u_{2,e})$.
In particular, considering a time interval of 10 seconds, the system maintains a constant speed

of 10 m/s for the first 5 seconds and then switches to a constant speed of 20 m/s for the remaining 5 seconds, as shown in 2.1:

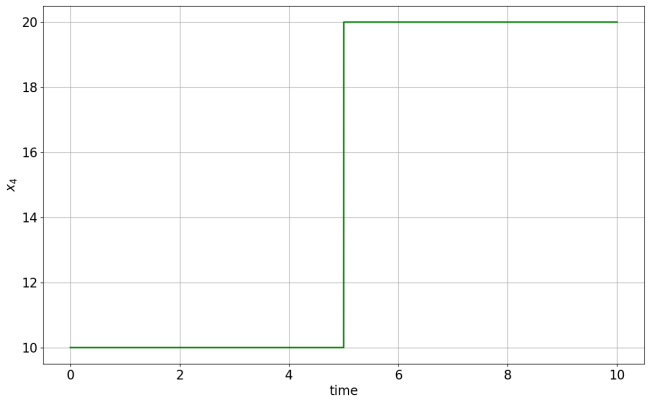

Figure 2.1: Reference curve

Instead, the position (x_1 and x_2) was calculated by forward integrating the dynamics for the first 5 seconds with the values of the first equilibrium $(x_{1,e}, u_{1,e})$ and for the following 5 seconds with the values of the second equilibrium $(x_{2,e}, u_{2,e})$. Therefore the reference curve defined is shown in the plot of Figure 2.2.

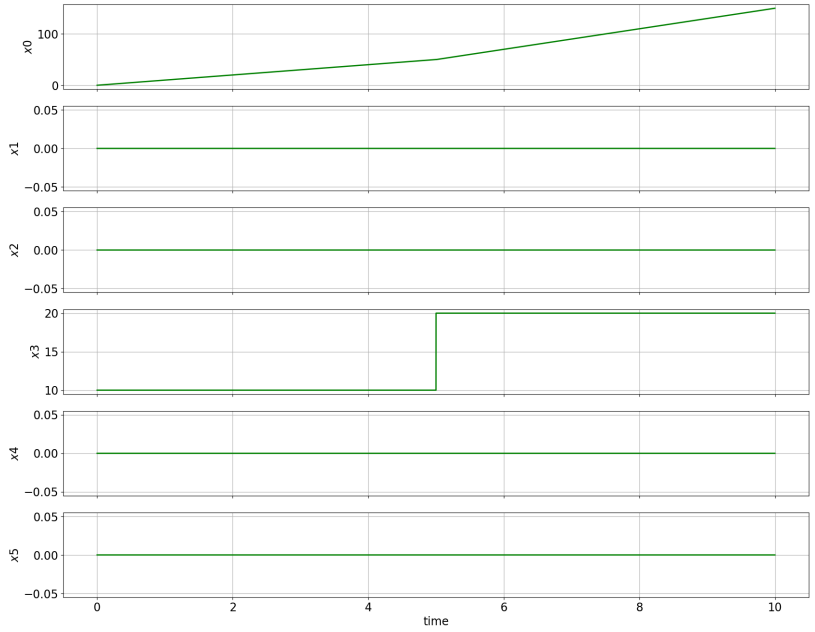

Figure 2.2: Reference curve

2.3 Linear Quadratic Regularization

To solve the optimal control problem it's been implemented Linear Quadratic Regularization (LQR).

2.3.1 Newton's method implementation

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

$$\nabla_1 f_t(x_k^t, u_k^t), \nabla_2 f_t(x_k^t, u_k^t), \nabla_1 l_t(x_k^t), \nabla_2 l_t(x_k^t, u_k^t), {}_T\,(x_k^T)$$

Compute the gradient of the reduced cost solving backwards the co-state equation, with $\lambda_k^T = \nabla l_T(x_k^T)$, 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,...,T-1:

$$\begin{aligned}\min_{\Delta x, \Delta u} \sum_{t=0}^{T-1} & \begin{bmatrix} \nabla_1 l_t(x_k^t, u_k^t \\ \nabla_2 l_t(x_k^t, u_k^t) \Delta u_t \end{bmatrix}^T \begin{bmatrix} \Delta x_t \\ \Delta u_t \end{bmatrix} \\ & + \frac{1}{2} \begin{bmatrix} \Delta x_t \\ \Delta u_t \end{bmatrix}^T \begin{bmatrix} Q_k^t & S_k^{t,T} \\ S_k^t & R_k^t \end{bmatrix} \begin{bmatrix} \Delta x_t \\ \Delta u_t \end{bmatrix} \\ & + \Delta l_T(x_k^T)^T \Delta_{xT} + \frac{1}{2} \Delta x_T^T Q_k^T \Delta x_T \end{aligned}$$

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 $x_0^{k+1} = x_{init}$

$$\begin{aligned} u_t^{k+1} &= u_t^k + K_t^k (x_t^{k+1} - x_t^k) + \gamma^k \sigma_t^k \\ x_{t+1}^{k+1} &= f_t(x_t^{k+1}, u_t^{k+1}) \end{aligned}$$

2.3.2 Costs definition

As can be seen from the following matrices, it was decided to give more weight to the speed and less weight to the force applied by the front wheel. In this case it was decided to keep the same cost also for the final cost.

$$Q_t = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \; R_t = \begin{bmatrix} 1 & 0 \\ 0 & 0.0001 \end{bmatrix} \; Q_T = Q_t$$

2.4 Results

2.4.1 Optimal Trajectory

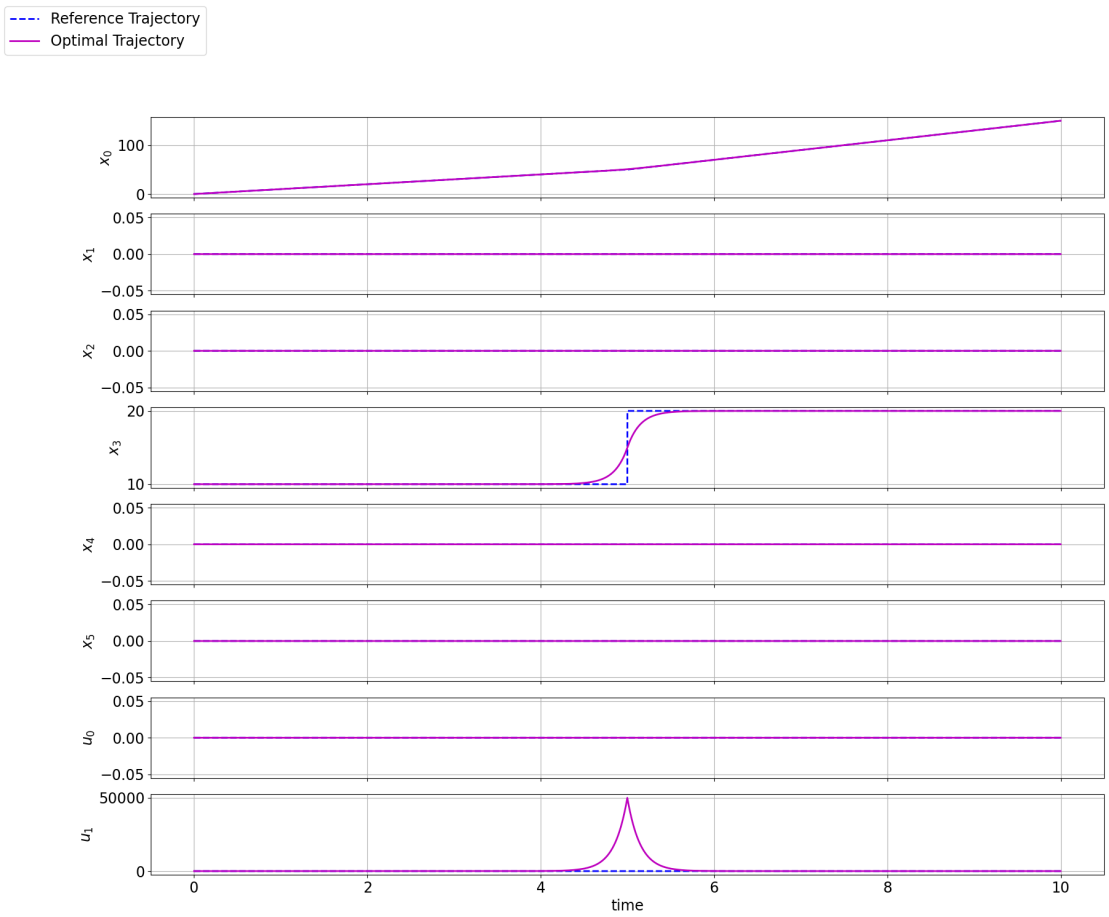

Figure 2.3: Optimal trajectory's plot

2.4.2 Armijo

As can be seen from Figure 2.4 and Figure 2.5, the algorithm converges to the optimal trajectory after just one iteration.

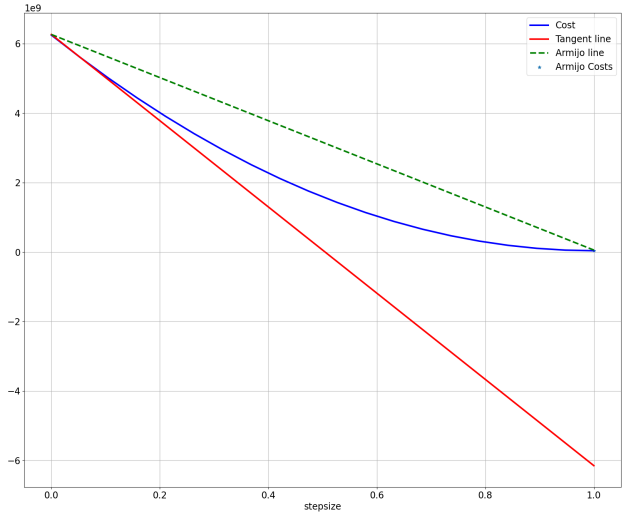

Figure 2.4: Armijo's plot iteration 0

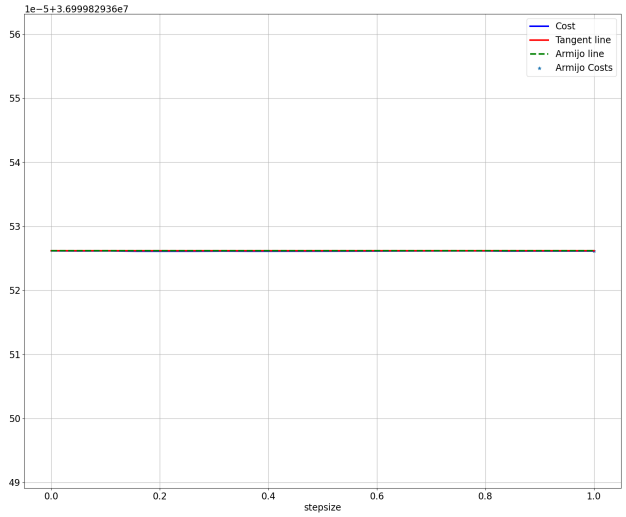

Figure 2.5: Armijo's plot iteration 1

2.4.3 Cost and descent direction

Following the execution, the plots related to the cost and direction generated are displayed.

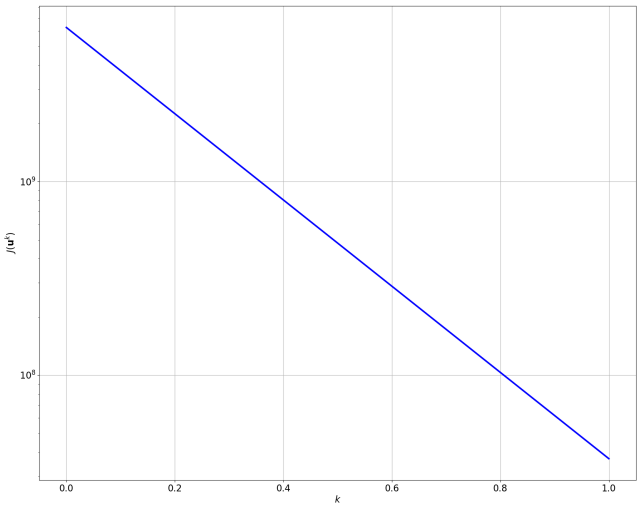

Figure 2.6: Cost's plot

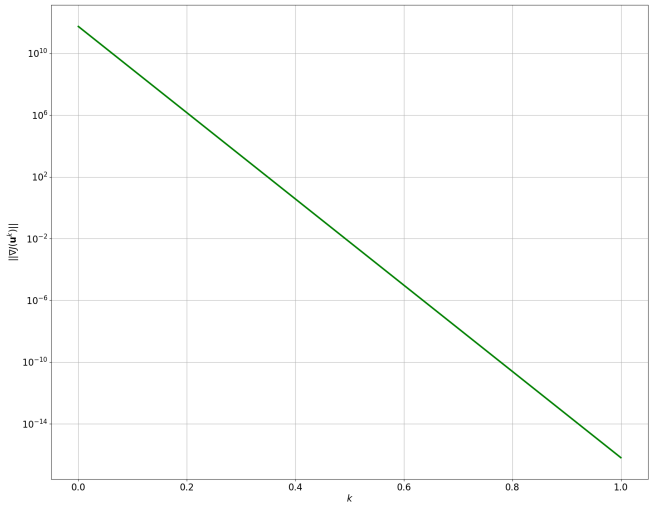

Figure 2.7: Descent direction's plot

Chapter 3

Task 2 - Trajectory generation (II)

Generate a desired (smooth) state-input curve and perform the trajectory generation task (chapter 2) on this new desired trajectory.

3.1 Reference curve definition

In task 2 it's required to generate a desired state-input curve and perform the trajectory generation task of Task 1 (chapter 2) 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 idea is to change the vehicle's lane while the speed stays constant.
The sigmoid function, often denoted as $\sigma(x)$, is defined as:

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

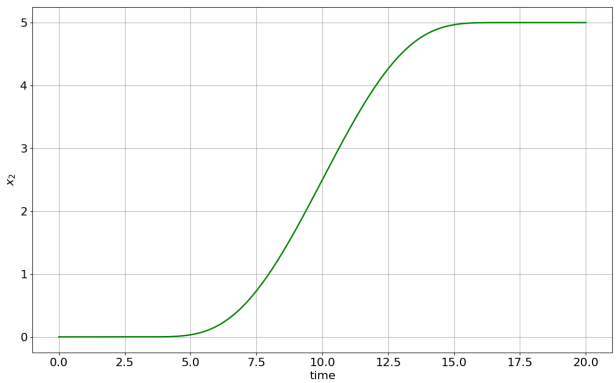

Figure 3.1: Reference curve's plot

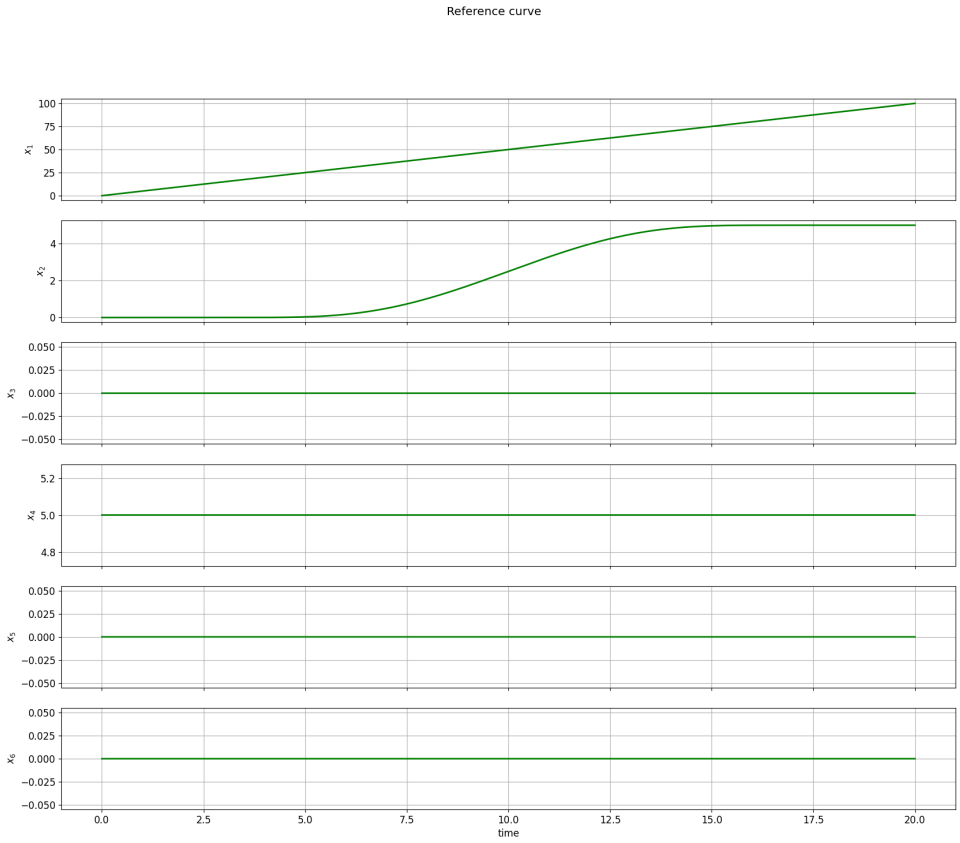

Figure 3.2: Reference curve's plot

3.2 Linear Quadratic Regularization

3.2.1 Costs definition

Below are reported the matrices representing the costs.
The values contained within have been defined following a weighted choice, considering more weight on $x_1(y)$ and on $u_1(\delta)$, less importance on $u_2(F_x)$ instead.

$$Q_t = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix} \quad R_t = \begin{bmatrix} 10000 & 0 \\ 0 & 0.0001 \end{bmatrix}$$

Q_T was chosen equal to the solution of the Riccati equation calculated for A_T and B_T, which represent the dynamics of the linearized system at instant T.

3.3 Results

3.3.1 Optimal Trajectory

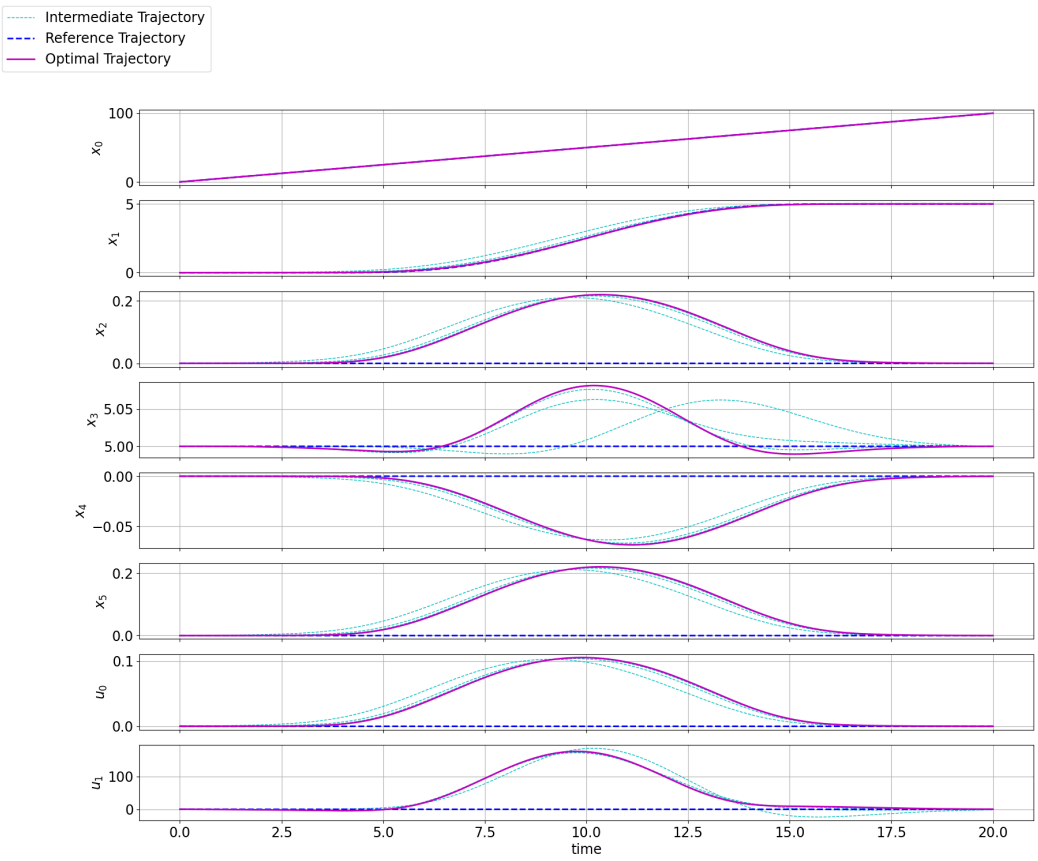

Figure 3.3: Optimal trajectory's plot

3.3.2 Cost and descent direction

Following the execution of Task 3, the plots related to the cost and direction generated are displayed.

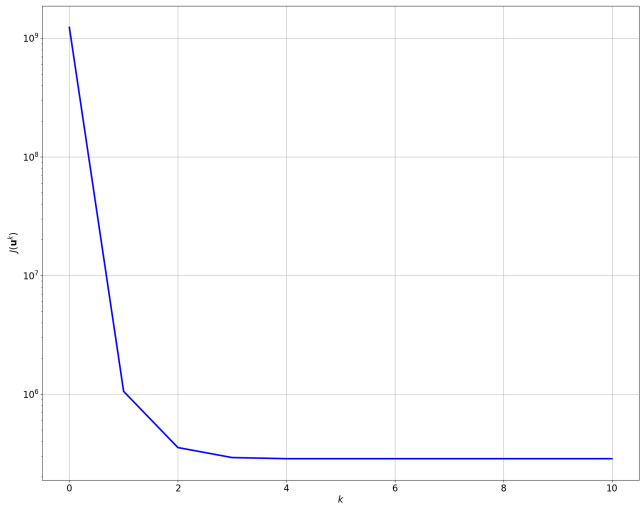

Figure 3.4: Cost's plot

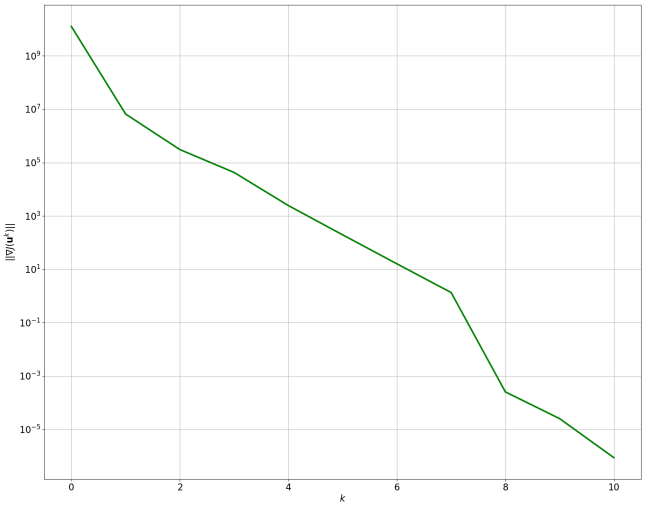

Figure 3.5: Descent direction's plot

3.3.3 Sub-optimal trajectories

The following first plot shows the optimal trajectory found at iterations 0, 2, and 10. From this image, it is possible to observe the improvement of the trajectory as the iterations progress.

The other ones shows for iterations 0, 2, and 10:

- Optimal trajectory including all states and inputs
- Armijo's plot

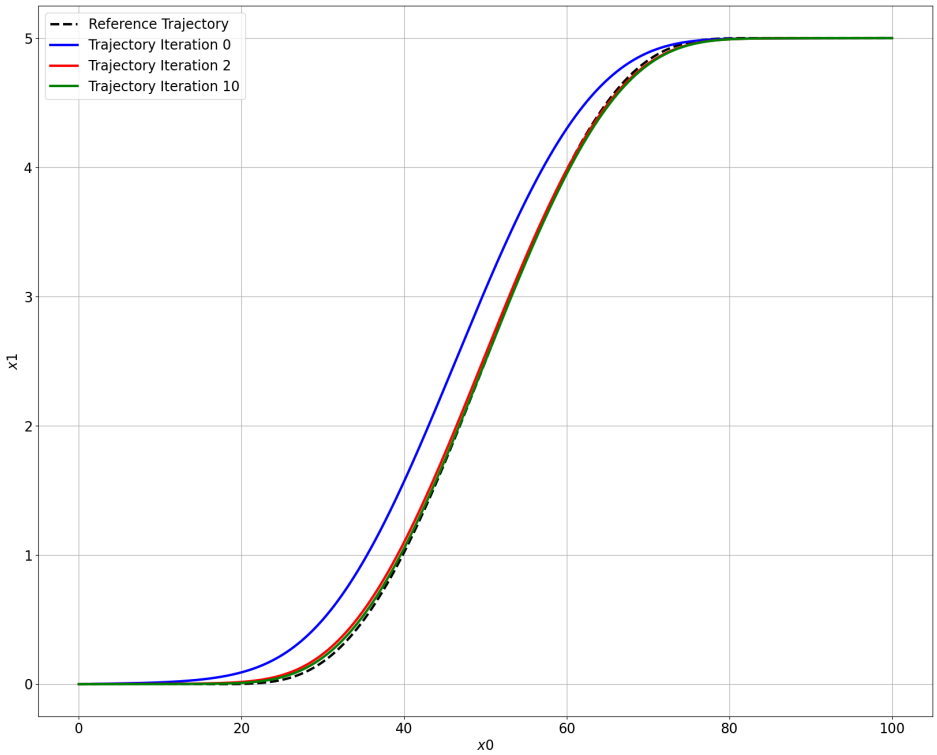

Figure 3.6: Sub-optimal trajectory for iteration 0,2 and 10

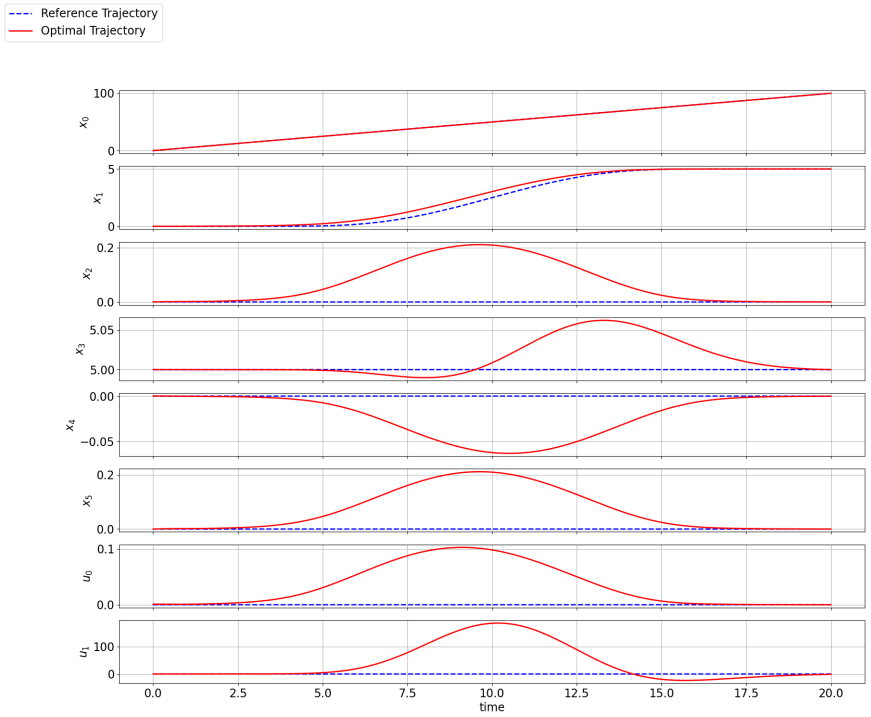

Figure 3.7: Optimal trajectory iteration 0

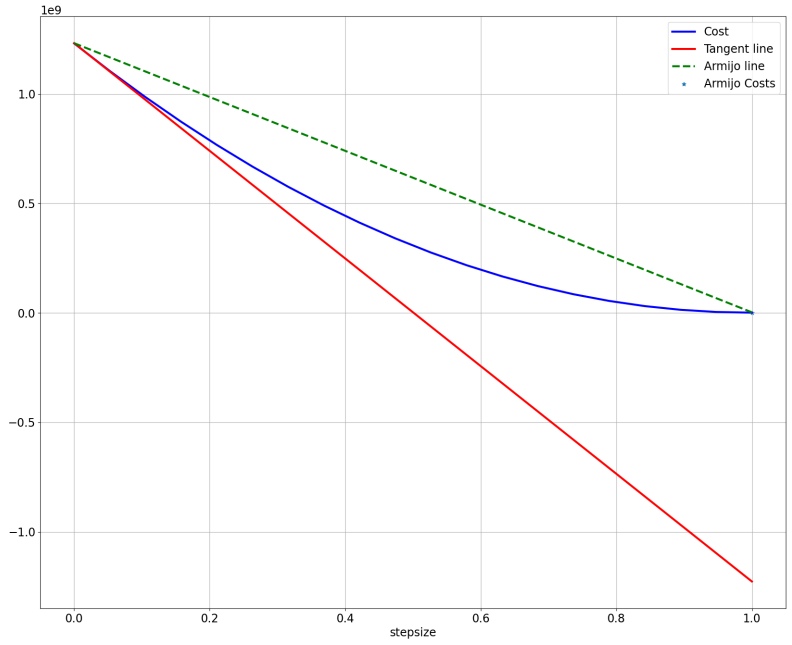

Figure 3.8: Armijo's plot iteration 0

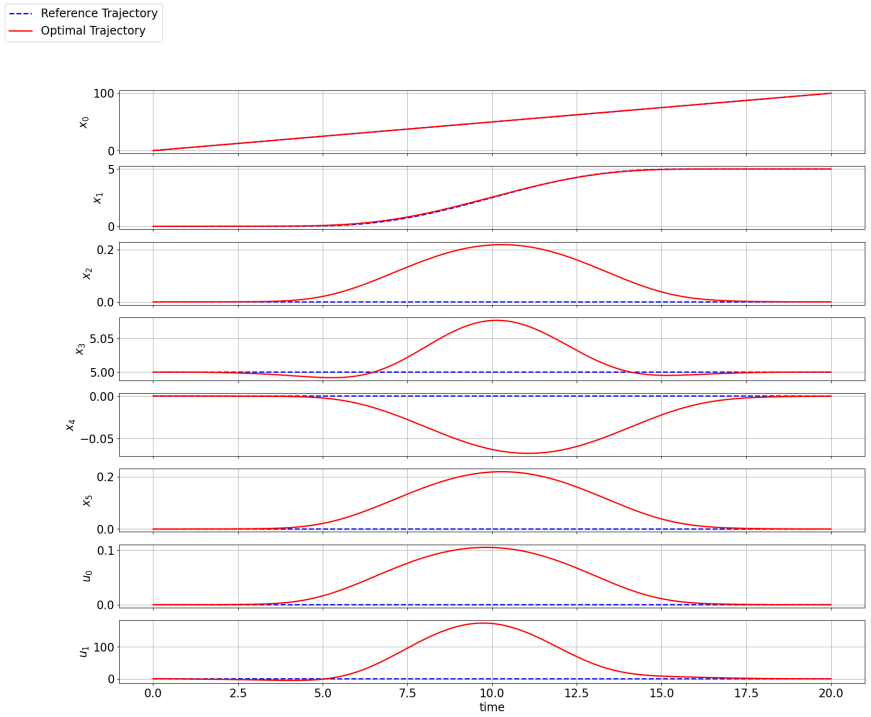

Figure 3.9: Optimal trajectory iteration 2

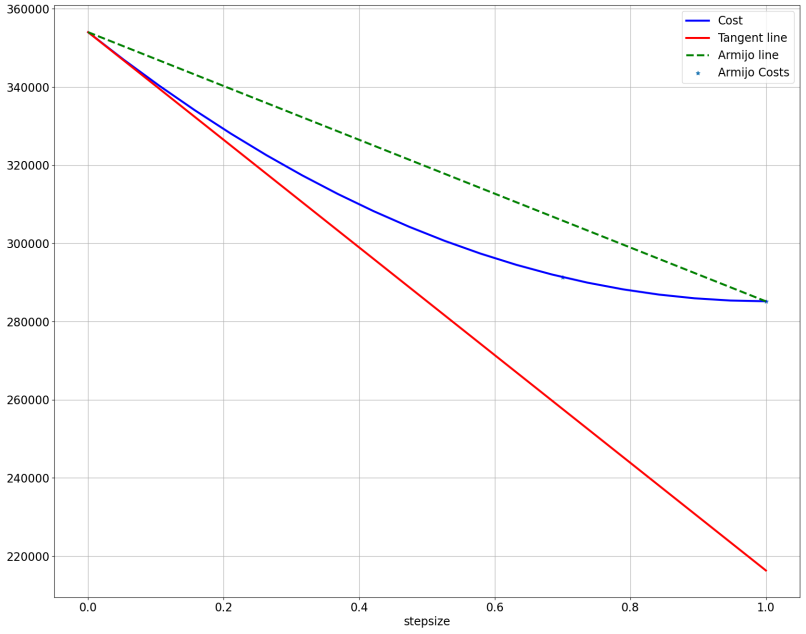

Figure 3.10: Armijo's plot iteration 2

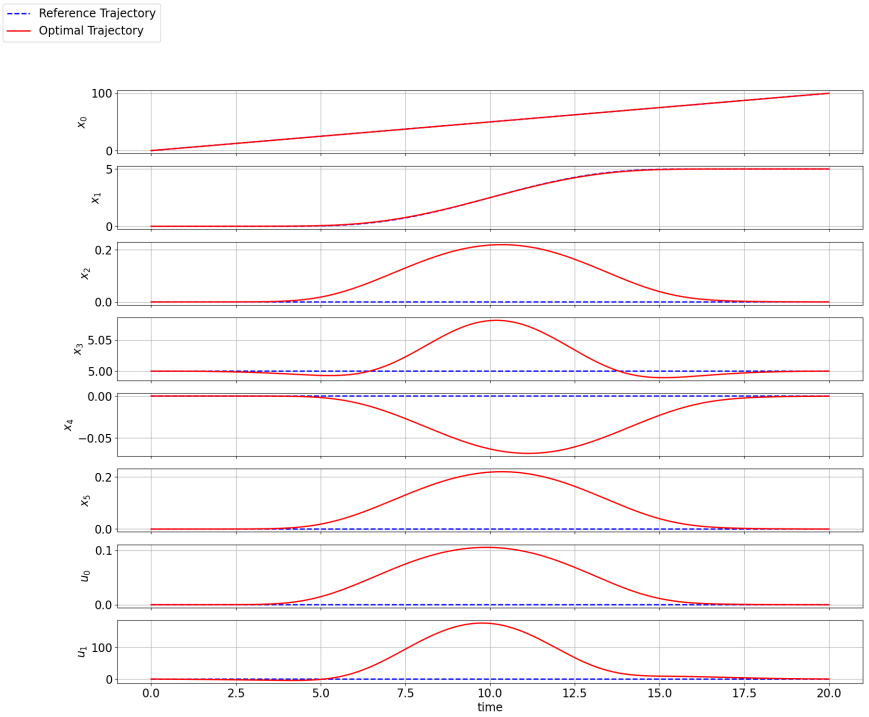

Figure 3.11: Optimal trajectory iteration 10

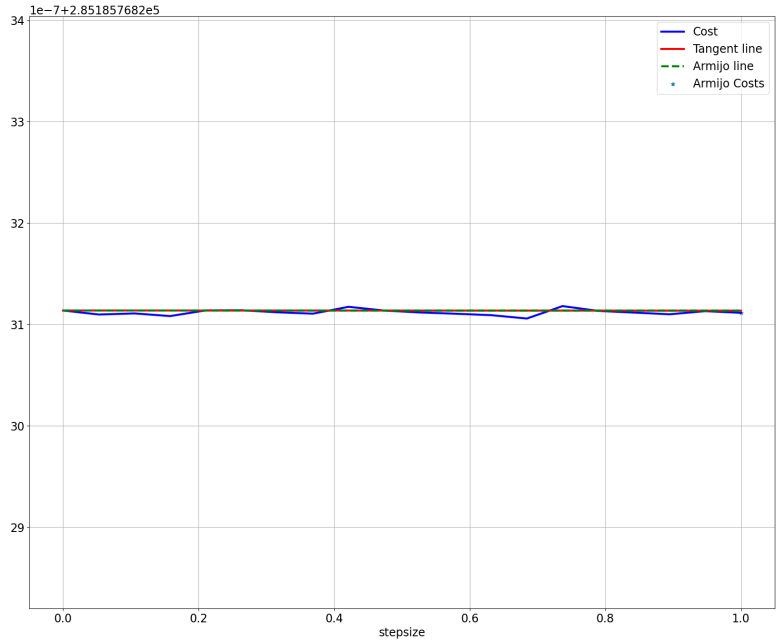

Figure 3.12: Armijo's plot iteration 10

Chapter 4

Task 3 - Trajectory tracking via LQR

Linearizing the vehicle dynamics about the (optimal) trajectory (x^{opt}, u^{opt}) computed in Task 2 (chapter 3), exploit the LQR algorithm to define the optimal feedback controller to track this reference trajectory.
In particular, you need to solve the LQ Problem

$$\min_{\substack{\Delta x_1,\cdots,\Delta x_T;\\ \Delta u_o,\cdots,\Delta u_{T-1}}} \sum_{t=0}^{T-1} \Delta x_t^T Q^{reg} \Delta x_t + \Delta u_t^T R^{reg} \Delta u_t + \Delta x_T^T Q_T^{reg} \Delta x_T$$

$$\text{subj.to } \Delta x_{t+1} = A_t^{opt} \Delta x_t + B_t^{opt} \Delta u_t \quad t = 0, ..., T-1$$
$$x_0 = 0$$

4.1 LQR-based trajectory tracking

The idea is to track the generated (optimal) trajectory (x^{opt}, u^{opt}) via a (stabilizing) feedback Linear Quadratic Regulator (LQR) on the linearization.
Applying the feedback controller designed on the linearization to the nonlinear system, it's possible to track (x_t, u_t).
Namely, for all $t = 0, ..., T-1$ [$T-1$ is finite horizon] it applies

$$u_t = u_t^{opt} + K_t^{reg}(x_t - x_t^{opt})$$
$$x_{t+1} = f_t(x_t, u_t)$$
$$\text{with x0 given.}$$

The nominal trajectory and the feedback gains have been computed on a finite horizon of length T.

4.2 Costs definition

The cost matrices employed in this task are the same as those used in Task 2 (chapter 3). This decision stems from the constancy in significance attributed to the state and input elements, which has persisted unchanged.

$$Q_t = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix} \quad R_t = \begin{bmatrix} 10000 & 0 \\ 0 & 0.0001 \end{bmatrix}$$

Q_T = solution of Riccati equation

4.3 Results

In Figure 4.1 can be observed the plot of the trajectory tracked via LQR in relation to the desired optimal trajectory for all states and inputs.
To assess how well the controller tracks the desired trajectory, the vehicle's starting position is intentionally varied from the optimal trajectory. The plot in Figure 4.1 illustrate instances of the controller's tracking performance.

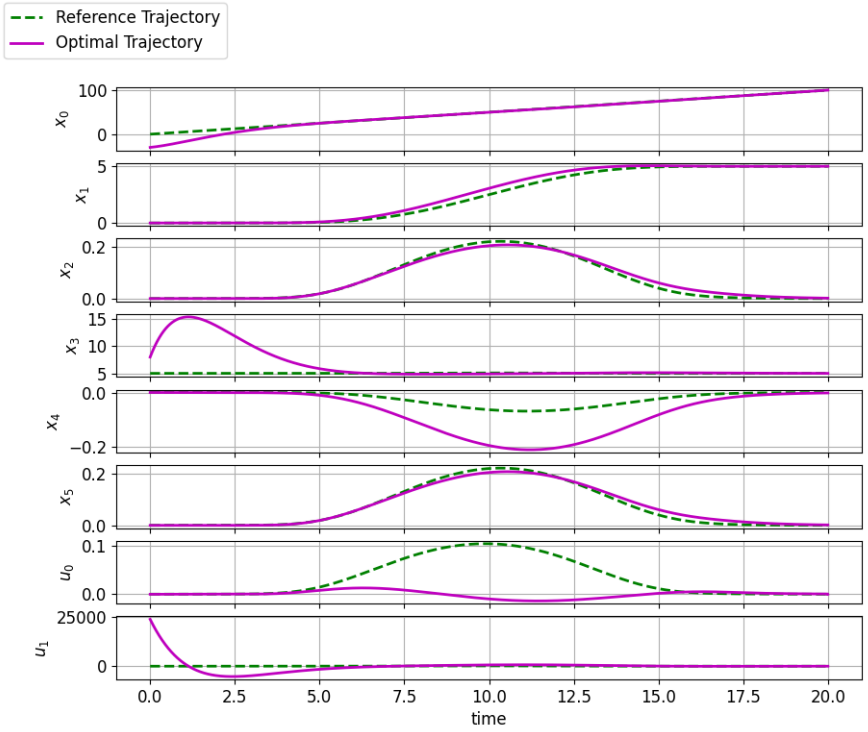

Figure 4.1: Optimal trajectory

Given the perturbed inputs as follows:

$$x_0 = [-30, 0, 0, 8, 0, 0]^T$$

The plot in Figure 4.2 shows the tracking error for different initial conditions.

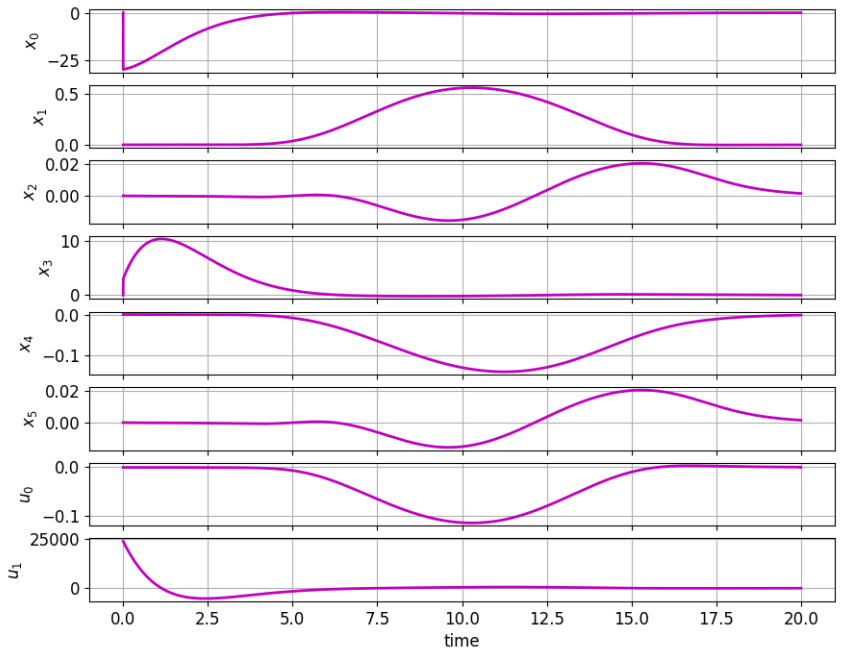

Figure 4.2: Error

Chapter 5

Task 4 - Trajectory tracking via MPC

Linearizing the vehicle dynamics about the (optimal) trajectory (x^{opt}, u^{opt}) computed in Task 2 (chapter 3), exploit an MPC algorithm to track this reference trajectory.

5.1 Model Predictive Control

MPC (Model Predictive Control) is a method that predicts the future behavior of the system and computes an optimal control input to minimize a predefined cost function, taking into account constraints on the system. At periodic intervals, MPC iteratively recomputes the optimal control input based on updated information, allowing for precise and responsive control over time.

The operation of MPC is as follows:
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^{meas}
- Apply the first control input $u*_{t|t}$
- Measure x_{t+1} and repeat

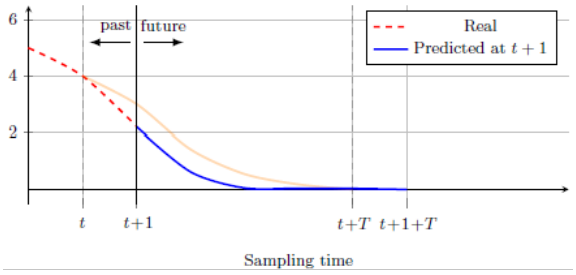

Figure 5.1: Model Predictive Control

5.2 Linear Quadratic Case

Consider the case where the dynamics is linear and the cost quadratic. At each time t, with initial measured state x_t^{meas}, solve the LQ problem.

$$\min_{\substack{x_1,\dots,x_{t+T} \\ u_1,\dots,u_{t+T-1}}} \sum_{\tau=t}^{t+T-1} x_\tau^T Q x_\tau + u_\tau^T R_\tau u_\tau + x_{\tau+T}^T Q_T x_{\tau+T}$$
$$\text{subject to } \mathrm{x}_{\tau+1} = A_\tau x_\tau + B_\tau u_\tau \ \forall \tau = t, \dots, t+T-1$$
$$\mathrm{x}_t = x_t^{meas}$$

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 $Q_\tau = Q_\tau^T >= 0 \ \forall \tau$
- $R_\tau \in \mathbb{R}^{mxm}$ and $R_\tau = R_\tau^T > 0 \ \forall \tau$
- $Q_T \in \mathbb{R}^{nxn}$ and $Q_T = Q_T^T >= 0$

5.3 Tracking of the Optimal Trajectory

The goal is to track the optimal trajectory through Model Predictive Control, therefore, the cost function to minimize is the one in which x_τ and u_τ represent the errors between the predicted trajectory (x^{mpc}, u^{mpc}) and optimal one (x^{mpc}, u^{mpc}).

The considered LQ problem is, consequently, as follows:

$$\min_{\substack{x_1,\dots,x_{t+T} \\ u_1,\dots,u_{t+T-1}}} \sum_{\tau=t}^{t+T-1} (x_\tau^{mpc} - x_\tau^{opt})^T Q_\tau (x_\tau^{mpc} - x_\tau^{opt}) +$$
$$+ (u_\tau^{mpc} - u_\tau^{opt})^T R_\tau (u_\tau^{mpc} - u_\tau^{opt}) +$$
$$+ (x_{t+T}^{mpc} - x_{t+T}^{opt})^T Q_T (x_{t+T}^{mpc} - x_{t+T}^{opt})$$

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

5.3.1 Costs definition

The matrices representing the costs, as shown below, have been kept identical to those used in Task 2 (chapter 3), as the significance associated with the state and input elements remains unchanged.

$$Q_t = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix} \quad R_t = \begin{bmatrix} 10000 & 0 \\ 0 & 0.0001 \end{bmatrix}$$

$$Q_T = \text{solution of Riccati equation}$$

5.4 Results

Following, the plot of the MPC trajectory in relation to the desired optimal trajectory for all states and inputs can be observed.

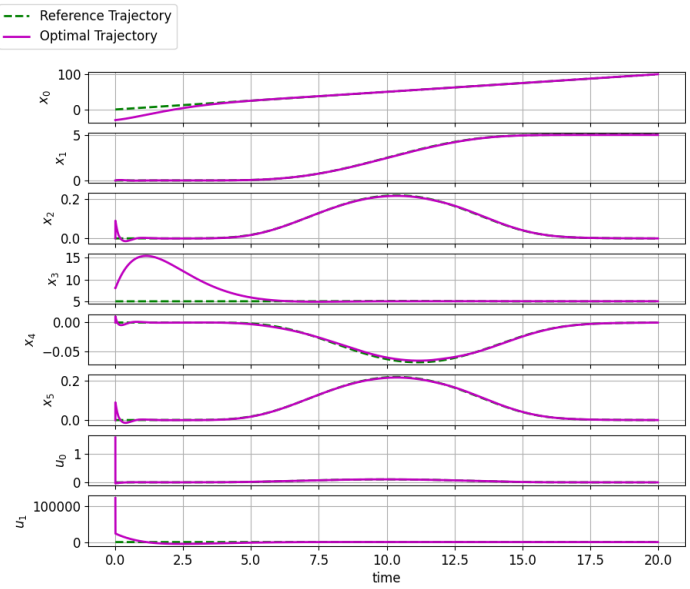

Figure 5.2: Optimal Trajectory

Given the perturbed inputs as follows:

$$x_0 = [-30, 0, 0, 8, 0, 0]^T$$

the following plot shows the tracking error for different initial conditions.

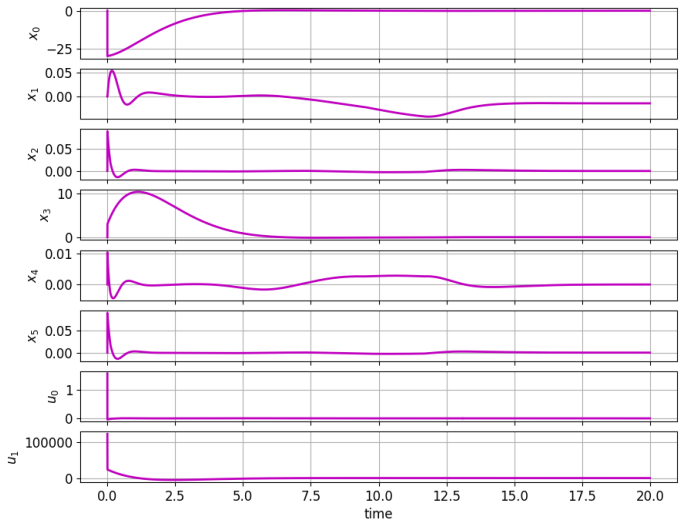

Figure 5.3: Error

Chapter 6

Task 5 - Animation

Produce a simple animation of the vehicle executing Task 3 (chapter 4).

6.1 Results

Given the optimal trajectories obtained in the computation of Task 3 (4), the animation was implemented using Python. Below, snapshots of the animation are presented.

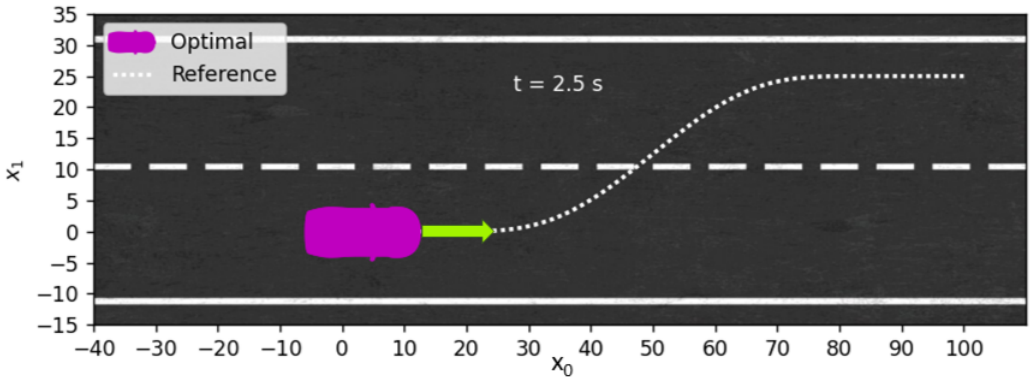

Figure 6.1: Animation of the vehicle at t=2.5

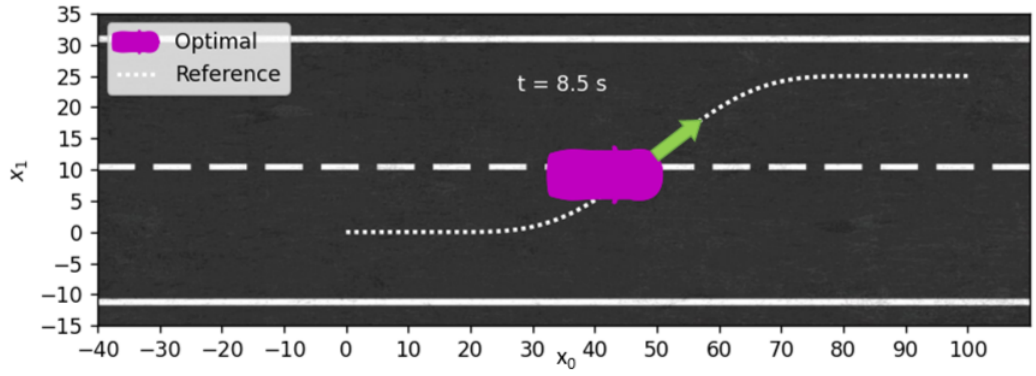

Figure 6.2: Animation of the vehicle at t=8.5

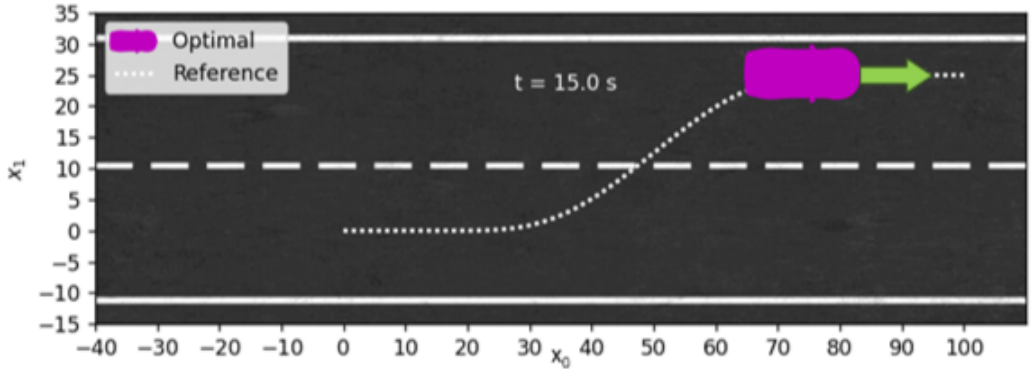

Figure 6.3: Animation of the vehicle at t=15

Conclusions

Through the successful execution of this project, various objectives were addressed, leading to the development of an optimal control system for a simple bicycle model with static load transfer for an autonomous vehicle. The project milestones include:

- **Task 0 - Problem setup**: discretization of the vehicle dynamics, writing of the discrete-time state-space equations and implementation of the dynamics function;
- **Task 1 - Trajectory Generation (I)**: Implemented a Newton-like algorithm to craft an optimal trajectory enabling the autonomous vehicle to smoothly transition from one equilibrium configuration to another. The Armijo algorithm was utilized for step size selection;
- **Task 2 - Trajectory Generation (II)**: Employed the Newton-like algorithm once more to formulate an optimal trajectory between two equilibria, incorporating a reference curve designed as a sigmoid function;
- **Task 3 - Trajectory Tracking via LQR**: Utilized the Linear Quadratic Regulator (LQR) method to track the optimal trajectory designed in Task 2. Additionally, demonstrated the controller's efficacy in reaching the target even when initiated from perturbed initial conditions;
- **Task 4 - Trajectory Tracking via MPC**: Applied the Model Predictive Control (MPC) algorithm to follow the optimal trajectory crafted in Task 2. Once again, substantiated the controller's ability to attain the target despite perturbed initial conditions.