UNIVERSITÀ DI BOLOGNA

School of Engineering

Master Degree in Automation Engineering

Optimal Control

Development of Optimal Control algorithms for a single-track car model

Professor: **Giuseppe Notarstefano**

Students: Alessandro Cecconi,
Marco Bugo,
Lorenzo Frangiamone

Academic year 2021/2022

Abstract

This report deals with the application of Optimal Control algorithms to a dynamic bicycle model in different significant conditions. The optimization algorithm used is the Differential Dynamic Programming, which we have applied to compute optimal trajectories in an open-loop fashion, and then we have developed a Linear Quadratic Regulator to track them by based on the common feedback structure of control systems.
The project is divided in several steps. At the beginning we have re-written the system equations in state-space form, discretizing and linearizing them. After that, we first computed by means of a simple PI controller the initial state-input trajectory to initialize the algorithm, and then applying the DDP we have computed the optimal trajectory for a lane-change maneuver. Moreover, with the same algorithm, we computed the optimal trajectory considering our car moving on a skidpad track. At the end, thanks to the LQR we have computed the feedback controller and tested its capabilities to force the system to follow the optimal trajectory, adding some noise on the initial conditions and during the motion. We have also provided a simple animation where we represent all the significant results. The whole process was not problem-free and we present some of them with our solutions in the next pages.

Contents

List of Figures

Chapter 1

Dynamics Model Definition - Task 0

In order to match the notation and to implement our Optimal Control methods, we have to rewrite the model in state-space form and discretize it. We started from the following set of equations

$$\begin{aligned} \dot{x} &= v_x \cos\psi - v_y \sin\psi \\ \dot{y} &= v_x \sin\psi + v_y \cos\psi \\ m(\dot{v}_x - \dot{\psi} v_y) &= F_x \cos\delta - F_{y,f} \sin\delta + F_{y,r} \\ m(\dot{v}_y + \dot{\psi} v_x) &= F_x \sin\delta + F_{y,f} \cos\delta + F_{y,r} \\ I_z \ddot{\psi} &= (F_x \sin\delta + F_{y,f} \cos\delta) a - F_{y,r} b \end{aligned} \tag{1.1}$$

which represent a dynamic bicycle model with static load transfers and lateral forces acting on the wheels. The state vector is $\mathbf{x} = [x\ y\ \psi\ v_x\ v_y\ \dot{\psi}]^T$ and the input vector is $\mathbf{u} = [\delta\ F_x]^T$. The first three components of the state $[x\ y\ \psi]$ represent the pose of the vehicle in the global reference frame, instead the other ones $[v_x\ v_y\ \dot{\psi}]$ are the velocities and yaw rate of the vehicle in its fixed reference frame. If we define the state vector and the input vector as

$$\begin{aligned} \mathbf{x} = \begin{bmatrix} x\ y\ \psi\ v_x\ v_y\ \dot{\psi} \end{bmatrix}^T := \begin{bmatrix} x_1\ x_2\ x_3\ x_4\ x_5\ x_6 \end{bmatrix}^T \\ \mathbf{u} = \begin{bmatrix} \delta\ F_x \end{bmatrix}^T := \begin{bmatrix} u_1\ u_2 \end{bmatrix}^T \end{aligned}$$

we can rewrite the system in state-space form as follows

$$\begin{aligned} \dot{x}_1 &= x_4 \cos(x_3) - x_5 \sin(x_3) \\ \dot{x}_2 &= x_4 \sin(x_3) + x_5 \cos(x_3) \\ \dot{x}_3 &= x_6 \\ \dot{x}_4 &= \tfrac{1}{m}\big(u_2 \cos(u_1) - F_{y,f} \sin(u_1) + m x_6 x_5\big) \\ \dot{x}_5 &= \tfrac{1}{m}\big(u_2 \sin(u_1) + F_{y,f} \cos(u_1) + F_{y,r} - m x_6 x_4\big) \\ \dot{x}_6 &= \tfrac{1}{I_z}\big(u_2 \sin(u_1) + F_{y,f} \cos(u_1)\big) a - \tfrac{1}{I_z} F_{y,r} b \end{aligned} \tag{1.2}$$

The lateral forces on the wheels are strongly related to the slip angles which are here introduced and written with the new notation

$$\begin{aligned} F_{y,f} &= \mu F_{z,f} \beta_f\ , \quad \beta_f = \delta - \frac{v_y + a\dot{\psi}}{v_x} = u_1 - \frac{x_5 + a x_6}{x_4} \\ F_{y,r} &= \mu F_{z,r} \beta_r\ , \qquad \beta_r = -\frac{v_y - b\dot{\psi}}{v_x} = -\frac{x_5 - b x_6}{x_4} \end{aligned} \tag{1.3}$$

where β_f and β_r are respectively the front and the rear slip angles. Instead, $F_{z,r}$ and $F_{z,f}$ represent the static load transfer which are defined as

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

Once we have defined the state-space form of our model we have to discretize it, because we are looking to use Discrete-Time Optimal Control strategies to control it. In order to do that, we use a forward Euler method to approximate the derivative as the difference between two consecutive states dived by the sampling time, i.e.

$$\dot{x} \approx \frac{x_{t+1} - x_t}{\delta}$$

By applying it, the discretized model will be

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \delta \begin{bmatrix} x_{4,t}\cos(x_{3,t}) - x_{5,t}\sin(x_{3,t}) \\ x_{4,t}\sin(x_{3,t}) + x_{5,t}\cos(x_{3,t}) \\ x_{6,t} \\ \frac{1}{m}\big(u_{2,t}\cos(u_{1,t}) - F_{y,f}\sin(u_{1,t}) + m x_{6,t}x_{5,t}\big) \\ \frac{1}{m}\big(u_{2,t}\sin(u_{1,t}) + F_{y,f}\cos(u_{1,t}) + F_{y,r} - m x_{6,t}x_{4,t}\big) \\ \frac{1}{I_z}\big(u_{2,t}\sin(u_{1,t}) + F_{y,f}\cos(u_{1,t})\big)a - \frac{1}{I_z}F_{y,r}b \end{bmatrix} \tag{1.4}$$

Where

- $\mathbf{x}_{t+1}$ represents the state vector at the next time instant;
- $\mathbf{x}_t$ represents the state vector at the current time instant;
- $\big(x_{\bullet,t}, u_{\bullet,t}\big)$ remarks the fact that we are evaluating a specific state variable or input at time instant t;
- δ represents the sampling time (not the steering angle).

It is clear visible that both the continuous-time (1.2) and the discrete-time models (1.4) are highly nonlinear. To overcome this issue, we will use methods that are able to iteratively linearize it to rely on linear optimal control methods. Before computing it, let's rewrite in a compact form the discrete-time model in 1.4 as

$$\mathbf{x}_{t+1} = \mathbf{f}\big(\mathbf{x}_t, \mathbf{u}_t\big) \tag{1.5}$$

where

$$\mathbf{f} := \begin{bmatrix} f_1(.) & f_2(.) & f_3(.) & f_4(.) & f_5(.) & f_6(.) \end{bmatrix}^T$$

Just to underline the fact that we are dealing with a nonlinear system which in our particular case is time-invariant.
Now, we can introduce the linearization of the model by using tools studied in System Theory courses and match it with the optimization framework. In particular, we can linearize a system by means of the *Jacobian Matrices*, defined as

$$A = \left.\frac{\partial \mathbf{f}(.)}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\bar{\mathbf{x}} \\ \mathbf{u}=\bar{\mathbf{u}}}} \quad \textit{and} \quad B = \left.\frac{\partial \mathbf{f}(.)}{\partial \mathbf{u}}\right|_{\substack{\mathbf{x}=\bar{\mathbf{x}} \\ \mathbf{u}=\bar{\mathbf{u}}}}$$

where $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ is generally an equilibrium pair. However, to be consistent with optimization theory, upon which optimal control methods are based, we will make use of the *gradients*. We can rewrite A and B as

$$A_t = \left.\frac{\partial \mathbf{f}(.)}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\mathbf{x}_t \\ \mathbf{u}=\mathbf{u}_t}} = \nabla_{x_t}\mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)^T$$

$$B_t = \left.\frac{\partial \mathbf{f}(.)}{\partial \mathbf{u}}\right|_{\substack{\mathbf{x}=\mathbf{x}_t \\ \mathbf{u}=\mathbf{u}_t}} = \nabla_{u_t}\mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)^T$$

We have previously considered $(\mathbf{x}, \mathbf{u})$ as an equilibrium pair, but we have to introduce the more general concept of *trajectory* that we are going to use throughout the project. A state-input curve $\bar{x}_t$ for $t = 0, \dots, T$ and $\bar{u}_t$ for $t = 0, \dots, T-1$ (i.e. $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$) and $\bar{x}_0 = x_0$ is said to be a *trajectory* of the system if it satisfies the dynamics as in 1.5. An equilibrium pair is a particular trajectory of the system: if I start there, I will remain there forever.
After this long introduction, let's compute the linearization of our model. First of all, since the lateral forces depend from the state because of the slip angles, we first compute their derivatives and then we put all together in the gradients of the dynamics. So, starting from the equation in 1.3 the partial derivatives of $F_{y,f}$ are

$$\frac{\partial}{\partial x_4} F_{y,f} = F_{z,f}\left(\frac{x_5 + a x_6}{x_4^2}\right), \quad \frac{\partial}{\partial x_5} F_{y,f} = -\frac{F_{z,f}}{x_5},$$
$$\frac{\partial}{\partial x_6} F_{y,f} = -\frac{a F_{z,f}}{x_4}, \qquad \frac{\partial}{\partial u_1} F_{y,f} = F_{z,f}$$

And for $F_{y,r}$ we will have

$$\frac{\partial}{\partial x_4} F_{y,r} = F_{z,r}\left(\frac{x_5 - b x_6}{x_4^2}\right), \quad \frac{\partial}{\partial x_5} F_{y,r} = -\frac{F_{z,r}}{x_4}, \quad \frac{\partial}{\partial x_6} F_{y,r} = \frac{b F_{z,r}}{x_4}$$

Now, we can compute the two linearized matrices of our system. First, the dynamical matrix A:

$$A = \begin{bmatrix} 1 & 0 & -\delta(x_4 \sin x_3 + x_5 \cos x_3) & \delta \cos x_3 & -\delta \sin x_3 & 0 \\ 0 & 1 & \delta(x_4 \cos x_3 - x_5 \sin x_3) & \delta \sin x_3 & \delta \cos x_3 & 0 \\ 0 & 0 & 1 & 0 & 0 & \delta \\ 0 & 0 & 0 & \frac{\partial f_4}{\partial x_4} & \frac{\partial f_4}{\partial x_5} & \frac{\partial f_4}{\partial x_6} \\ 0 & 0 & 0 & \frac{\partial f_5}{\partial x_4} & \frac{\partial f_5}{\partial x_5} & \frac{\partial f_5}{\partial x_6} \\ 0 & 0 & 0 & \frac{\partial f_6}{\partial x_4} & \frac{\partial f_6}{\partial x_5} & \frac{\partial f_6}{\partial x_6} \end{bmatrix}$$

where the symbolic terms are

$$\frac{\partial f_4}{\partial x_4} = 1 - \frac{\delta}{m}(\sin u_1 \frac{\partial F_{y,f}}{\partial x_4}) \qquad \frac{\partial f_4}{\partial x_5} = -\frac{\delta}{m}(\sin u_1 \frac{\partial F_{y,f}}{\partial x_5} - m x_6)$$
$$\frac{\partial f_4}{\partial x_6} = -\frac{\delta}{m}(\sin u_1 \frac{\partial F_{y,f}}{\partial x_6} - m x_5) \qquad \frac{\partial f_5}{\partial x_4} = \frac{\delta}{m}(\cos u_1 \frac{\partial F_{y,f}}{\partial x_4} + \frac{\partial F_{y,r}}{\partial x_4} - m x_6)$$
$$\frac{\partial f_5}{\partial x_5} = 1 + \frac{\delta}{m}(\cos u_1 \frac{\partial F_{y,f}}{\partial x_5} + \frac{\partial F_{y,r}}{\partial x_5}) \qquad \frac{\partial f_5}{\partial x_6} = \frac{\delta}{m}(\cos u_1 \frac{\partial F_{y,f}}{\partial x_6} + \frac{\partial F_{y,r}}{\partial x_6} - m x_4)$$
$$\frac{\partial f_6}{\partial x_4} = \frac{\delta}{I_z}(\frac{\partial F_{y,f}}{\partial x_4} a \cos u_1 - \frac{\partial F_{y,r}}{\partial x_4} b) \qquad \frac{\partial f_6}{\partial x_5} = \frac{\delta}{I_z}(\frac{\partial F_{y,f}}{\partial x_5} a \cos u_1 - \frac{\partial F_{y,r}}{\partial x_5} b)$$
$$\frac{\partial f_6}{\partial x_6} = 1 + \frac{\delta}{I_z}(\frac{\partial F_{y,f}}{\partial x_6} a \cos u_1 - \frac{\partial F_{y,r}}{\partial x_6} b)$$

And then the input matrix B

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -\frac{\delta}{m}(u_2 \sin u_1 + \frac{\partial F_{y,f}}{\partial u_1} \sin u_1 + F_{y,f} \cos u_1) & \frac{\delta}{m} \cos u_1 \\ \frac{\delta}{m}(u_2 \cos u_1 + \frac{\partial F_{y,f}}{\partial u_1} \cos u_1 - F_{y,f} \sin u_1) & \frac{\delta}{m} \sin u_1 \\ \frac{\delta}{I_z}(u_2 \cos u_1 + \frac{\partial F_{y,f}}{\partial u_1} \cos u_1 - F_{y,f} \sin u_1) a & \frac{\delta}{I_z} a \sin u_1 \end{bmatrix}$$

The different Hessians are explicitly written in the code and for sake of compactness we decide to not write them in the report. We have computed all the derivatives by hand because we encountered several errors by using the MATLAB symbolic toolbox, probably caused by the system dynamics complexity. To ensure that everything works fine, we have checked how the dynamics works by using a simple script where we imposed constant steering angle and longitudinal force and plotted the different variables.

Chapter 2

Lane Change - Task 1

The second task we were asked to perform is a lane change maneuver, by defining the optimal state-input trajectory using a DDP algorithm with the only condition of a zero heading angle at the beginning and at the end of it. In the scheme above 2 we briefly recap what is the algorithm implemented in our code. The main issue encountered has been properly initialized

Algorithm 1 DDP

Initialize using a system trajectory $(\mathbf{x}^0, \mathbf{u}^0)$;
for k=0,1,2,... **do**
  Set $p_T^k = \nabla_x l_T$ and $P_T^k = \nabla_{xx} l_T$
  for t=T-1,...,1 **do**
    Compute:
    $K_t^k = -(l_{uu,t}^k + f_{u,t}^k P_{t+1}^k {f_{u,t}^k}^T + f_{uu,t}^k p_{t+1}^k)^{-1}(l_{ux,t}^k + f_{u,t}^k P_{t+1}^k {f_{x,t}^k}^T + f_{ux,t}^k p_{t+1}^k)$
    $\sigma_t^k = -(l_{uu,t}^k + f_{u,t}^k P_{t+1}^k {f_{u,t}^k}^T + f_{uu,t}^k p_{t+1}^k)^{-1}(l_{u,t}^k + f_{u,t}^k p_{t+1}^k)$
    $P_t^k = (l_{xx,t}^k + f_{x,t}^k P_{t+1}^k {f_{x,t}^k}^T + f_{xx,t}^k p_{t+1}^k) - {K_t^k}^T (l_{uu,t}^k + f_{u,t}^k P_{t+1}^k {f_{u,t}^k}^T + f_{uu,t}^k p_{t+1}^k) K_t^k$
    $p_t^k = (l_{x,t}^k + f_{x,t}^k p_{t+1}^k) - {K_t^k}^T (l_{uu,t}^k + f_{u,t}^k P_{t+1}^k {f_{t+1}^k}^T + f_{uu,t}^k p_{t+1}^k) \sigma_t^k$
  end for
  Forward Simulation:
  for t=0,1,...,T-1 **do**
    Compute:
    $u_t^{k+1} = u_t^k + K_t^k (x^{k+1} - x_t^k) + \sigma_t^k$
    $x_{t+1}^{k+1} = f(x_t^{k+1}, u_t^{k+1})$, $x_0^{k+1} = x_{init}$
  end for
end for

the system dynamics to make possible the algorithm convergence. In fact, we were unable to analytically find a solution which satisfies the system equations because of all the nested dependencies. In particular, this was caused by the longitudinal velocity v_x, which being at the denominator in slip angles equations 1.3 causes several numerical errors. Without a proper initialization, we faced lots of singularities during the computation of σ_t and K_t, making impossible for us to simulate more than few seconds the movement of our car.
Then, to find a feasible system trajectory, we decide to develop a simple PI controller using as a reference a trapezoidal curve pretty similar to the lane change that we want to perform in the DDP. We decide to apply the PI controller on the longitudinal force to minimize the position error with respect to the reference line, while we develop a geometric approach for what concern the steering angle. We show first the computation of the error distance and the

PI control,

$$\begin{aligned} e_d(t) &= \sqrt{(x_{\text{ref}} - x_{\text{car}})^2 + (y_{\text{ref}} - y_{\text{car}})^2} \\ u_2(t) &= (K_p + K_i \tfrac{T_s}{2}) e_d(t) + (-K_p + K_i \tfrac{T_s}{2}) e_d(t-1) \end{aligned} \tag{2.1}$$

Where K_p and K_i are respectively the proportional and the integral gain, which has been chosen by following the rule: $K_i = 0.45 K_p$.
The approximated steering angle equation is defined as follows, taking into account the sketch in figure 2.1, we first define the geometric pointer as

$$\begin{aligned} \frac{y_{\text{d}} - y_{\text{c}}}{x_{\text{d}} - x_{\text{c}}} &= \tan(\delta + \psi) \\ \delta = -\psi + \arctan\left(\frac{y_{\text{d}} - y_{\text{c}}}{x_{\text{d}} - x_{\text{car}}}\right) &= -\psi + \gamma \end{aligned} \tag{2.2}$$

To limit the oscillating behaviour caused by the pointer when the car overcomes the reference, we have introduced a weight $w \in [0, 1]$ that balance the contribution of the pointer with the reference angle ψ_d. At the end the steering angle becomes,

$$\delta = -\psi + (1 - w)\gamma + w\psi_d \tag{2.3}$$

The result obtained by this simple control is shown the figure 2.2.

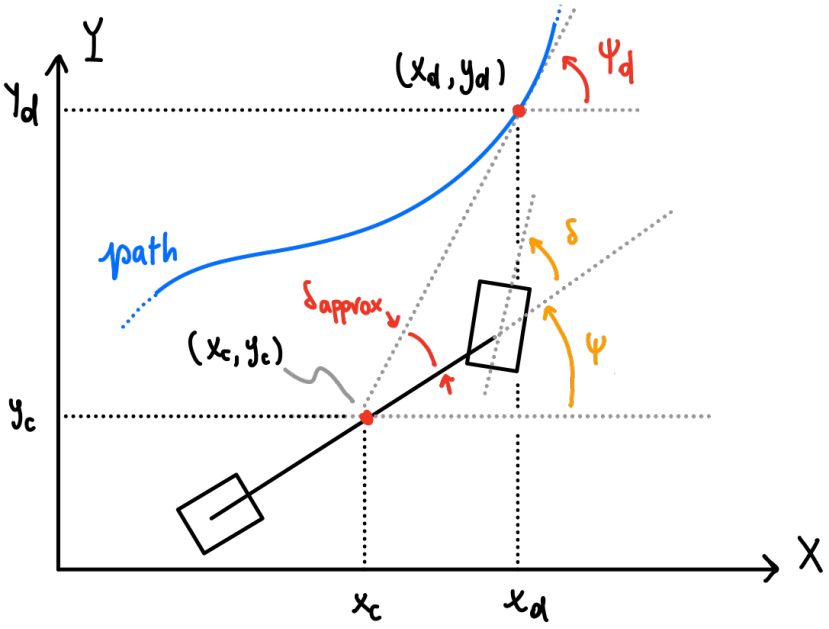


Figure 2.1: Definition of steering angle

Thanks to that, we have found an initial trajectory that satisfy the dynamics, and we have used it as initialization of the DDP algorithm. It was crucial to properly initialize the algorithm, otherwise it was impossible to complete even a single iteration to compare the results between different weights chosen. This point out the huge importance of the initial conditions to the convergence of this optimal control algorithm, which seems to be really sensitive to the starting point applied to it. At the end the algorithm works fine, both with fixed gamma and with Armijo's rule, they give in fact similar results. In particular the fixed gamma needs a lower number of iterations to converge. The difference between the two costs is visible in figure 2.3. All the results we have obtained are presented in figures 2.4, 2.5, 2.6, and 2.7.

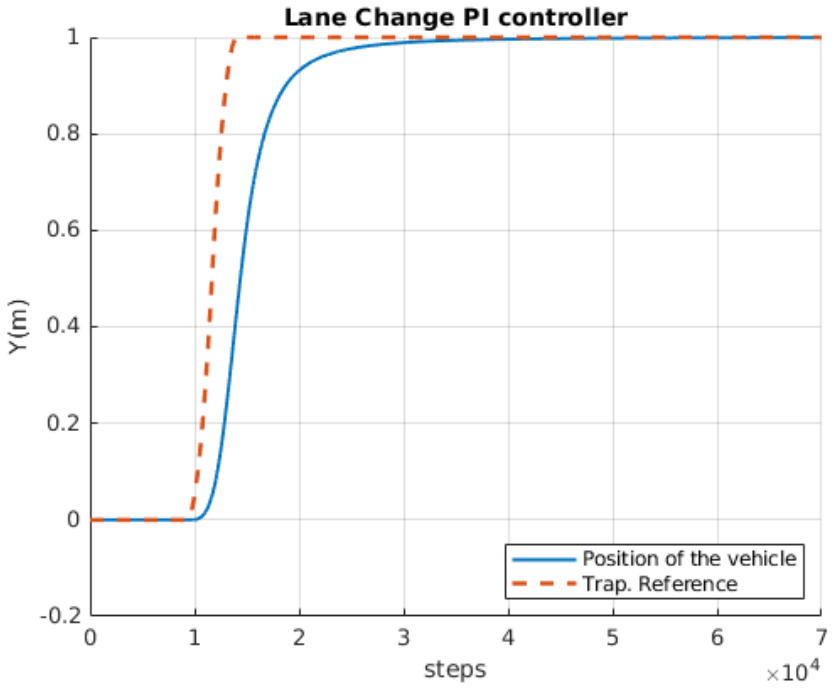


Figure 2.2: Lane Change Maneuver by using PI controller

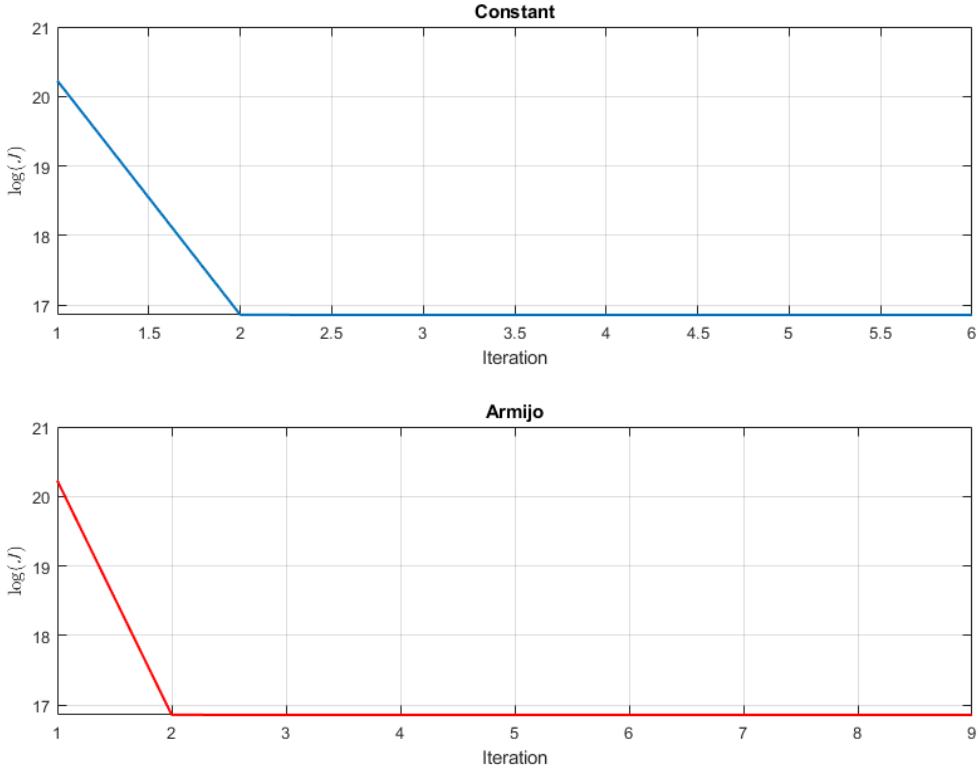


Figure 2.3: Fixed gamma vs Armijo's rule - Lane Change Maneuver

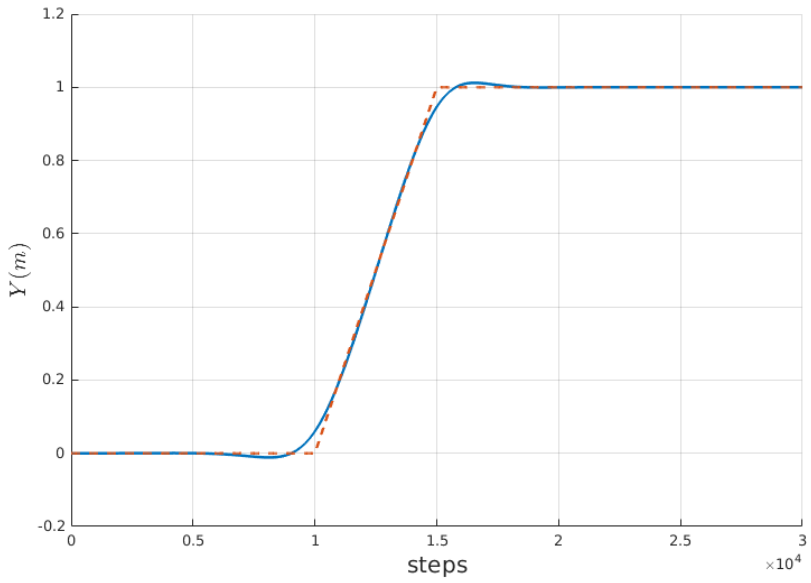


Figure 2.4: $Y(m)$ - Lane Change Maneuver

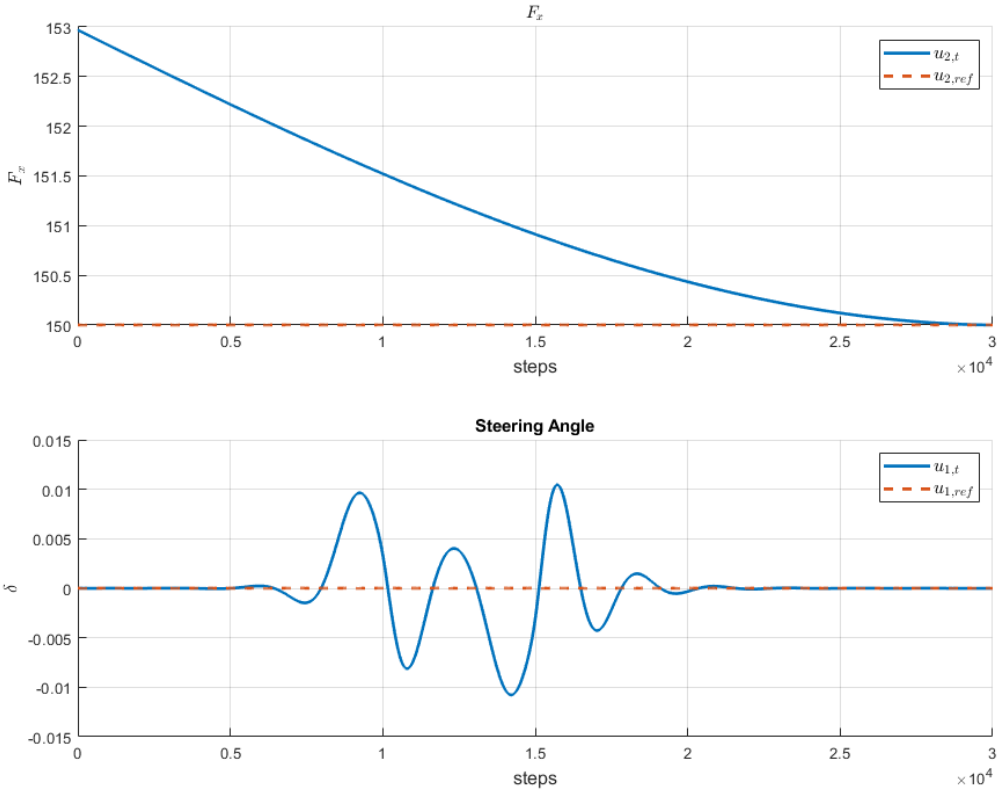


Figure 2.5: Longitudinal Force and Steering Angle - Lane Change Maneuver

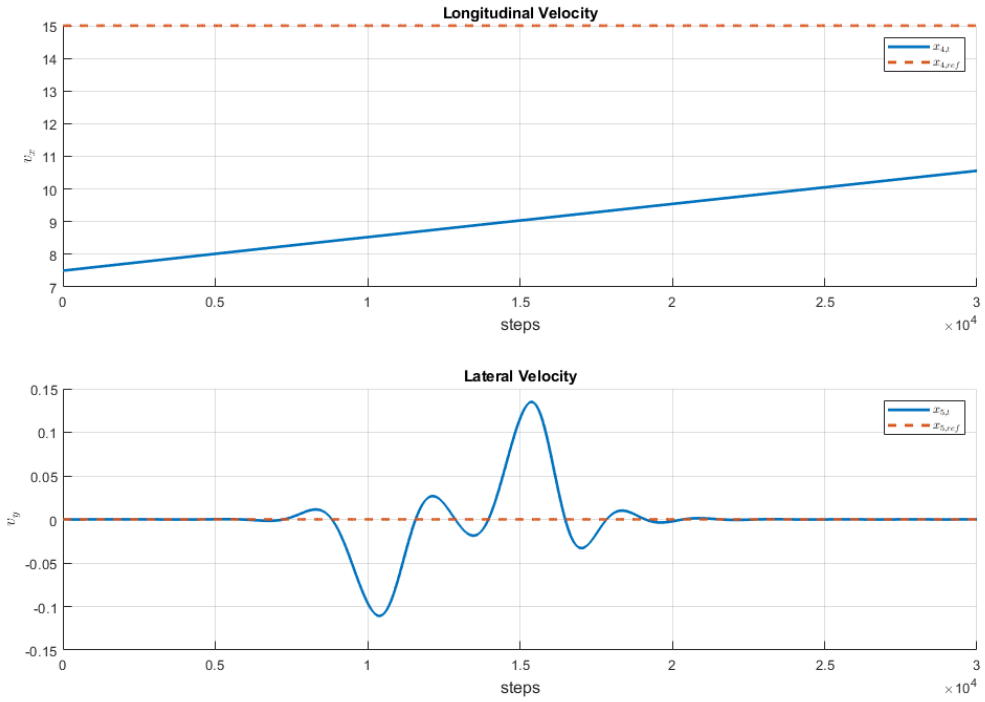


Figure 2.6: v_x and v_y - Lane Change Maneuver

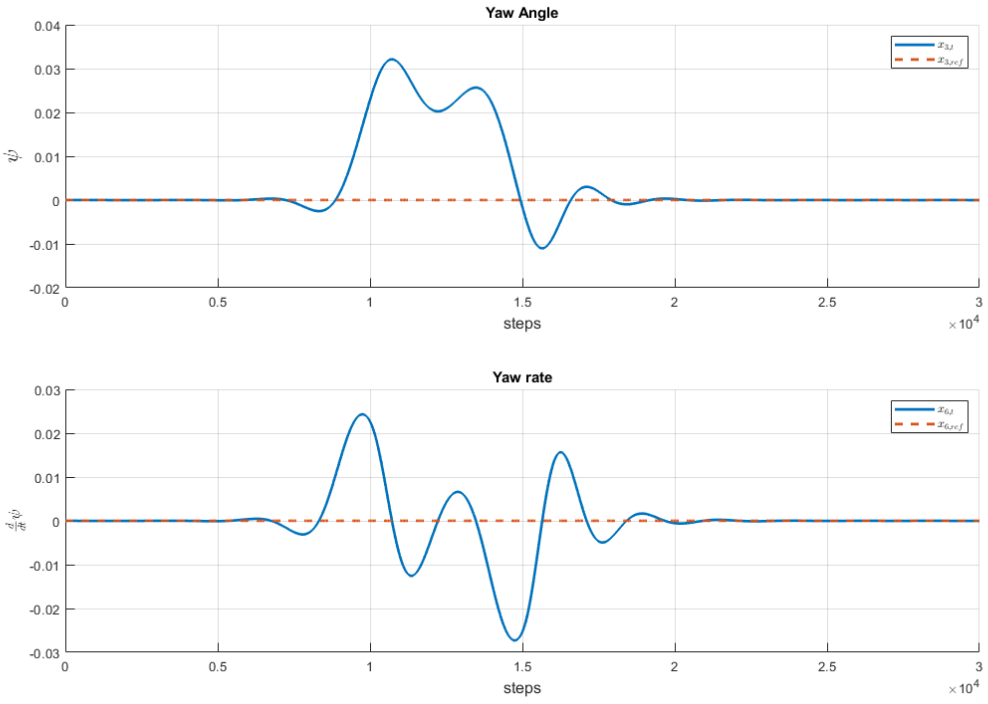


Figure 2.7: Yaw Rate and Yaw Angle - Lane Change Maneuver

Chapter 3

SkidPad - Task 2

In the second task we were asked to perform an optimal trajectory tracking along a skidpad using again the DDP introduced in 2. As in the previous case, the initialization is mandatory for the convergence of the algorithm. Unfortunately, during this task we faced several issues, both for what concern the definition of the initial trajectory and in the calculation of the optimal trajectory. Let's deal with it step-by-step.
First, we develop a script to graphically represent the circuit. Then, we have computed the trajectory to follow (i.e. the center of the track), by defining as reference points $\left[x_{traj}, y_{traj}, \psi_{traj} \right]$.
To do that, we tried two different approaches:

1. A trajectory based on the space travelled, which defines the distance of two consecutive reference points by considering the ds;

2. A pure geometric trajectory, which divide it by the steps of the simulation.

We had a better behaviour by using the second one, but both are included in the final folder. Once defined the reference, we had to create an initial trajectory for the DDP. In this case, we had to use a simpler approach for the steering angle with respect to geometric pointer in task 1, because the approximation used before seems to not properly work with a circular trajectory. We tried also other steering controllers presented in literature, but with no one gives good results. At the end, we have decided to force it by imposing a constant steering angle which changes its sign when the first circle has been completed. Instead, we used the same PI controller based on the error distance in equation 2.1 for what concern the longitudinal force F_x. The main result obtained is visible in figure 3.2, where we plot both the track and the trajectory obtained. We were able to track almost perfectly the reference trajectory, following a conservative approach since we have no boundary constraints forcing the car to remain on the track. However, we do not achieve the performances expected: the algorithm still goes in singularity when we tried to simulate the system under a simulation time of 55 seconds. The major drawback that can be appreciated is the velocity of the car around the circuit, which turns out to be very low. Also in this case we tried both with gamma fixed and Armijo's rule and the results obtained can be seen in figure 3.1, they are again pretty similar. All states and inputs variables obtained in task 2 are all presented in figures 3.3, 3.4, 3.5 and 3.6.

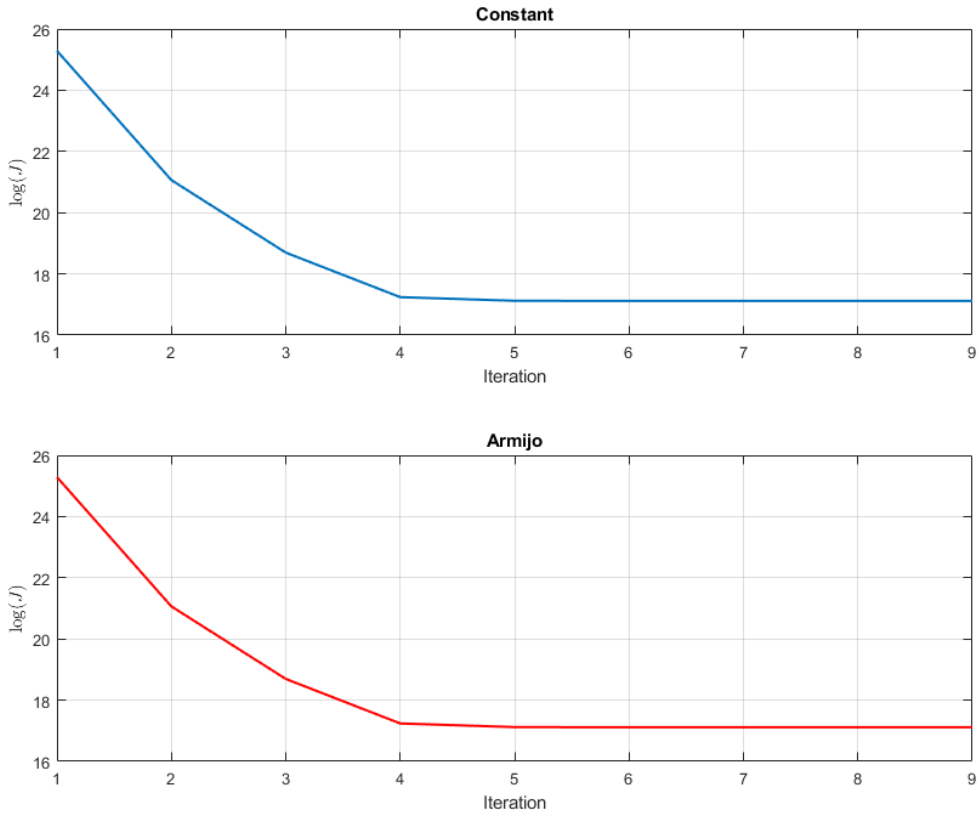


Figure 3.1: Fixed gamma vs Armijo's rule - Skidpad

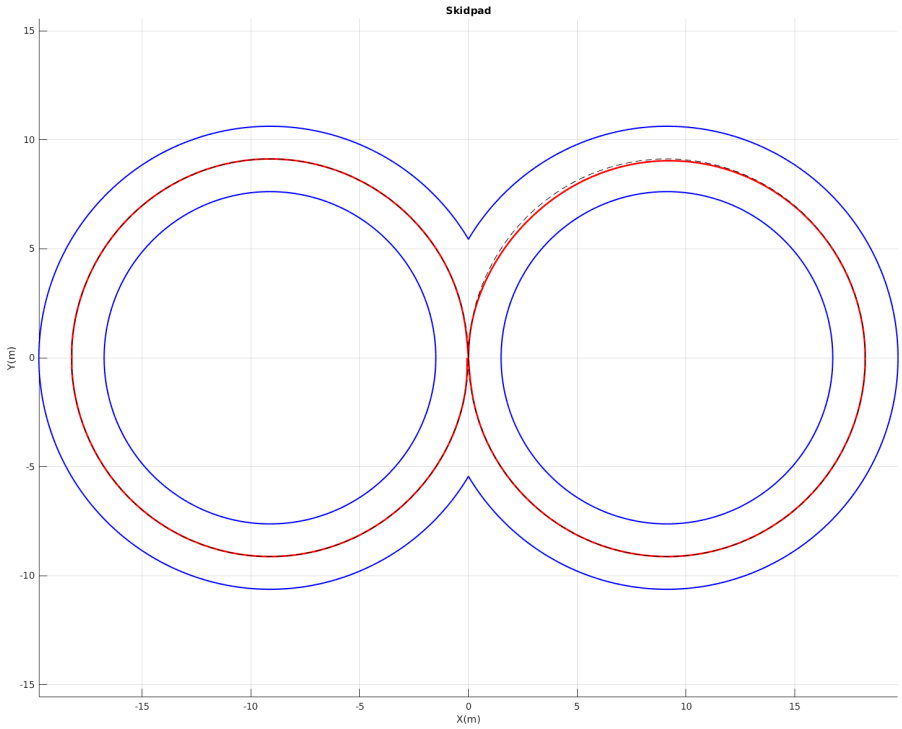


Figure 3.2: Skidpad trajectory by using DDP algorithm

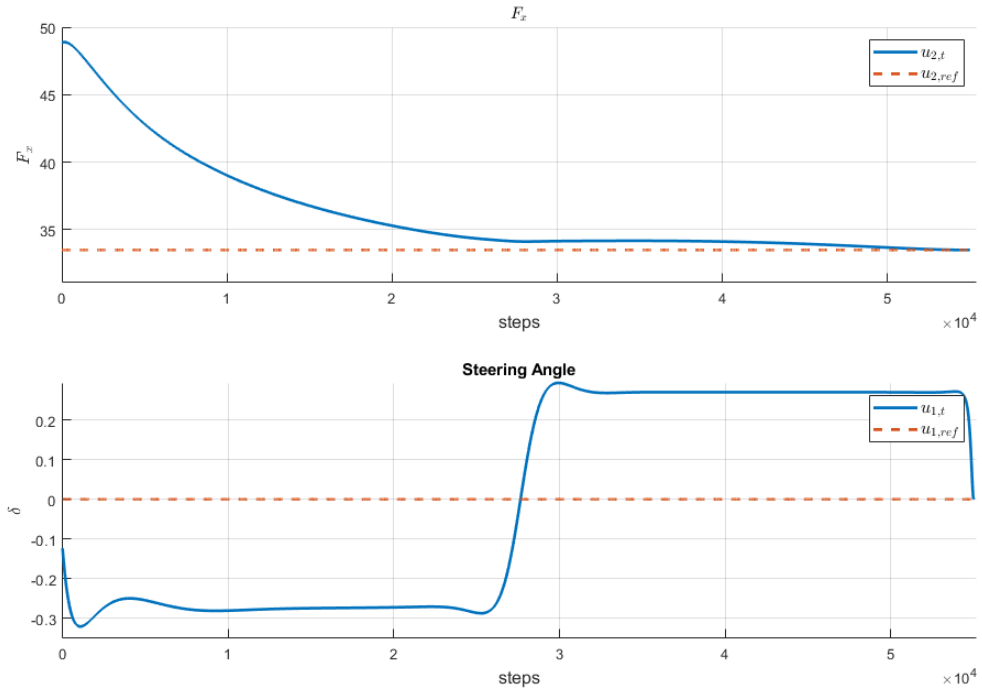


Figure 3.3: F_x and δ - Skidpad

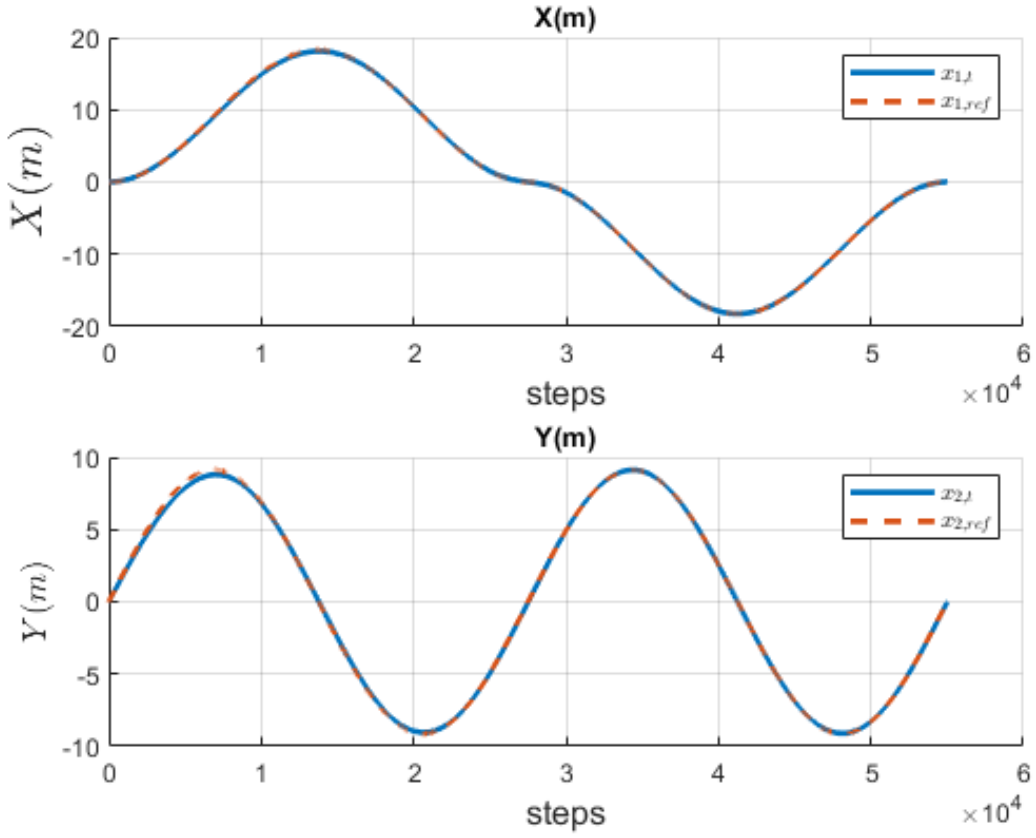


Figure 3.4: $X(m)$ and $Y(m)$ - Skidpad

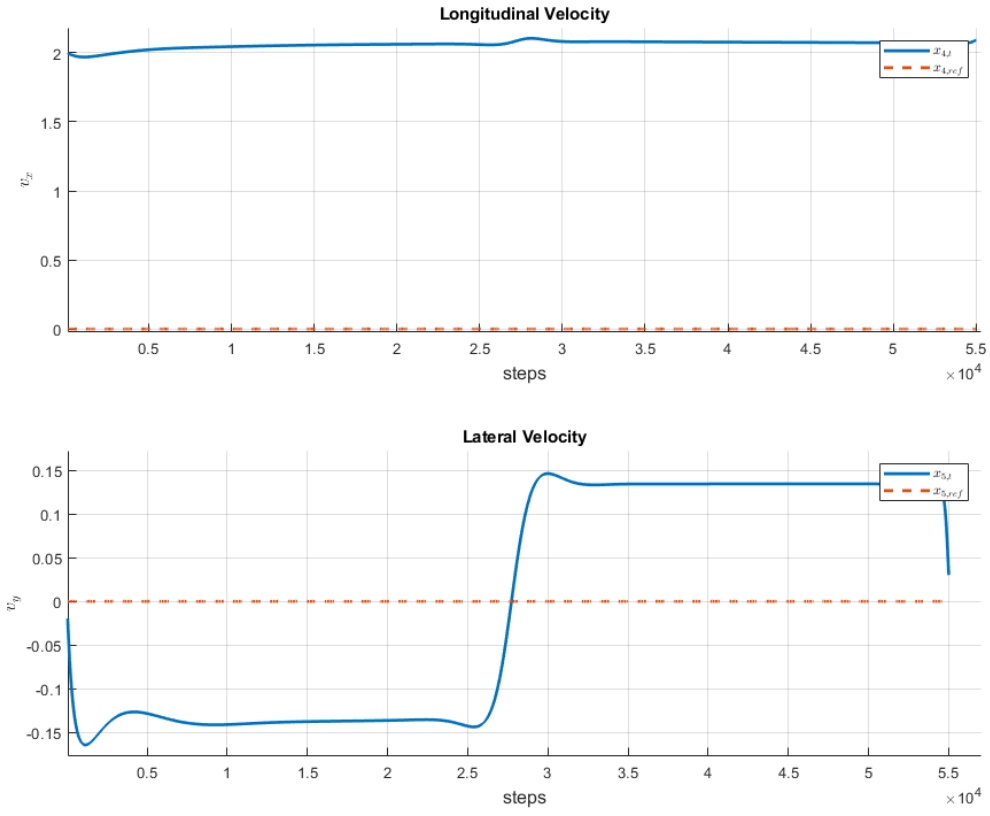


Figure 3.5: v_x and v_y - Skidpad

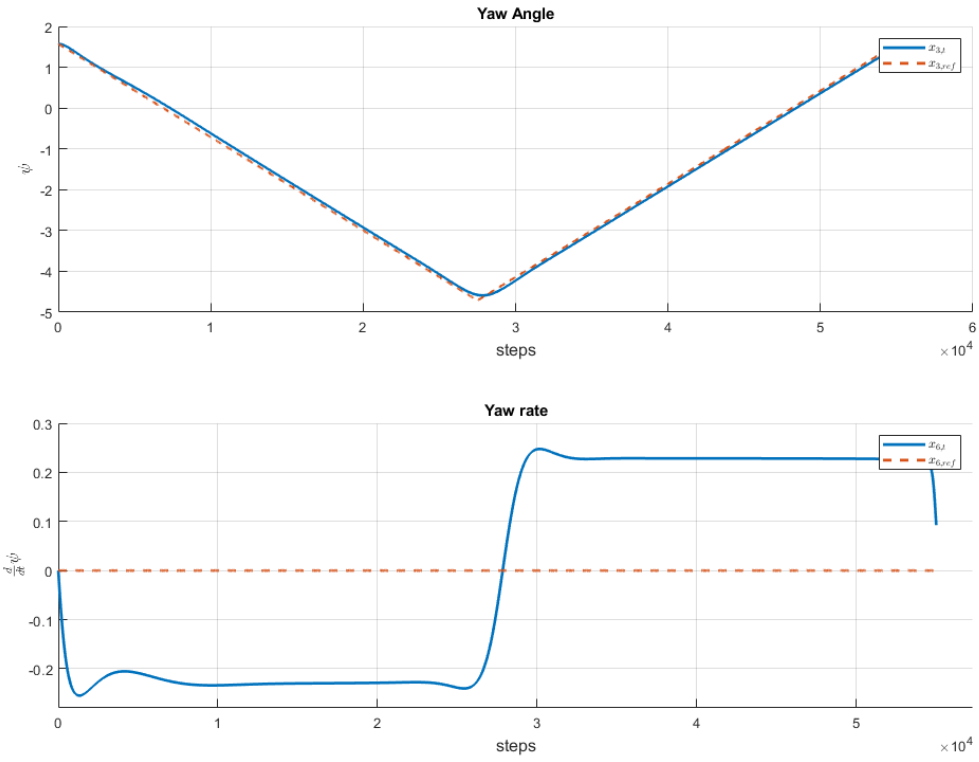


Figure 3.6: Yaw rate and Yaw angle - Skidpad

Chapter 4

LQR tracking and Animation - Task 3

In this last task we were asked to track the Optimal Trajectory computed using the DDP algorithm, which is an open-loop state-input trajectory, by defining a LQ feedback controller which stabilize the system. Then, we developed a simple animation of the car around the track. First of all, let's define the Finite Horizon LQ Optimal Control problem that we have used:

$$\begin{aligned} \min_{\Delta\mathbf{x},\,\Delta\mathbf{u}} \quad & \sum_{t=0}^{T-1} (\Delta x_t^T Q_t^{\text{reg}} \Delta x_t + \Delta u_t^T R_t^{\text{reg}} \Delta u_t) + \Delta x_T^T Q_T^{\text{reg}} \Delta x_T \\ \text{s.t.} \quad & \Delta x_{k+1} = A_t^{\text{opt}} \Delta x_t + B_t^{\text{opt}} \Delta u_t \quad t = 0, \dots, T-1, \\ & x_0 \text{ given} \end{aligned} \tag{4.1}$$

Then the pseudo-code of the algorithm is shown in 4. We introduced some noise in the initial

Algorithm 2 LQ Optimal Controller

0.*Compute* an Optimal State-Input Trajectory $(\mathbf{x}_{\text{opt}}, \mathbf{u}_{\text{opt}})$ using 2
1.*Linearize* the system around $(\mathbf{x}_{\text{opt}}, \mathbf{u}_{\text{opt}})$ via the LTV system:
for $t = 0, \dots, T$ **do**
  $A_t^{\text{opt}} := \nabla_{x_t} f(x^{\text{opt}_t}, u_t^{\text{opt}})^T$
  $B_t^{\text{opt}} := \nabla_{u_t} f(x^{\text{opt}_t}, u_t^{\text{opt}})^T$
  $\Delta x_{t+1} = A_t^{\text{opt}} \Delta x_t + B_t^{\text{opt}} \Delta u_t$
end for
2. *Calculate* the LQ Optimal Controller by solving the problem 4.1 as follows:
Set $P_T = Q_T^{\text{reg}}$ and
for t=T-1,...,0 **do**
  $P_t = Q_t^{\text{reg}} + A_t^{\text{opt}^T} P_{t+1} A_t^{\text{opt}} - (A_t^{\text{opt}^T} P_{t+1} B_t^{\text{opt}})(R_t^{\text{reg}} + B_t^{\text{opt}^T} P_{t+1} B_t^{\text{opt}})^{-1} (B_t^{\text{opt}^T} P_{t+1} A_t^{\text{opt}})$
end for
Compute the Feedback Gain
for t=0,...,T-1 **do**
  $K_t^{\text{reg}} := -(R_t^{\text{reg}} + B_t^{\text{opt}^T} P_{t+1} B_t^{\text{opt}})^{-1} (B_t^{\text{opt}^T} P_{t+1} A_t^{\text{opt}})$
end for
3. *Track* the generated Optimal Trajectory
for t=0,...,T-1 **do**
  $u_t = u_t^{\text{opt}} + K_t^{\text{reg}} (x_t - x_t^{\text{opt}})$
  $x_{t+1} = f(x_t, u_t)$
end for

conditions to test if the LQR was able to track the optimal trajectory also under uncertainties. It was surprisingly robust, even if the car starts outside the track boundaries. Then, to stress out its behaviour, we tried to introduce some noise in the position also during the motion of the car. In figure 4.1 can be appreciated the difference between the DDP trajectory and the one achieved by the LQR introducing a noise on the initial conditions x_0 and on the position of the car while moving. To conclude, we show in figure 4.2 the different trajectories obtained between the noisy LQR and the Optimal (DDP) algorithms.

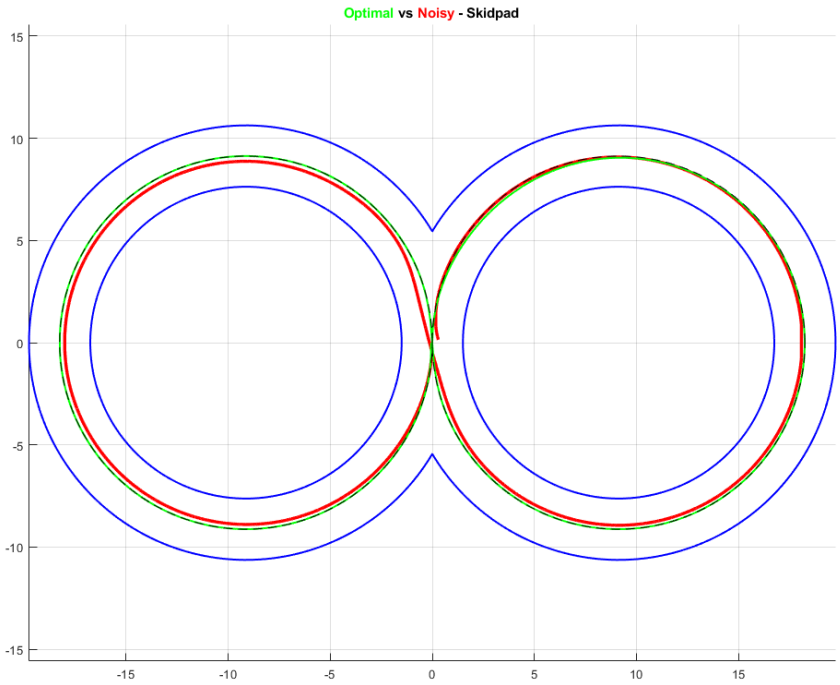
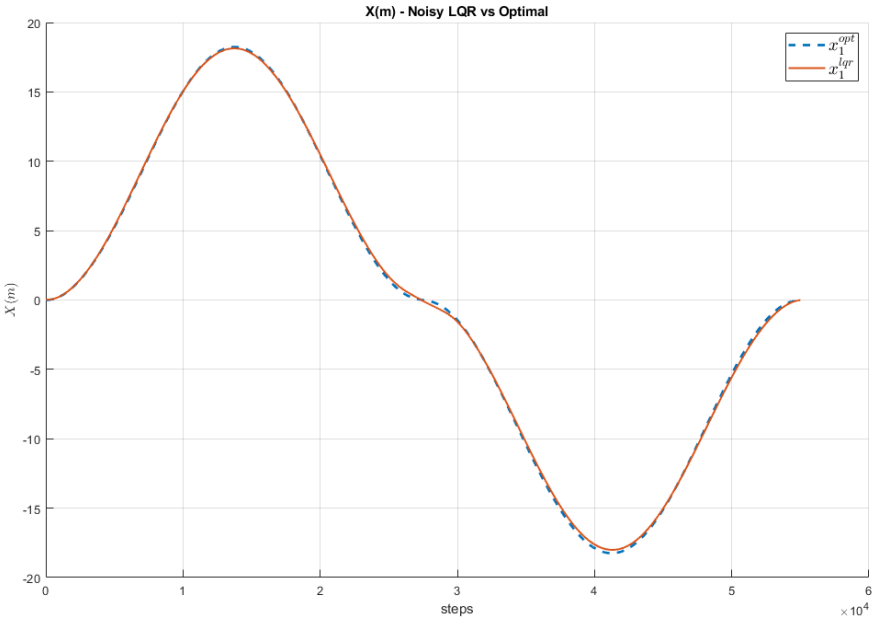
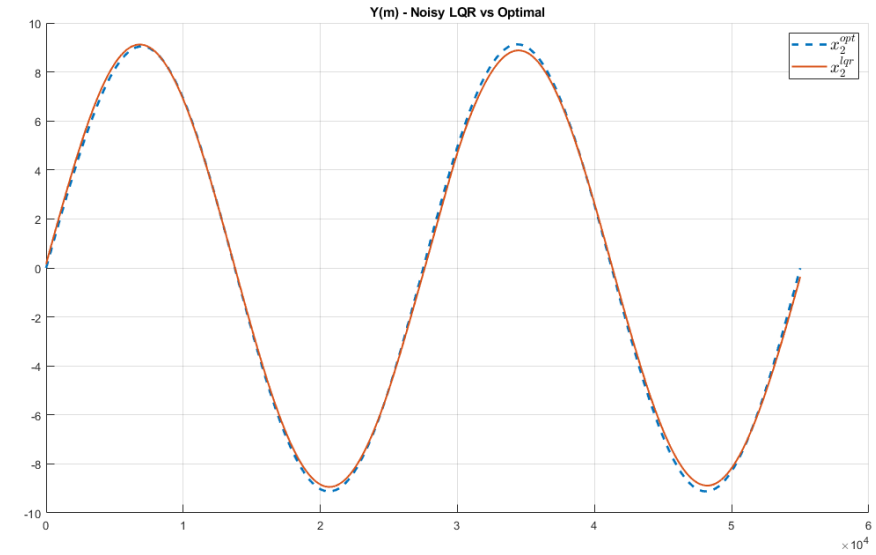


Figure 4.1: LQR controller with noisy x_0 w.r.t. DDP trajectory on the Skidpad



(a) $X(m)$



(b) $Y(m)$

Figure 4.2: Comparison between noisy LQR and Optimal trajectories

Conclusions

The project has been quite stimulating, and gives us the possibility to work on an interesting topic as the Optimal Control applied to vehicles.However, the performances achieved are not outstanding, in particular if we focus on the time needed to complete the skidpad. Further improvements on this topic could be obtained by changing the reference trajectory, applying the one with the velocity profile to reduce the lap time, and introducing track boundaries to push the car to its handling limits even if we do not exactly track the reference. We would to thanks professor Notarstefano for the course and the possibility given to work on this interesting topic and our tutor Marco Borghesi for his support during the development of the project.