POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Thesis

Design and experimental validation of vehicle dynamics estimators and trajectory tracking model predictive controller for scaled fully autonomous vehicles

Supervisors
Prof. Alessandro Vigliani
Prof. Angelo Domenico Vella
Dr. Umberto Montanaro
Prof. Aldo Sorniotti

Candidate
Nuccio lo Bello

Academic Year 2023-2024

Abstract

The progress of self-driving cars widely depends on the development of precise navigation systems and effective motion controllers that allow to track the desired trajectories provided by the motion planning stage. Both these critical aspects are addressed in this work: the enhancement of localization accuracy by using sensor fusion and estimation techniques and the design and experimental validation of trajectory tracking controllers for autonomous driving.
In particular, the first part of the thesis focuses on the design of Extended Kalman Filters (EKFs) that, by integrating measurements of multiple sensors such as Lidar, Inertial Measurement Unit (IMU) and encoder, allow to overcome the issues of splikes in Lidar-based pose measurements. These filters not only mitigate the impact of outliers in the Lidar measurements but also allow to estimate vehicle dynamics state variables that are not directly measured, e.g. vehicle side-slip angle. The improved localization and vehicle dynamics estimators are crucial for ensuring both reliability and accuracy of autonomous vehicles in diverse operational scenarios.
The second part of the work delves into the motion control stage which represents a fundamental module of an autonomous vehicles software stack, aiming to follow accurately the reference path by ensuring vehicle stability and robustness of control system performance. Specifically, two control strategies are presented: a static state feedback controller with optimization and a model predictive controller. All estimators and controllers have been firstly tested by using closed-loop simulations and then experimental validated employing a scaled fully autonomous vehicle prototype.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In earlier times, vehicles were characterized by their simplicity, primarily serving the purpose of transportation. However, as society progressed and technology evolved, vehicles began to encompass more than just transportation and it's started to prioritize comfort, safety, and convenience as well. This shift in focus prompted extensive research into enhancing vehicles by integrating technological innovations and advancements and the idea of autonomous vehicles (AVs) was soon conceived. According to the World Health Organization (WHO), about 1.3 million people die every year due to road accidents, by representing the major cause of death among persons aged 5-29 [1]. The main risk factors include high speeds, alcohol, distracted driving, unsafe vehicles and infrastructures. AVs have garnered considerable attention from researchers and manufacturers for their potential to assist in driving tasks such as sensing the surrounding environment, planning the shortest route, navigating, controlling speed and parking without human intervention. Although AVs are not yet widely adopted, their potential social and economic benefits are evident. They can play a crucial role in reducing road accidents by decreasing human errors, fuel consumption, and traffic congestion. These promising advantages represent a significant motivation for this work, whose aim is the design and experimental validation of vehicle dynamics estimators and trajectory tracking controllers for fully scaled autonomous vehicles. These model vehicles, designed for academic teaching and research, are presented in the following paragraph.

1.1 QCar

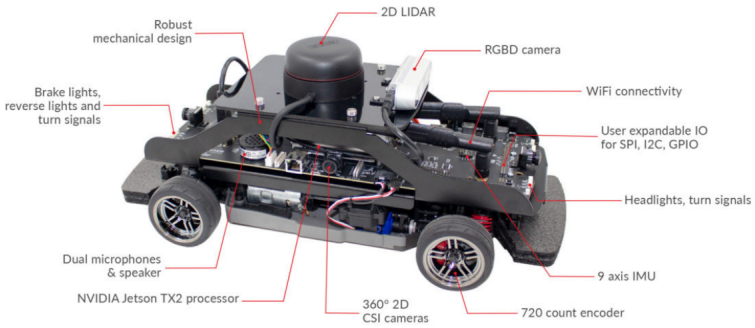


Figure 1.1: QCar

QCar, the feature vehicle of the Self-Driving Car Studio, is an open-architecture, scaled model vehicle designed for academic teaching and research. The vehicle is equipped with a wide range of sensors including lidar, 360 degrees vision cameras, depth sensor, inertial measurement unit (IMU) and one encoder, with the possibility to connect four encoders, one for each wheel. The QCar is supplied with a software stack where it's possible to modify or create new ROS nodes, allowing to interface directly with the vehicle avoiding the use of lower level programming language.
Additionally, a ground station and a desktop PC are provided to control the vehicles by running the desired control algorithm in the target hardware. The experimental setup is shown in the following figure.

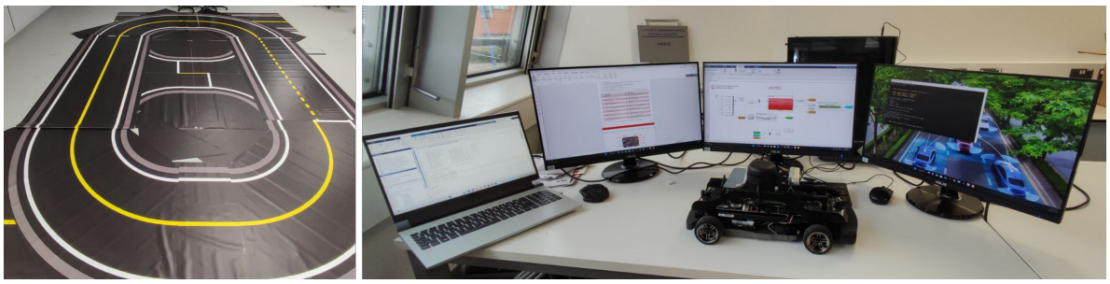
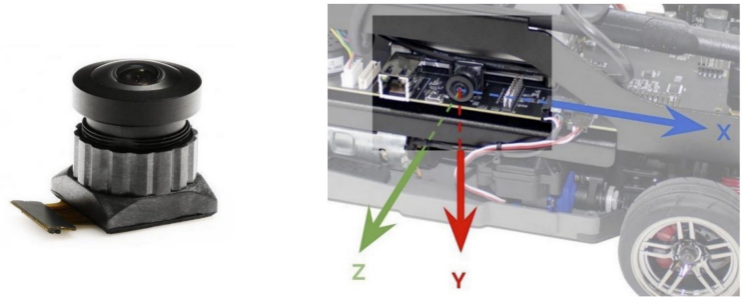


Figure 1.2: Experimental setup of the QLab

1.1.1 Hardware, sensors and actuators

The onboard computer of the QCar is a NVIDIA Jetson TX2, with Linux operating system, CPU: 2 GHz quad-core ARM Cortex-A57 64-bit and GPU: 256 CUDA core NVIDIA Pascal. The sensors and the actuators the vehicle is equipped with are:

- **CSI cameras** (Camera Serial Interface): the QCars is equipped with four CSI cameras providing a 360° vision of the surrounding environment. Each camera presents an horizontal Field-Of-View of 160° and a vertical Field-Of-View of 120°, a variable resolution from 3280x2464 to 820x410 and a frame rate with resolution ranging from 21 fps to 120 fps.



(a) CSI camera (b) Camera reference frame.

Figure 1.3: CSI camera and relative reference frame

Resolution	Max Frame Rate (FPS)	Horizontal FOV	Vertical FOV
3280x2464	21 Hz	160°	120°
1640x1232	80 Hz	160°	120°
1640x820	120 Hz	160°	80°
820x616	80 Hz	160°	120°
820x410	120 Hz	160°	80°

Figure 1.4: Available frame rates of CSI cameras

- RGB-D camera: the QCar is equipped with an Intel RealSense D435 RGB-D (Red, Gree, Blue and Depth) camera. It includes an IR projector and two IR imagers, which make this unit a stereo tracking solution. The camera can provide RGB, infrared data streams (left and right) and depth at different frame rates and resolutions. Moreover, the depth image allows to trace the distance of the objects that RealSense is framing, enabling objects detection in front of the vehicle which can be useful for creating an adaptive cruise control. The horizontal and vertical FOVs of this camera have higher values than the CSI one.

Figure 1.5: RGB-D camera

- IMU: the QCar is equipped with a 9-axis inertial measurement unit; specifically, three for the accelerometer to measure longitudinal, lateral and vertical accelerations, three for the gyroscope to measures angular velocities around each axis of the vehicle body reference frame, and three for the magnetometer to measure the magnetic field. The specifications of this sensor are reported below.

- DC steering motor: the QCar is equipped with only one motor for driving the four wheels though two differentials. The motor is the Titan 12T 550 manufactured by the US company Traxxas. Moreover, in order to manage the steering of the front wheels, a servo motor is used whose rotor is limited by physical constraints of [-0.5, 0.5] rad, i.e. $\pm$ 28.65°.

- Encoder: the QCar is equipped with an encoder placed on the rotor which is used to measure the angular position of the drive motor. The model of the encoder is E8T720-125 and it is produced by US Digital. It features a single-ended optical shaft that provides 720 counts per revolution. Additionally, it also provides the measurement of rotor speed based on the time that elapses between rising and falling edges of the signal.

RGB		*Infrared*		*Depth*	
Resolution	***Frame Rate (FPS)***	***Resolution***	***Frame Rate (FPS)***	***Resolution***	***Frame Rate (FPS)***
1920x1080	60	1280x800	30	1280x720	30
1280x720	60	1280x720	30	848x480	90
960x540	60	848x480	90	848x100	100
848x480	60	848x100	100	640x480	90
640x480	60	640x480	90	640x360	90
640x360	60	640x360	90	480x270	90
424x240	60	480x270	90	424x240	90
320x240	60	424x240	90	256x144	90
320x180	60	256x144	90	-	-

Figure 1.6: Resolution and frame rate of RGB-D camera

Sensor	**Description**
Accelerometer	16-bit with configuration range $\pm$2g to $\pm$16g
Gyroscope	Configurable range from $\pm$125°/s to $\pm$2000°/s
Magnetometer	Resolution of 0.3 μ/LSB

Figure 1.7: IMU specifications

Figure 1.8: Optical encoder

- Lidar: the lidar (Light Detection and Ranging) is a sensor that uses laser pulses to measure distances of the surrounding environment. Specifically, it consists of a laser emitter, a signal receiver and a data processing system. The emitter sends laser pulses in the desired direction, while the receiver gets back the signal reflected by the object. The data processing system computes the distance to the object based on the time it takes for the laser to return to the receiver. The Qcar is equipped with the RPLidar A2 (A2M8): this is a 2D planar lidar that supports up to 8000 samples per second with a scan rate of up to 15 Hz, providing a detection range of up to 18 m. The scanning frame rate and the corresponding samples per revolution are presented in the following.

Figure 1.9: RPLidar A2 (A2M8)

Frequency	Samples per revolution	Angular Resolution (Degrees)
5 Hz	1600	0.225°
10 Hz	800	0.45°
15 Hz	533	0.675°

Figure 1.10: Available frame rates and sample frequencies of the lidar sensor

Lidar, especially in indoor applications, is used for the SLAM (Simultaneous localization and mapping) service to trace the position of the vehicle and to map the system environment. In particular, Quanser has created a localization service for QCar using lidar, the algorithm is structured as follows: 1) an initial scan is carried out of the environment in which the QCar remains stationary. At the end of the scan the lidar data is saved in local memory and the initial position of the car or rather the position in which the scan took place coincides with the origin of the fixed reference system to which the position of the QCar will refer; 2) the position with the vehicle running is obtained by comparing the data that the lidar is acquiring with the data saved in memory during the first scan; depending on this comparison, the vehicle position and orientation are calculated and a score is also associated with this measurement.
However, the localization service provided by Quanser is still under development and therefore the vehicle pose provided may not be accurate. In particular, this problem is due to the fact that the initial scan takes place with the car stationary and therefore only the data of a very specific point is saved in memory. Indeed, to improve the first scan it is advisable to insert references into the environment, such as boxes or other objects, being able to obtain a measurement of the position more precise. Another issue arises from the lidar limited maximum sample frequency of 15 Hz, which could be not enough to ensure optimal performance for the path tracking control strategies that are based mainly on vehicle localization. To figure out these issues, Kalman filters have been implemented and presented in chapter 5.

Chapter 2

Vehicle models

In the context of vehicle dynamics and control, the precise modeling of vehicle motion dynamics plays a crucial role in the design of effective control algorithms. Within this scenery, two fundamental models, the kinematic model and the single-track model, allows us to comprehend and predict the vehicle behavior and become indispensable in the development of robust control systems for autonomous vehicles, where an accurate representation of the dynamics of the car is essential for ensuring safe navigation.
The kinematic model provides a simplified representation of the vehicle's motion, by focusing on its pose without introducing the complexities of forces and torques actions. It is a valuable abstraction for scenarios where dynamics can be overlooked in favor of a concise description of the vehicle's path.
On the other hand, the single-track model goes deeper into the dynamic aspects of vehicular motion, considering the forces and torques that influence the vehicle's behavior. This model, despite still represents a simplification, accounts for essential dynamics, making it suitable for tasks requiring a more detailed understanding of vehicle behavior.
In this chapter both of the models are presented and discussed since they are employed for Kalman filters and path tracking controllers design and implementation.

2.1 Reference frames

The aim of this paragraph is to present the two reference frames employed in the work. Reference frames serve as the basis for interpreting and responding to the surrounding world, allowing vehicles to understand their position, orientation, and movement relative to their surroundings. For localization purposes it's needed expressing the vehicle pose with respect to a fixed point in the space. Let's consider a mobile reference frame fixed on the vehicle itself and centered in its center of gravity that we call '*body reference frame*'. The '*inertial reference frame*' (centered in 'I' in the figure 2.1) is a fixed reference frame. The position and the orientation of the vehicle are referenced with respect to this latter:

$$p_I(t) = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix} \tag{2.1}$$

where p_I is the generic vector composed by x, y coordinates and yaw angle ψ. Considering that the body reference frame is rotating with respect to the fixed one by the angle ψ, the following velocity expressions hold true:

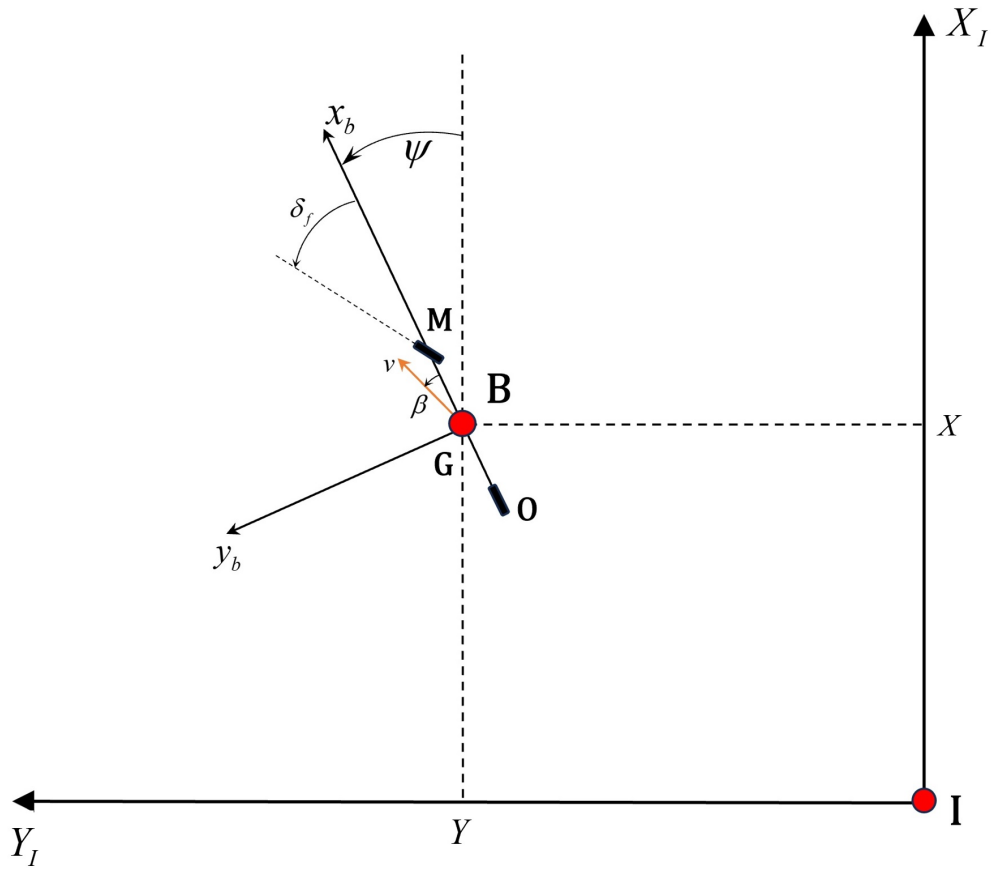


Figure 2.1: Inertial and body reference frames

$$\begin{aligned} v^B &= R_I^B v^I \\ v^I &= (R_I^B)^\top v^B \end{aligned} \tag{2.2}$$

where

$$R_I^B = \begin{pmatrix} \cos(\psi) & \sin(\psi) \\ -\sin(\psi) & \cos(\psi) \end{pmatrix} \tag{2.3}$$

and

$$R_B^I = \begin{pmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{pmatrix} \tag{2.4}$$

For what concern the position, it can be easily seen that the vehicle coordinates vector in the body reference frame is always zero since such reference system moves with the vehicle itself.

2.2 Kinematic model

The vehicle kinematic model, shown in figure 2.2, considers three degrees of freedom. The vehicle is considered to have front and rear wheels collapsed in single points, M and O respectively.

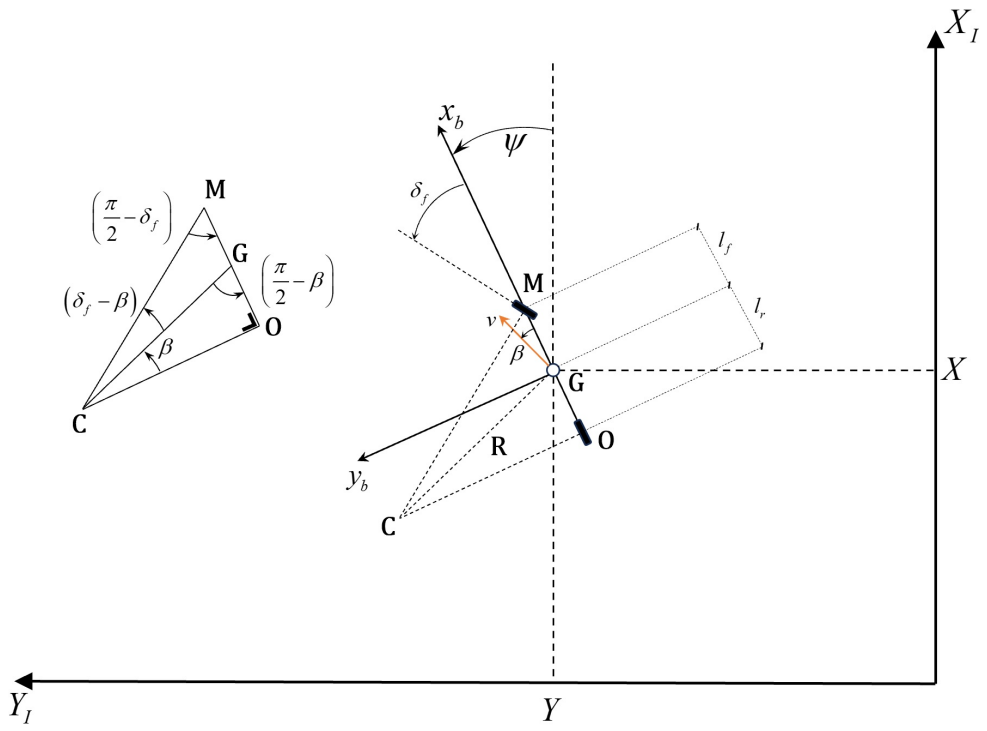


Figure 2.2: Vehicle kinematic model

For front-wheel-only steering vehicles, the steering angles are δ_f, that is the angle between vehicle longitudinal axis and front wheel longitudinal axis, and $\delta_r = 0$, that is the angle between vehicle longitudinal axis and rear wheel longitudinal axis . The vehicle mass is assumed to be concentrated in its center of gravity (c.o.g), which corresponds to point G. The distances of M and O from the center of gravity are l_f and l_r respectively and their sum equals $L = l_f + l_r$ and represents the vehicle wheelbase. Under the above assumptions, the vehicle moves along a circle of radius R, with the center of the curvature being C. This point coincides also with the istantaneous centre of rotation of the body and can be found as the intersection of the normal-segment to the longitudinal plane of the front and rear wheels MC and OC. The velocity of the system is perpendicular to the line CG and the angle between vehicle velocity vector and the longitudinal axis of the vehicle is called vehicle side slip angle β. The angle between the longitudinal axis and the vehicle heading is called yaw angle or heading angle ψ. The main assumptions for the kinematic model [2] are the following:

- Vehicle is moving in 2D plane, so roll, pitch and lift dynamics are neglected;
- The wheel slip angles are considered to be zero. This implies that velocities vectors v_M and v_O are oriented along the direction of the front and rear wheels respectively. In the presence of low vehicle velocity ($v < 5$ m/s), since the lateral forces are small, this assumption is

reasonable.

By assuming that the curvature radius changes slowly in the presence of low vehicle velocity, the vehicle yaw rate shall be equal to the vehicle angular velocity:

$$\dot{\psi} = \omega = \frac{v}{R}. \tag{2.5}$$

Applying the sine rule to the triangle CGM we get:

$$\frac{\sin(\delta_f - \beta)}{l_f} = \frac{\sin\left(\frac{\pi}{2} - \delta_f\right)}{R} \tag{2.6}$$

Applying the sine rule to the triangle CGO we get:

$$\frac{\sin(\beta)}{l_r} = \frac{\sin(\frac{\pi}{2})}{R} = \frac{1}{R} \tag{2.7}$$

Multiplying both sides of 2.6 by $\frac{l_f}{\cos(\delta_f)}$ we get:

$$\frac{\sin(\delta_f)\cos(\beta) - \cos(\delta_f)\sin(\beta)}{l_f} = \frac{\cos(\delta_f)}{R} \Rightarrow \tan(\delta_f)\cos(\beta) - \sin(\beta) = \frac{l_f}{R} \tag{2.8}$$

In the same way, multiplying both sides of 2.7 by l_f we get:

$$\sin(\beta) = \frac{l_r}{R} \tag{2.9}$$

If we combine 2.8 and 2.9 it holds true the following expression:

$$\cos(\beta)\tan(\delta_f) = \frac{l_f + l_r}{R} \tag{2.10}$$

Finally, substituting 2.10 in 2.5, the vehicle yaw rate can be expressed as follows:

$$\dot{\psi} = \frac{v\cos(\beta)}{l_f + l_r}\left(\tan(\delta_f)\right). \tag{2.11}$$

The equation 2.11 represents the differential equation of the kinematic model that describes the evolution of the vehicle yaw angle trajectory. In order to write the remaining differential equations of the kinematic model that describe the evolution of the X and Y coordinates of the vehicle, the vehicle velocity vector shall be expressed in the inertial frame. The velocity vector described in the body reference frame is:

$$v = \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} v \cdot \cos(\beta) \\ v \cdot \sin(\beta) \end{bmatrix} \tag{2.12}$$

Since the body reference frame rotates of the angle ψ with respect to the inertial reference frame, the vehicle velocity vector in the inertial frame can be expressed as:

$$V^{(I)} = \begin{bmatrix} V_X \\ V_Y \end{bmatrix} = \begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = R_B^I v^{(B)} = \begin{pmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{pmatrix} \cdot \begin{bmatrix} v \cdot \cos(\beta) \\ v \cdot \sin(\beta) \end{bmatrix} \tag{2.13}$$

It follows:

$$V_X = v \cdot \cos(\psi) \cdot \cos(\beta) - v \cdot \sin(\psi) \cdot \sin(\beta) = v \cdot \cos(\psi + \beta) \tag{2.14}$$

$$V_Y = v \cdot \sin(\psi) \cdot \cos(\beta) + v \cdot \cos(\psi) \cdot \sin(\beta) = v \cdot \sin(\psi + \beta) \tag{2.15}$$

The two equations presented above represent the differential equations of the kinematic model that describe the evolution of the vehicle coordinates in the inertial reference frame.
Finally, since all the kinematic model's differential equations 2.11, 2.14, and 2.15 depend on the vehicle side-slip angle, this shall be written as a function of the model inputs. To do so, it is possible to combine equation 2.8 and equation 2.9 multiplied respectively by l_r and l_f:

$$\begin{aligned} l_r(\tan(\beta)\cos(\beta) - \sin(\beta)) &= \tfrac{l_r \cdot l_f}{R} \\ l_f \cdot \sin(\beta) &= \tfrac{l_r \cdot l_f}{R} \end{aligned} \tag{2.16}$$

Developing the above equations, the vehicle side-slip angle can be written as:

$$\beta = \tan^{-1}\left(\frac{l_r \tan(\delta_f)}{l_r + l_f}\right) \tag{2.17}$$

Summarizing, the state variables of the vehicle kinematic model coincide with the vehicle pose X, Y, ψ and the input variables are the front steering angle δ_f and the vehicle velocity in the body reference frame v.

$$\begin{cases} \dot{X} = v\cos(\psi + \beta) \\ \dot{Y} = v\sin(\psi + \beta) \\ \dot{\psi} = \frac{v\cos(\beta)}{l_f + l_r}(\tan(\delta_f)) \\ \beta = \tan^{-1}\left(\frac{l_r \tan(\delta_f)}{l_r + l_f}\right) \end{cases} \tag{2.18}$$

2.3 Dynamic single-track model

In contrast to the simpler vehicle kinematic model, which primarily focuses on geometric relationships without accounting for the tire characteristic, as well as forces and torques that are applied, the dynamic single-track model provides a more comprehensive and realistic description of the vehicle motion's in the presence of higher vehicle velocity. Indeed, at higher speeds the assumption that tire side-slip angles are equal to zero, i.e. that the velocity vector at each wheel is in the direction of the wheel longitudinal axle, cannot be made since it does not approximate properly the vehicle's dynamics, so a different approach is needed.
In order to build the dynamic vehicle single-track model, the description of the forces acting on the vehicle is needed. For what concerns the forces due to tire-road interactions, thanks to experimental analysis, it is observed that the lateral tire force is proportional to the tire side-slip

angle α, which represents the angle between the velocity vector of the wheel and its longitudinal axis. The tire side-slip angles for both front and rear wheels can be obtained by writing a balance between the longitudinal and lateral components of the velocities of the tires and the vehicle chassis.

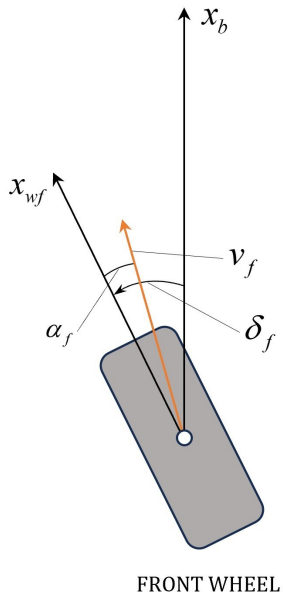


Figure 2.3: Front tire side-slip angle

The front tire side-slip angle (figure 2.3) can be obtained by writing a velocity balance in both lateral and longitudinal directions as follows:

$$v_f \sin(\delta_f - \alpha_f) = v \sin(\beta) + l_f \dot{\psi} \tag{2.19}$$

$$v_f \cos(\delta_f - \alpha_f) = v \cos(\beta) \tag{2.20}$$

The ratio between 2.19 and 2.20 leads to:

$$\tan(\delta_f - \alpha_f) = \frac{l_f \dot{\psi} + v \sin \beta}{v \cos \beta} \tag{2.21}$$

Under the assumption of small vehicle side-slip angle $\beta \leq 10°$, it's possible to consider $\sin(\beta) \cong \beta$, $\cos(\beta) = 1$ and $\tan(\beta) = \beta$. In such case equation 2.21 becomes:

$$\alpha_f = \delta_f - \frac{l_f \dot{\psi} + v\beta}{v} \tag{2.22}$$

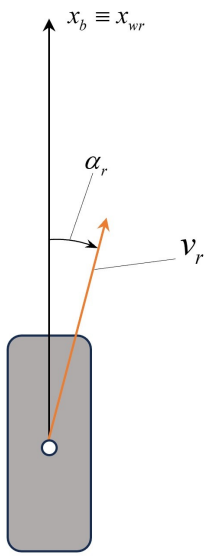


Figure 2.4: Rear tire side-slip angle

By using the same approach, the rear tire side-slip angle (figure 2.4) can be obtained by writing a velocity balance in lateral and longitudinal directions:

$$v_r \sin(\alpha_r) = -v \sin(\beta) + l_r \dot{\psi} \tag{2.23}$$

$$v_r \cos(\alpha_r) = v \cos(\beta) \tag{2.24}$$

The ratio between 2.23 and 2.24 gives:

$$\tan(\alpha_r) = \frac{l_f \dot{\psi} - v \sin \beta}{v \cos \beta} \tag{2.25}$$

and again assuming β small, the rear tire side-slip angle is equal to:

$$\alpha_r = \frac{l_r \dot{\psi} - v\beta}{v} \tag{2.26}$$

Once the tire side-slip angles are defined with equations 2.22 and 2.26, the lateral forces generated due to the interaction between the tire and the road, highlighted in figure 2.5, can be expressed as:

$$F_{yf} = C_{\alpha_f} \cdot \alpha_f = (C_{\alpha_{fl}} + C_{\alpha_{fr}}) \cdot \alpha_f \tag{2.27}$$

$$F_{yr} = C_{\alpha_r} \cdot \alpha_r = (C_{\alpha_{rl}} + C_{\alpha_{rr}}) \cdot \alpha_r \tag{2.28}$$

C_{α_f} and C_{α_r} are respectively the cornering stiffness of the front and rear tire and for the vehicle single-track model they correspond to the sum of the cornering stiffness of the two front and rear wheels.

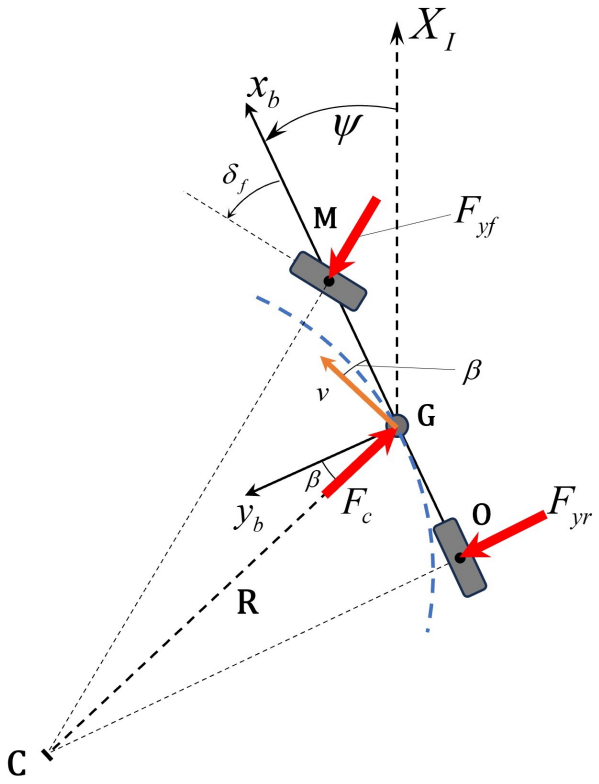


Figure 2.5: Single-track model cornering dynamics

Therefore the lateral frictional forces depend on the tire side-slip angle as they are generated due to tire deformation, whose magnitude depends on the lateral velocity and on the time spent by the tire's treads in the tire-road contact area [3]. For small lateral velocities, and hence for small slip angles, the lateral forces are linearly proportional to the slip angle. When the vehicle lateral velocity increases, also the tire deformation rises as well, and the relation between lateral force and tire side-slip angle becomes highly non-linear. An indicative shape of the tire lateral characteristic is presented in figure 2.6. The tire cornering stiffness C_α corresponds to the tangent of the curve (α, F_y). For small tire side-slip angles, the cornering stiffness can be considered constant and it is denoted as $C_{\alpha 0}$.

At this point it is possible to derive the differential equations that describe the vehicle dynamics considering the tire-road interactions. According to figure 2.5, we can write the equilibrium of forces along the y axis in the body reference frame as follow:

$$m \cdot a_y = F_c \cos\beta = F_{yr} + F_{yf} \cos(\delta_f) \tag{2.29}$$

Considering the vehicle velocity in the body reference frame:

$$v^{(b)} = \begin{bmatrix} v\cos(\beta) \\ v\sin(\beta) \\ 0 \end{bmatrix}$$

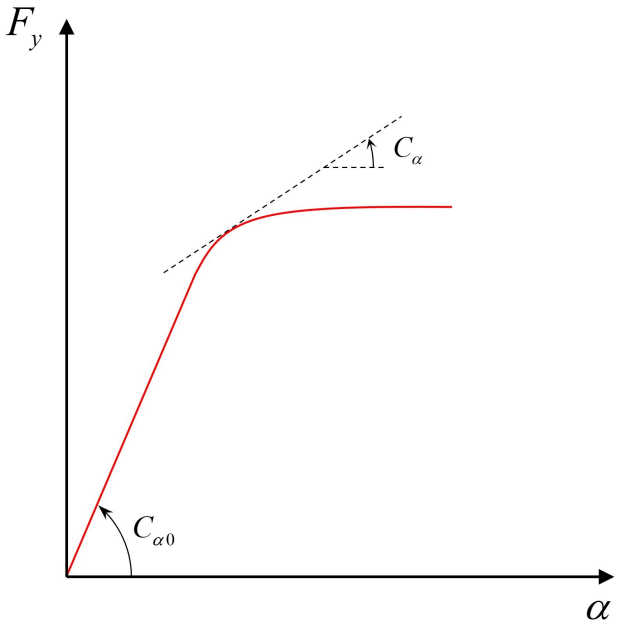


Figure 2.6: Lateral tire-road force as function of tire side-slip angle

the vehicle acceleration in the same reference frame can be computed as the sum of two terms: the acceleration due to motion along the y axis and the centripetal acceleration due to rotational motion along the z axis [2].

$$\begin{aligned} a^b = \tfrac{dv^b}{dt} + \omega^b \wedge v^b = \begin{bmatrix} -v\sin(\beta)\cdot\dot{\beta} \\ v\cos(\beta)\cdot\dot{\beta} \\ 0 \end{bmatrix} + \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix} \wedge \begin{bmatrix} v\cos(\beta) \\ v\sin(\beta) \\ 0 \end{bmatrix} = \\ = \begin{bmatrix} -v\sin(\beta)\cdot\dot{\beta} \\ v\cos(\beta)\cdot\dot{\beta} \\ 0 \end{bmatrix} + \begin{bmatrix} -v\sin(\beta)\cdot\dot{\psi} \\ v\cos(\beta)\cdot\dot{\psi} \\ 0 \end{bmatrix} = \begin{bmatrix} -v\sin(\beta)\cdot(\dot{\beta}+\dot{\psi}) \\ v\cos(\beta)\cdot(\dot{\beta}+\dot{\psi}) \\ 0 \end{bmatrix} \end{aligned} \tag{2.30}$$

The magnitude of the vehicle acceleration vector is equal to:

$$|a^b| = {a_n}^b = v\cdot(\dot{\beta}+\dot{\psi}) \tag{2.31}$$

Under the assumption of small vehicle side-slip angle β, the lateral acceleration can be expressed as:

$$a_y = v\cos\beta(\dot{\beta}+\dot{\psi}) \cong v(\dot{\beta}+\dot{\psi}) \tag{2.32}$$

Substituting the expression of the lateral acceleration presented above into the Newton's second law we get:

$$mv\cdot(\dot{\beta}+\dot{\psi}) = F_c\cos\beta = F_{yr} + F_{yf}\cos(\delta_f) \tag{2.33}$$

Additionally, substituting expressions 2.22, 2.27, 2.26, 2.28 in the previous one it's obtained:

$$\begin{aligned} mv \cdot (\dot{\beta} + \dot{\psi}) &= C_{\alpha r}\alpha_r + C_{\alpha f}\alpha_f \cos(\delta_f) = \\ &= C_{\alpha r}\left(\tfrac{l_r\dot{\psi}}{v} - \beta\right) + C_{\alpha f}\cos(\delta_f)\left(\delta_f - \tfrac{l_f\dot{\psi}}{v} - \beta\right) \end{aligned} \tag{2.34}$$

Considering the assumptions of small front steering angle, the previous equation can be rewritten as:

$$mv\dot{\beta} + \frac{\dot{\psi}}{v}\left(mv^2 - C_{\alpha r}l_r + C_{\alpha f}l_f\right) + \beta\left(C_{\alpha f} - C_{\alpha r}\right) = C_{\alpha f}\delta_f \tag{2.35}$$

Equation 2.35 is the first differential equation of the dynamic single-track model in which the state variable corresponds to vehicle side-slip angle β.
Based on the balance of the moments around the z-axis of the vehicle's body reference frame it is possible to write:

$$I_z\ddot{\psi} = l_f F_{yf} - l_r F_{yr} \tag{2.36}$$

Substituting equations 2.22, 2.27, 2.26, 2.28 in the previous equation:

$$I_z\ddot{\psi} = l_f F_{yf} - l_r F_{yr} = l_f C_{\alpha f}\left(\delta_f - \frac{l_f\dot{\psi}}{v} - \beta\right) - l_r C_{\alpha r}\left(\frac{l_r\dot{\psi}}{v} - \beta\right) \tag{2.37}$$

The above equation can be rewritten as:

$$I_z\ddot{\psi} + \frac{\dot{\psi}}{v}\left(\frac{C_{\alpha f}{l_f}^2}{v} + \frac{C_{\alpha r}{l_r}^2}{v}\right) + \beta\left(C_{\alpha f}l_f - C_{\alpha r}l_r\right) = C_{\alpha f}l_f\delta_f \tag{2.38}$$

Equation 2.38 is the second differential equation of the dynamic single-track model in which the state variable corresponds to vehicle yaw rate $\dot{\psi}$.

In conclusion, the equations that describe the vehicle dynamics in the presence of the assumptions of the dynamic single-track model are rewritten below:

$$\begin{cases} mv\dot{\beta} + \frac{\dot{\psi}}{v}\left(mv^2 - C_{\alpha r}l_r + C_{\alpha f}l_f\right) + \beta\left(C_{\alpha f} - C_{\alpha r}\right) = C_{\alpha f}\delta_f \\ I_z\ddot{\psi} + \frac{\dot{\psi}}{v}\left(\frac{C_{\alpha f}{l_f}^2}{v} + \frac{C_{\alpha r}{l_r}^2}{v}\right) + \beta\left(C_{\alpha f}l_f - C_{\alpha r}l_r\right) = C_{\alpha f}l_f\delta_f \end{cases} \tag{2.39}$$

The state variables of the dynamic single-track model are the vehicle side-slip angle β and the yaw rate $\dot{\psi}$, and the input variables are the front wheel road steering angle and the vehicle velocity in the body reference frame.

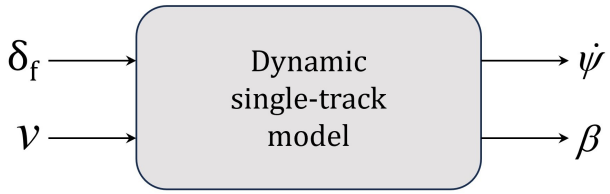


Figure 2.7: Block diagram of dynamic single-track model

2.4 Vehicle model for path tracking

To design trajectory tracking control algorithms, it is necessary to employ a dynamic model of the vehicle in which the lateral error and the heading error with respect to the road are used as state variables. So, the dynamic single-track model defined in 2.3 is re-written in terms of:

- e_y: distance of the c.g. of the car from the center line of the road;
- e_ψ: heading error of the car with respect to the road.

A car moving with constant longitudinal velocity v_x on a road with constant curvature ρ is considered. As usual, the curvature is assumed to be small so that the small angles assumptions can be made. The desired yaw rate of the vehicle is defined as:

$$\dot{\psi}_{des} = v_x \rho \tag{2.40}$$

Then, the desired acceleration of the vehicle can be expressed as:

$$v_x^2 \rho = v_x \dot{\psi}_{des} \tag{2.41}$$

The following variables can be defined [3]:

$$\begin{cases} \ddot{e}_y = \ddot{y} + v_x(\dot{\psi} - \dot{\psi}_{des}) \\ e_\psi = \psi - \psi_{des} \end{cases} \tag{2.42}$$

Since the longitudinal velocity is assumed to be constant, the following expression can be obtained from the above equation:

$$\dot{e}_y = \dot{y} + v_x e_\psi \tag{2.43}$$

If the velocity is not constant, the derivative of the lateral error with respect to the time has to be computed by using the following equation:

$$\dot{e}_y = \dot{y} + \int v_x e_\psi \cdot dt \tag{2.44}$$

by leading to a nonlinear and time-varying model that would not be suitable for control purposes. By substituting expressions of e_ψ and $\dot{e}_y$ into 2.39, it can be written:

$$\begin{cases} m\ddot{e}_y = \dot{e}_y(-\frac{C_{\alpha f}}{v_x} - \frac{C_{\alpha r}}{v_x}) + e_\psi(C_{\alpha f} + C_{\alpha r}) + \dot{e}_\psi(\frac{-C_{\alpha f} l_f}{v_x} + \frac{C_{\alpha r} l_r}{v_x}) + \\ \quad + \dot{\psi}_{des}(\frac{-C_{\alpha f} l_f}{v_x} + \frac{C_{\alpha r} l_r}{v_x}) + C_{\alpha f}\delta \\ I_z \ddot{e}_\psi = C_{\alpha f} l_f \delta + \dot{e}_y(-\frac{C_{\alpha f} l_f}{v_x} + \frac{C_{\alpha r} l_r}{v_x}) + e_\psi(C_{\alpha f} l_f - C_{\alpha r} l_r) + \\ \quad + \dot{e}_\psi(-\frac{C_{\alpha f} l_f^2}{v_x} - \frac{C_{\alpha r} l_r^2}{v_x}) - I_z \dot{\psi}_{des}(-\frac{C_{\alpha f} l_f^2}{v_x} - \frac{C_{\alpha r} l_r^2}{v_x} \end{cases} \tag{2.45}$$

The above equations can be written in the following state space form:

$$\dot{x} = Ax + B_1\delta + B_2\rho \tag{2.46}$$

where the state vector is $x = \begin{bmatrix} e_y & \dot{e}_y & e_\psi & \dot{e}_\psi \end{bmatrix}^T$ and the matrices A, B_1, B_2 are:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{C_{\alpha f}+C_{\alpha r}}{mv_x} & \frac{C_{\alpha f}+C_{\alpha f}}{m} & \frac{-C_{\alpha f}l_f+C_{\alpha r}l_r}{mv_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{C_{\alpha f}l_f-C_{\alpha r}l_r}{I_z v_x} & \frac{C_{\alpha f}l_f-C_{\alpha r}l_r}{I_z} & -\frac{C_{\alpha f}l_f^2+C_{\alpha r}l_r^2}{I_z v_x} \end{bmatrix}$$

$$B_1 = \begin{bmatrix} 0 \\ \frac{C_{\alpha f}}{m} \\ 0 \\ \frac{C_{\alpha f}l_f}{I_z} \end{bmatrix} \qquad B_2 = v_x \cdot \begin{bmatrix} 0 \\ -\frac{C_{\alpha f}l_f-C_{\alpha r}l_r}{mv_x} - v_x \\ 0 \\ -\frac{C_{\alpha f}l_f^2+C_{\alpha r}l_r^2}{I_z v_x} \end{bmatrix}$$

The state-space model presented above is used to design the trajectory tracking controllers, as the main objective is to minimize both the lateral error and the orientation error with respect to an imposed desired path that the vehicle has to follow.

2.5 Longitudinal model

The vehicle longitudinal model equations are obtained employing the dynamics of the DC motor. The two following equations are derived respectively from the Kirchhoff's law and from the dynamic equilibrium for the rotational motion.

$$\begin{cases} V_a = R_a i_a + L_a \frac{di_a}{dt} + E \\ C_m - C_r = J\frac{d\omega}{dt} + B\omega \end{cases} \tag{2.47}$$

where,

- V_a is the armature voltage;
- $R_a i_a$ is the voltage drop across the equivalent armature resistance;
- $L_a \frac{di_a}{dt}$ is the voltage drop due to equivalent inductance during transients. This contribution can be neglected when steady-state conditions are considered.
- E is the back electro-motive-force;
- C_m is the driving torque provided by the EM;
- C_r is the load torque due to resistance contributions;
- $J\frac{d\omega}{dt}$ is the product between motor's inertia and angular acceleration;
- $B\omega$ is the product between viscous friction and angular speed.

For a DC motor the following equations hold true:

$$\begin{cases} C_m = K_m \Phi i_a \\ E = K_e \Phi \omega \end{cases} \tag{2.48}$$

where,

- K_m is a constant that depends on the motor characteristics;
- Φ is the excitation flux;
- ω is the motor angular velocity;
- K_e is a constant that depends on the motor characteristics.

Combining 2.47, 2.48 and considering that $K_t = K_m \Phi$ and $K_v = K_e \Phi$ it is possible to write:

$$\begin{cases} i_a = \frac{V_a - K_v \omega}{R_a} \\ K_t i_a - C_r = J \frac{d\omega}{dt} + B\omega \end{cases} \tag{2.49}$$

Substituting the above expression of armature current i_a in the second equation we get:

$$K_t \frac{V_a - K_v \omega}{R_a} - C_r = J \frac{d\omega}{dt} + B\omega \tag{2.50}$$

and rewriting the equation:

$$\dot{\omega} = \frac{1}{J}\left(\frac{K_t}{R_a}(V_a - K_v \omega) - B\omega - C_r\right) \tag{2.51}$$

Equation 2.51 describes the dynamics of the EM. Nevertheless, if we consider small vehicle side-slip angle and we assume zero slip at the ground contact point, this differential equation can be used to describe the longitudinal dynamics of the QCar by introducing the transmission ratio τ:

$$v_{wh} = \frac{1}{\tau} \cdot \omega r_{wh} \tag{2.52}$$

Equation 2.51 can be rewritten as:

$$\dot{\omega} = \frac{K_t}{J \cdot R_a} V_a + \left(\frac{K_t K_v}{J R_a} - B\right)\omega - \frac{C_r}{J} \tag{2.53}$$

and, equivalently:

$$\dot{\omega} = P_1 V_a - P_2 \omega - P_3 \tag{2.54}$$

where

$$\begin{cases} P_1 = \frac{K_t}{J \cdot R_a} \\ P_2 = \frac{K_t K_v}{J R_a} - B \\ P_3 = \frac{C_r}{J} \end{cases} \tag{2.55}$$

Chapter 3

Reference trajectories generation

This chapter aims to present the procedure employed to generate the reference curvature ρ used as reference signal in the path tracking controllers. In this work, five different trajectories are taken into account: U-shaped trajectory, S-shaped trajectory, Circular-shaped trajectory, Eight-shaped trajectory and Obstacle avoidance trajectory.
All the reference trajectories have been designed such that they can be properly performed in the experimental tests' track present in the laboratory, whose plan is shown in figure 3.1.

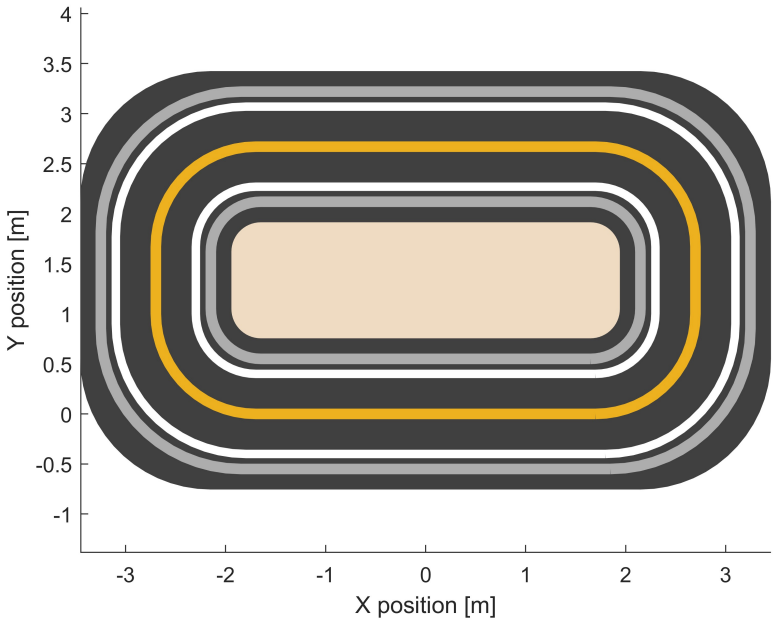


Figure 3.1: Laboratory tests track plan

3.1 U-shaped trajectory

The U-shaped trajectory, which starts in the point of coordinates (0,0), is built by connecting two straight horizontal segments with a semi-circular arc. The numerical values presented below are provided in meters.

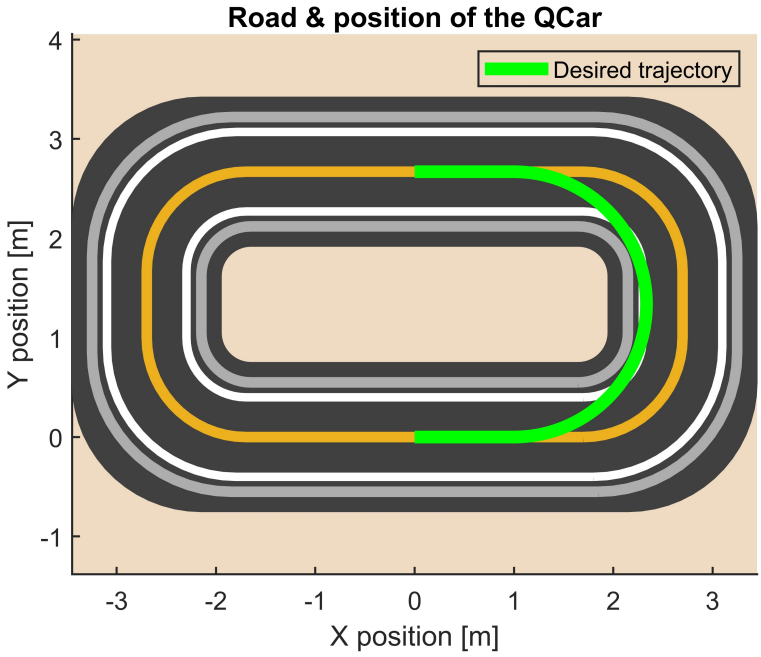


Figure 3.2: U-shaped trajectory on the test track

Specifically, the measurements of the path are reported below.

- Straight segment starting at (X,Y) = (0,0), ending at (X,Y) = (1, 0);
- semi-circumference with radius r = 1.335 starting at (X,Y) = (1,0) and ending at (X,Y) = (1, 2.67);
- straight segment starting at (X,Y) = (1,2.67) and ending at (X,Y) = (0,2.67).

These three segments are combined together to provide the reference coordinates vectors, as shown in the fragment of code below.

```
%  Initial straight segment
y_in = zeros(1, 1000);
x_in = linspace(0,0.999, 1000);

% Semicircumference
r = 1.335;
theta = linspace(-pi/2, pi/2, 1000);
x = r*cos(theta) + 1 ; % x coordinates
y = r*sin(theta) + r; % y coordinates

% Final straight segment
y_fin = 2.67*ones(1, 1000);
```

```
x_fin = linspace(0.999, 0, 1000);

% Combined coordinates
xRef = [x_in, x, x_fin]';
yRef = [y_in, y, y_fin]';

% Coordinates vector
refpos = [xRef, yRef];
```

Listing 3.1: Computation of the U-shaped trajectory's reference coordinates

At this point, the points of the reference path are interpolated using a defined number of samples. Additionally, the reference heading angle is computed.

```
%% Distance computation
distancematrix = squareform(pdist(refpos));
distancesteps = zeros(length(refpos)-1,1);

% Distance between consecutive points
for i = 2:length(refpos)
    distancesteps(i-1,1) = distancematrix(i,i-1);
end

% Total distance
totalDistance = sum(distancesteps);

% Cumulative distance
distbp = cumsum([0; distancesteps]);

% Curvilinear abscissa
s = linspace(0, totalDistance, 300);

xRef2 = interp1(distbp,xRef,s,'pchip');
yRef2 = interp1(distbp,yRef,s,'pchip');
yRef2s = smooth(s,yRef2);
xRef2s = smooth(s,xRef2);

%% Calculate psi_ref
psiRef = zeros(length(s),1);
for i = 2:length(s)
    psiRef(i,1) = atan2d((yRef2s(i) - yRef2s(i-1)),...
            (xRef2s(i) - xRef2s(i-1)));
end
psiRefs = smooth(s, psiRef);
```

Listing 3.2: Computation of the U-shaped trajectory's smoothed reference coordinates and heading angle

In the end, the reference curvature is computed as the inverse of the curvature radius [4]:

$$\rho(s) = \frac{1}{R} = \frac{x'y'' - x''y'}{(x'^2 + y'^2)^{3/2}} \tag{3.1}$$

where:

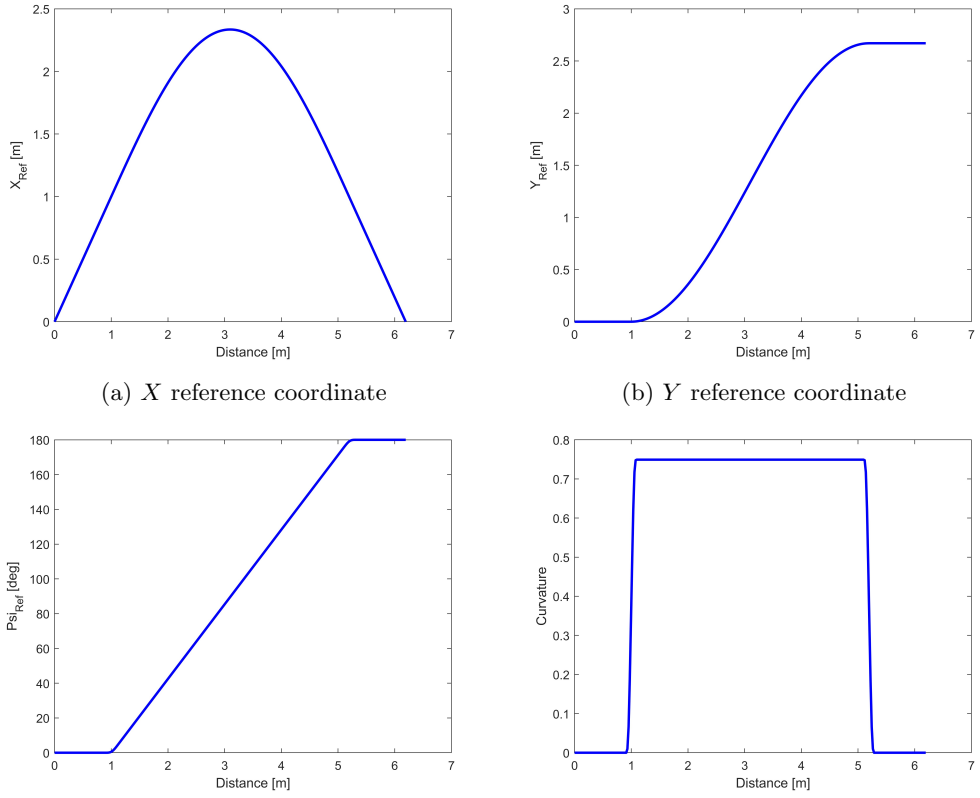
$$x' = \frac{dx}{ds} \tag{3.2}$$

$$x'' = \frac{dx'}{ds} \tag{3.3}$$

The fragment of code employed to compute the path curvature is reported below, as well as the reference variables of the U-shaped trajectory.

```
% Calculate curvature vector through curvature function
curvature = getCurvature(xRef2s, yRef2s);

% Curvature Function

function curvature = getCurvature(xRef,yRef)
% Calculate gradient by the gradient of the X and Y vectors
DX = gradient(xRef);
D2X = gradient(DX);
DY = gradient(yRef);
D2Y = gradient(DY);
curvature = (DX.*D2Y - DY.*D2X) ./(DX.^2+DY.^2).^(3/2);
end
```

Listing 3.3: Computation of the U-shaped trajectory's path curvature



(a) X reference coordinate

(b) Y reference coordinate

(c) Reference heading angle ψ_{Ref}

(d) Reference curvature ρ

Figure 3.3: Reference pose and curvature for U trajectory

3.2 S-shaped trajectory

The procedure used to build the S-shaped trajectory is the same to the one presented above for the U-shaped trajectory. It is based on a first horizontal segment, three adjacent semi-circumferences and a final vertical segment. Specifically, the measurements in meters of these trajectory sections are:

- straight segment starting at (X,Y) = (0,0) and ending at (X,Y) = (1.334, 0);
- a quarter of circumference with radius r = 1.335 starting at (X,Y) = (1.335,0) and ending at (X,Y) = (0, 1.335);
- semi-circumference with radius r = 1.335 starting at (X,Y) = (0, 1.335) and ending at (X,Y) = (-2.67, 1.335);
- semi-circumference with radius r = 1.335 starting at (X,Y) = (0, 1.335) and ending at (X,Y) = (-2.335, 1.335);
- straight segment starting at (X,Y) = (-2.67, 1.335) and ending at (X,Y) = (-2.67, 2.335).

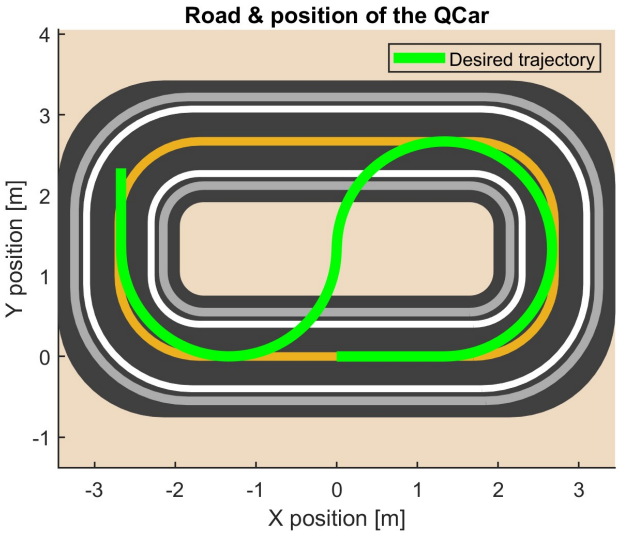


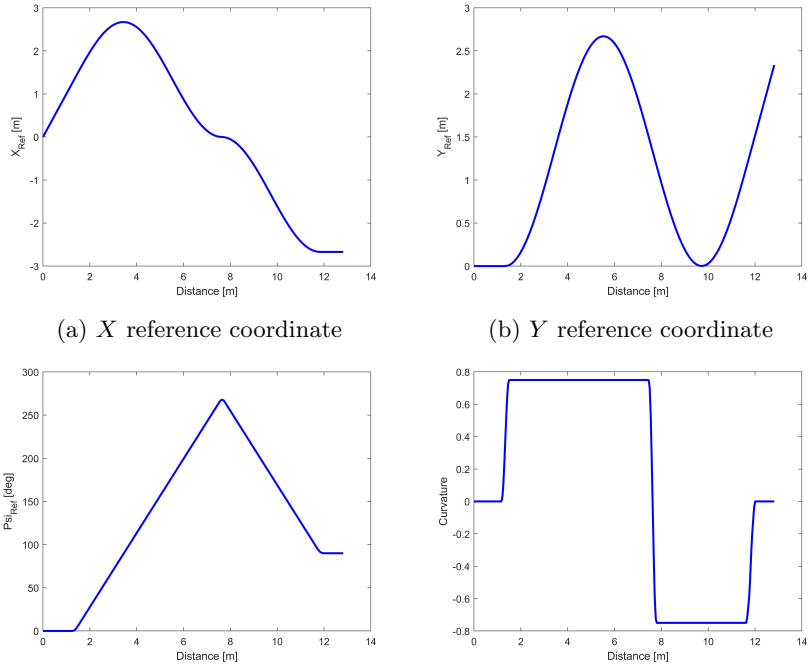
Figure 3.4: S-shaped trajectory on the test track

As before, the code used to compute the reference coordinates is presented. Additionally, the code employed to smooth the coordinates vector and to calculate the reference heading angle and the path curvature is always the same and for this reason it's not shown.

```
% Initial horizontal segment
y_in = zeros(1, 1000);
x_in = linspace(0, 1.334, 1000);

% First quarter of circumference
r = 1.335;
theta0 = linspace(-pi/2, 0, 1000);
x0 = r*cos(theta0) + 1.335 ;
```

```
y0 = r*sin(theta0) + r;

% Second semicirfumference
theta1 = linspace(0.001, pi, 1000);
x1 = r*cos(theta1) + 1.335;
y1 = r*sin(theta1) + r;

% Third semicircumference
theta2 = linspace(0.001, pi, 1000);
x2 = r*cos(theta2) - 1.335;
y2 = -r*sin(theta2) + r;

% Final vertical line
y_fin = linspace(r+0.001, r+1, 1000);
x_fin = -2*r*ones(1, 1000)- 0.001;

% Combined coordinates
xRef = [x_in, x0, x1, x2, x_fin]';
yRef = [y_in, y0, y1, y2, y_fin]';

% Coordinates vector
refpos = [xRef, yRef];
```

Listing 3.4: Computation of the S-shaped trajectory's reference coordinates



(a) X reference coordinate

(b) Y reference coordinate

(c) Reference heading angle ψ_{Ref}

(d) Reference curvature ρ

Figure 3.5: Reference pose and curvature for S trajectory

3.3 Circular trajectory

The circular trajectory is built by combining an initial horizontal segment, a circumference and final horizontal section. The measurements of these elements are:

- Horizontal segment starting at (X,Y) = (0,0) and ending at (X,Y) = (1.535, 0);
- circumference with radius r = 1.335 centered in (X,Y) = (1.535, 1,335) starting and ending at (X,Y) = (1.535,0)
- Horizontal segment starting at (X,Y)=(1.535, 0) and ending at (X,Y) = (2.5, 0).

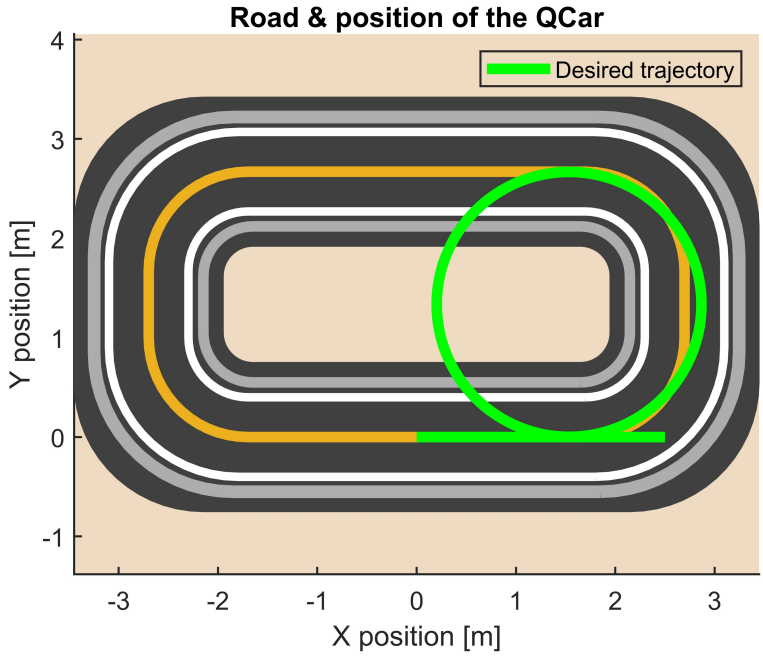


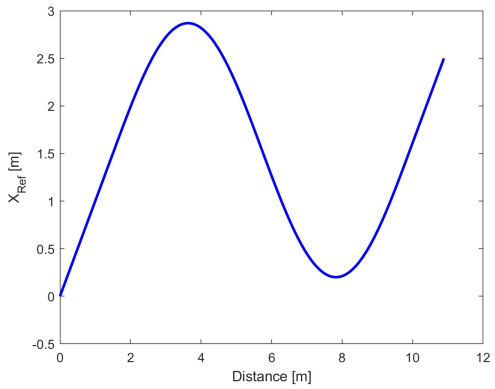
Figure 3.6: Circular trajectory on the test track

The code used to compute the reference coordinates is presented. As usual, the code employed to smooth the coordinates vector and to calculate the reference heading angle and the path curvature is always the same and for this reason it's not shown.

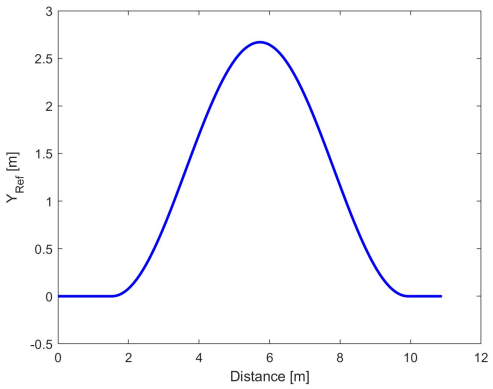
```
% Initial horizontal segment
y_in = zeros(1, 1000);
x_in = linspace(0,r+0.199, 1000);

% Circumference
r = 1.335;
theta = linspace(-pi/2, +3*pi/2-0.001, 1000);
x = r*cos(theta) + r + 0.2;
y = r*sin(theta) + r;

% Final horizontal segment
```

```
y_fin = zeros(1, 1000);
x_fin = linspace(x(end)+0.001, 2.5, 1000);

% Combined coordinates
xRef = [x_in, x, x_fin]';
yRef = [y_in, y, y_fin]';

% Coordinates vector
refpos =[xRef, yRef];
```

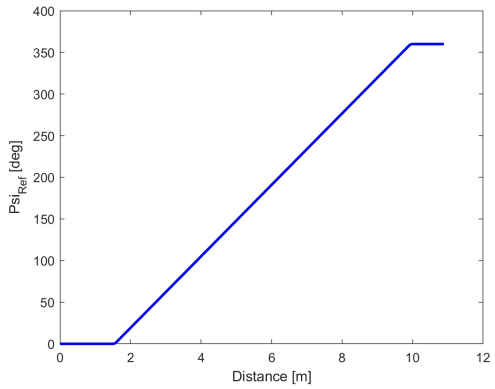
Listing 3.5: Computation of the circular trajectory's reference coordinates



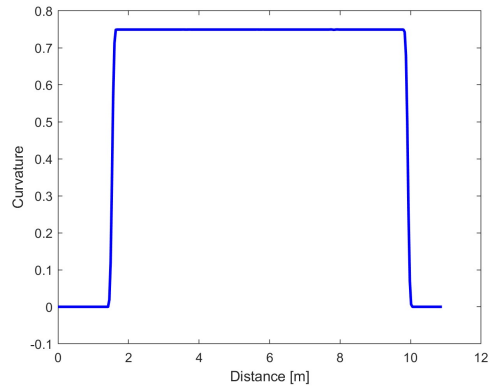
(a) X reference coordinate



(b) Y reference coordinate



(c) Reference heading angle ψ_{Ref}



(d) Reference curvature ρ

Figure 3.7: Reference pose and curvature for circular trajectory

3.4 Eight-shaped trajectory

The eight-shaped trajectory is built by combining an initial horizontal segment, a series of semi-circumferences and final horizontal segment. As usual, the measurements of all the sections are reported below in meters.

- First horizontal segment starting at (X,Y) = (0,0) and ending at (X,Y) = (1.335, 0);
- quarter of circumference with radius r = 1.335 centered in (X,Y) = (1.335, 1.335) starting at (X,Y) = (1.335, 0) and ending at (X,Y) = (2.67, 1.335);
- semi-circumference with radius r = 1.335 centered in X,Y) = (1.335, 1.335) starting at (X,Y) = (2.67, 1.335) and ending at (X,Y) = (0, 1.335);
- semi-circumference with radius r = 1.335 centered in (X,Y) = (-1.335, 1.335) starting at (X,Y) = (0, 1.335) and ending at (X,Y) = (-2.67, 1.335);
- semi-circumference with radius r = 1.335 centered in (X,Y) = (-1.335, 1.335) starting at (X,Y) = (-2.67, 1.335) and ending at (X,Y) = (1.335, 0);
- quarter of circumference with radius r = 1.335 centered in (X,Y) = (1.335, 1.335) starting at (X,Y) = (0, 1.335) and ending at (X,Y) = (1.335, 0);
- final straight segment starting at (X,Y) = (1.335, 0) and ending at (X,Y) = (2.5, 0).

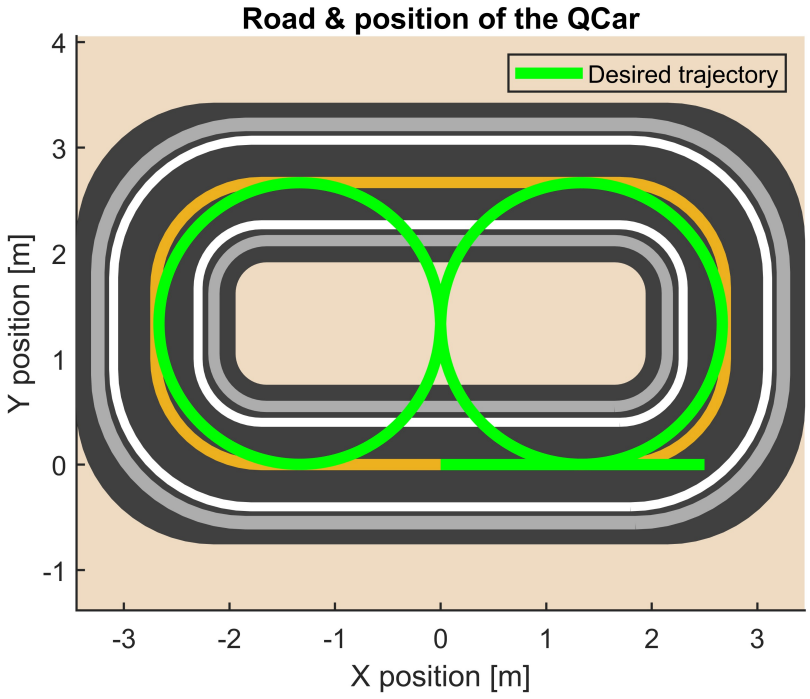
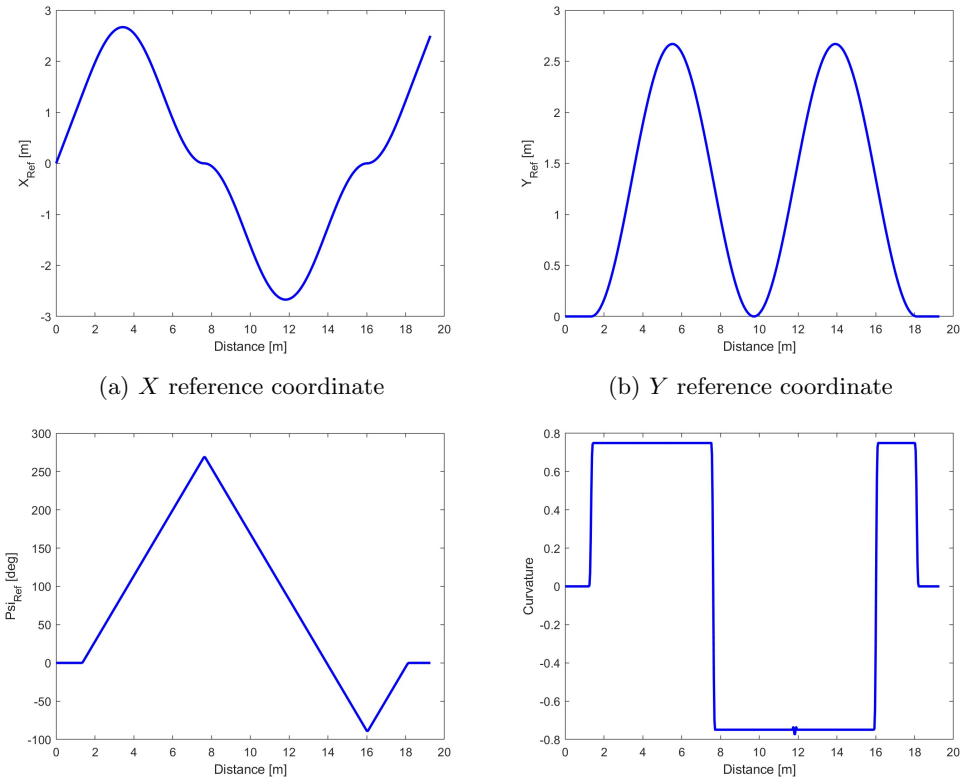


Figure 3.8: Eight-shaped trajectory on the test track

The code used to compute the reference coordinates is presented. As usual, the code employed to smooth the coordinates vector and to calculate the reference heading angle and the path curvature is always the same and for this reason it's not shown.

```
% Horizontal segment
y_in = zeros(1, 1000);
x_in = linspace(0, 1.349, 1000);

% Quarter of circumference
r = 1.335;
theta0 = linspace(-pi/2, 0, 1000);
x0 = r*cos(theta0) + r ;
y0 = r*sin(theta0) + r;

% Semicirfumference
theta1 = linspace(0.001, pi, 1000);
x1 = r*cos(theta1) + r;
y1 = r*sin(theta1) + r;

% Semicircumference
theta2 = linspace(0.001, pi, 1000);
x2 = r*cos(theta2) - r;
y2 = -r*sin(theta2) + r;

% Semicircumference
theta3 = linspace(pi+0.001, 0.001, 1000);
x3 = r*cos(theta3) - r;
y3 = r*sin(theta3) + r;

% Quarter of semicircumference
theta4 = linspace(pi, pi/2, 1000);
x4 = r*cos(theta4) + r;
y4 = -r*sin(theta4) + r;

% Horizontal segment
y_fin = zeros(1, 1000);
x_fin = linspace(r+0.001, 2.5, 1000);

% Combined coordinates
xRef = [x_in, x0, x1, x2, x3, x4, x_fin]';
yRef = [y_in, y0, y1, y2, y3, y4, y_fin]';

% Coordinates vector
refpos = [xRef, yRef];
```

Listing 3.6: Computation of the eight trajectory's reference coordinates



(a) X reference coordinate

(b) Y reference coordinate

(c) Reference heading angle ψ_{Ref}

(d) Reference curvature ρ

Figure 3.9: Reference pose and curvature for eight trajectory

3.5 Obstacle-avoidance manoeuvre

To properly build the obstacle-avoidance manoeuvre, the international standard ISO3888 was taken as reference [5]. Nevertheless, the shape of the trajectory is properly adapted by accounting for the dimension of the QCar used for the experimental tests and the dimensions of the test track. Specifically, since the QCar is in scale 1:10, the measures of the obstacle-avoidance trajectory pointed out in the standard are scaled by a factor of 10 as well.

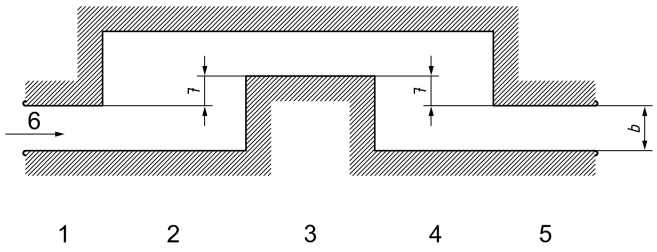


Figure 3.10: Obstacle avoidance trajectory's sections from ISO3888 standard.

The figure 3.10 shows the different parts of the trajectory according to the normative. The dimensions of all the sections are reported in the following table.

Section	**Length**	**Lane offset**	**Width b**
1	0,4	-	1,1 · vehicle width + 0,25
2	1,35	-	1,1 · vehicle width + 0,25
3	1,1	1	vehicle width + 1
4	1,25	-	vehicle width + 1
5	0,12	-	1,3 · vehicle width + 0,25, but not less than 0,3

Table 3.1: Obstacle avoidance trajectory dimensions in meters

Additionally, the trajectory reported in figure 3.10 is completed with the remaining sections listed below in order to preserve the QCar starting point at the center of the track's lower straight section, where the first scan of the LiDar allows an optimal detection of the surrounding environment.

- Horizontal segment starting at (X,Y) = (0,0) and ending at (X,Y) = (1,0);
- Semi-circumference with radius r = 1.335 starting at (X,Y) = (1,0) and ending at (X,Y) = (1, 2.67).

The final trajectory is shown in figure 3.11

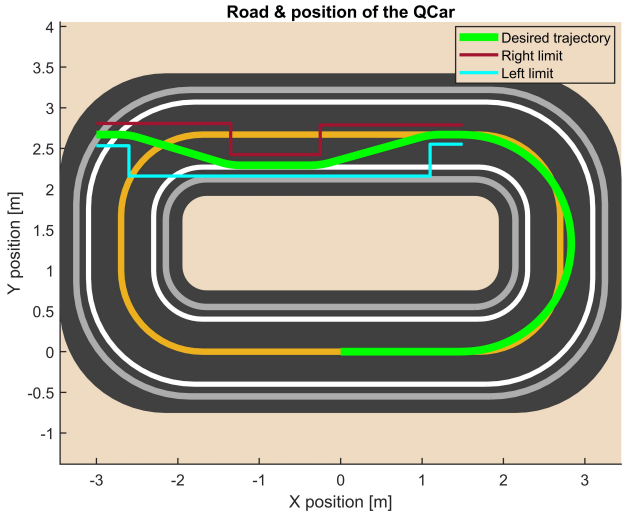


Figure 3.11: Obstacle-avoidance trajectory on the test track

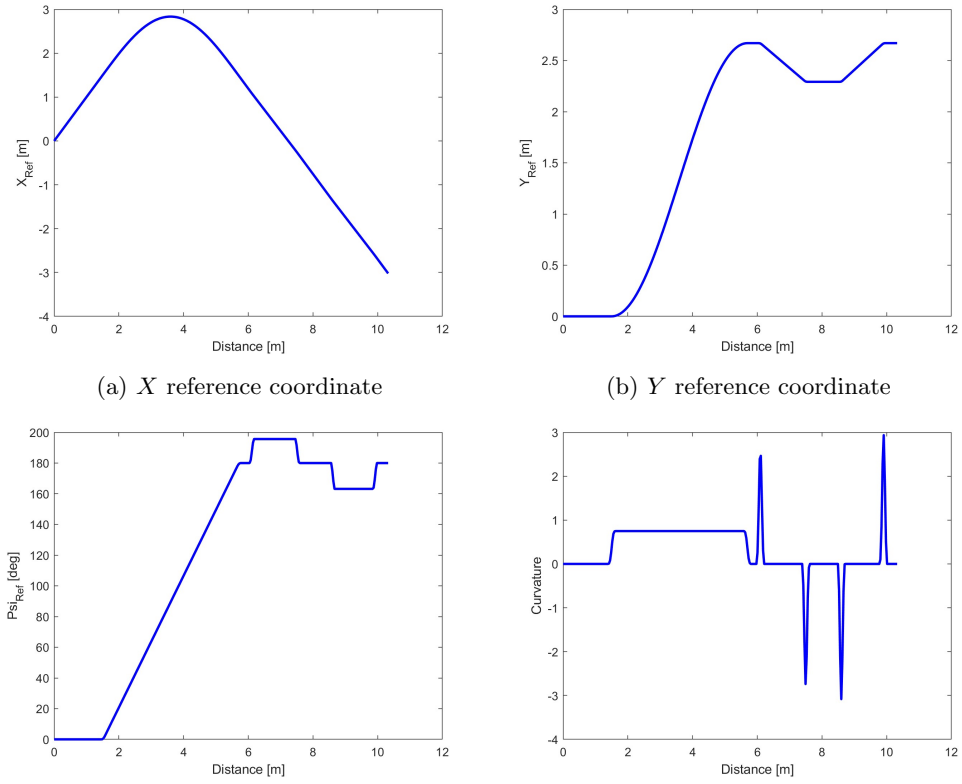
The code used for the generation of the trajectory is presented below.

```
% First initial segment
interpPoints = 1000;
y_in = zeros(1, 1000);
x_in = linspace(0,1.499, interpPoints);
```

```
5
6  % Semicircumference
7  r = 1.335;
8  theta = linspace(-pi/2, pi/2, interpPoints);
9  x = r*cos(theta) + 1.5 ;
10 y = r*sin(theta) + r;
11
12 %% Obstacle avoidance trajectory
13 s_init = 0;
14 lane_offset = 0.1;
15
16 % Test limit definition
17 vehicle_width = 0.192;
18 x_init = x(end) - 0.001;
19 x_1 = x_init - 0.4;
20 x_2 = x_1 - 1.35;
21 x_3 = x_2 - 1.1;
22 x_4 = x_3 - 1.25;
23 x_5 = x_4 - 0.3;
24 x_end = x_5 - 0.12;
25
26 % Right limit
27 reference_path_tmp.x_right = [x_init;x_init-10^-10;x_1 ;x_1-10^-10;
28                               x_2; x_2-10^-10;x_3; x_3-10^-10; x_4;
29                               x_4-10^-10; x_5; x_end];
30 y_right_1 = y(end) + (1.1*vehicle_width+0.025)/2;
31 y_right_2 = y_right_1;
32 y_right_3 = y(end) - lane_offset - ((vehicle_width+0.1)/2);
33 y_right_4 = y(end) + (0.3)/2;
34 y_right_5 = y_right_4;
35 reference_path_tmp.y_right = [y_right_1;y_right_1;y_right_1;
36                               y_right_2; y_right_2; y_right_3;
37                               y_right_3;y_right_4; y_right_4;
38                         y_right_5; y_right_5;y_right_5];
39
40 % Left limit
41 reference_path_tmp.x_left = [x_init;x_init-10^-10;x_1 ;x_1-10^-10;
42                              x_2;x_2-10^-10; x_3; x_3-10^-10; x_4;
43                              x_4-10^-10; x_5; x_end];
44 y_left_1 = y(end) - (1.1*vehicle_width+0.025)/2;
45 y_left_2 = y_left_1 - lane_offset - (vehicle_width+0.1);
46 y_left_3 = y_left_2;
47 y_left_4 = y_left_2;
48 y_left_5 = y(end) - (0.3)/2;
49 reference_path_tmp.y_left=[y_left_1;y_left_1;y_left_1;y_left_2;
50                            y_left_2;y_left_3;y_left_3;y_left_4;
51                            y_left_4;y_left_5;y_left_5;y_left_5];
52
53 % Mid line
54 reference_path_tmp.x_mid = reference_path_tmp.x_left;
55 y_mid_1 = mean([y_left_1 y_right_1]);
56 y_mid_2 = mean([y_left_2 y_right_2]);
57 y_mid_3 = mean([y_left_3 y_right_3]);
58 y_mid_4 = mean([y_left_4 y_right_4]);
59 y_mid_5 = mean([y_left_5 y_right_5]);
60 reference_path_tmp.y_mid = [y_mid_1;y_mid_1;y_mid_1;y_mid_1;
61                             y_mid_3;y_mid_3;y_mid_3;y_mid_3;
62                             y_mid_5;y_mid_5;y_mid_5;y_mid_5];
63
64 % Calculate reference vectors with the same length
65 reference_path_tmp.x_ref=reference_path_tmp.x_mid;
66 reference_path_tmp.y_ref=reference_path_tmp.y_mid;
```

```
67
68 % Interpolate the original vectors
69 x_interp = interp1(1:numel(reference_path_tmp.x_ref),...
70           reference_path_tmp.x_ref,...
71      linspace(1, numel(reference_path_tmp.x_ref),...
72      interpPoints));
73 y_interp = interp1(1:numel(reference_path_tmp.y_ref),...
74           reference_path_tmp.y_ref,...
75      linspace(1, numel(reference_path_tmp.y_ref),
76           interpPoints));
77
78 % Combined coordinates
79 xRef = [x_in, x, x_interp]';
80 yRef = [y_in, y, y_interp]';
81
82 % Coordinates vector
83 refpos = [xRef, yRef];
```

Listing 3.7: Computation of the obstacle avoidance trajectory's reference coordinates



(a) X reference coordinate

(b) Y reference coordinate

(c) Reference heading angle ψ_{Ref}

(d) Reference curvature ρ

Figure 3.12: Reference pose and curvature for obstacle-avoidance trajectory

Chapter 4

Kalman filters

The use of Kalman filters for estimating a vehicle's state variables has drawn a lot of interest lately. For many applications, including autonomous driving, navigation, and vehicle control, the knowledge of vehicle's state variables are essential. The vehicle's dynamics uncertainties, sensor noise, and measurement mistakes make it difficult to estimate these variables precisely. An overview of the use of Kalman filters in the context of estimating vehicle status is provided in this work.
Based on a set of noisy observations, Kalman filters offer a recursive technique to estimate the system's state. Prediction and update are the two phases of the filter's operation. Based on the prior estimate and the plant's dynamics, the filter predicts the state variables in the first stage using a model of the vehicle. An uncertainty covariance matrix that reflects the prediction's level of confidence is linked to the predicted state.
In the update stage, the filter employs the measurements from sensors in order to correct the predicted state. By generating a weighted average, where the weights are based on the uncertainties of both the predicted state variables and the measurements, the Kalman filter offers a better illustration of the actual state.
In this work, KFs are employed to correct the vehicle's pose, velocity and acceleration respectively provided by lidar, encoder and IMU. Indeed, despite the localization system produces precise measurement of the vehicle's pose, often happens that the measurements differ considerably from the actual pose. These "spikes" are due to inaccurate matching which therefore leads to inappropriate measured data that have to be corrected.

4.1 Kalman Filters algorithm

Consider the discrete-time nonlinear system [6]:

$$\begin{cases} x_{k+1} = f(x_k, u_k) + d_k \\ y_k = h(x_k) + d_k^y \end{cases}$$

Suppose that x_k, d_k, d_k^y are unknown and y_k, u_k are measured. The goal is to obtain an accurate estimate $\hat{x_k}$ of x_k from current and past measurements of y_k and u_k.

Notation part 1:

- x_k is the state;

- u_k is the input;
- y_k is the output;
- d_k is the process disturbance;
- d_k^y is the measurement noise.

The elements x_k, d_k, d_k^y are supposed to be unknown whereas y_k, u_k to be measured. The aim is to have an estimate $\hat{x_k}$ of the state x_k from current and past measurements of y_k and u_k.

Notation part 2:

- x_k^p: prediction of x_k (computed at step k-1);
- $\hat{x}_k$: estimate of x_k (computed at step k);
- $\delta x_k = x_k - x_k^p$: state prediction error;
- $\tilde{x}_k$=$x_k - \hat{x}_k$: state estimation error;
- $P_k^p = E[\delta x_k \delta x_k^T]$: covariance matrix of the prediction error;
- $P_k = E[\tilde{x}_k \tilde{x}_k^T]$: covariance matrix of the estimation error;
- $Q = E[d_k d_k^T]$: covariance matrix of process noise d_k;
- $R = E[d_k^y (d_k^y)^T]$: covariance matrix of measurement noise d_k^y.

Assumptions:

- The noises are independent, identically distribuited and white:
  - Zero-mean: $\begin{cases} E[d_k] = 0 \\ \\ E[d_k^y] = 0 \end{cases}$;
  - Bounded variance: $\begin{cases} \text{var}(d_k) < \infty \\ \\ \text{var}(d_k^y) < \infty \end{cases}$;
- Noise cross-uncorrelation: $E[d_k (d_k^y)^T] = 0$;
- Input-noise cross-uncorrelation: $E[d_k u_k^T] = 0$, $E[d_k^y u_k^T] = 0$;
- The system is globally observable.

The algorithm, as previously mentioned, is composed by two steps:

1. **Prediction**: at time t_{k-1}, compute a prediction x_k^p of the state x_k using the prediction model:

$$x_k^p = f(\hat{x}_{k-1}, u_{k-1})$$

$$P_k^p = F_{k-1} P_{k-1} F_{k-1}^T + Q$$

2. **Update**: at time t_k, the prediction x_k^p is corrected using the output y_k, providing more accurate estimate:

$$S_k = H_k P_k^p H_k^T + R$$
$$K_k = P_k^p H_k^T S_k^{-1}$$
$$\delta y_k = y_k - h(x_k^p)$$
$$\hat{x}_k = x_k^p + K_k \delta y_k$$
$$P_k = (I - K_k H_k) P_k^p$$

In the linear case K_k is computed with the aim to minimize the variance of the estimation error norm, while in the non-linear case is computed from F_k and H_k matrices, in turn obtained linearizing the system around the current estimate:

$$F_k = \frac{\partial f}{\partial x}(\hat{x}_k, u_k)$$
$$H_k = \frac{\partial h}{\partial x}(\hat{x}_k)$$

For what concerns the initial conditions, the estimated initial state is typically set to $\hat{x}_0 = 0$ and the estimated initial covariance matrix is typically set to $P_0 = \mathcal{I}$.
In this work, two extended Kalman filters have been designed for estimating the vehicle state variables of interest that are employed for control purposes: a kinematic extended Kalman filter (KEKF) and a dynamic extended Kalman filter (DEKF). The details of both filters are presented in the following sections.

4.2 Kinematic Extended Kalman filter

The first Kalman filter which has been developed uses the equation of uniformly accelerated motion. By writing the equations of motion in the continuous time domain and applying the Euler discretization method, the discrete state-space representation of the considered plant is

$$\begin{cases} X(k+1) = X(k) + V_x(k)T_s + a_X(k)\frac{T_s^2}{2} \\ Y(k+1) = Y(k) + V_y(k)T_s + a_Y(k)\frac{T_s^2}{2} \\ V_x(k+1) = V_x(k) + a_X(k)T_s \\ V_y(k+1) = V_y(k) + a_Y(k)T_s \\ \psi(k+1) = \psi(k) + \dot{\psi}(k)T_s \end{cases} \tag{4.1}$$

where T_s is the sampling time. The longitudinal and lateral accelerations are expressed in the inertial reference frame; since the acceleration vector provided by the IMU is referred to the body reference frame, a rotation is needed to compute the accelerations used in the model equations:

$$\begin{bmatrix} a_X \\ a_Y \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} a_x \\ a_y \end{bmatrix} \tag{4.2}$$

where $a_x \equiv a_{x,IMU}$, $a_y \equiv a_{y,IMU}$.
By substituting 4.2 in 4.1, the following discrete time nonlinear model is obtained:

$$\begin{cases} X(k+1) = X(k) + V_x(k)T_s + (a_x \cos(\psi(k)) - a_y \sin(\psi(k)))\frac{T_s^2}{2} \\ Y(k+1) = Y(k) + V_y(k)T_s + (a_x \sin(\psi(k)) + a_y \cos(\psi(k)))\frac{T_s^2}{2} \\ V_x(k+1) = V_x(k) + (a_x \cos(\psi(k)) - a_y \sin(\psi(k)))T_s \\ V_y(k+1) = V_y(k) + (a_x \sin(\psi(k)) + a_y \cos(\psi(k)))T_s \\ \psi(k+1) = \psi(k) + \dot{\psi}(k)T_s \end{cases} \tag{4.3}$$

The state and the input vectors of the system are:

$$x(k) = \begin{bmatrix} X(k) & Y(k) & V_x(k) & V_y(k) & \psi(k) \end{bmatrix}^T, u = \begin{bmatrix} a_x(k) & a_y(k) & \dot{\psi}(k) \end{bmatrix}^T$$

As measurements, vehicle's pose (X, Y, ψ) provided by the lidar and vehicle's longitudinal velocity v_x computed starting from the motor rotational velocity provided by the encoder are employed to correct the predicted state. Instead, the IMU measurements, suitably rotated as indicated in (2), are used as model's input. Under the assumption of small velocity and vehicle side slip angle, it can be assumed that:

$$v = \sqrt{V_x^2 + V_y^2} \simeq v_x \tag{4.4}$$

Therefore, the output equations are:

$$\begin{cases} y_1(k) = X(k) \\ y_2(k) = Y(k) \\ y_3(k) = \psi(k) \\ y_4(k) = \sqrt{V_x^2 + V_y^2} \end{cases}$$

The filtered state vector is $\hat{x} = \begin{bmatrix} \hat{X} & \hat{Y} & \hat{V}_x & \hat{V}_y & \hat{\psi} \end{bmatrix}^T$.
Additionally, considering the following static equations:

$$\begin{cases} \begin{bmatrix} \hat{v}_x \\ \hat{v}_y \end{bmatrix} = \begin{bmatrix} \cos(\hat{\psi}) & -\sin(\hat{\psi}) \\ \sin(\hat{\psi}) & \cos(\hat{\psi}) \end{bmatrix}^T \begin{bmatrix} \hat{V}_x \\ \hat{V}_y \end{bmatrix} \\ \hat{v} = \sqrt{\hat{v}_x^2 + \hat{v}_y^2} \\ \hat{\beta} = \tan^{-1}(\frac{\hat{v}_y}{\hat{v}_x}) \end{cases}$$

the augmented filtered state vector $\hat{x}_a = \begin{bmatrix} \hat{X} & \hat{Y} & \hat{\psi} & \hat{v} & \hat{v}_x & \hat{v}_y & \hat{\beta} \end{bmatrix}^T$ is obtained.
An overview scheme of the KEKF is presented below.

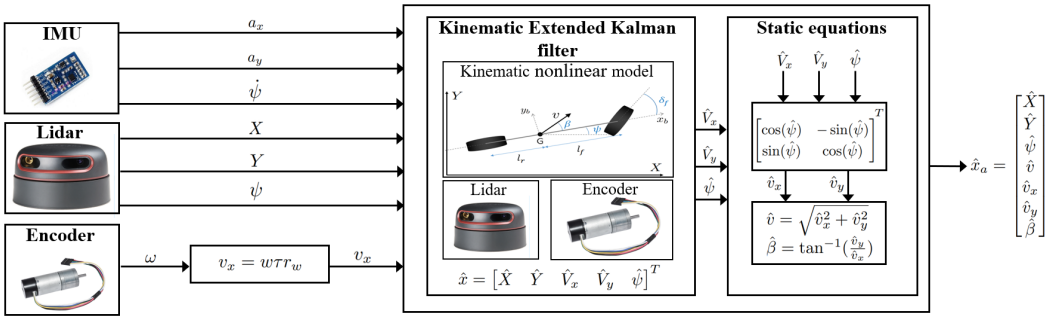


Figure 4.1: Kinematic Extended Kalman filter scheme

4.2.1 Dynamic Extended Kalman filter

In the dynamic extended Kalman filter (DEKF), the single track dynamic model, also known as bicycle model, is used with the assumption of small slip-angles in order to consider lateral tire force proportional to the tire slip angle. The equations that describe the time evolution of the lateral velocity and yaw rate are:

$$\begin{cases} \dot{v}_y = \frac{-(C_{\alpha f}+C_{\alpha r})}{mv_x} v_y - (v_x + \frac{(C_{\alpha f} l_f + C_{\alpha r} l_r)}{mv_x})\dot{\psi} + \frac{C_{\alpha f}}{m}\delta \\ \ddot{\psi} = \frac{-(C_{\alpha f} l_f - C_{\alpha r} l_r)}{I_z v_x} v_y - \frac{C_{\alpha f} l_f^2 + C_{\alpha r} l_r^2}{I_z v_x}\dot{\psi} + \frac{C_{\alpha f} l_f}{I_z}\delta \end{cases} \tag{4.5}$$

where $C_{\alpha f}$ and $C_{\alpha r}$ are respectively the equivalent cornering stiffness of front and rear axles. The time derivatives of the global coordinates X and Y associated with the centre of gravity and the yaw angle can be expressed as:

$$\begin{cases} \dot{X} = v_x \cos(\psi) - v_y \sin(\psi) \\ \dot{Y} = v_x \sin(\psi) + v_y \cos(\psi) \\ \dot{\psi} = r \end{cases} \tag{4.6}$$

where r is the yaw rate.
The differential equation that describes the electric motor dynamics is:

$$\dot{\omega} = P_1 V_a - P_2 \omega - P_3 \tag{4.7}$$

By considering the transmission ratio τ between the motor and the wheels, and the wheel radius r_w, if there is no wheel slip in the longitudinal direction of the vehicle it can be written:

$$v_x = \omega \tau r_w \tag{4.8}$$

So it can be obtained:

$$\dot{v}_x = \tau r_w (P_1 V_a - \frac{P_2}{\tau r_w} v_x - P_3) \tag{4.9}$$

By combining the equations written above and applying Euler discretization method, the complete model's equations are:

$$\begin{cases} X(k+1) = X(k) + (v_x(k)\cos(\psi(k)) - v_y(k)\sin(\psi(k)))T_s \\ Y(k+1) = Y(k) + (v_x(k)\sin(\psi(k)) + v_y(k)\cos(\psi(k)))T_s \\ \psi(k+1) = \psi(k) + \dot{\psi}(k)T_s \\ v_x(k+1) = v_x(k) + (\tau r_w(P_1 V_a(k) - \frac{P_2}{\tau r_w}v_x(k) - P_3))T_s \\ v_y(k+1) = v_y(k) + (\frac{-(C_{\alpha f}+C_{\alpha r})}{m v_x(k)}v_y(k) - (v_x(k) + \frac{(C_{\alpha f}l_f + C_{\alpha r}l_r)}{m v_x(k)})\dot{\psi}(k) + \frac{C_{\alpha f}}{m}\delta(k))T_s \\ \dot{\psi}(k+1) = \dot{\psi}(k) + (\frac{-(C_{\alpha f}l_f - C_{\alpha r}l_r}{I_z v_x(k)}v_y(k) - \frac{C_{\alpha f}l_f^2 + C_{\alpha r}l_r^2}{I_z v_x(k)}\dot{\psi}(k) + \frac{C_{\alpha f}l_f}{I_z}\delta(k))T_s \end{cases} \tag{4.10}$$

As measurements, vehicle pose provided by the lidar, longitudinal velocity computed starting from the measurement of motor speed provided by the encoder and yaw rate provided the IMU are employed. The output equations are:

$$\begin{cases} y_1(k) = X(k) \\ y_2(k) = Y(k) \\ y_3(k) = \psi(k) \\ y_4(k) = v_x(k) \\ y_5(k) = \dot{\psi}(k) \end{cases} \tag{4.11}$$

The filtered state vector is: $\hat{x} = \begin{bmatrix} \hat{X} & \hat{Y} & \hat{\psi} & \hat{v}_x & \hat{v}_y & \hat{\dot{\psi}} \end{bmatrix}^T$.
Additionally, considering the following static equations:

$$\begin{cases} \hat{v} = \sqrt{\hat{v}_x^2 + \hat{v}_y^2} \\ \hat{\beta} = \tan^{-1}(\frac{\hat{v}_y}{\hat{v}_x}) \end{cases} \tag{4.12}$$

the augmented filtered state vector $\hat{x}_a = \begin{bmatrix} \hat{X} & \hat{Y} & \hat{\psi} & \hat{v} & \hat{v}_x & \hat{v}_y & \hat{\beta} & \hat{\dot{\psi}} \end{bmatrix}^T$ is obtained.
An overview scheme of the DEKF is shown below.

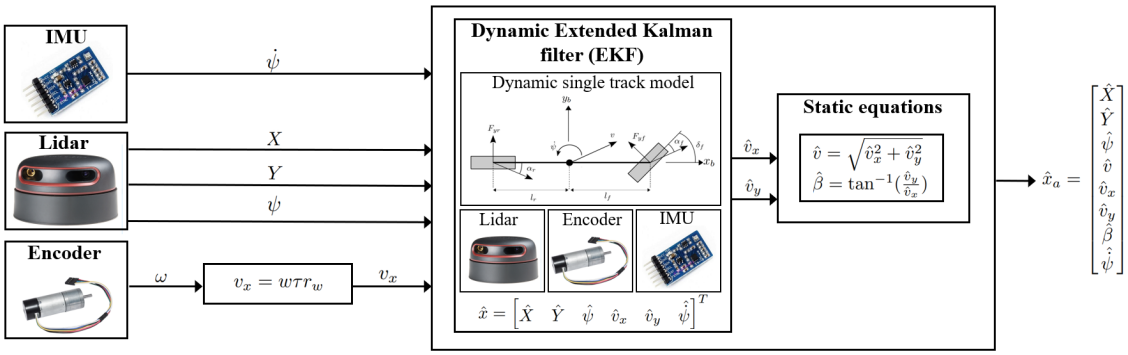


Figure 4.2: Dynamic Extended Kalman filter scheme

4.3 Simulation results

The goal of the simulation tests is to compare the two Kalman filters in the same environment, where the exact system and the statistical information about disturbances are available to the filters. Although this situation is unrealistic in experimental practice, it allows to realize a right comparison of the two ideally tuned filters with no model uncertainty affecting the results.
The output measurements are generated by summing white noises to the model's output vector y_m. The considered noises variances are presented below.

Signal	Variance	UoM
a_x	0.05	m^2/s^4
a_y	0.05	m^2/s^4
$\dot{\psi}$	0.005	rad^2/s^2
X	0.2	m^2
Y	0.01	m^2
ψ	0.05	rad^2
v_x	0.001	m^2/s^2

Table 4.1: White noises variances

The maneuver is simulated in open-loop imposing constant longitudinal velocity and a steering angle with sine-wave shape. In order to numerically quantify the goodness of the two filters performances, the root-mean-square errors between the filtered states and the ideal states coming out from the model are evaluated.

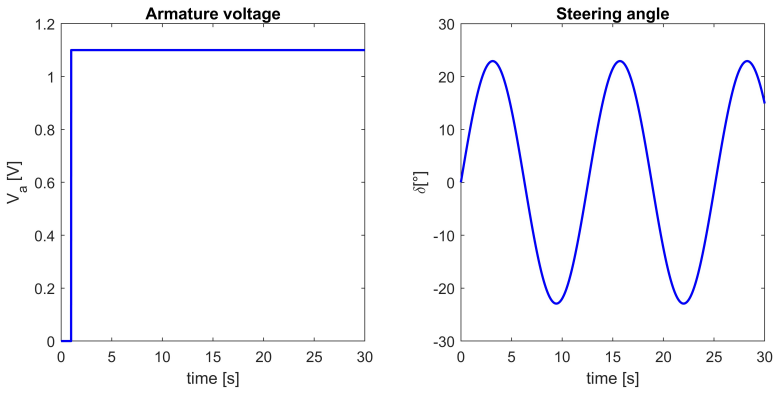


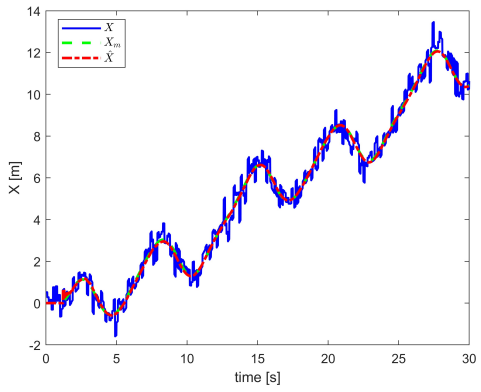
Figure 4.3: Model's input in simulation

4.3.1 Kinematic Extended Kalman Filter

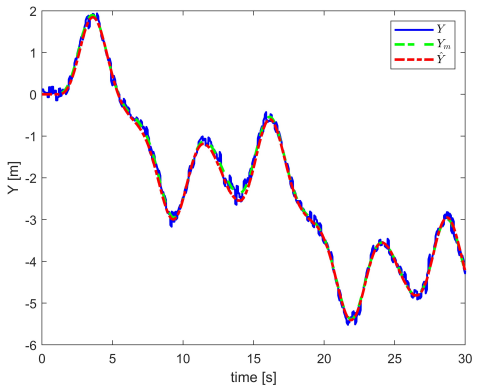
By using the covariance matrices Q and R presented in the following table, the simulation results of the KEKF are shown below.

Q	R
$10^{-7}\mathcal{I}_5$	diag(0.2,0.01,0.05,0.001)

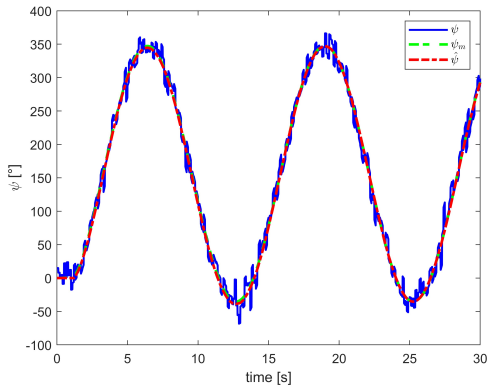
Table 4.2: Covariance matrices KEKF



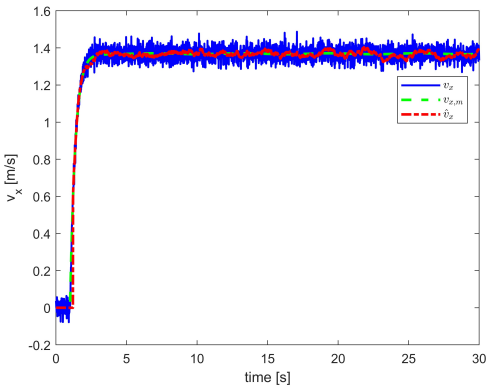
(a) X position



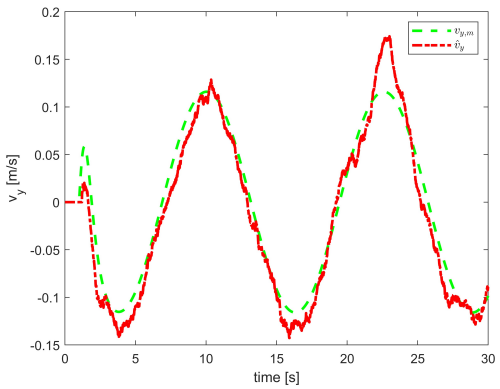
(b) Y position



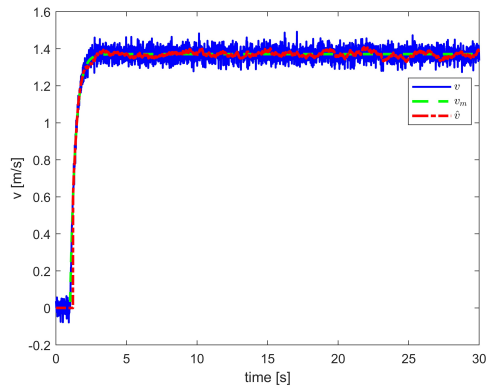
(c) Yaw angle



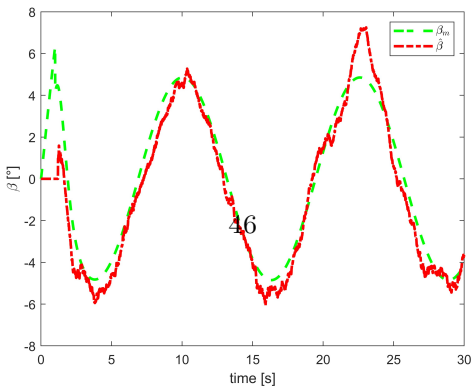
(d) Longitudinal velocity



(e) Lateral velocity



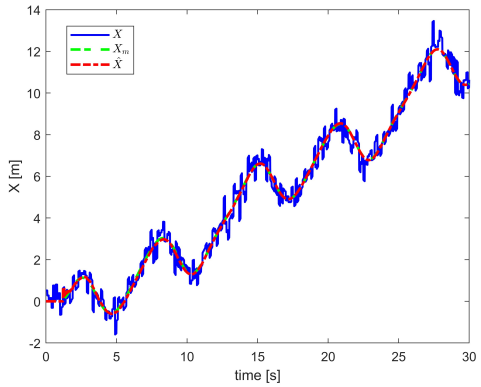
(f) Total velocity



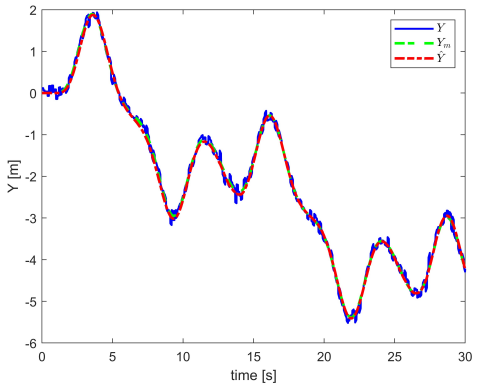
(g) Vehicle sideslip angle

4.3.2 Dynamic Extended Kalman Filter

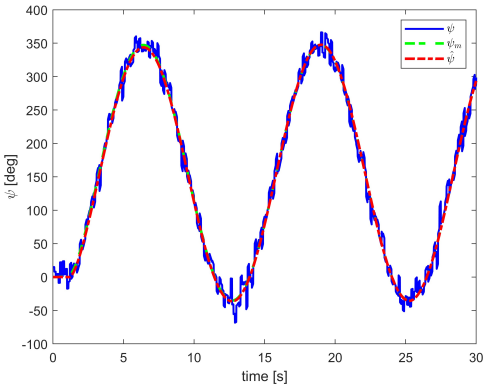
By using the covariance matrices Q and R presented in the following table, the simulation results of the first DEKF are shown below.



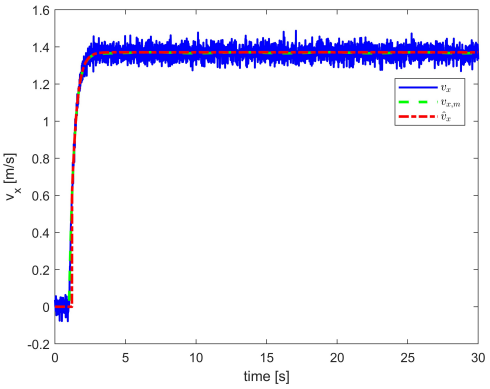
(a) X position



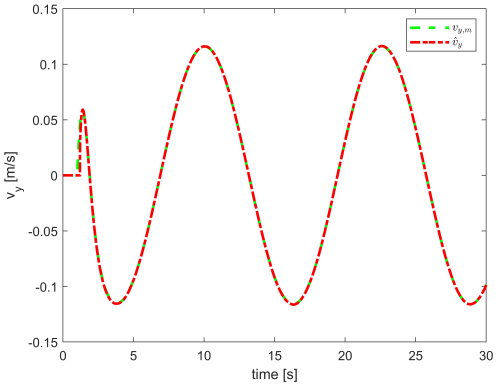
(b) Y position



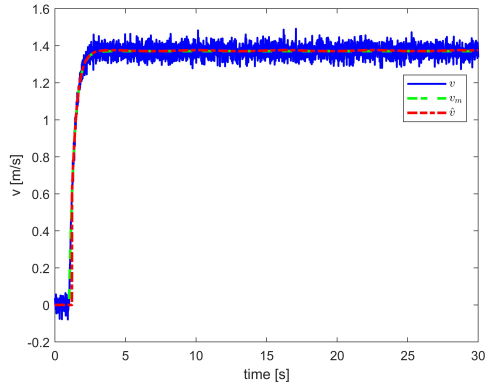
(c) Yaw angle



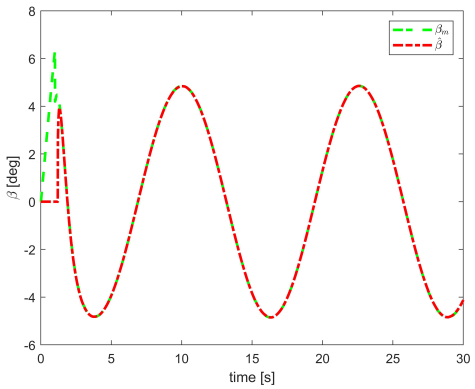
(d) Longitudinal velocity



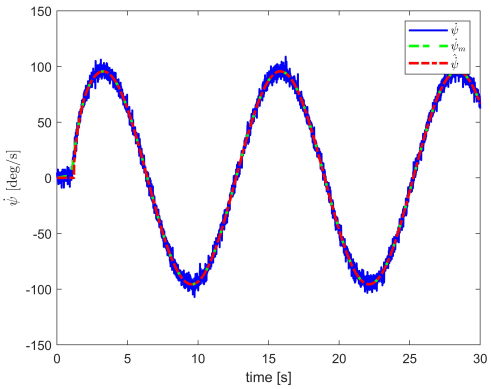
(e) Lateral velocity



(f) Total velocity



(a) Vehicle sideslip angle



(b) Yaw rate

Q	R
$10^{-7}\mathcal{I}_6$	diag(0.2,0.01,0.05,0.001,0.005)

Table 4.3: Covariance matrices DEKF

4.3.3 Comparison of simulation results

In order to compare the performance of both KEKF and DEKF, the root mean square errors between the model state variables and the filtered state variables are evaluated. The results are presented in the following table.

	KEKF	DEKF	UoM
RMSE$(\hat{X}, X_m)$	0.0604	0.0523	m
RMSE$(\hat{Y}, Y_m)$	0.0713	0.0460	m
RMSE$(\hat{\psi}, \psi_m)$	0.0367	0.0337	rad
RMSE$(\hat{v}_x, v_{x,m})$	0.0331	0.0303	m/s
RMSE$(\hat{v}_y, v_{y,m})$	0.0196	0.0024	m/s
RMSE$(\hat{v}, v_m)$	0.0331	0.0304	m/s
RMSE$(\hat{\beta}, \beta_m)$	0.0202	0.0135	rad
RMSE$(\hat{\dot{\psi}}, \dot{\psi}_m)$	-	0.0193	rad/s

Table 4.4: KPIs comparison

As expected, the DEKF leads to better results with respect to the KEKF that employs only a kinematic model of the car, especially for the estimation of the vehicle side-slip angle and equivalently of the vehicle lateral velocity.

4.4 Experimental results

The Kalman filters that have been designed are based on the measurements of the LiDar for the vehicle pose, the motor encoder for the longitudinal speed and the IMU for the yaw rate. The

LiDar equipped on the QCar provides a localization system developed by Quanser that measures the pose of the car based on a matching with some data (ranges and angles) saved in memory during an initial setup where the sensor scans the surrounding environment for few seconds. During this stage, the point in which the LiDar performs the scan is considered as origin of the inertial reference frame with respect to which the pose is expressed.
Although the localization system provides a position's precise measure, sometimes, in the case of a wide area,measurements diverge from the actual values of the vehicle pose. These "spikes" are caused by wrong matching that generate completely wrong values. The LiDar Simulink block

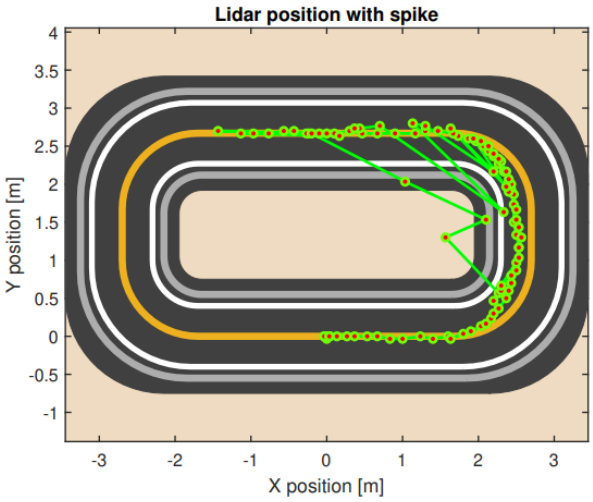


Figure 4.7: Pose provided by the LiDar with spikes

designed by Quanser provides a quality index of the measurement accuracy of the sensor named LiDar score that can reach a maximum value of 200. An high value of the LiDar score implies an accurate pose measurement provided by the sensor. So, the entries of the pose measurement noise's covariance matrix are time-varying and strongly depend on the score value. The following sigmoid functions are employed to define the entries of the LiDar covariance matrix:

$$\begin{cases} i = \frac{200}{|LiDarScore|} \\ R_{xx,0} = R_{yy,0} = 80 \cdot \tanh(\frac{i}{0.6} - 3) + 81 \\ R_{\psi\psi,0} = 9 \cdot \tanh(\frac{i}{0.7} - 3) + 9.3 \end{cases} \tag{4.13}$$

In this way, as the accuracy of the pose measurement increases the respective entry of the covariance matrix decreases in order to trust more the sensor data with respect to the model's prediction. However, in the presence of spikes, the LiDar score does not penalize the wrong measurement enough and this leads to bad performance due to one or more values of $R_{xx}, R_{yy}, R_{\psi\psi}$ still too high. In order to improve filters performance by further penalizing the spikes presence, the respective entry of the LiDar measurement noise's covariance matrix is multiplied by a factor $\alpha_X, \alpha_Y, \alpha_\psi > 1$ when a spike is detected respectively in the X, Y and ψ measurement. The

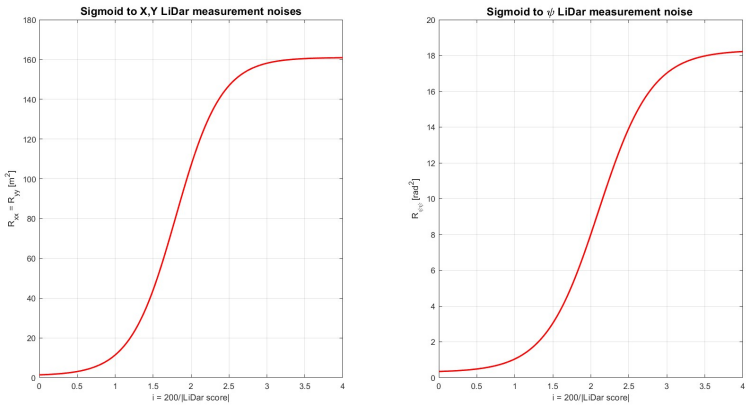


Figure 4.8: Sigmoid functions for LiDar measurement noises

condition that allows the spike identification is the following:

$$\begin{cases} |X(k) - X(k-1)| > 0.1 \rightarrow SPIKE_X \rightarrow R_{xx} = \alpha_X R_{xx,0} \\ |Y(k) - Y(k-1)| > 0.1 \rightarrow SPIKE_Y \rightarrow R_{yy} = \alpha_Y R_{yy,0} \\ |\psi(k) - \psi(k-1)| > 0.05 \rightarrow SPIKE_\psi \rightarrow R_{\psi\psi} = \alpha_\psi R_{\psi\psi,0} \end{cases} \tag{4.14}$$

The values of the multiplied factors for the two filters are presented below.

Multiplied factors	KEKF	DEKF
α_X	200	20
α_Y	1000	1000
α_ψ	10000	10000

Table 4.5: Multiplied factors for spike management

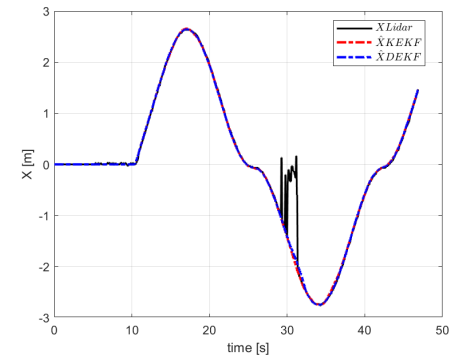
For the motor encoder and IMU measurement noises and the process disturbances covariance entries, the values are shown in the following table.

	KEKF	DEKF
R_{v_x,v_x}	0.025	0.025
$R_{\dot{\psi},\dot{\psi}}$	-	0.04
Q	diag(1,0.6,100,60,80)	diag(0.05,0.05,0.0628,0.01,0.0001,0.1257)

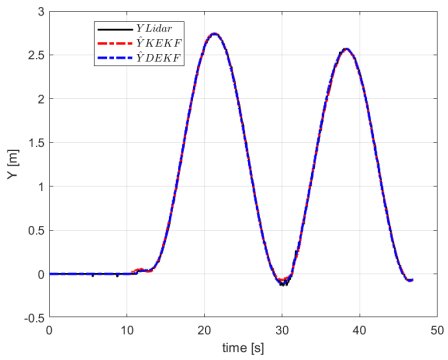
Table 4.6: Covariance matrix's entries for encoder and IMU measurement noise and process disturbances

The experimental test is performed in closed-loop by using one of the benchmark path tracking controllers presented in the following chapters, imposing a eight trajectory's tracking. With the numerical values presented above for matrices R and Q, the following experimental results are obtained. As already observed in the simulation results, also through experimental tests it can

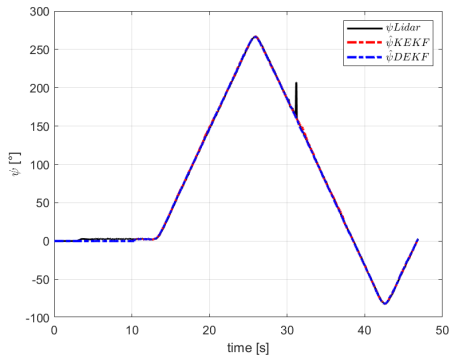
be seen that the performance of the two filters are comparable for the estimation of the vehicle pose but the DEKF provides smoother and more accurate estimation of the vehicle side-slip angle and vehicle lateral velocity. For this reason, for the design of the trajectory tracking controller presented in the next chapter, the DEKF has been chosen for the evaluation of the vehicle state variables for control purposes.



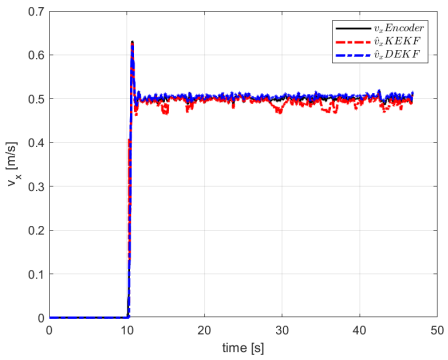
(a) X position



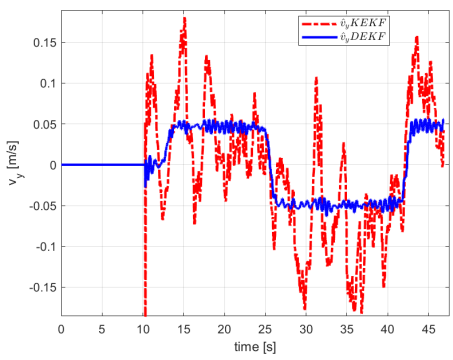
(b) Y position



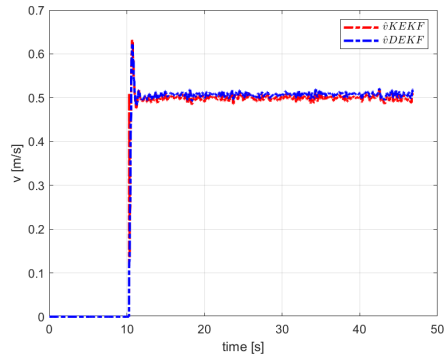
(c) Yaw angle



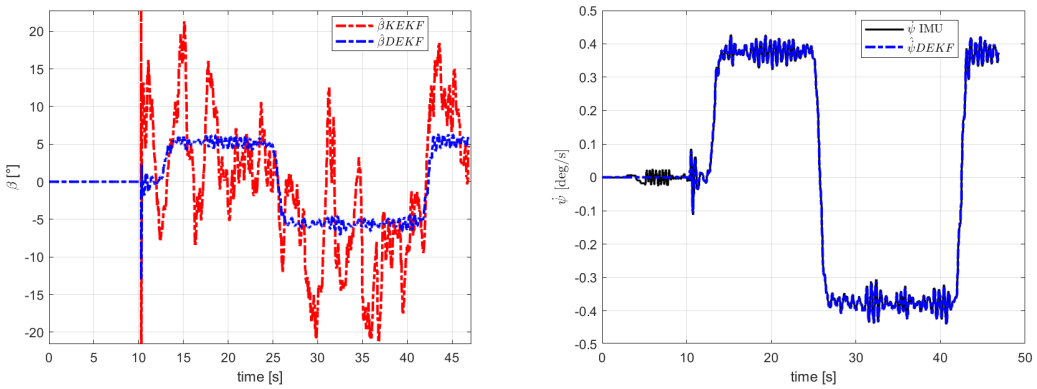
(d) Longitudinal velocity



(e) Lateral velocity



(f) Total velocity



(a) Vehicle sideslip angle

(b) Vehicle yaw rate

Figure 4.10: KEKF and DEKF performance comparison in experimental tests

Chapter 5

Trajectory tracking controllers design

In this chapter the design of trajectory tracking control algorithms for the QCar is discussed. Firstly, a pole placement controller is presented, followed by a model predictive controller (MPC). Once each controller has been designed and validated by using closed-loop simulations, the control solution has been integrated into the scaled vehicle's ECU for experimental tests in real scenario environment.

5.1 Pole placement controller solution

5.1.1 Control design via static state feedback

Let's consider a LTI dynamic system in the state-space representation:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases} \tag{5.1}$$

where $x \in \mathcal{R}^n$ is the state vector, $y \in \mathcal{R}^p$ is the output vector and $u \in \mathcal{R}^p$ is the input vector. As widely known, the dynamics of an LTI system depend on the eigenvalues of the dominant matrix A, so if we would desire to control the dynamics of the system we need to look for an input that can affect the structure of the matrix A in order to change its eigenvalues.
In this regard, the following control input law is considered [7]:

$$u(t) = -Kx(t) + Nr(t), K \in \mathcal{R}^{p,n}, r(t) \in \mathcal{R}^p, N \in \mathcal{R}^{p,p}$$

When this control input is applied, the state equation becomes:

$$\dot{x}(t) = Ax(t) + Bu(t) = (A - BK)x(t) + BNr(t) \tag{5.2}$$

The above differential equation represents the state equation of the closed-loop dynamics, where the dynamical properties of the (controlled) system depend on the eigenvalues of the matrix $A - BK$, where the matrix K is referred to as state gain and $r(t)$ is a reference signal that the output has to track. A suitable choice of the state gain allows to change the dominant matrix's eigenvalues and in turn to modify the dynamic properties of the system, e.g. its stability. On the other hand, the reference gain N can be chosen in order to impose desired performance at

the steady state.
It's important to highlight that it's not always possible to find a state gain K able to arbitrarily assign the eigenvalues of the closed-loop dominant matrix $A - BK$. In particular, we can report the following result:

Result 1 *Given the state space equation*

$$\dot{x}(t) = Ax(t) + Bu(t), A \in \mathcal{R}^{n,n}, B \in \mathcal{R}^{n,p}$$

a state gain K exists such that the n eigenvalues of $A - BK$ can be arbitrarily assigned, real or complex-conjugate, locations if and only if

$$\rho(M_R) = n$$

The matrix M_R is referred to as the reachability matrix and it is defined as

$$M_R = \begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix}$$

The reachability (or controllability) property describes the capabilities of the control input to affect the system dynamics. If $\rho(M_R) = n$ the dynamic system is said reachable, or equivalently the couple (A, B) is reachable and it's possible to find a matrix K to modify the dynamical properties of the system. Typically, K is computed by placing the eigenvalues of the controlled system to obtain asymptotic stability (eigenvalues with strictly negative real part) and desired damping and rapidity properties of the transient.Instead, N can be chosen to make unitary the dc-gain of the controlled system ensuring zero steady state tracking error in the presence of a constant reference signal.
The architecture of this control technique is shown before, assuming that the matrix D is null.

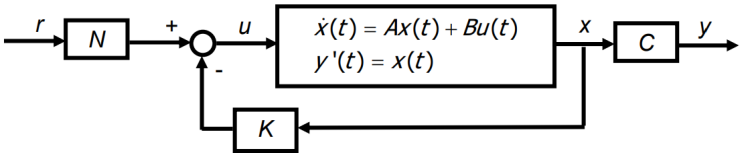


Figure 5.1: Static state feedback control law

5.1.2 Pole placement path tracking design

As described in 2.4, considering a bicycle model assumptions, the lateral dynamics of a vehicle written with respect to the desired lane errors can be described by

$$\dot{x} = Ax + B_1\delta + B_2\rho \tag{5.3}$$

with $x = \begin{bmatrix} e_y & \dot{e}_y & e_\psi & \dot{e}_\psi \end{bmatrix}^T$, where e_y is the vehicle's center of gravity's lateral position error, e_ψ is the yaw angle error between the road and the car, δ is the front wheel steering angle and it represents the control input to the system and ρ is the desired trajectory's curvature.
By substituting the numerical values in the open-loop matrix A defined in the paragraph 2.4 and considering $v_x = 1$ m/s, it results

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -7.5287 & 7.5287 & 0.0091 \\ 0 & 0 & 0 & 1 \\ 0 & 0.5931 & -0.5931 & -7.8412 \end{bmatrix}$$

This matrix has two eigenvalues at the origin with minimal polynomial multiplicity equal to 2, so the system is unstable. Moreover, as shown in the previous paragraph, the couple (A, B_1) is reachable, so it's possible to control the system using a static state feedback control law of the form

$$\delta = -K_x x + k_r \rho$$

where K_x is the state gain and K_r is the reference gain. The closed-loop dynamics using this static state feedback control input is

$$\dot{x} = (A - B_1 K_x)x + (B_1 K_r + B_2)\rho = A_{cl}x + B_{cl}\rho \tag{5.4}$$

where the eigenvalues of the matrix $A_{cl} = A - B_1 K_x$ affect the dynamics of the controlled system. Before approaching the choice of the desired eigenvalues of the closed-loop dominant matrix, let's consider first how the reference gain is computed. The feed-forward matrix K_r is chosen in such a way that the steady state value of the lateral error is equal to zero in the presence of constant desired curvature. In the detail, considering as output of the system the entire state vector, we can define $C_{cl} = \mathcal{I}_4$ in such a way $y = C_{cl}x = x$. The closed-loop transfer function's set can be computed as

$$G(s) = C_{cl}(s\mathcal{I}_4 - A_{cl})^{-1}B_{cl} \tag{5.5}$$

where s is the Laplace variable and $G(s)$ corresponds to

$$G(s) = \begin{bmatrix} G_1(s) & G_2(s) & G_3(s) & G_4(s) \end{bmatrix}^T = \begin{bmatrix} \frac{e_y(s)}{\rho(s)} & \frac{\dot{e}_y(s)}{\rho(s)} & \frac{e_\psi(s)}{\rho(s)} & \frac{\dot{e}_\psi(s)}{\rho(s)} \end{bmatrix}^T \tag{5.6}$$

Employing the final value theorem, the steady state value of the lateral error can be computed as

$$|e_y(t)|_\infty = |\lim_{t\to\infty} e_y(t)| = |\lim_{s\to 0} s \cdot e_y(s)| = |\lim_{s\to 0} s \cdot G_1(s) \cdot \rho(s)| \tag{5.7}$$

Considering a constant desired curvature $\rho(t) = \bar{\rho} \cdot \mathcal{E}(t)$, its Laplace transform is $\rho(s) = \frac{\bar{\rho}}{s}$. Substituting this expression in 5.7 we have

$$|e_y(t)|_\infty = |\lim_{s\to 0} s \cdot G_1(s) \cdot \frac{\bar{\rho}}{s}| = \bar{\rho} \cdot |\lim_{s\to 0} G_1(s)| \tag{5.8}$$

In order to achieve $|e_y(t)|_\infty = 0$, we have to impose that $|G_1(0)| = 0$. From this constraint, the feed-forward gain K_r can be computed as a function of the vehicle's parameters and an entry of the state gain K_x. To do this, as first step the transfer function $G_1(s)$ can be computed by using the Matlab symbolic toolbox, then the DC gain is evaluated with the Matlab command $G_1(0) = subs(G_1, s, 0)$ and lastly the expression of K_r to obtain a zero steady state lateral error can be found using the Matlab command $K_r = solve(G_1(0), K_r)$. With the last step, we impose to solve the equation $G_1(0) = 0$ with respect to the parameter K_r. With this procedure, the expression of K_r that has been obtained is the following

$$K_r = \frac{m v_x^2}{L}\left(\frac{l_r}{C_{\alpha f}} - \frac{l_f}{C_{\alpha r}} + \frac{l_f}{C_{\alpha r}} K_{x,3}\right) + L - l_r K_{x,3} \tag{5.9}$$

As it can be seen from the above equation, K_r depends on the vehicle's parameters and only on the third entry of the state gain K_x.
For what concerns the state gain, it is computed by employing a set of desired eigenvalues obtained through an optimization procedure. Since the controller has been designed to achieve a zero steady state value of the lateral error, other three performance index are considered.

In particular, I would desire to minimize the maximum lateral error and the maximum orientation error and its steady state value during the maneuver. So the objective is to minimize $||G_1(s)||_\infty, ||G_3(s)||_\infty, |G_3(0)|$, where

$$||G_1(s)||_\infty = \min_w |G_1(jw)|$$

$$||G_3(s)||_\infty = \min_w |G_3(jw)|$$

$$|G_3(0)| = |\lim_{s\to 0} G_3(s)|$$

Notice that the closed-loop transfer function's $G_1(s)$ and $G_3(s)$ depend on the eigenvalues of the matrix $A - B_1K_x$, so the goal is to find a set of desired eigenvalues λ_{des} which minimize the following quadratic cost function

$$J(\lambda_{des}) = \alpha_1||G_1(s)||^2_\infty + \alpha_2||G_3(s)||^2_\infty + \alpha_3|G_3(0)|^2 \tag{5.10}$$

where $\alpha_1, \alpha_2, \alpha_3$ are positive constant used to manage the trade-off between the three above three quantities. The minimization of the cost function is subject to the constraint that the desired eigenvalues coming from the optimization problem must have strictly negative real part to ensure the stability of the system. Since the matrix A and B_2 depend on the vehicle's longitudinal velocity, a set of optimized eigenvalues is considered for each value of v_x used in the simulated and experimental maneuvers. Then, the optimization problem that has to be solved is formulated in the following way

$$\begin{aligned}
&\lambda^*{}_{des} = \arg\min_{\lambda_{des}} J(\lambda_{des}) \\
&J(\lambda_{des}) = \alpha_1||G_1(s)||^2_\infty + \alpha_2||G_3(s)||^2_\infty + \alpha_3|G_3(0)|^2 \\
&\text{subject to:} \\
&\quad G_1(s) = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}(s\mathcal{I}_4 - A_{cl})^{-1}B_{cl} \\
&\quad G_3(s) = \begin{bmatrix}0 & 0 & 1 & 0\end{bmatrix}(s\mathcal{I}_4 - A_{cl})^{-1}B_{cl} \\
&\quad \mathcal{R}e\{\lambda_{des,i}\} < 0, \qquad i = 1,...,4
\end{aligned} \tag{5.11}$$

5.2 Model predictive controller solution

5.2.1 NMPC algorithm

Model predictive control (MPC) is a optimal control strategy. The fundamental principle of MPC is to use a dynamic model to predict the system's behavior and to provide the best decision to control the system in terms of optimal control action at the current time instant. Therefore, prediction models are crucial for every kind of MPC, since it involves the employment of dynamic models and optimization strategies. Despite the prediction model employed in this work to control the lateral dynamics of the vehicle is a simple single-track model, in this section the nonlinear model predictive control (NMPC) is presented since it represents a general and flexible technique to control nonlinear dynamic systems, e.g. vehicle's two-track model.
The key point of the NMPC is that it allows to directly consider state,input and output constraints and to manage systematically the trade-off between performance and command effort. The general approach is based on a first prediction of the system's state trajectory over a given time horizon by using a prediction model of the plant that is embedded into the controller; at this point, the control input is computed in order to have the prediction as close as possible to the desired behavior, by means of some on-line optimization algorithms.
Let's consider a general MIMO dynamic system

$$\begin{cases} \dot{x} = f(x,u) \\ y = h(x,u) \end{cases} \tag{5.12}$$

where $x \in \mathcal{R}^n$ is the state vector, $u \in \mathcal{R}^{n_u}$ is the command input and $y \in \mathcal{R}^{n_y}$ is the output.
As assumption, we consider that the state is measured in real-time with a sampling time T_s and the measurements are expressed with

$$x(t_k), \quad t_k = T_s k, k = 0,1,...$$

If the state is not measured, either an observer or a filter has to be used. The model predictive control is based on two essential operations: prediction and optimization. At each time $t = t_k$, the state and the output of the system are predicted over the time interval $[t, t+T_p]$ by the integration of the dynamic equations of the system (or much more likely a model of it), where $T_p \geq T_s$ is called prediction horizon. At any time $\tau \in [t, t+T_p]$, the predicted output $\hat{y}(\tau)$ is a function of the "initial" state $x(t)$ and the input signal $u(t:\tau)$, that represents the command input in the interval $[t, \tau]$.
At each time $t = t_k$, an input signal $u(t:\tau) = u^*(t:\tau)$ is computed such that the prediction $\hat{y}(u^*(t:\tau)$ has the desired behavior over the considered prediction horizon. But what does "desired behavior" mean? The concept of desired behavior is formalized by considering a cost function [8]

$$J(u(t:t+T_p)) = \int_t^{t+T_p} (||\tilde{y}_p(\tau)||_Q^2 + ||u(\tau)||_R^2) d\tau + ||\tilde{y}_p(t+T_p)||_P^2$$

where $\tilde{y}_p(\tau) = r(\tau) - \hat{y}(\tau)$ is the predicted tracking error, $r(\tau) \in \mathcal{R}^{n_y}$ is the reference signal to track, $||\cdot||_X$ and their integrals are respectively vector norms and signal norms. The optimal control input sequence $u^*(t:t+T_p)$ is selected to minimize the cost function $J(u(t:t+T_p))$. Usually, the main goal is to minimize, at each time t_k, the tracking error square norm $||\tilde{y}_p(\tau)||_Q^2$ over the prediction horizon; the term $||u(\tau)||_R^2$ is used to manage the trade-off between performance and command effort; the term $||\tilde{y}_p(t+T_p)||_P^2$ is used to give further importance to the tracking error at the final time instant of the considered time interval $[t, t+T_p]$.

It's crucial to highlight that the tracking error $\tilde{y}_p(\tau)$ depends on the predicted system's output $\hat{y}(\tau)$, which in turn is computed by integrating the model differential equations. Therefore, the minimization of the cost function J is subject to the constraints

$$\begin{cases} \dot{\hat{x}}(\tau) = f(\hat{x}(\tau), u(\tau)), & \tau \in [t, t+T_p] \\ \hat{y}(\tau) = h(\hat{x}(\tau), u(\tau)) \end{cases}$$

so, the predicted output and state have to satisfy the dynamic equations of the prediction model that is embedded into the controller. The MPC allows also to directly consider further constraints that could be present on the predicted state, the output and/or the input

$$\hat{x}(\tau) \in X_c, \quad \hat{y}(\tau) \in Y_c, \quad u(\tau) \in U_c, \quad \tau \in [t, t+T_p]$$

Indeed, in real applications it's very likely to have constraints on the state/output of a system, e.g. obstacles, collision avoidance, and on the command input, e.g. saturation of the actuator due to physical limitations.
To summarize, at each time $t = t_k$, for $\tau \in [t, t+T_p]$, the following optimization problem is solved

$$\begin{aligned} & u^*(t : t+T_p) = \arg\min_{u(\cdot)} J(u(t : t+T_p)) \\ & J(u(t : t+T_p)) = \textstyle\int_t^{t+T_p} (||\tilde{y}_p(\tau)||_Q^2 + ||u(\tau)||_R^2) d\tau + ||\tilde{y}_p(t+T_p)||_P^2 \\ & \text{subject to:} \\ & \qquad \dot{\hat{x}}(\tau) = f(\hat{x}(\tau), u(\tau)) \\ & \qquad \hat{y}(\tau) = h(\hat{x}(\tau), u(\tau)) \\ & \qquad \hat{x}(\tau) \in X_c, \hat{y}(\tau) \in Y_c, u(\tau) \in U_c \end{aligned} \tag{5.13}$$

where T_s is the sampling time, T_p is the prediction horizon, with $0 \leq T_s \leq T_p$.
The optimization problem presented above is in general non-convex, this means that the numerical algorithms used to solve the problem usually provide in general a local minimum solution as optimal control action. Moreover, the optimization problem must be solved on-line at each time t_k and this makes the MPC very computationally demanding.
By solving the optimization problem 5.13, the optimal input sequence $u^*(t : t+T_p)$ is computed. Now, suppose to apply the entire sequence for the time interval $[t, t+T_p]$; this does not perform a feedback action and so it's not possible neither to reduce error and disturbance effects or to adapt to new and varying environments. This is why $u^*(t : t+T_p)$ represents an open-loop input as it depends on $x(t)$ but not on $x(\tau)$, $\tau > t$. For this reason, the NMPC feedback controller is obtained by employing the *receding horizon strategy*: at time $t = t_k$, the entire optimal input sequence $u^*(t : t+T_p)$ is computed by solving the optimization problem and at this point only the first value of the input signal $u^*(t = t_k)$ is applied by keeping it constant in the interval $[t_k, t_{k+1}]$.
By recalling that the cost function $J(\cdot)$ depends on the state $x(t_k)$, assuming that the model and the objective function are time invariant, the receding horizon strategy implicitly defines a

nonlinear time invariant static state feedback control law of the form:

$$u(t_k) = \mathcal{K}(x(t_k)) \tag{5.14}$$

Unfortunately, in general the analytic expression of the control law $\mathcal{K}(x(t_k))$ cannot be computed.

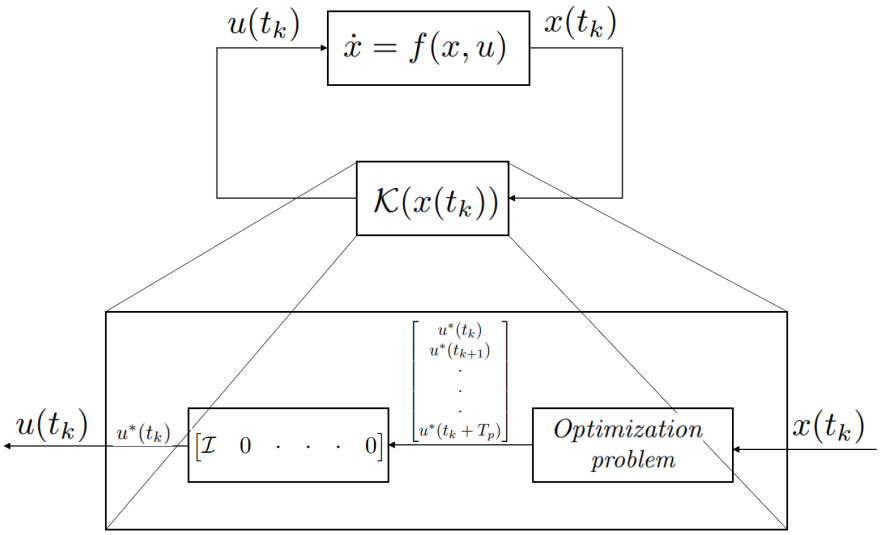


Figure 5.2: Receding horizon strategy

For what concerns the choice of the design parameters, the sampling time T_s is often given and cannot be arbitrarily chosen, as in this application since it coincides with the sampling time of the Kalman filter (10 ms). The prediction horizon $T_p = T_s N_h$, where N_h represents the number of prediction steps, can be chosen through a sensitivity analysis in simulation, by ensuring an optimal trade-off between performance and computational time. In general, a large T_p increases the closed-loop stability properties; however, a too large T_p may reduce the short-time tracking accuracy. For what regards the choice of the weight matrices Q, R, P, they can be set as diagonal non-negative matrices (like LQR controller) and the values of Q_{ii}, R_{ii} and P_{ii} are changed by using trial and error procedure, until the given requirements are fulfilled.

increasing Q_{ii}, P_{ii}	$\Rightarrow$	decreasing the energy of x_i, y_i	$\Rightarrow$	reducing oscillations and convergence time
increasing R_{ii}	$\Rightarrow$	decreasing the energy of u_i	$\Rightarrow$	reducing command effort and "energy consumption"

5.2.2 MPC path tracking design

In order to properly design a model predictive controller for trajectory tracking application, firstly let's consider the dynamic model that is employed to predict the system's behavior of the vehicle over the prediction horizon. As in the case of pole placement controller, the considered

plant model is the linear lateral dynamics model of the vehicle written with respect to the desired lane errors

$$\dot{x} = Ax + B_1\delta + B_2\rho$$

where $x = \begin{bmatrix} e_y & \dot{e}_y & e_\psi & \dot{e}_\psi \end{bmatrix}^T$, δ is the steering angle and ρ represents the desired curvature. In order to limit steering angle oscillations, the model has been slightly modified by adding the steering angle as fifth state variable and imposing its derivative as control input. By making this choice, it's possible to minimize the steering angle floating behavior by tuning the R weight in the cost function. Therefore, the following augmented system is considered as prediction model

$$\dot{\hat{x}}_a = A_a\hat{x}_a + B_{1a}u + B_{2a}\rho$$

where $\hat{x}_a = \begin{bmatrix} \hat{x} & \delta \end{bmatrix}^T$ is the augmented state, u represents the time derivative of the steering angle and the matrices A_a,B_{1a} and B_{2s} are

$$A_a = \begin{bmatrix} [A & B_1] \\ O & \end{bmatrix} \in \mathcal{R}^{5x5}, \qquad B_{1a} = \begin{bmatrix} O \\ 1 \end{bmatrix} \in \mathcal{R}^{5x1}, \qquad B_{2a} = \begin{bmatrix} B_2 \\ 0 \end{bmatrix} \in \mathcal{R}^{5x1}$$

As output of the system, the entire augmented state in considered

$$\hat{y} = \hat{x}_a$$

The considered cost function is

$$J(u(t:t+T_p)) = \int_t^{t+T_p} (||\tilde{y}_p(\tau)||_Q^2 + ||u(\tau)||_R^2)d\tau + ||\tilde{y}_p(t+T_p)||_P^2$$

where $\tilde{y}_p(\tau) = r(\tau) - \hat{y}(\tau)$ is the predicted tracking error and $r(\tau) \in \mathcal{R}^{5x1}$ is the reference signal to track and in this case is set as a zero vector. The weight matrices $Q \in \mathcal{R}^{5x5}$ and $R \in \mathcal{R}^{1x1}$ are set by considering the longitudinal velocity of the trajectory tracking test and so their values are chosen by using trial and error procedure. The weight matrix $P \in \mathcal{R}^{5x5}$ is set to a zeros matrix O^{5x5} because I did not give importance to the tracking error at the final time instant of the considered time horizon. As further constraint, mechanical saturation of the steering actuation system is considered; indeed, the maximum possible value of the steering angle's magnitude that can be actuated by the QCar's steering system is 0.5 radians which corresponds to 28.65°

$$|\delta| = |\hat{x}_{a,5}| \leq 0.5 \tag{5.15}$$

For what concerns the choice of the design parameters T_s, T_p and N_h, we have that T_s is chosen based on the Kalman filter sampling time equal to $10ms$ while T_p is selected by performing a set of simulations in order to evaluate a good trade-off between performance and computation time. At this point, once the prediction model has been presented, as well as the cost function, the reference signal to track and the constrained state domain $\mathcal{X}_c$, the optimization problem 5.13 can be solved and the receding horizon principle is employed to realize a feedback control action. The implementation of the presented model predictive controller has been done in Matlab/Simulink environment by using the ACADO toolkit and its Matlab interface [9].
ACADO Toolkit is a software environment and algorithm collection written in C++ for dynamic optimization and controls design. It provides a general framework to deal with a great variety of algorithms for direct optimal control, including model predictive control as well as Runge-Kutta and BDF integrators for the simulation of ODE's and DAE's. ACADO Toolkit has been designed to be an open-source and user-friendliness software; additionally, no external packages are required since it is designed in a completely self-contained manner.
ACADO Toolkit is able to interact with Matlab/Simulink thanks to "ACADO for Matlab", that represents a Matlab interface which allows to bring ACADO integrators and algorithms for direct optimal control to Matlab. The most important characteristics of ACADO for Matlab are:

- Same properties of ACADO Toolkit: in the interface we have no new algorithms, nor any further functionality.
- No C++ required, familiar with Matlab: we do not need any knowledge of C++, neither syntax or compiling rules, to use the interface. This is one of the main reasons why ACADO for Matlab is a valid way to start using ACADO Toolkit if you are familiar with Matlab but don't have any C++ knowledge or experience. Indeed, the interface makes use of Matlab style notations, as well as it allows to directly use variables and matrices stored in the workspace.
- Use Matlab black box models: Although the ACADO Toolkit supports a symbolic syntax to write down differential and algebraic equations, the interface allows to link existing Matlab black box models to ACADO Toolkit.
- Cross-platform: The interface works on Windows, Mac and Linux OS.

After installing and configuring ACADO for Matlab, the S-function representing the model predictive controller used in Simulink environment is generated by using the Matlab script presented in the following page. Since both the prediction model and the state constraints are linear and the objective function is quadratic, just one quadratic program (QP) needs to be solved at each sampling instant. This is why in general QP solvers play a crucial role in looking for the optimal control inputs considering system dynamics and ensuring constraints' fulfillment. In this work, the solver *qpOASES* has been used to solve QP optimization problem [10].

```
%% --- Definition of AKADO s function --- %%
clear all, close all, clc

load("QCar_param.mat");

EXPORT = 1;
COMPILE = 1;

Ts = 0.01;            % MPC sampling time
Ts_pm = 0.001;        % Prediction model sampling time
N = 40;               % Prediction horizon steps
N_disc = N*Ts/Ts_pm;  % Integrator steps number

% States of the system
DifferentialState e1 e1d e2 e2d delta;

% Control input
Control delta_dot;

% Extra input
OnlineData curvature;

%% Parameters definition

Ca_F       = QCar.Ca_lin; % [N/rad] Front tire cornering stiffness
Ca_R       = QCar.Cp_lin; % [N/rad] Rear tire cornering stiffness
m          = QCar.m;      % [kg] Vehicle mass
Vx         = QCar.long_speed; % [m/s] Longitudinal speed
I_z        = QCar.Iz;         % [kg*m^2] Vehicle inertia
l_f        = QCar.Wb_half_a;  % [m] CoG - front tire distance
l_r        = QCar.Wb_half_b;  % [m] CoG - rear tire distance
L          = QCar.Wb_half_a + QCar.Wb_half_b; % [m] Wheelbase

%% Control action bounds

Lb_delta = -0.5; % [rad]
Ub_delta = 0.5; % [rad]

%% State equations

x1d = is(e1d);

x2d = is((e2*(Ca_F + Ca_R))/m - Vx*curvature*(Vx +...
      (Ca_F*l_f - Ca_R*l_r)/(Vx*m)) + (Ca_F*delta)/m + ...
      - (e1d*(Ca_F + Ca_R))/(Vx*m) - (e2d*(Ca_F*l_f -...
      Ca_R*l_r))/(Vx*m));

x3d = is(e2d);

```

```
x4d = is((e2*(Ca_F*l_f - Ca_R*l_r))/I_z - (curvature*(Ca_F*l_f^2 + ...
      Ca_R*l_r^2))/I_z + (Ca_F*delta*l_f)/I_z - ...
      (e2d*(Ca_F*l_f^2 + Ca_R*l_r^2))/(I_z*Vx) - ...
      (e1d*(Ca_F*l_f - Ca_R*l_r))/(I_z*Vx));

control_action = is(delta_dot);

f = [dot(e1);dot(e1d);dot(e2);dot(e2d);dot(delta)]==...
                        [x1d;x2d;x3d;x4d;control_action];

%% Define cost function

h = {e1;e1d;e2;e2d;delta;control_action};

hN = {e1};

%% MPC export

acadoSet('problemname', 'NMPC');

ocp = acado.OCP( 0.0, N*Ts, N );

%These two variables are just for the compilation.
% The ones implemented in the Simulink Model are used in the
    simulation
W = acado.BMatrix(eye(length(h)));
WN = acado.BMatrix(eye(length(hN)));

ocp.minimizeLSQ(W, h);
ocp.minimizeLSQEndTerm(WN, hN);

%% Constraints (hard)

% State constraints (steering angle)
ocp.subjectTo(is(Lb_delta - (delta)) <= 0);
ocp.subjectTo(is(Ub_delta - (delta)) >= 0);

%% Settings
ocp.setModel( f );

mpc = acado.OCPexport( ocp );
mpc.set( 'HESSIAN_APPROXIMATION','GAUSS_NEWTON');
mpc.set( 'DISCRETIZATION_TYPE','MULTIPLE_SHOOTING' );
mpc.set( 'SPARSE_QP_SOLUTION','FULL_CONDENSING_N2');
mpc.set( 'INTEGRATOR_TYPE','INT_IRK_GL4');
mpc.set( 'NUM_INTEGRATOR_STEPS', N_disc);
mpc.set( 'QP_SOLVER', 'QP_QPOASES3');
mpc.set( 'GENERATE_SIMULINK_INTERFACE', 'YES' );
```

```
mpc.set( 'LEVENBERG_MARQUARDT',1e-4);

%% Export, Compilation and S-function generation
if EXPORT
    mpc.exportCode( 'export_NMPC' );
end
if COMPILE
    global ACADO_;
    copyfile([ACADO_.pwd '/../../external_packages/qpoases3'],...
        'export_NMPC/qpoases3')

    make_custom_solver_sfunction

end
```

Once the MPC s-function has been generated by running the Matlab script presented above, it is employed in the Simulink file where the trajectory tracking model predictive controller is designed. The crucial variables and the MPC design parameters that have been considered for the design of controller are:

- n: number of parameters in the cost function J. In this case $n = 6$, since the entire state vector (5 state variables) and the unique control input (time derivative of the steering angle) are considered.
- N_h: number of prediction horizon steps. In this case $N_h = 40$. This value comes out from a sensitivity analysis that has been made in simulation based on finding a good trade-off between performance and computational time. In order to realize this sensitivity analysis, the controller sampling time has been fixed to $T_s = 10$ ms and both performance and computational time have been evaluated as the prediction horizon changed, leading to an optimal value of the design parameter N_h. Indeed, the prediction horizon is computed as $T_p = T_s N_h$, where T_s is the controller sampling time. The results of the sensitivity analysis are presented below. As expected, as the prediction horizon's length rises the performance improve (both the lateral and heading error decrease) but at the same time the computational time increases. A good trade-off between these two can be obtained for $N_h = 40$, so for a prediction horizon equal to $T_p = 400$ ms.
- W: stage cost matrix. It is defined as $W = \text{diag}(Q, R)$, where Q and R are the weight matrices used in the cost function.
- W_N: terminal cost matrix. I set this parameter equal to a zeros matrix since I did not give importance to the tracking error at the final time instant of the considered prediction horizon $t + T_p$.
- N_{iter}: number of the iterations of the control algorithm to give the optimal control input that has to be applied to the system. As this value increases, it's more likely that the found solution is optimal, in the sense that it approaches the absolute minimum of the cost function, but on the other hand the computational effort increases as well. I set $N_{iter} = 3$ because this represents a good trade-off between solution's accuracy and computational effort.

After setting the controller parameters and generating the controller s-function using the Matlab script presented above, it is employed in the Simulink file when the path tracking model predictive

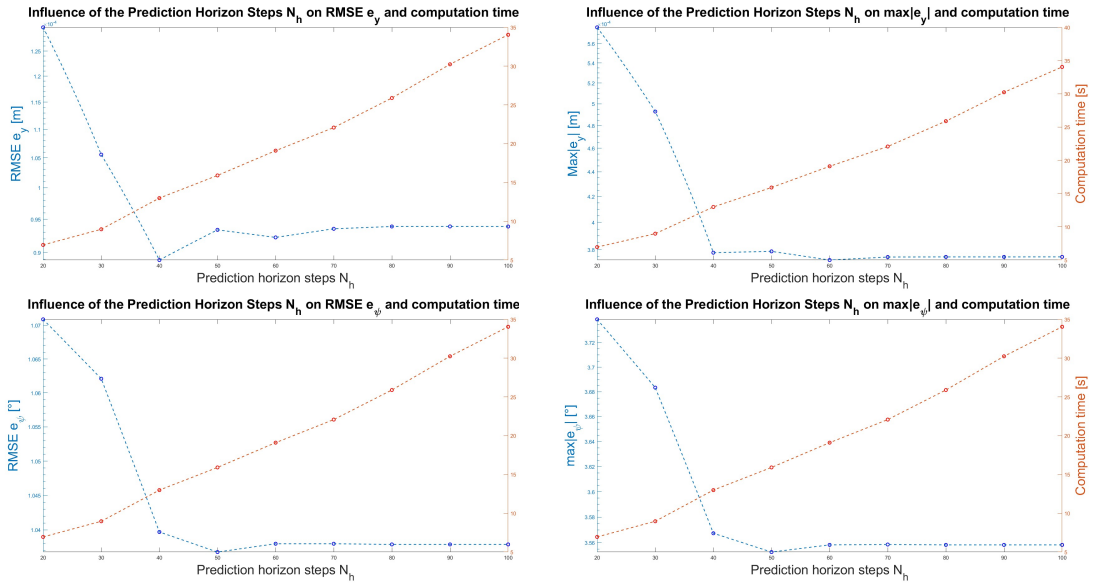


Figure 5.3: Sensitivity analysis

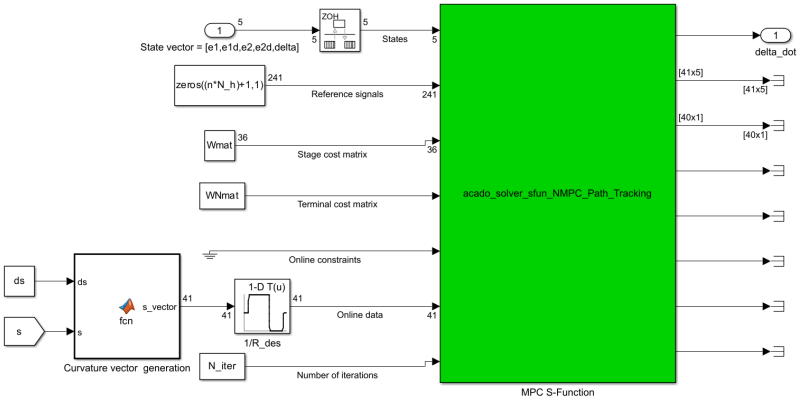


Figure 5.4: MPC block in Simulink

controller is designed. Therefore, the s-function block represents the controller in Simulink and it takes as input:

- The current state vector of the plant.
- The reference signals over the entire prediction horizon, that coincides with a zeros' vector having length $nN_h + 1$.
- The stage cost matrix, written as column vector.
- The terminal cost matrix, written as column vector.
- Possible online constraints, if any.

- Online data over the prediction horizon: in this case, the desired curvature for the entire time interval $[t, t + T_p]$ has to be provided since, as presented in the Matlab script, the curvature is defined as an online data. In order to compute the desired curvature over the prediction horizon, the assumption of constant longitudinal speed is used. The algorithm that I used to build this vector is the following

Algorithm 1 Curvature prediction over the prediction horizon at each T_s

Set N_h ▷ Prediction horizon steps
$s_v = \text{zeros}(N_h + 1,1)$ ▷ Distance vector initialization
$s_v(1) = s$ ▷ s: distance covered until the actual time instant
for $i = 2 : N_h + 1$ **do**
    $s_v(i) = s + (i - 1)ds$
end for
ρ_v is the output of a look-up table that maps travelled distance and desired curvature.

- Number of iterations of the controller.

Chapter 6

Simulation results

6.1 Computation of the lateral and heading errors in real-time for control purposes

Since the plant model employed for the design of both pole placement controller and model predictive controller contains the lateral displacement error e_y and the heading angle error e_ψ as state variables, it becomes fundamental to evaluate these signals in real-time for control purposes. It's important to notice that both quantities cannot be directly measured, i.e. it's not present a sensor which provides e_y and e_ψ measurements, but these variables need to be evaluated in an accurate way to ensure the correct behavior of the controllers. The goal of this paragraph is to present the procedure for the accurate computation of e_y and e_ψ in real-time for both simulation and experimental tests.
The reference trajectory is expressed in terms of travelled distance s_{path}, reference curvature ρ, reference yaw angle ψ_{ref} and reference coordinates in the inertial frame X_{ref}, Y_{ref}. For simulation and experimental purposes, these variables are provided to the control architecture through look-up-tables as function of actual distance travelled by the vehicle s, as shown in figure 6.1. If the car would travel on the reference path, position, curvature and yaw angle would be equal to the reference ones. Nevertheless, in general the vehicle deviates from the reference path and it's needed to evaluate this difference by using the lateral displacement error e_y and the heading angle error e_ψ.

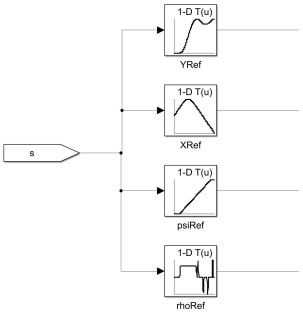


Figure 6.1: Look-Up-Tables used to provide the reference variables to the control architecture, 's' is the vehicle travelled distance.

The lateral distance between vehicle and reference path is given by [11]:

$$d = e_y = (Y - Y_{ref})\cos(\psi_{ref}) - (X - X_{ref})\sin(\psi_{ref}) \tag{6.1}$$

On the other hand, the heading angle error is simply calculated as the difference between current heading angle of the vehicle and reference heading angle for the vehicle at the travelled distance s.

$$e_\psi = (\psi - \psi_{ref}) \tag{6.2}$$

A crucial step is how the actual travelled distance is computed. At each sampling time, a segment ds_1 is calculated considering the Euclidean distance between two consecutive points $(X_k, X_{k-1}), (Y_k, Y_{k-1})$ as reported in equation 6.3.

$$ds_1 = \sqrt{(X_k - X_{k-1})^2 + (Y_k - Y_{k-1})^2} \tag{6.3}$$

This segment corresponds to the distance travelled by the car on the longitudinal direction and can be related to the space travelled on the reference trajectory ds with this formula:

$$ds = \frac{ds_1}{1 - \frac{e1}{\rho}} \tag{6.4}$$

Then, at each sampling time the total distance travelled up to that specific time instant s is the sum of all the previous differential segments ds and is fed into the LUT to obtain the reference variables ρ, ψ_{ref}, X_{ref} and Y_{ref}.

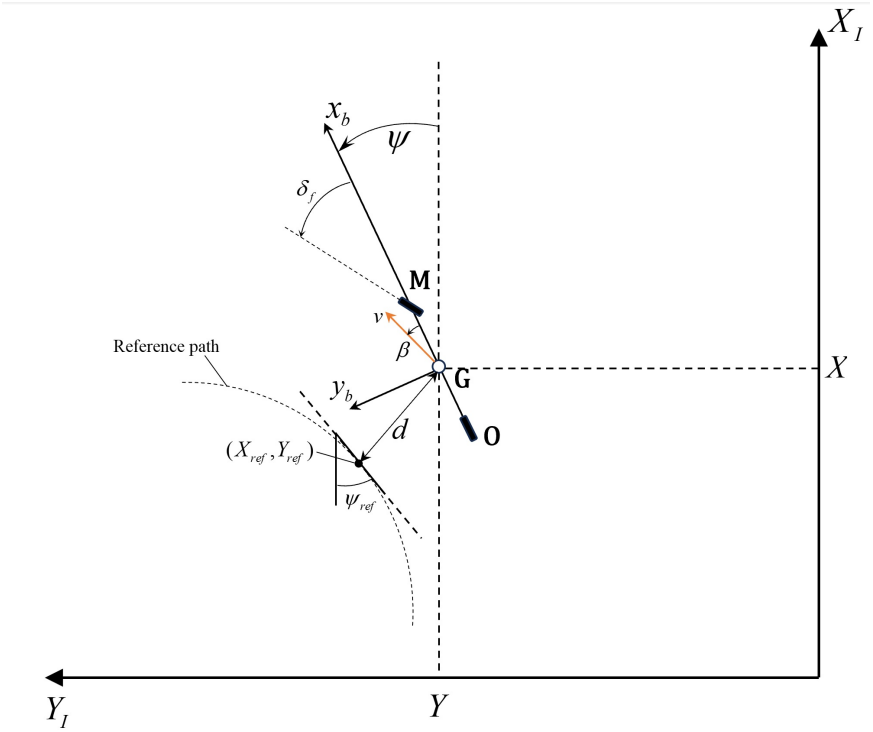


Figure 6.2: Definition of variables used for calculation of distance travelled along the path

6.2 Presentation of trajectory tracking controllers simulation results

This paragraph aims to present the simulation results of the designed trajectory tracking controllers. Several simulations have been performed for different vehicle longitudinal velocity, starting from 0.5 up to 1.5 mps, and for different trajectories. For sake of brevity and clarity, only the simulations realized with 0.5, 0.7 and 1 mps are presented for the considered trajectories. In the figures below, the following variables are plotted:

- Reference and actual trajectories;
- Lateral error;
- Heading error;
- Steering angle.

6.2.1 Pole placement with optimization simulation results

This section aims to present the simulation results and the tracking performance achieved by the static state feedback controller with optimization. As expected, both lateral and heading angle errors increase as the velocity rises and more evasive manoeuvres such as eight trajectory or obstacle avoidance are considered.

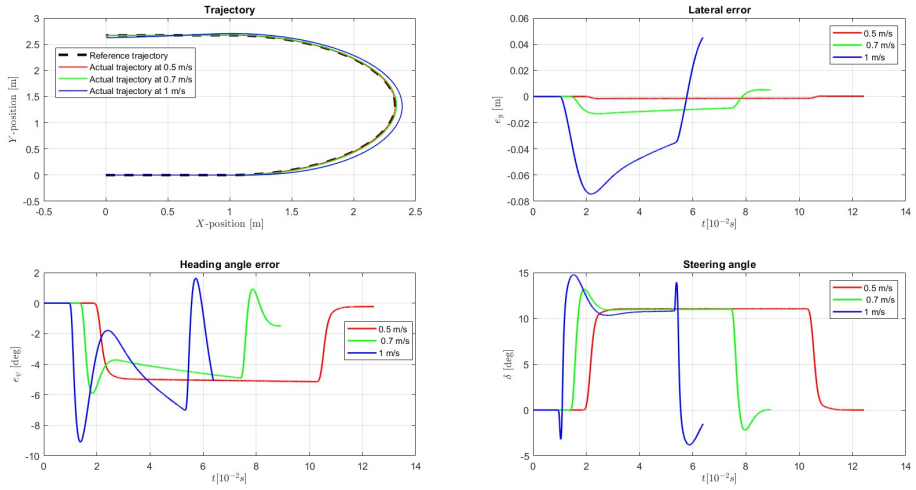


Figure 6.3: Simulation results of pole placement controller for U-shaped trajectory

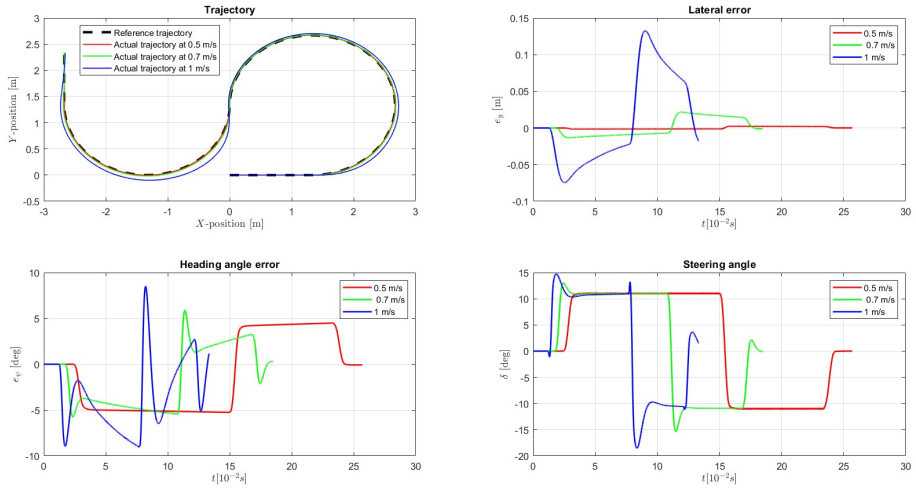


Figure 6.4: Simulation results of pole placement controller for S-shaped trajectory

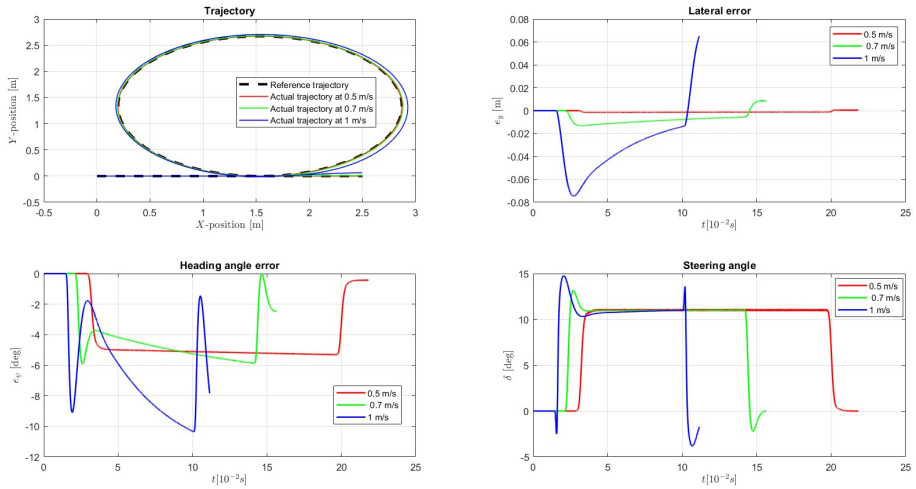


Figure 6.5: Simulation results of pole placement controller for circle trajectory

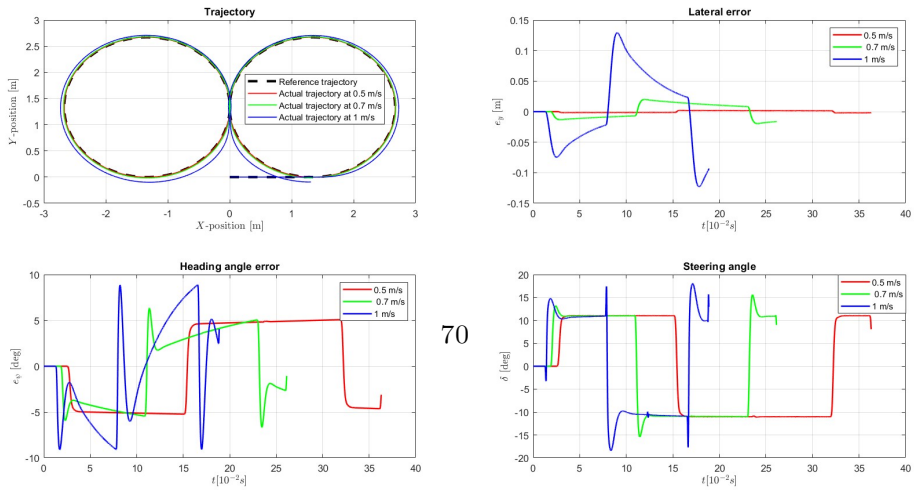


Figure 6.6: Simulation results of pole placement controller for eight trajectory

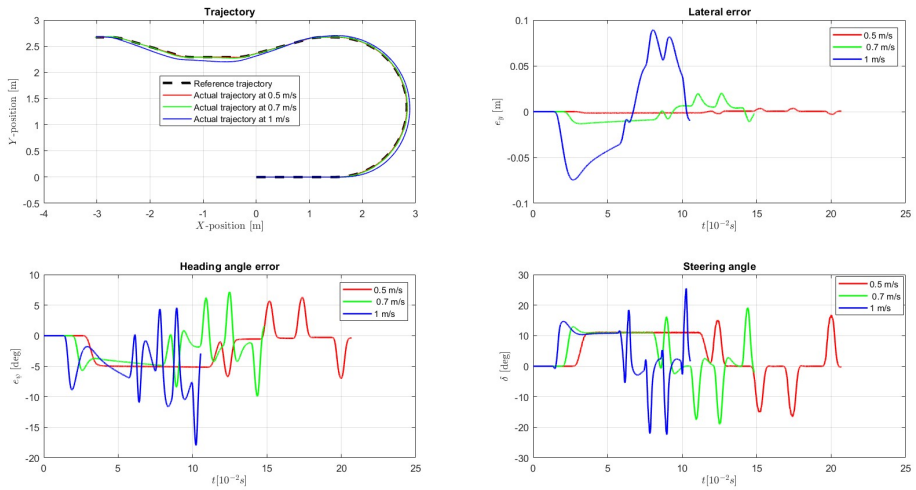


Figure 6.7: Simulation results of pole placement controller for obstacle avoidance trajectory

6.2.2 Model predictive controller simulation results

This section aims to present the simulation results and the tracking performance achieved by the model predictive controller. Since the weight matrices of the cost function have been tuned considering a nominal velocity equal to 1 mps, the controller reaches better performance for such value of longitudinal velocity. Moreover, the MPC allows to have very good performance even when higher velocities are considered.

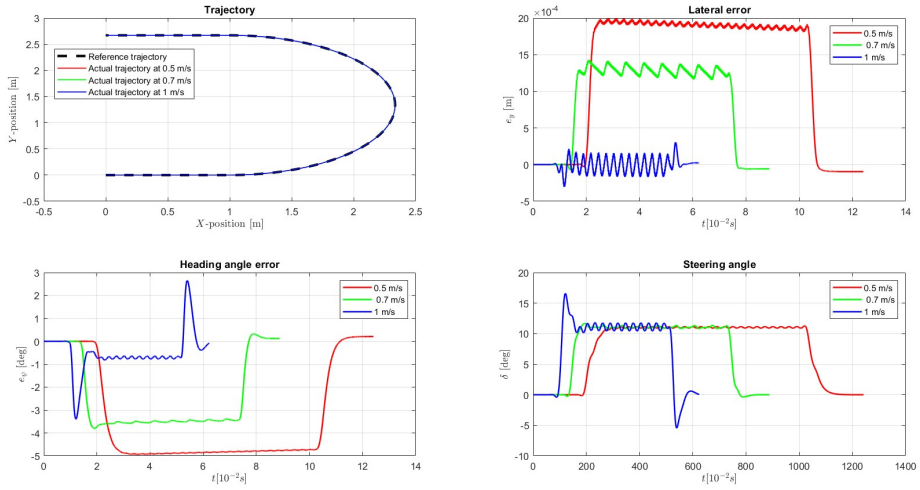


Figure 6.8: Simulation results of model predictive controller for U-shaped trajectory

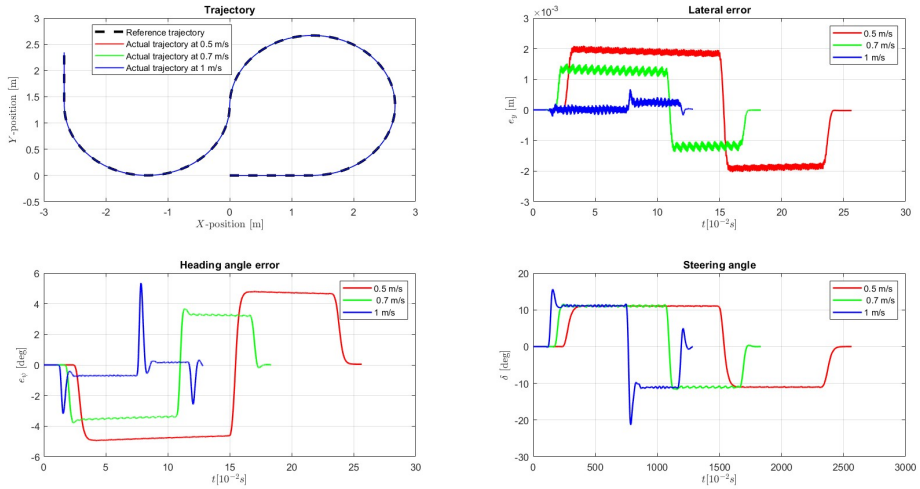


Figure 6.9: Simulation results of model predictive controller for S-shaped trajectory

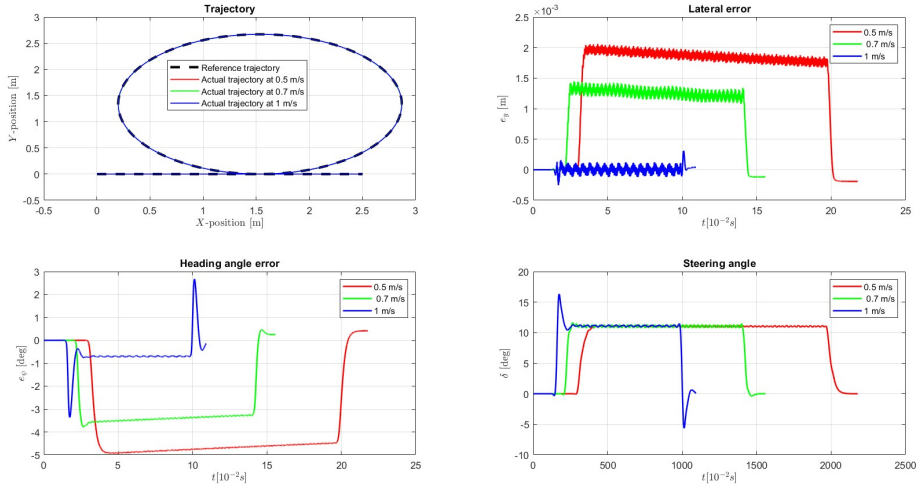


Figure 6.10: Simulation results of model predictive controller for circle trajectory

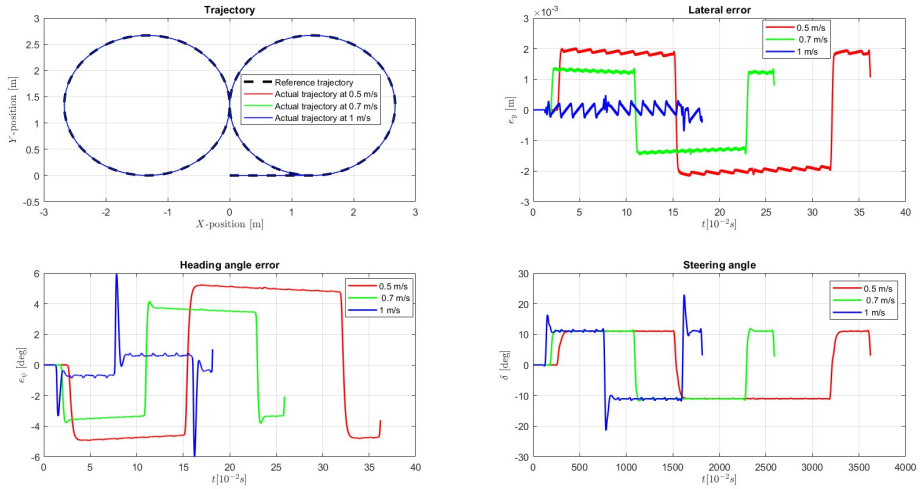


Figure 6.11: Simulation results of model predictive controller for eight trajectory

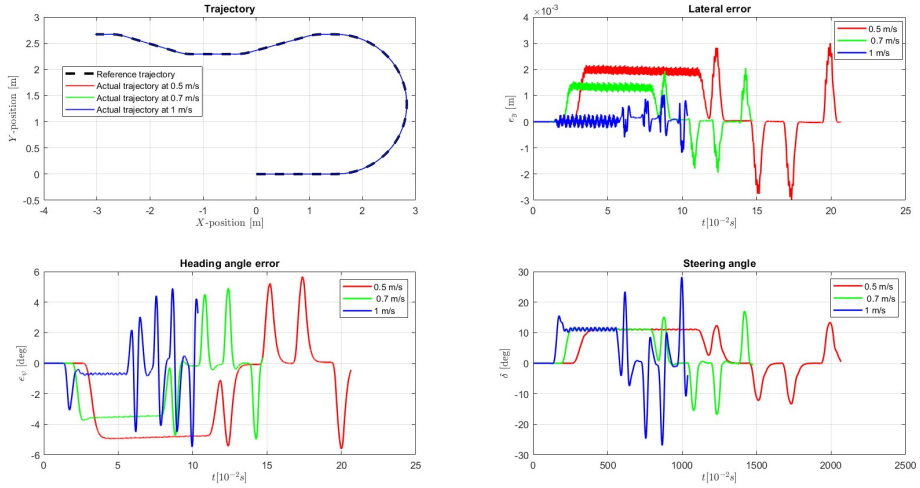


Figure 6.12: Simulation results of model predictive controller for obstacle avoidance trajectory

Chapter 7

Experimental results

In this chapter, the experimental performance of both controllers for different velocities and trajectories on the test track are presented. The considered velocities are 0.5, 0.6, 0.7, 0.8, 0.9, 1, 1.2 and 1.5 m/s; however, for brevity and clarity, in the following figures only the results for 0.5, 1, 1.5 m/s are reported. Additionally, for the closed trajectories, i.e. circle and eight, three laps were made at each velocity to compare the performance with other trajectory tracking controllers developed in the same research group, e.g. an enhanced model reference adaptive controller with neural network [12]. The plots show the reference and the actual trajectory of the vehicle and the measured lateral and heading errors with respect to the reference path as function of the travelled distance.

7.1 Pole placement with optimization experimental results

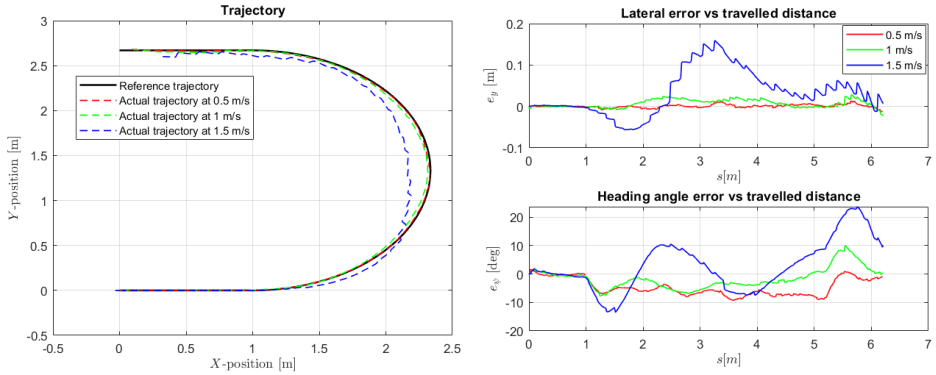


Figure 7.1: Experimental results of pole placement controller for U-shaped trajectory at 0.5, 1 and 1.5 mps

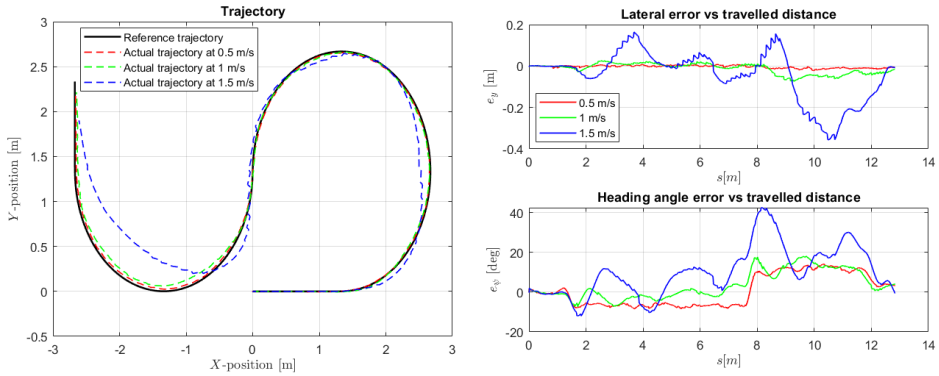


Figure 7.2: Experimental results of pole placement controller for S-shaped trajectory at 0.5, 1 and 1.5 mps

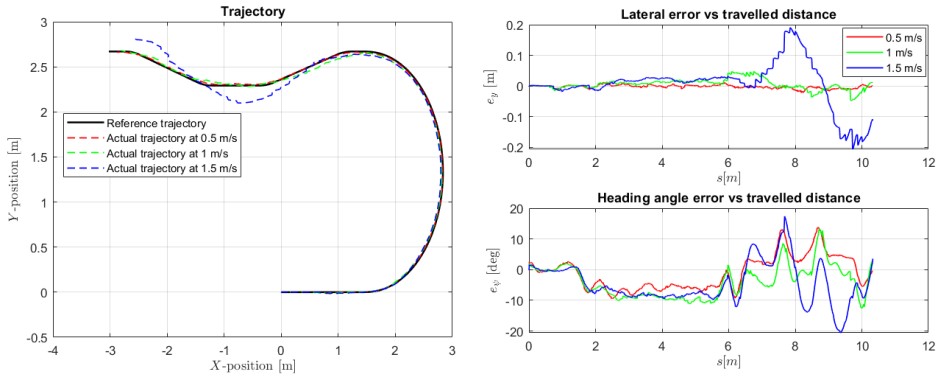


Figure 7.3: Experimental results of pole placement controller for obstacle avoidance trajectory at 0.5, 1 and 1.5 mps

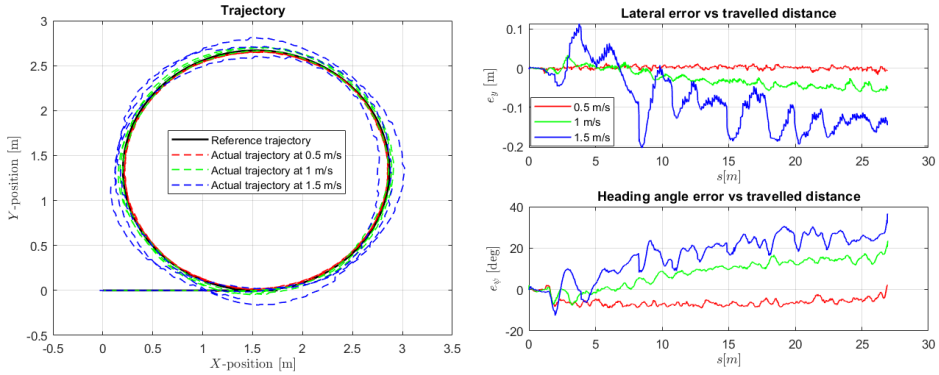


Figure 7.4: Experimental results of pole placement controller for circle trajectory at 0.5, 1 and 1.5 mps

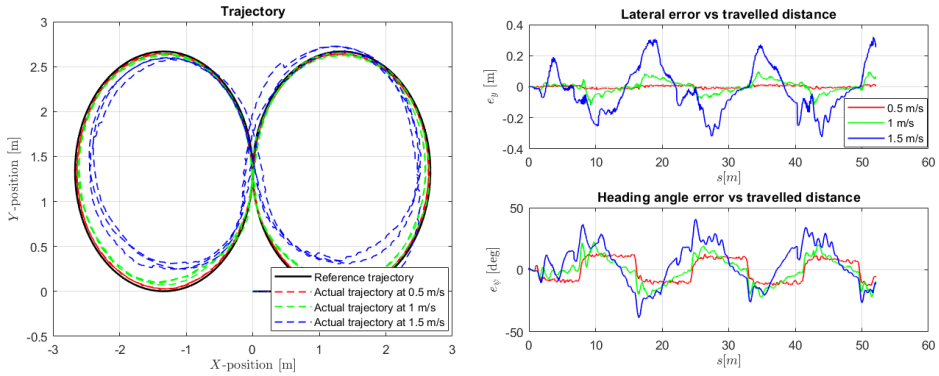


Figure 7.5: Experimental results of pole placement controller for eight trajectory at 0.5, 1 and 1.5 mps

7.2 Model predictive controller experimental results

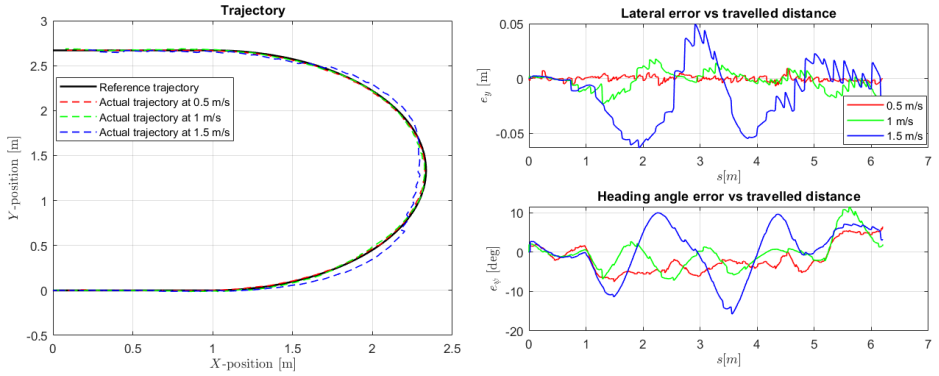


Figure 7.6: Experimental results of model predictive controller for U-shaped trajectory at 0.5, 1 and 1.5 mps

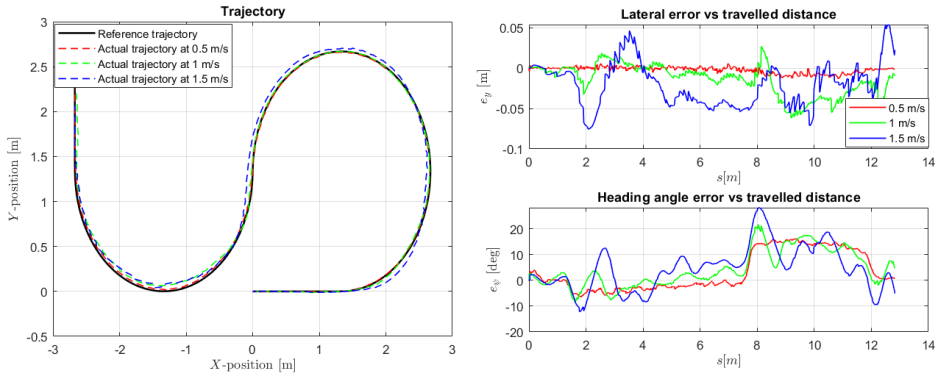


Figure 7.7: Experimental results of model predictive controller for S-shaped trajectory at 0.5, 1 and 1.5 mps

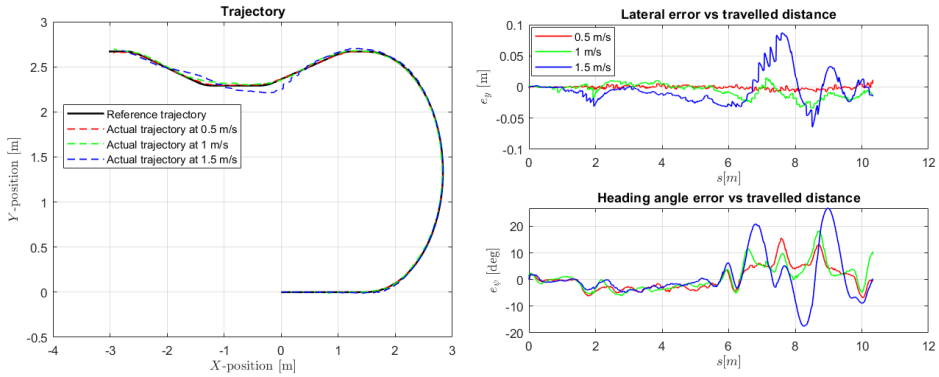


Figure 7.8: Experimental results of model predictive controller for obstacle avoidance trajectory at 0.5, 1 and 1.5 mps

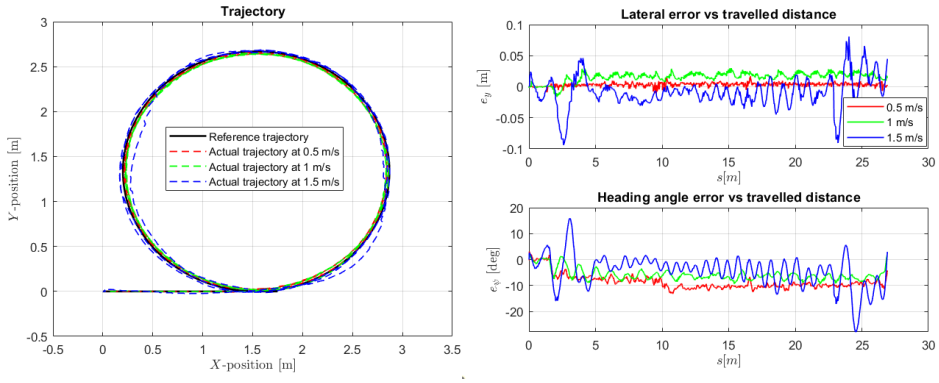


Figure 7.9: Experimental results of model predictive controller for circle trajectory at 0.5, 1 and 1.5 mps

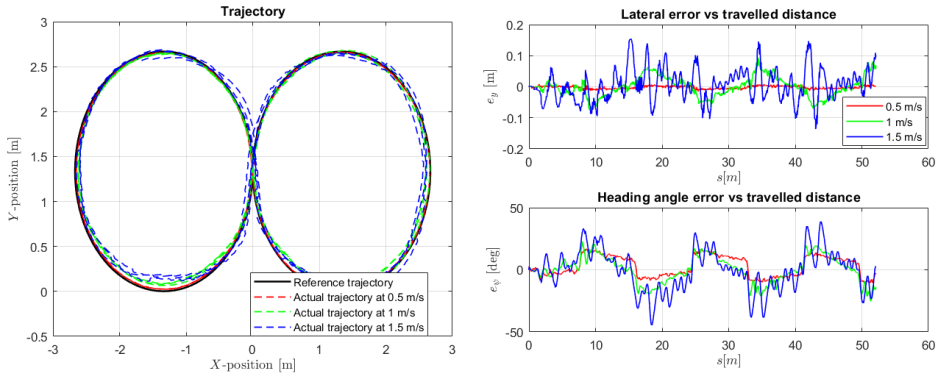


Figure 7.10: Experimental results of model predictive controller for eight trajectory at 0.5, 1 and 1.5 mps

7.3 Comparison of experimental results

The figures above for the five trajectories at 0.5, 1 and 1.5 mps show how the model predictive controller leads to a consider improvement of the tracking performance especially at higher speeds. However, since not all the experimental tests for each velocity are shown for brevity in the plots, performance were evaluated by using five Key Performance Indicators (KPIs) and presented using histograms. The considered KPIs and their analytical definitions are:

- Maximum lateral error;
- Maximum heading angle error;
- Root-Mean-Squared-lateral Error;
- Root-Mean-Squared-heading angle Error;
- Integral of the Absolute value of the Control Action variations (IACA);

$$MAX_{e_y} = \max(e_y) \tag{7.1}$$

$$RMS_{e_y} = \sqrt{\frac{1}{t_{fin} - t_{in}} \int_{t_{in}}^{t_{fin}} (e_y)^2 dt} \tag{7.2}$$

$$MAX_{e_\psi} = \max(e_\psi) \tag{7.3}$$

$$RMS_{e_\psi} = \sqrt{\frac{1}{t_{fin} - t_{in}} \int_{t_{in}}^{t_{fin}} (e_\psi)^2 dt} \tag{7.4}$$

$$IACA_\delta = \frac{1}{t_{fin} - t_{in}} \int_{t_{in}}^{t_{fin}} |\Delta\delta| \, dt \tag{7.5}$$

7.3.1 Comparison of controllers experimental results for U-shaped trajectory

Figure 7.11: Comparison of controllers performance in experimental tests with U trajectory at different vehicle velocities

7.3.2 Comparison of controllers experimental results for S-shaped trajectory

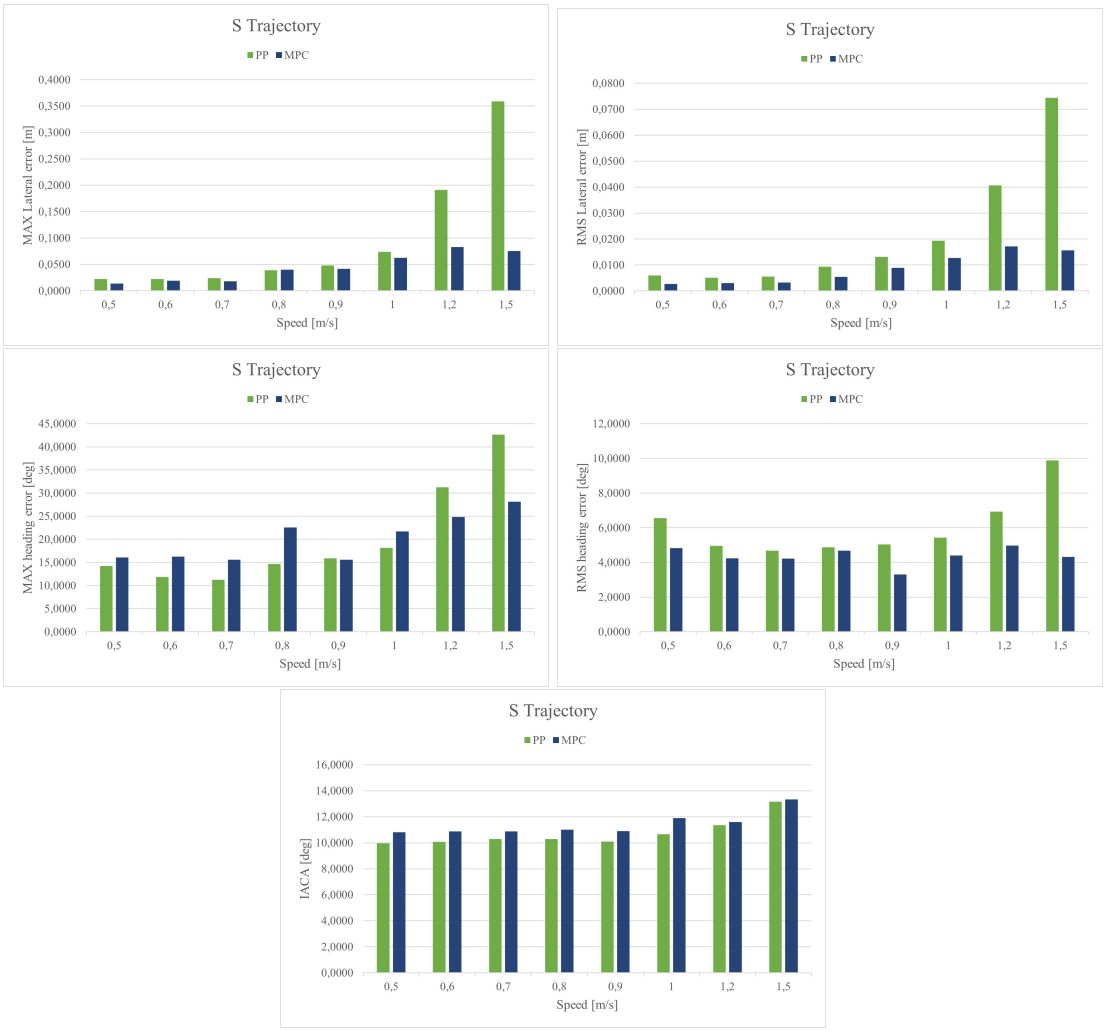


Figure 7.12: Comparison of controllers performance in experimental tests with S trajectory at different vehicle velocities

7.3.3 Comparison of controllers experimental results for obstacle avoidance trajectory

Figure 7.13: Comparison of controllers performance in experimental tests with eight trajectory at different vehicle velocities

7.3.4 Comparison of controllers experimental results for circle trajectory

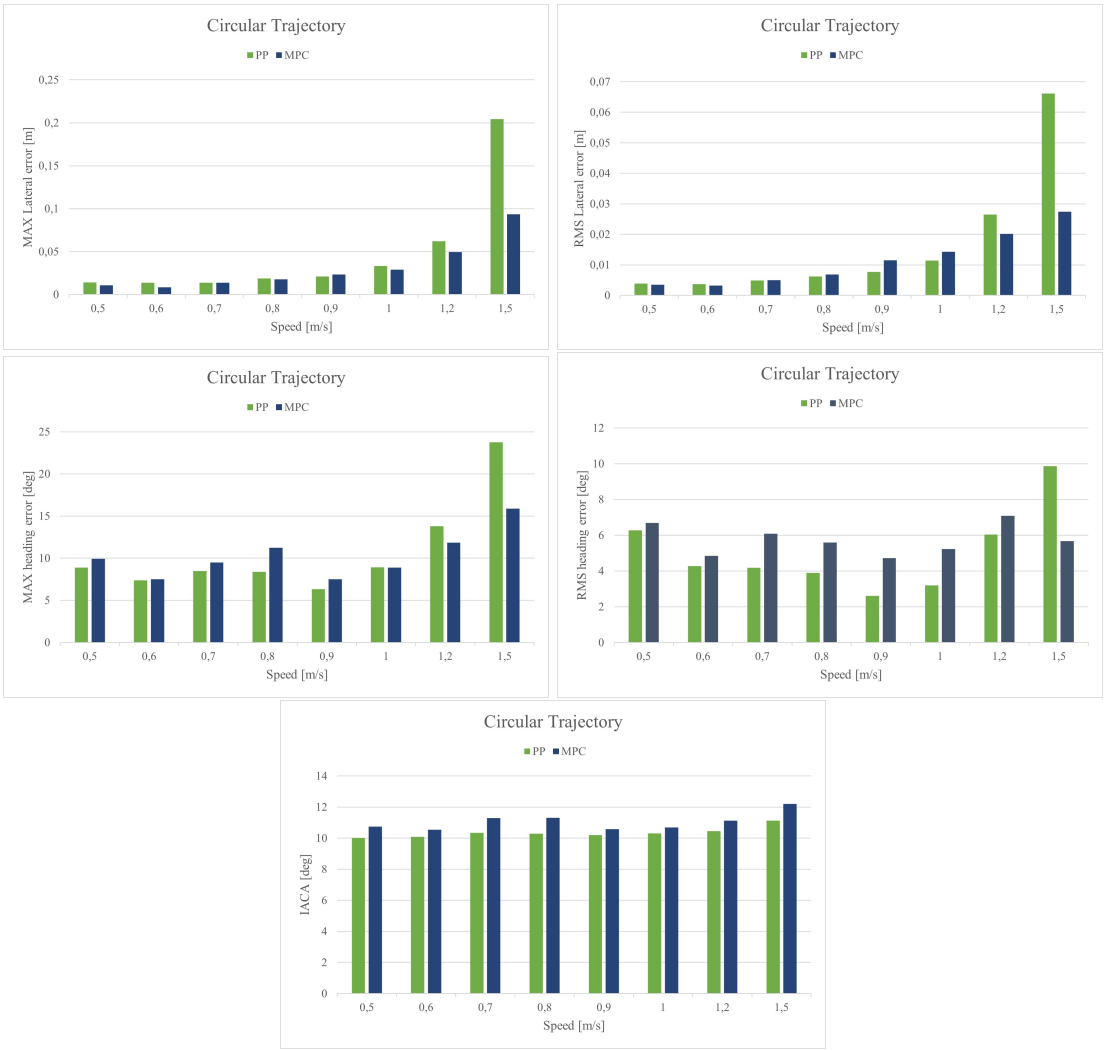


Figure 7.14: Comparison of controllers performance in experimental tests with circular trajectory at different vehicle velocities

7.3.5 Comparison of controllers experimental results for eight trajectory

Figure 7.15: Comparison of controllers performance in experimental tests with eight trajectory at different vehicle velocities

As can be seen in the above figures and as expected according to the preliminary simulation results, the performance of both trajectory tracking controllers in terms of maximum and mean lateral error worsen with increasing speeds. However, the model predictive controller allows to heavily improve the tracking performance especially at higher vehicle velocities, by leading to a decrease of the maximum lateral error in the range of [50,70]% for 1.5 mps depending on the considered manoeuvre.

Conclusion

This thesis has presented an application of estimation and control techniques for self-driving cars. In chapter 1, the QCar has been presented for giving an overview of the model vehicle employed for the experimental validation. In chapter 2, the vehicle dynamic models were discussed, by highlighting the paramount importance and assumptions of the dynamic single-track model employed together with the longitudinal dynamic model for both filtering and control purposes. The path-tracking control application has been introduced and discussed in chapter 3, where the generation of the reference trajectories considered for this work is presented. In chapter 4, a kinematic and a dynamic Extended Kalman Filter have been designed, implemented and validated to significantly improve the accuracy of the vehicle pose measured by the lidar by allowing the rejection of the spikes of this sensor, providing an estimate of the vehicle state variables of interest, including some signals that are not directly measured such as the vehicle side-slip angle, at the desired sample frequency needed to achieve better tracking results in the controller design. Both filters allows to achieve exceptional performance for spikes rejection and localization improvements; however the dynamic EKF leads to better results for the estimation of lateral velocity and vehicle side-slip angle and this is why this last one has been used for the controller validation. In chapter 5, the design of two trajectory tracking controllers was tackled: the first benchmark controller based on a static state feed-back control law in which the poles of the closed-loop dynamic system are computed by solving an optimization problem, and a model predictive controller which allows to heavily improve the tracking performance for higher velocities in all the considered trajectories, by leading to a decrease of the maximum lateral error of up to 70 %. Specifically, the simulation and the experimental results of the controllers are presented respectively in chapter 6 and 7.
The work carried out at the University of Surrey turned out to be fundamental for applying vehicle dynamics, filtering and model based controller design concepts in the field of autonomous driving. A possible future work deriving from this activity could be the usage of a neural network as prediction model embedded into the model predictive controller algorithm in order to overcome the limits of the simple linear single-track model at higher velocities leading potentially to better tracking performance in such scenarios.

Nomenclature

Symbol	Variable	Unit of measure
s	Distance covered along the path	m
ρ	Road curvature	1/m
e_y	Lateral offset error	m
e_ψ	Heading angle error	rad

Table 7.1: List of path tracking symbols

Symbol	Variable	Unit of measure
x_b, y_b	Body axes	m
X_I, Y_I	Global axes	m
X, Y	Coordinates of the c.g. of vehicle in the inertial frame	m
V	Total velocity at c.g. of vehicle in the inertial frame	m/s
V_x	Total velocity at c.g. of vehicle in the inertial frame	m/s
V_y	Lateral velocity at c.g. of vehicle in the inertial frame	m/s
v	Total velocity at c.g. of vehicle in the body frame	m/s
v_x	Longitudinal velocity at c.g. of vehicle in the body frame	m/s
v_y	Lateral velocity at c.g. of vehicle in the body frame	m/s
a_x	Longitudinal acceleration at c.g. of vehicle in the body frame	m/s^2
a_X	Longitudinal acceleration at c.g. of vehicle in the inertial frame	m/s^2
a_y	Lateral acceleration at c.g. of vehicle in the body frame	m/s^2
a_Y	Lateral acceleration at c.g. of vehicle in the inertial frame	m/s^2
ψ	Yaw angle of vehicle in global axes	rad
$\dot{\psi}$	Yaw rate of vehicle	rad/s
F_y	Lateral tire force	N
F_{yf}	Lateral tire force on front tires	N
F_{yr}	Lateral tire force on rear tire	N
m	Total mass of vehicle	kg

Symbol	Variable	Unit of measure
I_z	Yaw moment of inertia of vehicle	kg m^2
l_f	Longitudinal distance from c.g. to front tires	m
l_r	Longitudinal distance from c.g. to rear tires	m
δ_f	Front wheel steering angle	rad
δ_r	Rear wheel steering angle	rad
α_f	Front wheel slip angle	rad
α_r	Rear wheel slip angle	rad
C_{α_f}	Cornering stiffness of front tire	N/rad
C_{α_r}	Cornering stiffness of rear tire	N/rad
β	Slip angle at vehicle c.g	rad
r_w	Wheel radius	m
τ	Transmission ratio	-
V_a	Armature voltage	V
ω	Motor speed	rad/s
R_a	Terminal resistance	Ω
K_t	Torque constant	N m/A
K_v	Motor back-emf constant	V s/rad
J	Inertia equivalent to the motor	kg m^2
B	Coefficient of viscous friction	N m s
C_r	Static friction torque	N m

Table 7.2: List of symbols

Acknowledgements

Questo elaborato sancisce la fine del mio percorso accademico, durante il quale diverse persone sono state fondamentali.
Prima di tutto vorrei ringraziare la mia fidanzata. Sharon, sei stata, sei e sarai vitale per il mio benessere. Ti ringrazio per tutti i momenti passati insieme durante questi cinque anni e per le lunghe chiamate e la tua visita durante il mio soggiorno in Inghilterra, dove questo lavoro e' stato partorito. Sono fiero di essere cresciuto insieme a te, e non vedo l'ora di festeggiare questo e tanti altri traguardi insieme. Piu' di ogni altra cosa, di te ammiro e cerco di assorbire l'immensa energia positiva e la luce che emani. Sei stata essenziale durante questo percorso.
Ringrazio la mia famiglia. Senza il vostro supporto questo traguardo non esisterebbe, grazie per le chiamate giornaliere, per i consigli e per non avermi mai fatto sentire solo nonostante i mille chilometri di distanza che ci hanno separato in questi anni.
Ringrazio i ragazzi della residenza universitaria Borsellino, troppi per essere citati individualmente. E' stato stupendo condividere con voi questi anni. I ricordi delle cene, delle uscite, delle riflessioni e delle serate passate insieme sono tra i piu' belli che mi portero' dietro per sempre.
Ringrazio i ragazzi con cui ho condiviso il mio soggiorno a Guildford. Giulio, Matteo, Paolo (x2), Edoardo, i legami di amicizia creati qui rimarranno la cosa piu' bella che questa esperienza mi ha regalato.
E infine, ringrazio me stesso per aver cercato sempre di raggiungere il massimo durante questi anni di studi, per aver creduto nelle mie capacita' e nelle mie ambizioni.

Bibliography

[1] P. Darsh, P. Nishi, R. Aakash, C. Manisha, G. P. J. Neeraj Kumar, and W. Cho, "A review on autonomous vehicles: Progress, methods and challenges," *electronics*, vol. 11, no. 2162, 2022.

[2] D. Schramm, M. Hiller, and R. Bardini, "Vehicle dynamics," *Modeling and Simulation. Berlin, Heidelberg*, vol. 151, 2014.

[3] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[4] "Curvature," 2016. Accessed on July 14, 2023.

[5] "Passenger cars - Test track for a severe lane change manoeuvre - Part 2: Obstacle avoidance," standard, International Organization for Standardization, Geneva, CH, 2015.

[6] C. Novara, "Nonlinear control and aerospace applications, lecture notes.," 2017. Politecnico di Torino.

[7] M Canale, "Automatic control, lecture notes.." Politecnico di Torino.

[8] M. Canale, "Digital control technologies and architectures, lecture notes.." Politecnico di Torino.

[9] B. Houska, H. Ferreau, and M. Diehl, "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[10] "Qpoases user's manual," 2014.

[11] R. S. Sharp, D. Casanova, and P. Symonds, "A mathematical model for driver steering control, with design, tuning and performance results," *Vehicle system dynamics*, vol. 33, no. 5, pp. 289–326, 2000.

[12] P. Timis, "Path tracking control solutions via enhanced model reference adaptive control algorithms augmented with neural networks and their experimental validation in scaled fully autonomous vehicles.," 2023.