UNIVERSITÀ DI BOLOGNA

School of Engineering

Master Degree in Automation Engineering

Optimal Control

OPTIMAL CONTROL OF A VEHICLE

Professor: **Giuseppe Notarstefano**

Students: Bertamé Sebastiano
Rapallini Antonio

Academic year 2023/2024

Abstract

This project consist in the design and implementation of an optimal control law for a vehicle described by simple bicycle model.

The work is structured into a series of tasks, each building upon the last to evolve our understanding and control of the vehicle model. We start by discretizing the dynamics and proceed to generate optimal trajectories, ensuring the stability and efficiency of vehicle motion. The subsequent tasks involve tracking the trajectory through Linear Quadratic Regulator (LQR) and Model Predictive Control (MPC) strategies, culminating in a visualization of the vehicle executing the tracking of the optimal trajectory evaluated with the LQR.

The job has been done entirely in Python, and this report documents our approach, our analyses, and implemented solutions for addressing the challenges posed by optimal vehicle control. We present our findings, supported by graphical representations for clarity and visualization.

Contents

List of Figures

Introduction

In the field of high-performances autonomous driving, our project focuses on the complex challenge of generating optimal vehicle trajectories that consider the dynamics of the vehicle.

In our case this involves the design and the implementation of an optimal control law for a simplified bicycle model which is represented in Figure 1. The project aims to navigate through the complexity of vehicle dynamics, using the mathematical and computational tools of optimal control theory.

At the base of our exploration lies the vehicle's state space and its dynamic model. The state space is composed by the following quantities $x = (x, y, \psi, V, \beta, \dot{\psi})$ where:

- x, y, ψ are Cartesian coordinates of the center of mass of the vehicle
- V is the velocity modulus in the body fixed reference frame
- β is the angle between the speed and the body fixed reference frame
- $\dot{\psi}$ is the yaw rate

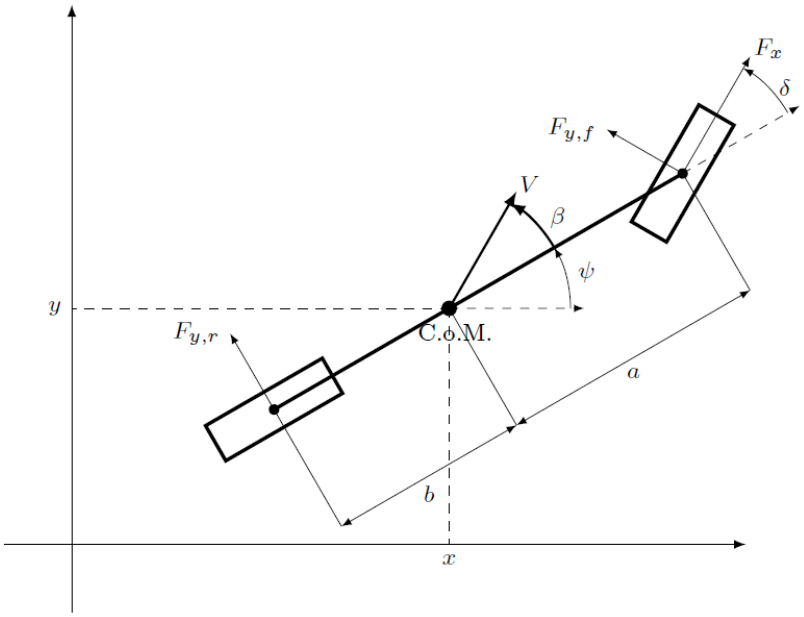

Figure 1: Bicycle Model

The control input is $u = (\delta, F)$ where:

- δ is the steering angle
- F is the force apply by the front wheel

The dynamic model instead is described by the following equations:

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

where all the quantities and mechanical parameters where given.

Task 0

Problem setup

In addressing a problem of this nature, the initial step involves discretizing the dynamics, as the algorithms we subsequently implement operate in discrete time. Therefore we have evaluated the discrete-time equations and their derivatives (in order to compute the matrices A and B), into the file *Dynamics.py*.

For evaluating the discrete-time version of the dynamics we have used the forward Euler method, i.e:

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

where $\delta > 0$ is a sufficiently small discretization step (we have used $\delta = 0.01$ and $f_{CT}(\cdot, \cdot, \cdot)$) the continuous-time dynamics.

We have evaluated such procedure for each dynamic equation of the vehicle and afterwards we exploited the derivatives of the dynamics w.r.t. to $x(fx)$, and $u(fu)$. With those terms we later evaluated the matrices A and B as their transposes. We firstly tried to compute the derivatives by hand, and later computed them with a built-in python function contained in the SymPy library. This dual verification process ensures the precision of our dynamic calculations and their corresponding derivatives.

0.1 Test

Once we had implemented the dynamics, we proceeded with a testing approach to validate the accuracy of both the dynamics and their derivatives. To assess the correctness of the dynamics themselves, we conducted an open-loop test by providing constant inputs for a defined period and verifying whether the resulting trajectory matched our expected outcomes. For example given a velocity of 5 $\frac{m}{s}$ for 10 sec with steering angle $\delta = 0$ and starting from the point $(x, y) = (0, 0)$ the final pose should be in $(x, y) = (50, 0)$.

Later we have used the following approach to check the correctness of

the derivatives.

$$f(\bar{x} + \delta_{t+1}) - f(\bar{x}) \simeq \left.\frac{\partial f}{\partial x}\right|_{\bar{x}} \delta x$$

Choosing a small δx and perform this calculation, using as f our dynamics and as $\frac{\partial f}{\partial x}$ our evaluated descrete time derivatives, if the difference between the terms is almost zero, the evaluated derivatives are correct. We have done the same thing also for the inputs u.

Task 1

Trajectory generation (I)

After completing the discretization process, we proceeded to generate a reference trajectory. This involved establishing two system equilibria and connecting them with a curve. Subsequently, we utilized a Newton-like algorithm for optimal control to calculate the optimal transition between these equilibria.

1.1 Equilibrium evaluation

In line with the suggested approach, we employed the cornering equilibria method to assess the equilibria. This method involves determining equilibria related to system configurations in which $\dot{\beta}$, $\dot{V}$ and $\ddot{\psi}$ are 0.

After completing this step, we designated specific values for both velocity and β. Subsequently, we computed the remaining equilibria using the 'fsolve' method. To obtain the values of x, y, and psi, we performed a forward integration using the descrete time dynamic equations.

1.2 Definition of the reference curve

After calculating two equilibria, we established a connection between them using a step reference. Instead, for the trajectory of x, y, and psi, we determined it through integration taking into account that for half of the time we had an equilibrium and for the rest of the time the other one. The results are depicted in the figure 1.1. Using the evaluated values for x and y we evaluated the reference space trajectory for our bicycle, that is reported in figure 1.2.

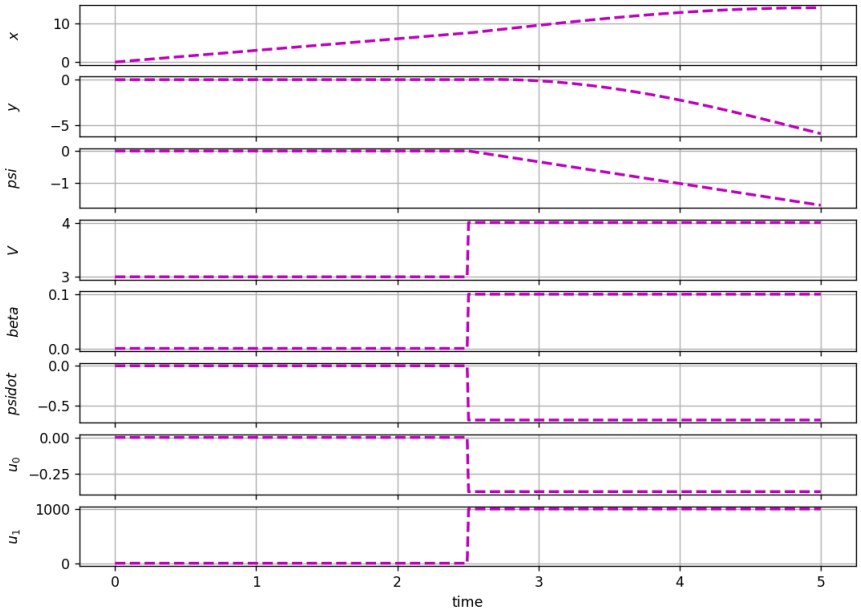

Figure 1.1: Trajectory reference

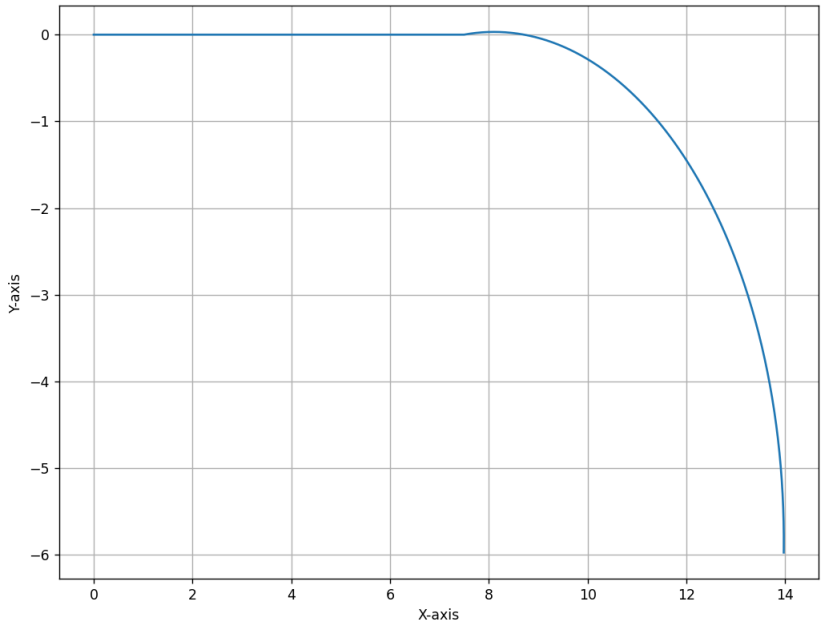

Figure 1.2: Vehicle reference trajectory

1.3 Newton's method evaluation

As optimization algorithm, we have used the (regularized) Newton's method for optimal control which is based on the Hessians of the cost only.

Our methodology involved progressively testing the algorithm, initially applying it to a simplified dynamic model and then gradually incorporating complexity over time. Initially, we applied the gradient method to a pendulum model. Following the success of this approach, we transitioned to Newton's method. Once we achieved positive results with Newton's method, we implemented our bicycle dynamics model. We initially tested it on a quasi-static trajectory, starting the algorithm in close proximity to this trajectory. After ensuring successful operation on this simple path, we moved on to define the trajectory as shown in the figure 1.2.

Subsequently, we tested the algorithm using a constant step size of 0.1. Once everything functioned smoothly with this setting, we proceeded to implement the Armijo rule for updating the step size

Next, we present graphics that illustrate the evaluation of Newton's method using the reference trajectory shown in figure 1.2. The results shown are derived from a simulation that utilized the Armijo rule for updating the step size. We have included several plots from the initial phases of the Armijo rule application, a few from the later stages, and some from intermediate evaluations of the trajectories.

The parameters used for the simulation are the following:

- $dt = 0.01$ (discretization stepsize - Forward Euler)
- $tf = 5$ (final time in seconds)
- $TT = \frac{tf}{dt} = 500$ (number of discrete time samples)
- number of iterations = 35

Moreover, for the armijo we have used the following parameters:

- $c = 0.5$
- $\beta = 0.7$
- $stepsize_0 = 1$
- $armijo\_maxiters = 20$

After several tries we decided to implement the closed-loop update in the Newton's method since it was for us the solution that worked better. The update of the u and x are the following:

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

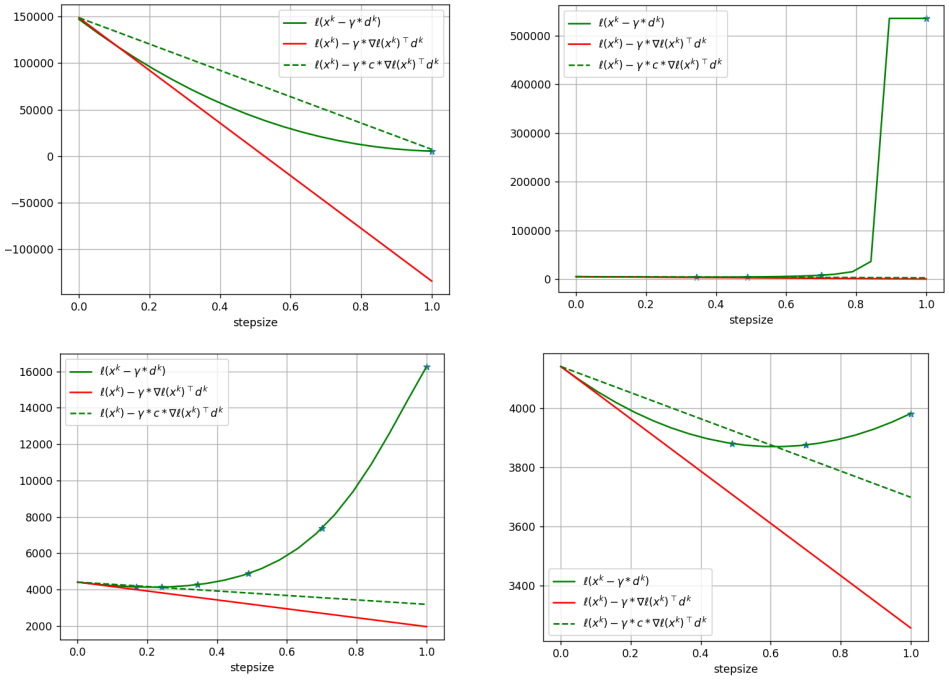

Figure 1.3: First 4 Armijo iterations

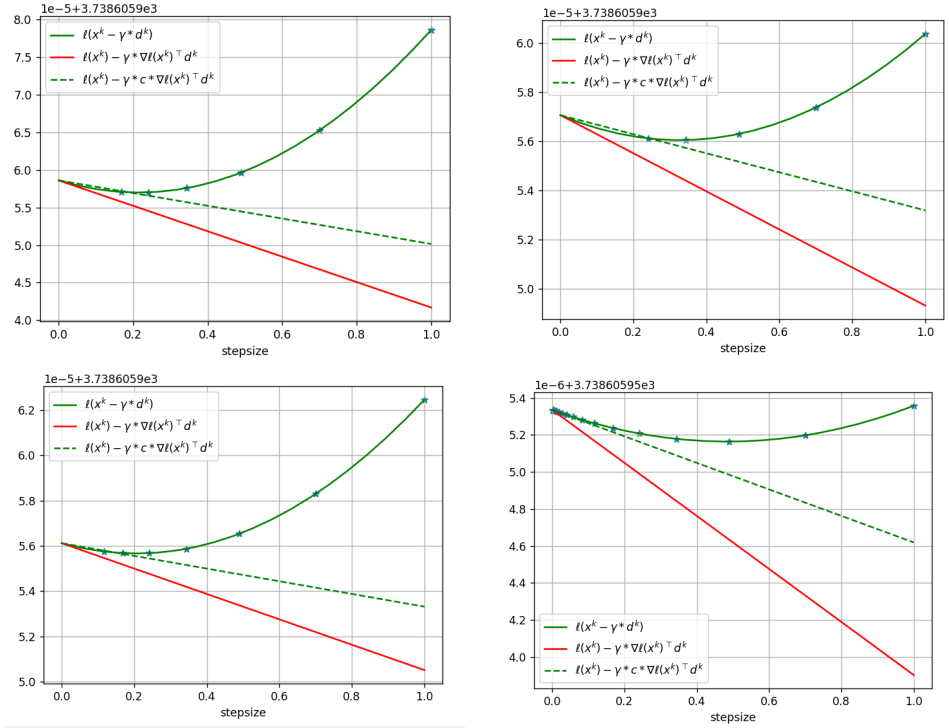

Figure 1.4: Last 4 Armijo iterations (before convergence)

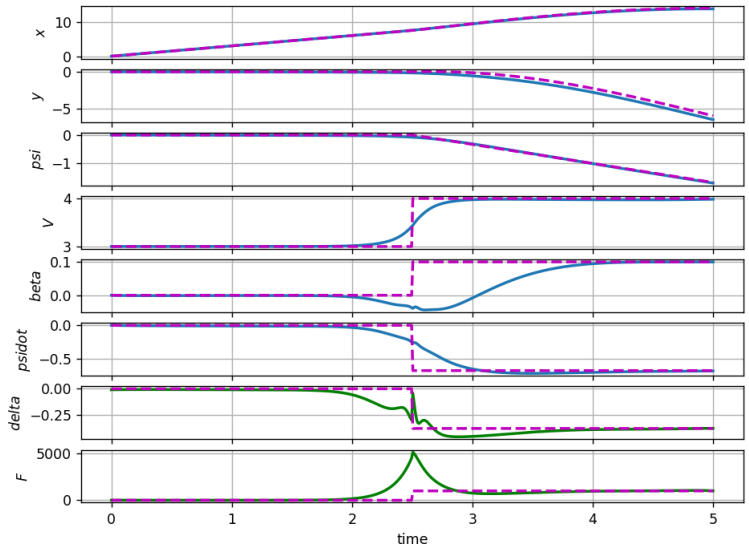

Figure 1.5: Trajectory obtained with Newton's method after 5 iterations

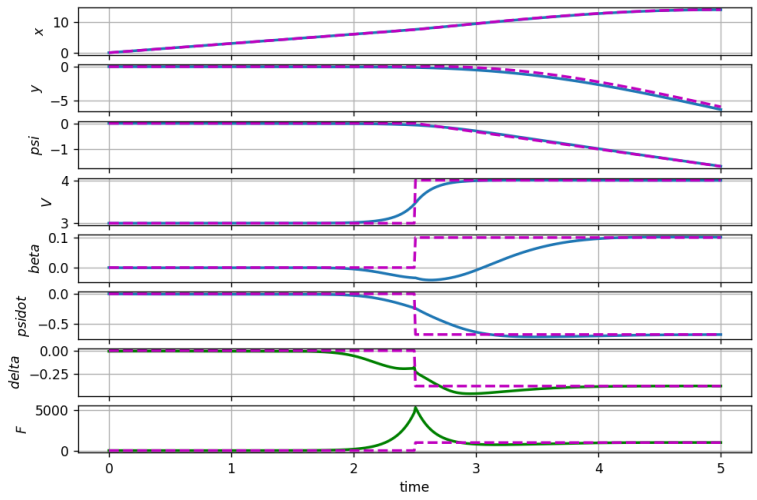

Figure 1.6: Trajectory obtained with Newton's method after 10 iterations

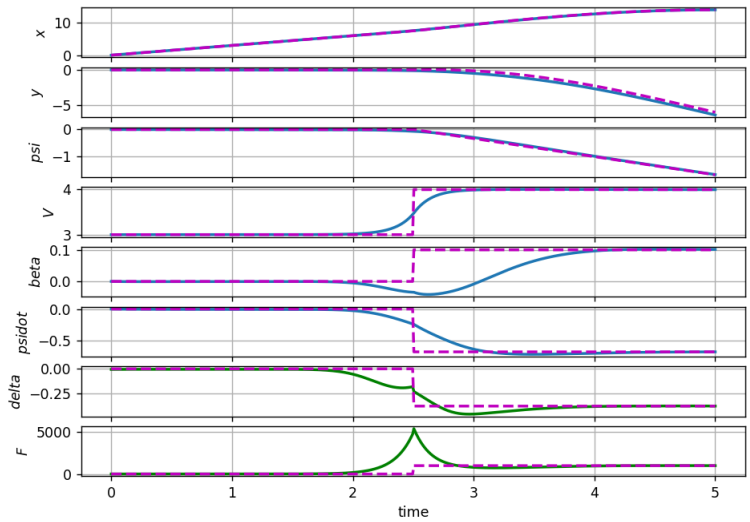

Figure 1.7: Trajectory obtained with Newton's method after 20 iterations

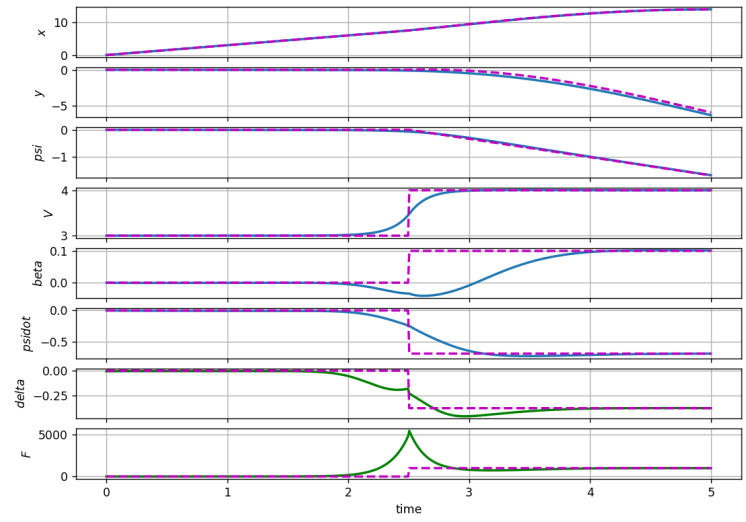

Figure 1.8: Final trajectory obtained with Newton's method (35 iterations)

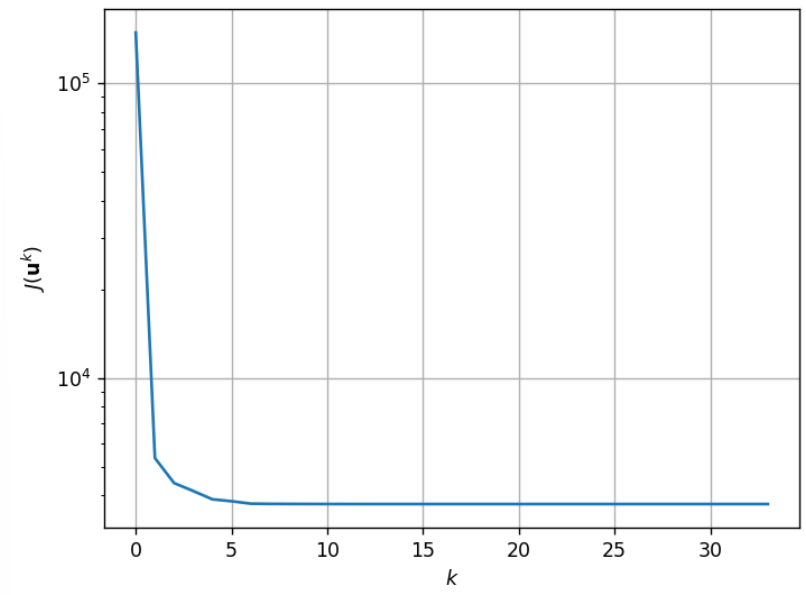

Figure 1.9: Cost

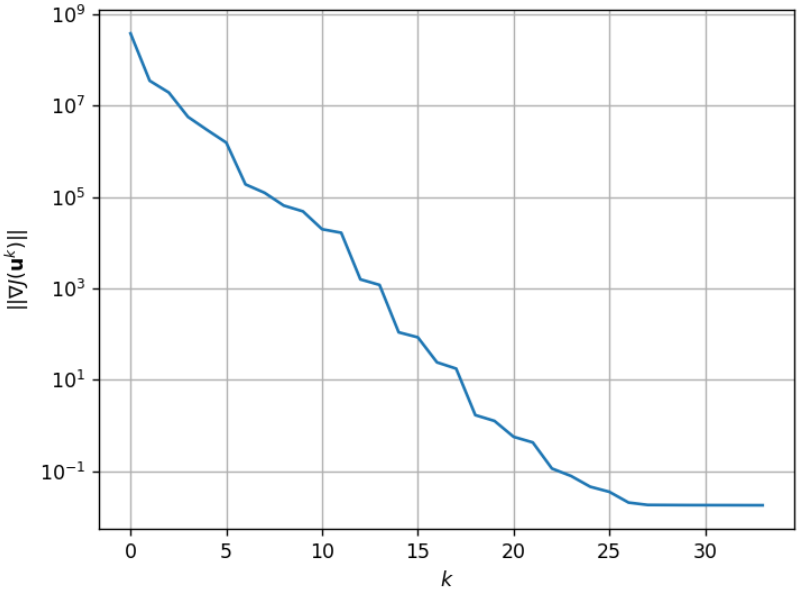

Figure 1.10: Descent direction

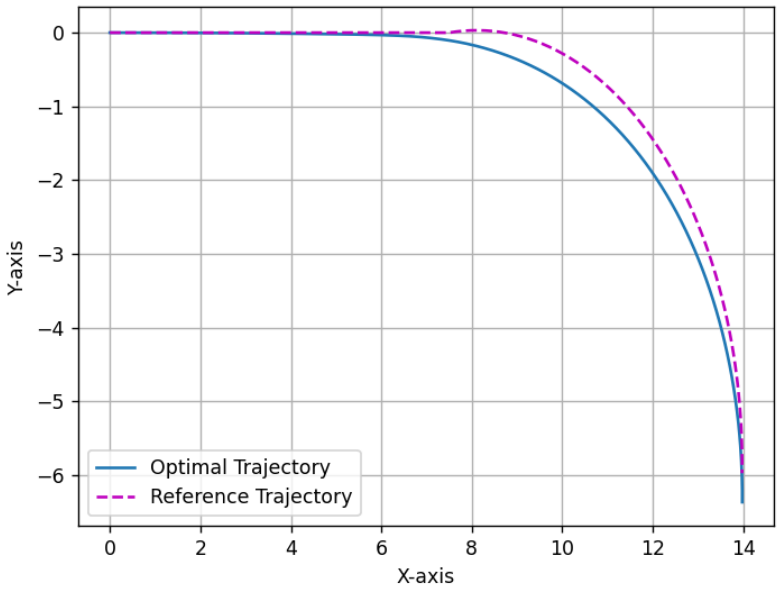

Figure 1.11: Obtained trajectory

Task 2

Trajectory generation (II)

2.1 Smoothing the trajectory reference

As next step we evaluated a "smoother" state-input curve on which performed again the Newton's method. The smoother curve has been obtained using the PCHIP spline, the new obtained reference is the one reported in figure 2.1.

2.2 Newton's method evaluation

Once we have created the "smoothed" trajectory reference we have perform the Newton's method algorithm evaluated for the previous task for this new reference trajectory.

From the following plots is possible to notice how the descent direction converge much faster, compared to the one before. So using the smoothed trajectory is possible to run the algorithm for a reduce number of iterations. Comparing the figure 2.9 with the figure 1.10 we can see how, in the case of a step reference the algorithm, it requires almost 30 iterations to converges, while with the smoothed trajectory, after 10 iterations it already converges. Despite that, we simulated in both cases for 35 iterations to better compare them.

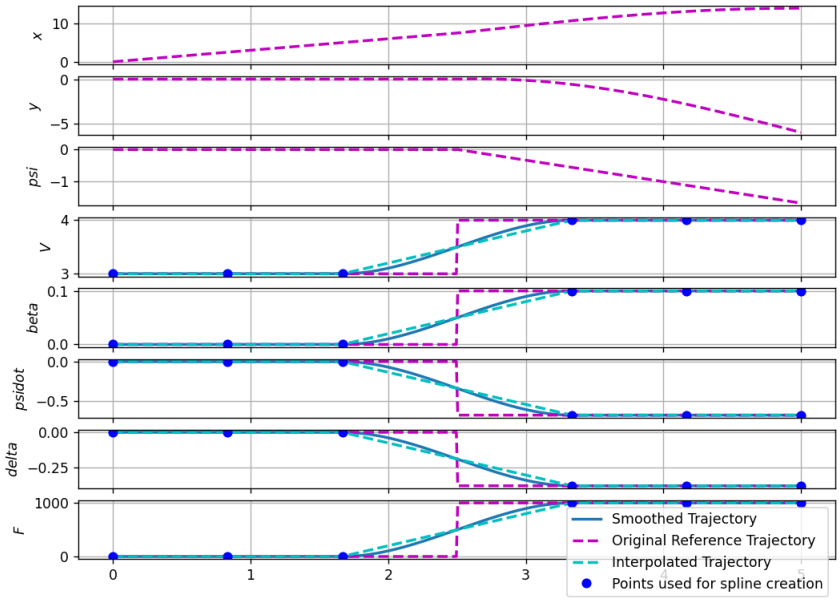

Figure 2.1: Smoothed Trajectory reference

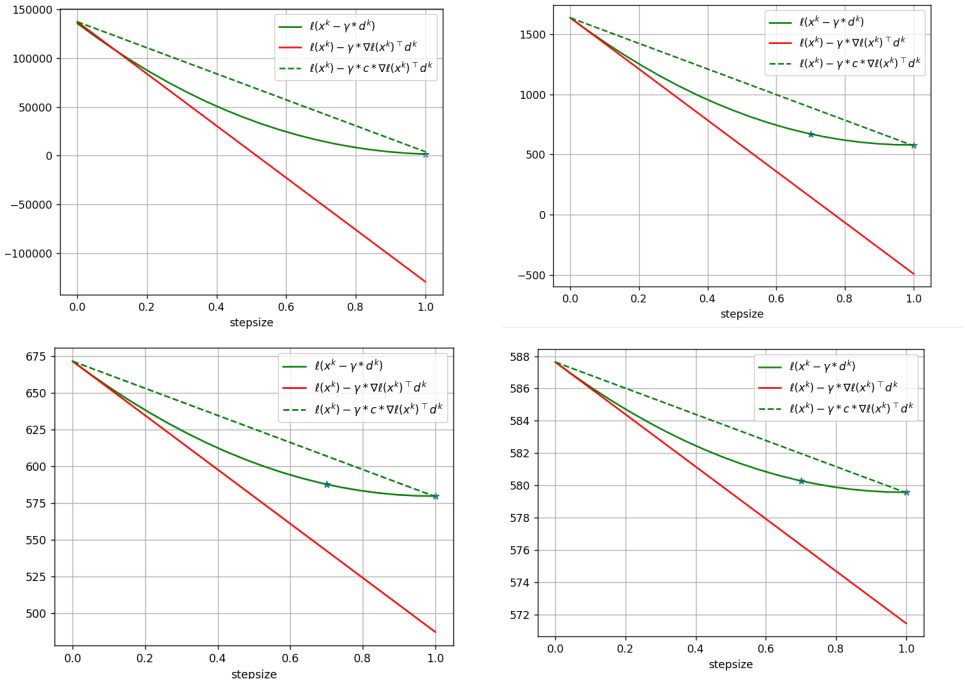

Figure 2.2: First 4 Armijo iterations

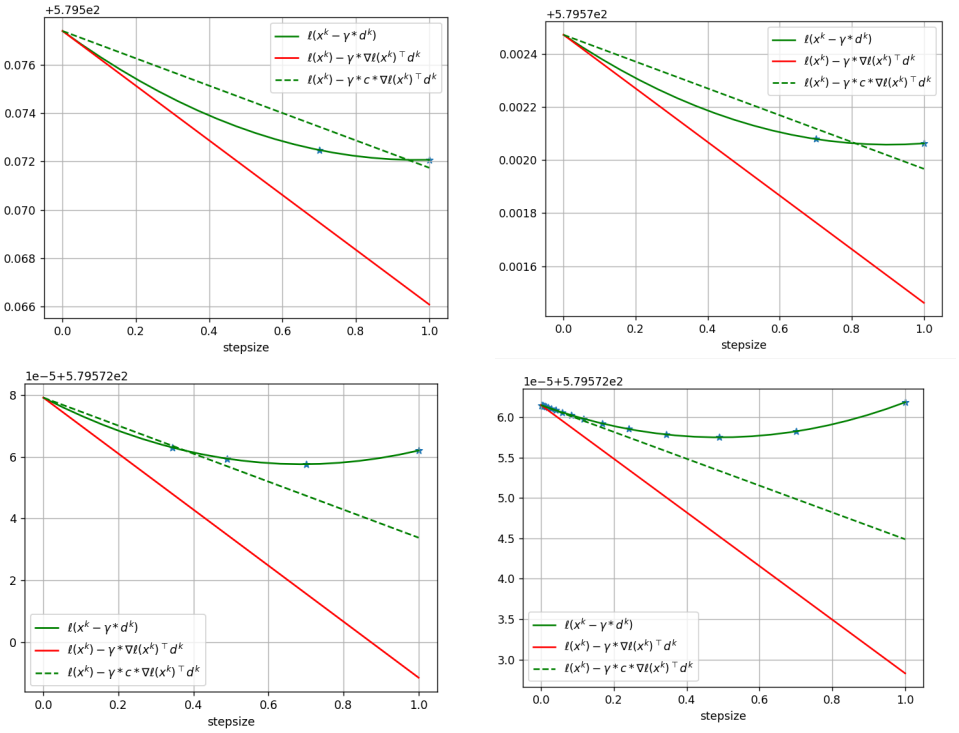

Figure 2.3: Last 4 Armijo iterations (before convergence)

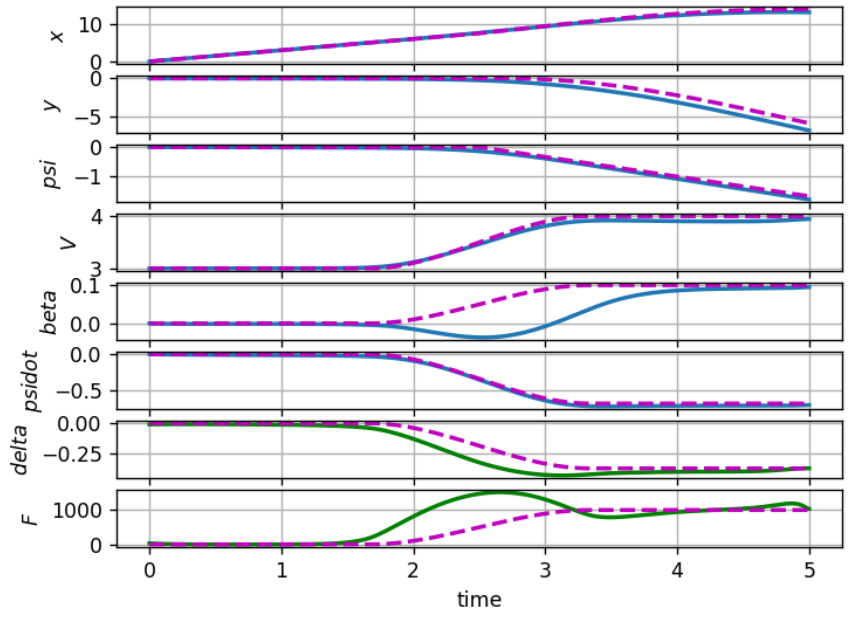

Figure 2.4: Trajectory obtained with Newton's method after 2 iterations

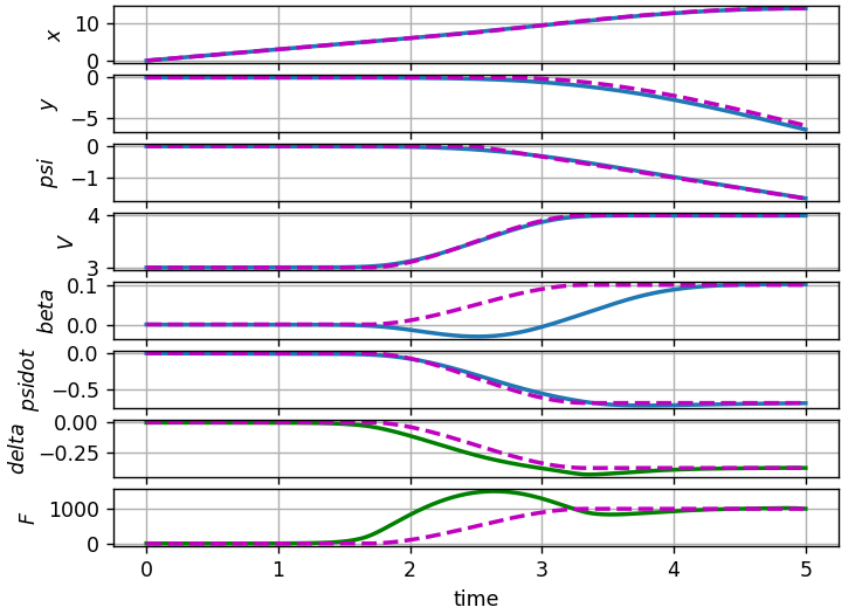

Figure 2.5: Trajectory obtained with Newton's method after 4 iterations

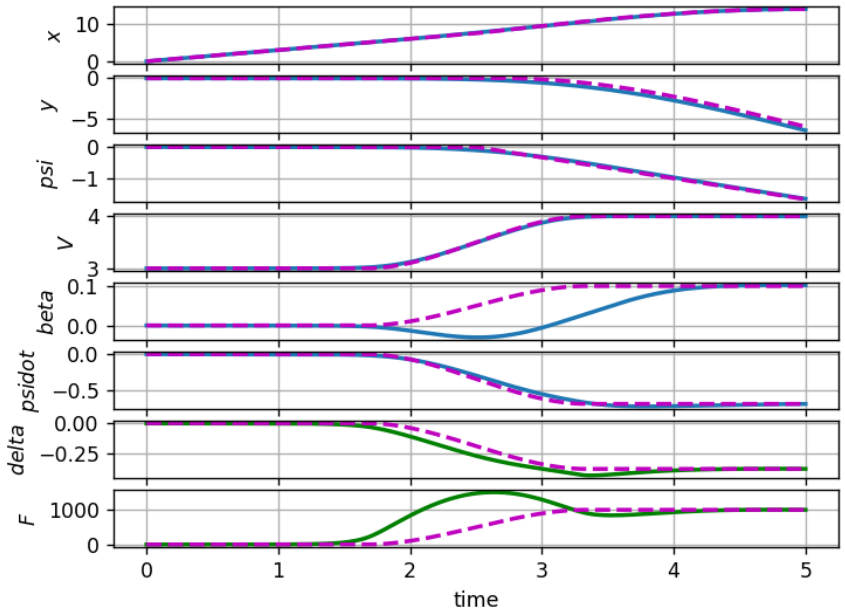

Figure 2.6: Trajectory obtained with Newton's method after 7 iterations

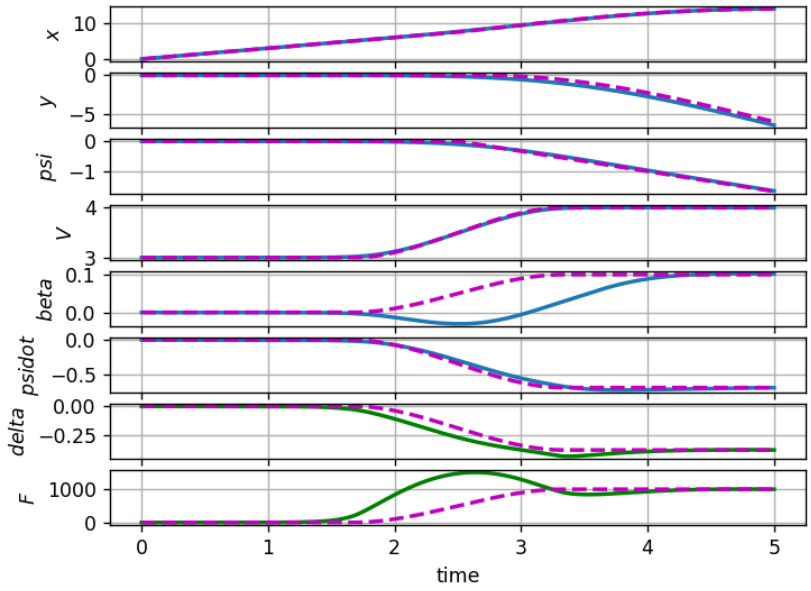

Figure 2.7: Final trajectory obtained with Newton's method (35 iterations)

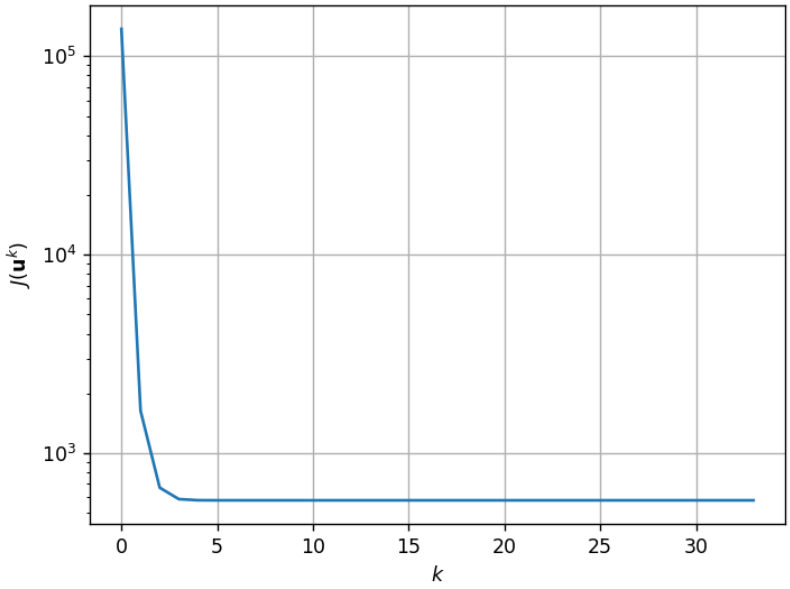

Figure 2.8: Cost

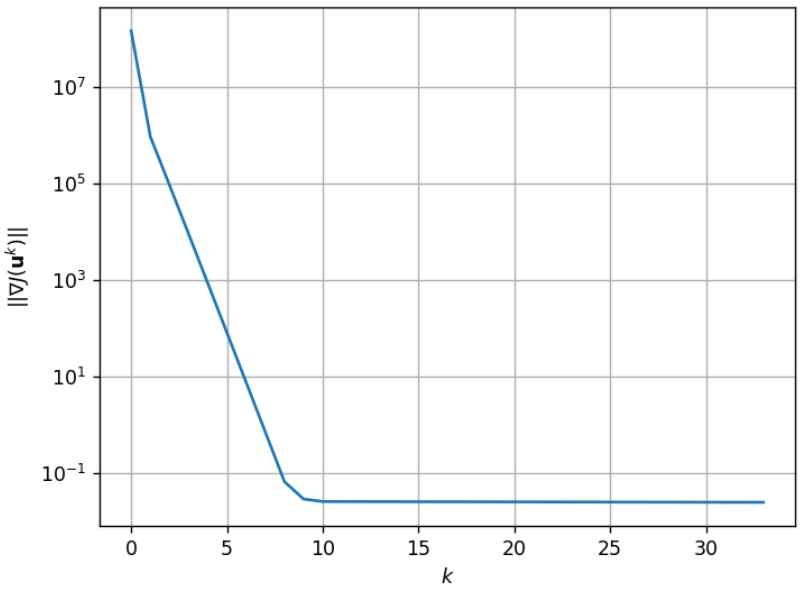

Figure 2.9: Descent direction

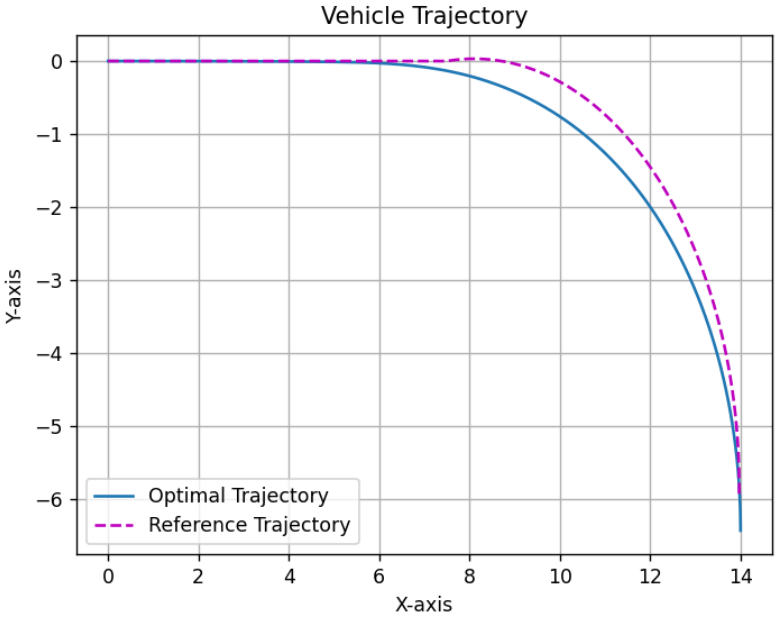

Figure 2.10: Obtained trajectory

Task 3

Task 3 – Trajectory tracking via LQR

For this task, the objective was to linearize the vehicle dynamics with respect to the (optimal) trajectory (x_{opt}; u_{opt}) determined in Task 2. We utilized the LQR algorithm to establish an optimal feedback controller that would enable the tracking of this reference trajectory. Specifically, we tackled the following Linear Quadratic (LQ) Problem:

$$\begin{aligned} \min_{\substack{\Delta x_1,\dots,\Delta x_T \\ \Delta u_0,\dots,\Delta u_{T-1}}} & \sum_{t=0}^{T-1} \left(\Delta x_t^\top Q^{reg} \Delta x_t + \Delta u_t^\top R^{reg} \Delta u_t \right) + \Delta x_T^\top Q^{reg} \Delta x_T \\ \text{subj. to} \quad & \Delta x_{t+1} = A_t^{opt} \Delta x_t + B_t^{opt} \Delta u_t \quad t = 0, \dots, T-1 \\ & x_0 = 0 \end{aligned}$$

Here, the matrices A_t^{opt} and B_t^{opt} represent the linearization of the system with respect to the optimal trajectory. Although the cost matrices for the regulator, denoted as Q^{reg} and R^{reg}, are arbitrary, we have chosen to maintain them consistent with those employed in previous tasks due to their demonstrated effectiveness.

In order to showcase the enhanced performance of this feedback-driven approach, we present simulation outcomes below, using a range of initial conditions. Notably, the algorithm consistently converges to the prescribed trajectory after a brief transition period. Additionally, it is worth noticing that we maintain uniform initial conditions for the vehicle, specifically for its position in terms of x, y, and ψ, ensuring that the vehicle starts from the same position in every simulation run.

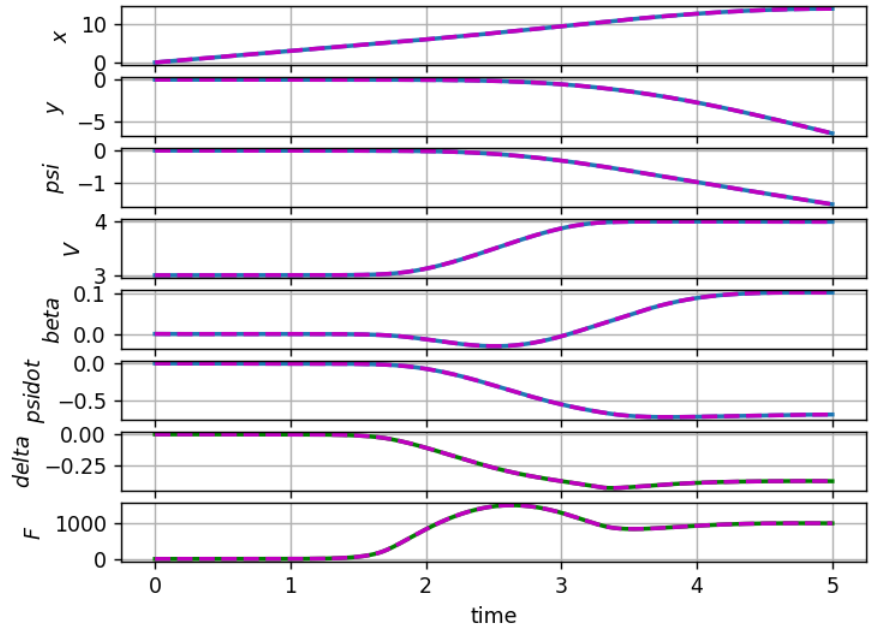

Figure 3.1: Trajectory tracking using LQR starting from the reference

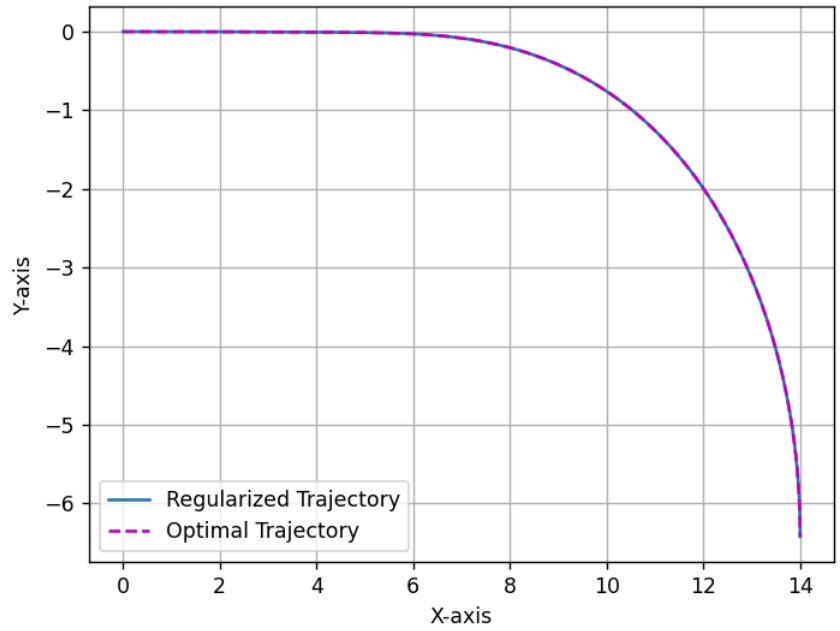

Figure 3.2: Obtained trajectory

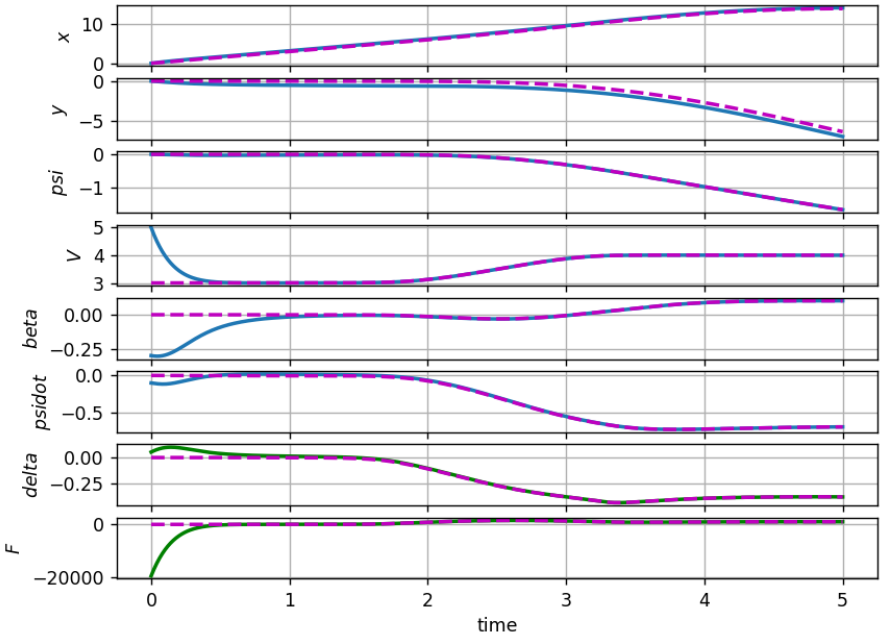

Figure 3.3: Trajectory tracking using LQR and perturbed initial conditions

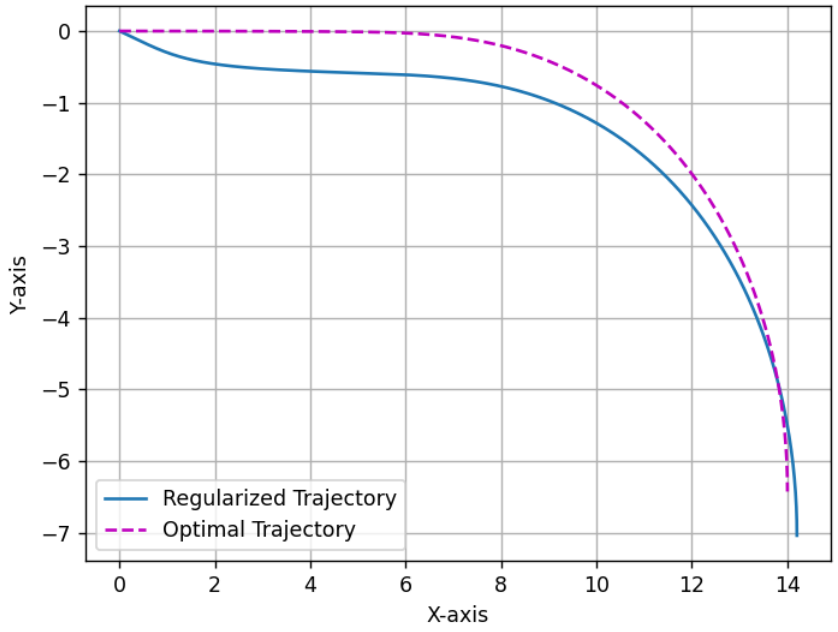

Figure 3.4: Obtained trajectory (perturbed initial conditon

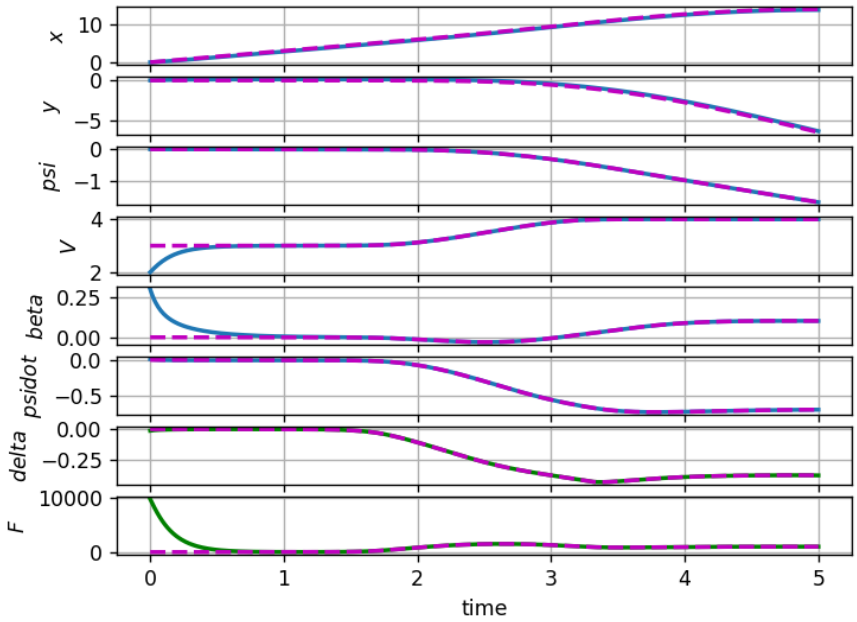

Figure 3.5: Trajectory tracking using LQR and perturbed initial condition

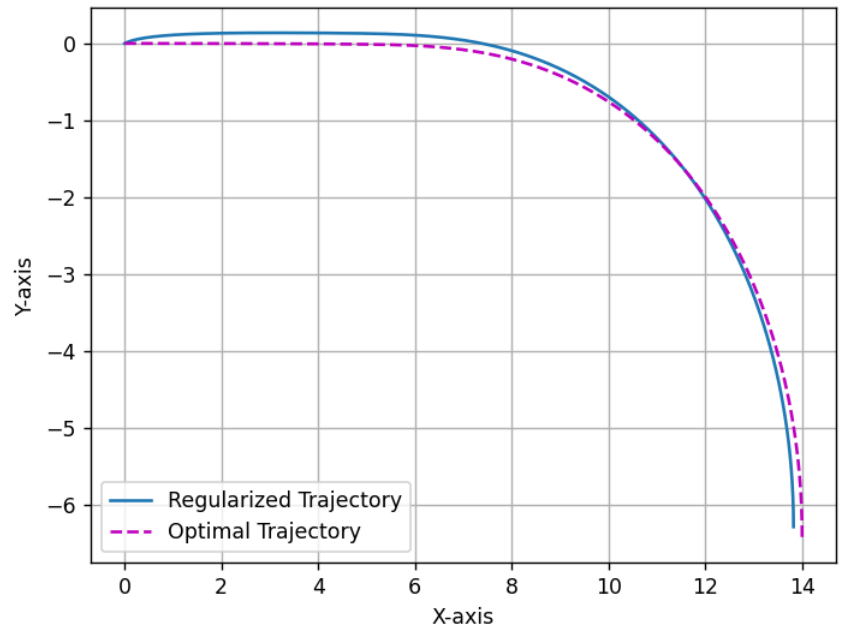

Figure 3.6: Obtained trajectory (perturbed initial condition)

Task 4

Trajectory tracking via MPC

We have finally implemented the Model Predictive Control (MPC) on the linearized vehicle dynamics with respect to the (optimal) trajectory (x_{opt}; u_{opt}) determined in Task 2. To demonstrate the tracking performance, we present various tests below, each with distinct initial conditions different from x_0^{star}.

Once more, we maintain the identical values for the matrices Q and R, as they have consistently yielded favorable results in our experiments. Following a series of tests, we've opted to employ a prediction horizon of 30 time instances for our Model Predictive Control (MPC). This choice strikes a balance between swift evaluation and effective performance.

In cases involving perturbations, we slightly extend the prediction horizon to mitigate oscillations in the steering input (δ).

Additionally, it's worth noting that the final segment of the trajectory exhibits some inaccuracies. This is because we conclude the estimation using predictions computed at the time step T_sim minus T_pred, which can lead to imprecision in the final trajectory segment.

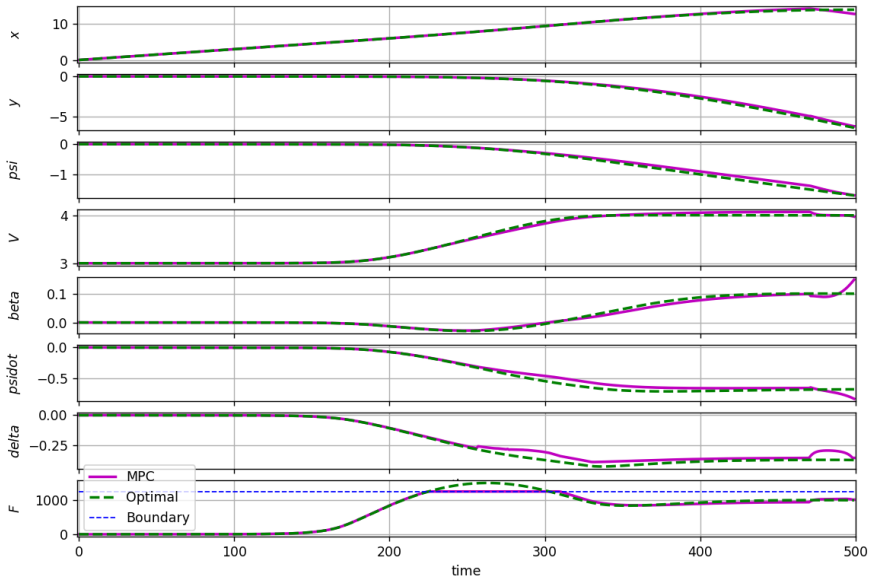

Figure 4.1: Trajectory tracking using MPC starting from the reference

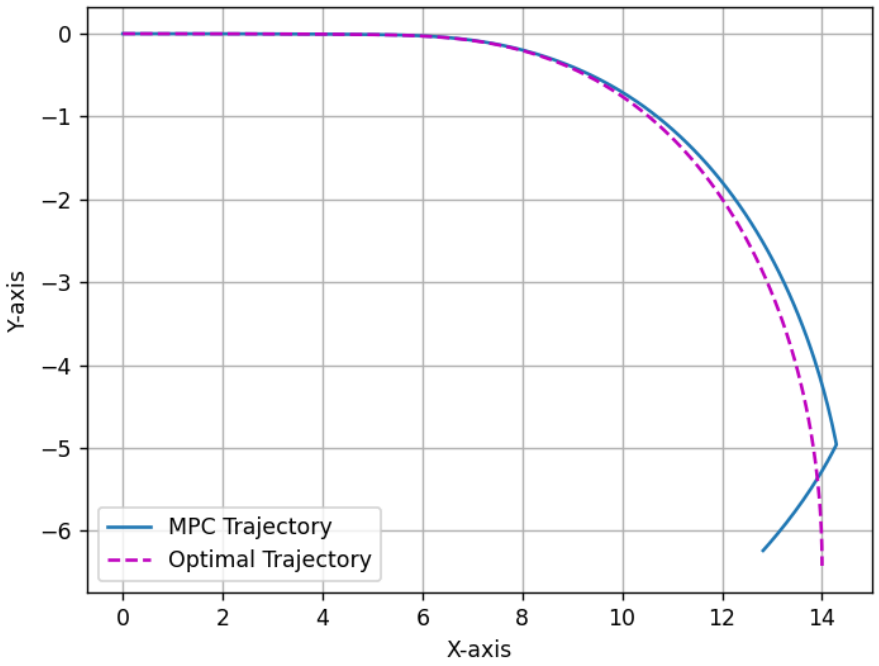

Figure 4.2: Obtained trajectory

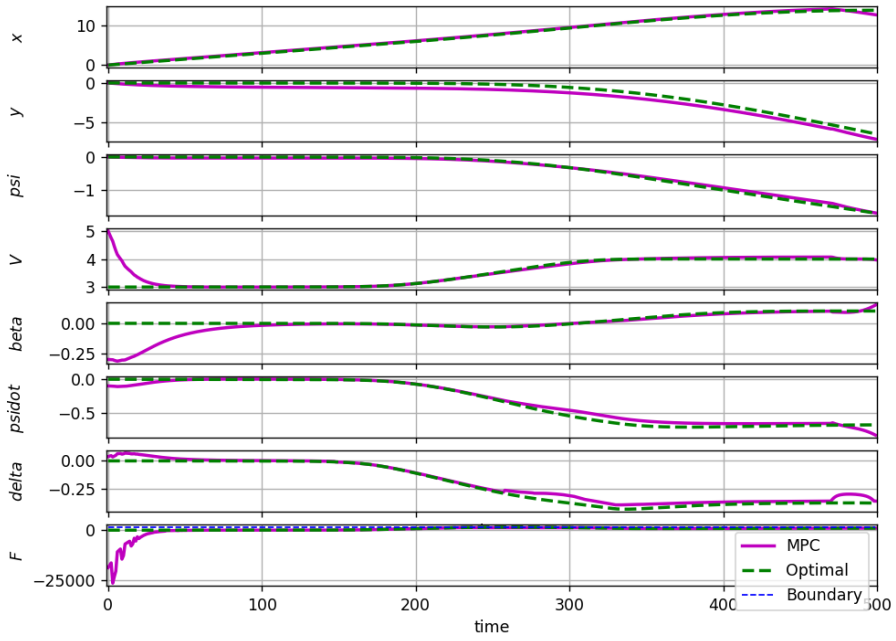

Figure 4.3: Trajectory tracking using MPC and perturbed initial conditions

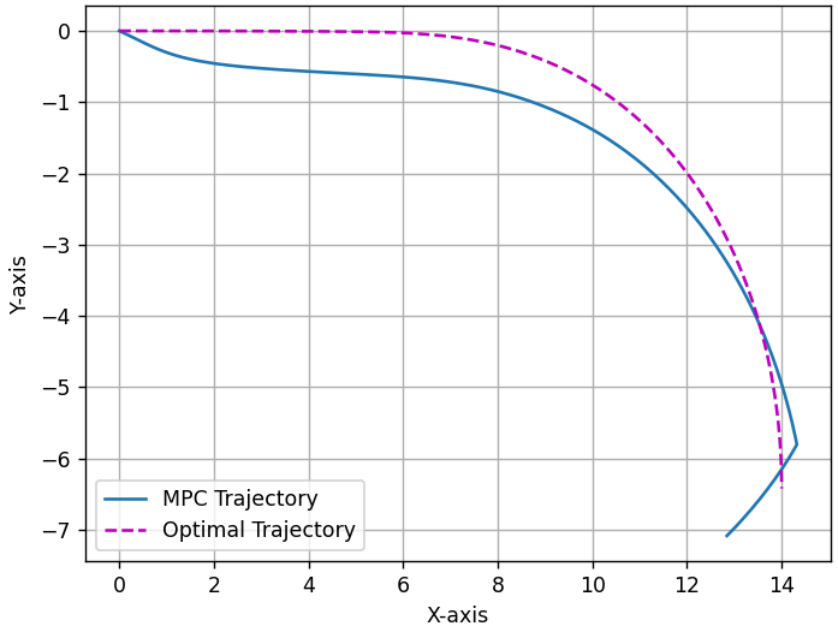

Figure 4.4: Obtained trajectory (perturbed initial condition

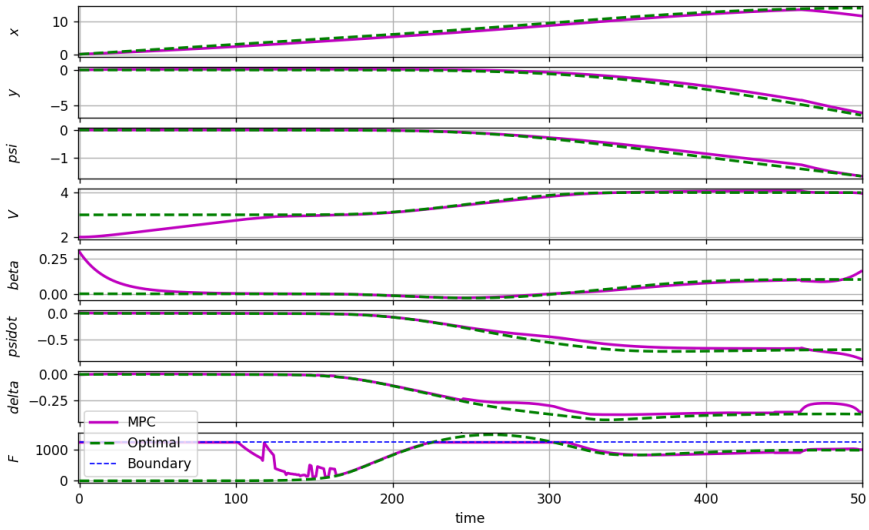

Figure 4.5: Trajectory tracking using MPC and perturbed initial condition

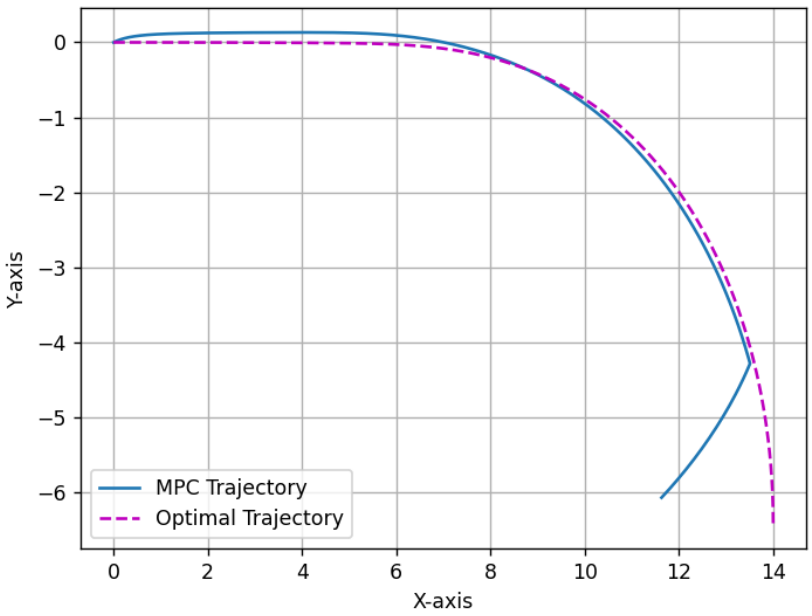

Figure 4.6: Obtained trajectory (perturbed initial condition)

Task 5

Animation

Finally we perform an animation of the trajectory performed by the car in task 3.

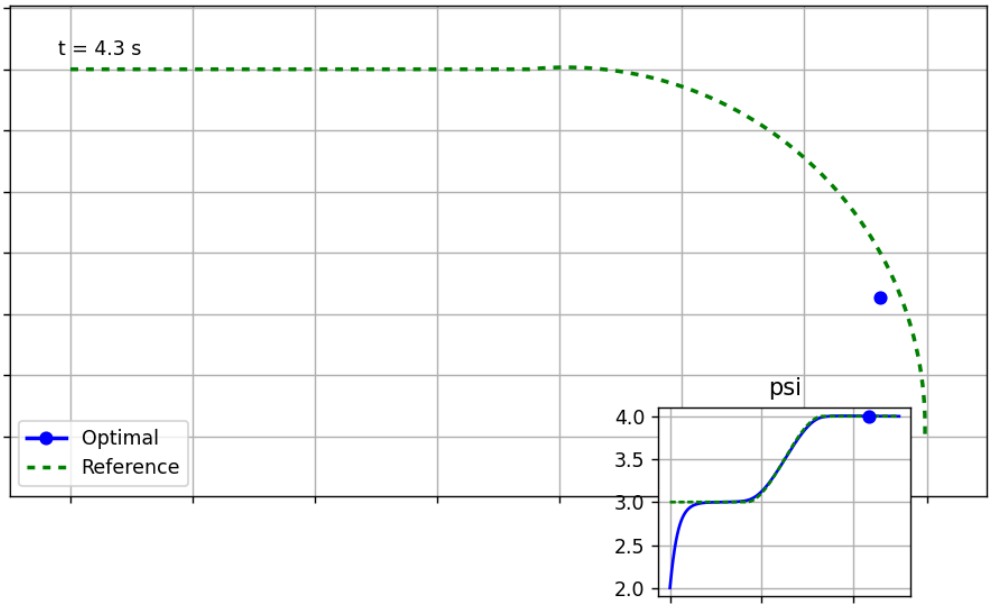

Figure 5.1: Animation

Conclusions

This project effectively managed the challenges of designing and implementing an optimal control strategy for vehicle navigation using a bicycle model. It involved various key components, such as applying Newton's method and Model Predictive Control (MPC), discretizing vehicle dynamics, establishing system equilibria, and creating a reference trajectory. Linear Quadratic Regulator (LQR) and MPC were employed for trajectory tracking, culminating in an animation demonstrating the vehicle's proficient handling of the LQR task, thereby validating our theoretical frameworks.

Furthermore, we observed how the provided reference trajectory significantly influences performance, highlighting the crucial role it plays. We also demonstrated the substantial performance improvement achievable through feedback tracking and explored alternative approaches, such as MPC, which also exhibited impressive performance.

This research emphasizes the importance of systematic and innovative approaches in tackling intricate control problems, laying the groundwork for future advancements in vehicle dynamics and control systems.