POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Thesis

Positioning and Path Tracking control strategies experimentally validated through fully autonomous scaled vehicle

Supervisors
Prof. Alessandro Rizzo
Dr. Umberto Montanaro
Prof. Aldo Sorniotti

Candidate
Scattolini Giulio

Academic Year 2022-2023

Summary

Autonomous driving is experiencing a rapid evolution in recent years. Basing on a combination of sensors, actuators and softwares, the final aim is to completely replace human driver in its operations, while improving safety and reliability. This thesis work has focused on the study and implementation of algorithms aiming at improving vehicle localization and aiming at the implementation of path tracking application. Regarding the first objective, Linear and Non-Linear Kalman Filters have been developed to estimate some essential variables and to improve the accuracy of the measurement, particularly those provided by the LiDar, allowing outliers to be neglected. For the second objective a PI controller and a LQR controller have been designed, letting the car to be able to follow a pre-determined trajectory with the use of data registered by LiDar and IMU sensors together with odometric informations. All the developed solutions have been designed and validated at first in simulation environment then validated and tested also in an experimental setup using the QCar, a real scaled fully autonomous vehicle prototype.

Acknowledgements

Alla mia famiglia
e a tutti coloro che mi hanno supportato in questo percorso

Contents

List of Tables

List of Figures

Chapter 1

Nomenclature

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
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
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 1.1: List of symbols

Chapter 2

Introduction

Over the last twenty years there has been a growing interest in the field of autonomous driving by research groups and automotive companies. The main aim of control systems for automated driving [1] is to make driving safer by reducing accidents. Additionally, a secondary focus is about the optimization of traffic flows and the optimization of energy consumption.
The early autonomous driving research lead to the development of the first driver assistance systems such as Adaptive Cruise Control (ACC), Lane keeping assist (LKA) and Lane Departure Warning (LDW). The former is related to control of the longitudinal dynamics of the vehicle whereas the second is related to lateral dynamics control and is achieved essentially through steering actuation. The third one simply gives warnings to the drivers if the vehicle starts to drift from the current line. One of the more recent evolution of this systems is the Lane Change Assist (LCA), which computes and tracks safe and comfortable trajectories for lane changing manoeuvres in highway scenarios.
Autonomous driving represents so a quickly evlving technology but there are still several technical and legislative challenges before this technology can become spread on large scale.
In 2016 the Society for Automotive Engineers (SAE) revised its technology for autonomous vehicles and defined several layers basing on the level of automation. As reported in figure 2.1, there are six layers starting from level 0 (no automation) ending with level 5(complete automation) in which the vehicle can handle every situation, also the emergency ones.

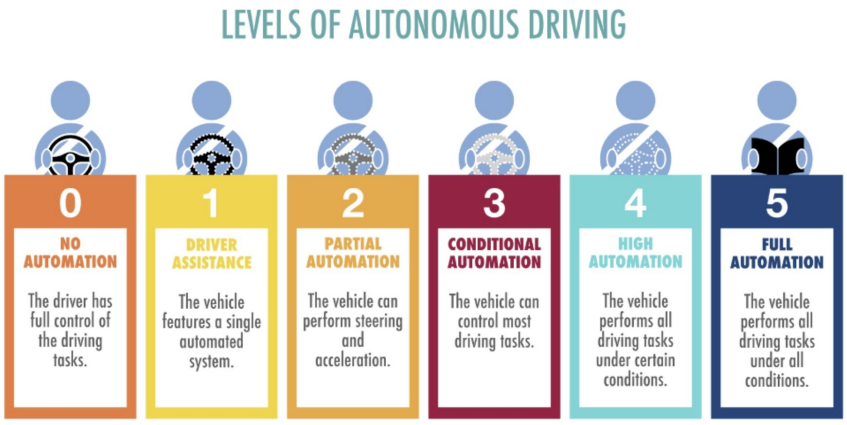


Figure 2.1: Levels for autonomous driving defined by SAE International.

As for now, the majority of series vehicles can be classified at level 2 or level 3. One of the challenges that autonomous vehicle encounter is the capability to operate in complex environments such as urban scenarios. In addition also the problem of data security is also very important, considering that a lot of sensible information are collected by autonomous vehicles during their working.

2.1 Localization and Path Tracking

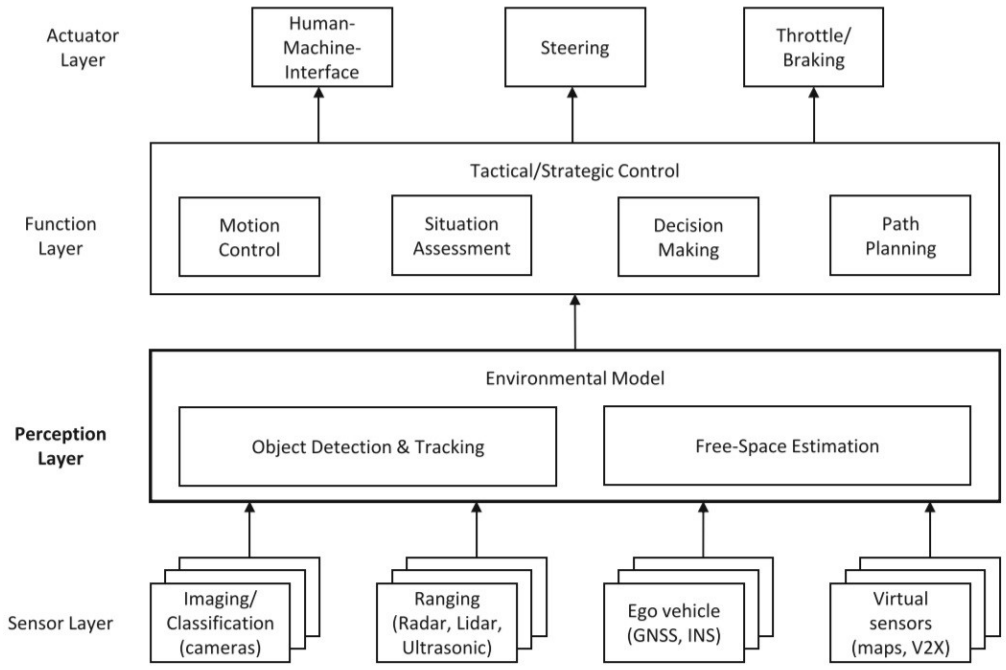


Figure 2.2: Structural layers for autonomous driving strategies. From "Automated Driving: Safer and More Efficient Future Driving", by D. Watzenig and M. Horn, 2017, p.162, Springer International.

From a more practical point of view, the main aim for autonomous vehicles is to navigate in a safe and efficient way from one point to another avoiding obstacles and making smart driving decisions. This task can be divided into three parts: first, planning the best route from point A to point B; second, making decisions about how to drive in response to the environment and figuring out the exact path the car should follow; third making the vehicle follow that planned trajectory and this is what this thesis work focused on in the second part.
Prior to the development of the path tracking control strategies, the initial part of the work focused on enhancing the vehicle localization through the development of Kalman Filters. This effort, that can be included in the perception layer showed in figure 2.2, is essential for two purposes: first to estimate such variables that cannot be directly measured from the plant but are essential for the control architecture, such as the vehicle lateral speed; second, using also data coming from other sensors provides estimations at a higher sampling frequency compared to the sensor's original data output. This is essential since a specific sampling frequency requirement must be met for the controller to work properly.

The work has been developed at University of Surrey (Guildford, UK) under the guidance of Dr. Umberto Montanaro, Prof. Aldo Sorniotti and Prof. Alessandro Rizzo. Valuable support was also provided PhD students Pietro Stano and Carmine Caponio.

2.2 Introduction to QCar

The vehicle used for the experimental tests is a 1:10 scaled prototype for indoor applications and designed for academic purposes. As it is possible to see in figure 2.3, the vehicle is provided with a wide variety of sensors including LiDAR, 360° cameras, depth camera, IMU, encoders, signalling devices, set of I/O ports to expand the funcitonalities. All this hardware is handled by a central on-board computer, the NVIDIA Jetson TX2, based on Linux OS: CPU 2 GHz quad-core ARM Cortex-A57 64-bit, GPU 256 CUDA core NVIDIA Pascal.
The manufacturing company also provides a set of Simulink libraries and Python scripts through which is possible to create ROS nodes, facilitating communication between the on-board computer and the ground station, allowing the user to directly interface with the vehicle, avoiding the need of low level programmin languages. In particular, all the work within this thesis has been developed in Matlab/Simulink environment. Additionally, they provide also a virtual environment for the development of mixed-reality scenario applications.

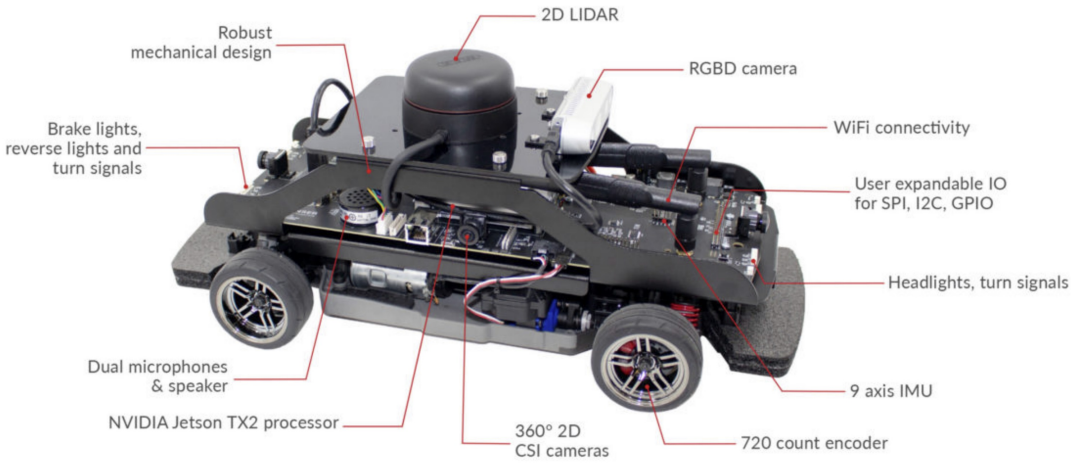


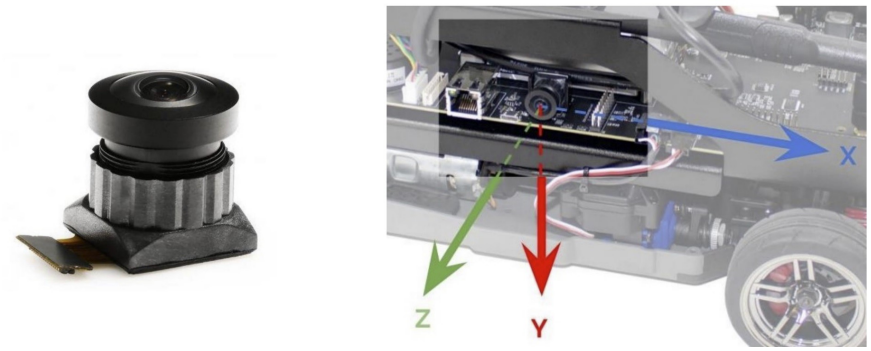
Figure 2.3: Qcar vehicle with its sensor equipment.

2.2.1 On-board hardware

This subsection to explain more in details about sensors and actuators the vehicle is equipped with:

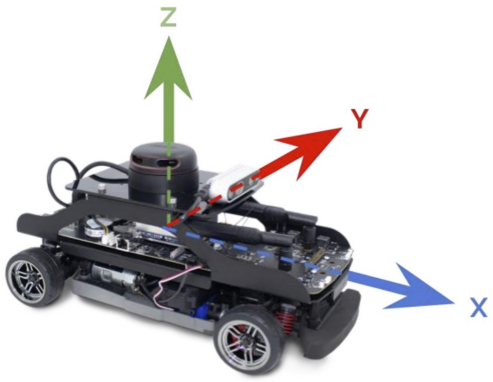
- CSI (Camera Serial Interface) cameras: Qcar mounts four of this cameras that are able together to provide a 360° vision of the environment. Each of this camera has an horizontal Field-Of-View of 160° and a vertical Field-Of-View of 120°. Moreover them are characterized by a variable resolution and a resolution dependant frame

rate. In the following an image of CSI camera (2.4a) with highlight on its reference system (2.4c) and a table (2.1) where the technical specifications are summarized.



(a) CSI camera

(b) Camera reference frame.



(c) QCar reference frame.

Figure 2.4: CSI camera the vehicle is equipped with, its reference frame and vehicle 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°

Table 2.1: Achievable frame rates for CSI cameras.

- Inertial-Measurement-Unit (9-axis):
  - Three for the accelerometer to measure linear accelerations;
  - Three for the gyroscope to measure angular speeds around each axis of the global reference frame;
  - Three for the magnetometer, to measure the magnetic field along each axis of the global reference frame.

The main problem of the accelerometer is that it provides highly noisy measurements. On the other end, the magnetometer is particularly slow and susceptible to environmental factors, such as the presence of iron in reinforced concrete floors. Characteristics of the sensor are reported in table 2.3.

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

Table 2.2: IMU sensor specifications.

- DC motor (Titan 12T 550 produced by Trsxxas) for wheels actuation, made possible thanks to a drive shaft and two differentials.

Symbol	Description	Value
R_m	Terminal resistance	0.470Ω
k_t	Torque constant	0.0027 N-m/A
k_m	Motor back-emf constant	0.0027 V/(rad/s)

Table 2.3: DC motor for wheels actuation parameters.

- Steering servo actuator with limited steering in the range ±30° and a time constant $\tau = 0.16$s;
- Single-ended optical shaft encoder **??** used to measure the angular position of the drive motor. Produced bu US Digital, provides 720 counts per revolution or 2880 counts per revolution in quadrature mode. The same device provides also the angular speed of the shaft using the time difference between encoder edges. This variable is measured in count/s and the following relation is used to convert it in rad/s:

$$\omega_{rad/s} = \omega_{count/s} \cdot \frac{4}{720} \cdot 2\pi \tag{2.1}$$

Figure 2.5: Optical encoder used to measure drive shaft angular speed.

- RGB-Depth Camera: RealSense D435 manufactured by Intel. It includes also one IR IR projector and two IR imagers, making this unit a stereo tracking solution.

This device can provide RGB, IR and depth stream of data at several frame rates and resolutions, depending on the specific needs. The informations about distance from the obstacle in front of the car can be useful for the development of platooning or adaptive cruise control applications.

Figure 2.6: Intel RealSense D435 RGB-D camera.

- LiDAR (Light-Detection-And-Ranging, figure 2.7) measures the distances with objects in the neighborhood through laser beams which are received upon reflection. The system calculates the distance between the LiDAR and an object based on the time it takes for the laser beam to travel from the emitter to the receiver. The technical specifications of the device the QCar is equipped with, are reported in table 2.4

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

Table 2.4: DC motor for wheels actuation parameters.

Additionally, the measuring range is between 0.2m to 12m.

Figure 2.7: RPLiDAR A2M8 the QCar is equipped with.

For indoor applications, the LiDAR is usually employed for Simultaneous Localization and Mapping (SLAM) services. The algorithm developed by the manufacturing company to determinate the vehicle position within a simultaneously mapped environment is structured as follows:

- Initially, a scan is performed with vehicle at a standstill. Data about the mapped environment are stored in memory and this point, is taken as reference for the global fixed reference frame;
- As the vehicle moves, real-time data from the sensor is compared to the stored data to determine the vehicle position in fixed reference frame coordinates. Additionally, this comparison provides also informations about current heading.

The main problems encountered are related to inaccurate measurements and outliers, due to the fact that this algorithm is still under development by the company. Another issue arises from the sensor limited maximum data output frequency of 15 Hz, which is not enough to ensure optimal performance for the path tracking control strategies that rely heavily on vehicle positioning. To tackle these issues, Kalman filters have been designed and discussed in chapter 5.

2.3 QCar in the academic research

Thanks to its practicality of use, the QCar has been used in several research works. In [2], the authors discuss about collaborative mapping approach using occupancy maps. These maps help multiple entities work together to explore faster the environment. They tackle the challenge of matching maps from different robots by using a feature-based method with adaptive filtering to process occupancy data. The approach demonstrated to work properly when merging six maps to create a single one for simultaneous localization and mapping.
J. Hu, Y.Zhang and S. Rakheja, "Adaptive Lane Change Trajectory Planning Scheme for Autonomous Vehicles under Various Road Frictions and Vehicle Speeds" [3], introduces a dynamic lane change trajectory planning system for autonomous driving, taking into account road friction and vehicle speed. It prioritizes both the safety of the maneuver and the comfort of the passengers. To further demonstrate the effectiveness of the proposed idea, experimental validation through QCars has been performed.
In [4], the authors present a cooperative adaptive cruise control system by leveraging V2V communication and integrated LiDAR. Firstly they develop an Adaptive-Cruise-Control then this algorithm is enhanced through an agent trained by Deep Q learning. Experimental tests with QCars showed that this solution can decrease the average inter-vehicular distance.
A cooperative UAV-UGV control mechanism is presented in [5]. Thanks to the collaborative operation between Unmanned ground vehicles and Unmanned aerial vehicles, the trajectory tracking is achieved through leader-follower strategy established together with the applicative scenario.
Additionally, the article proposed by J. Hu, Y. Zhang and S. Rakheja, "Adaptive Trajectory Tracking for Car-Like Vehicles With Input Constraints" ([6]), proposes an adaptive control scheme for trajectory tracking on low speed scenarios. The design of the adaptive gains is aimed to achieving a better convergence rate for the tracking errors while maintaining the commanded inputs within certain boundaries.

Chapter 3

System modelling: Kinematical and Dynamical analysis

In this chapter the aim is to describe the physical models used for the modellization of the behaviour of the QCar vehicle. For what concerns the longitudinal dynamics, either the kinematic either the dynamic models have been object of study and application. In particular, the two models have been used in the realization of the two Kalman filters descrived in subsequent chapter.

3.1 Single-track model

Single-track model (figure 3.1) of the vehicle has been considered in this work, as suggested by the QCar vehicle manufacturing company. It considers three degrees of freedom and if compared with other advanced models allows to reduce system complexity. The vehicle is considered to have front and rear wheels collapsed in a single entity at point M and point O respectively. 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 $\delta_r = 0$. The vehicle mass is assumed to be concentrated at point G, which is its center of gravity (c.o.g). The distances of M and O from the center of gravity are l_f and l_r respectively and their sum equals $L = l_f + l_r$, the vehicle wheelbase. Other assumptions [7] are at the foundation of this modeling:

- Vehicle is operating in 2D plane: roll, pitch and lift are neglected;
- The longitudinal component of the vehicle's centre of gravity velocity is constant in module;
- From the assumption of constant longitudinal velocity, the longitudinal forces on the tires are neglected.
- The load of the vehicle is supposed to be equally distributed between front and rear axles;

Whithin this set of hypothesis, 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 velocity and longitudinal vehicle axis is called body side slip angle β. The angle between the abscissa-axis and the vehicle heading is called yaw angle or heading angle of the vehicle ψ.

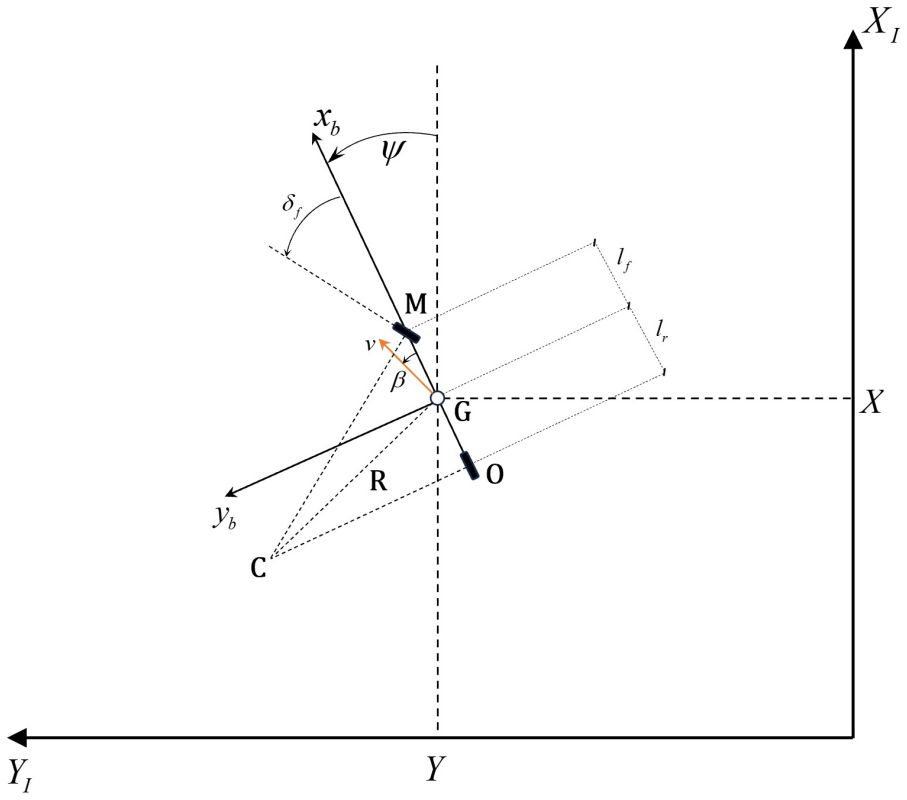


Figure 3.1: Bycicle model schematization

3.2 Reference systems

This section has been added to explain the two reference systems that have been used in the development of the work. As it can be imagined, in localization problems there is the need to express the vehicle position and orientation with respect to a fixed point in the space.

To this aim, considering that the vehicle is moving in the space, with a mobile reference frame fixed on the vehicle itself (centered in 'B' in the figure 3.2) that we call '*body reference frame*'. The '*inertial reference frame*' (centered in 'I' in the figure 3.2) is the fixed one. All the informations regarding the position and orientation of the vehicle are referenced with respect to this latter:

$$p_I(t) = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix} \tag{3.1}$$

Where p_I is the generic vector composed by x, y coordinates and value of yaw angle ψ. In order to move from one reference system to another, simple analytical procedure has to be followed. Considering that the body reference frame is rotating with respect to the fixed one by the angle ψ, vectors in one or in the other reference frame can be related by the following expressions:

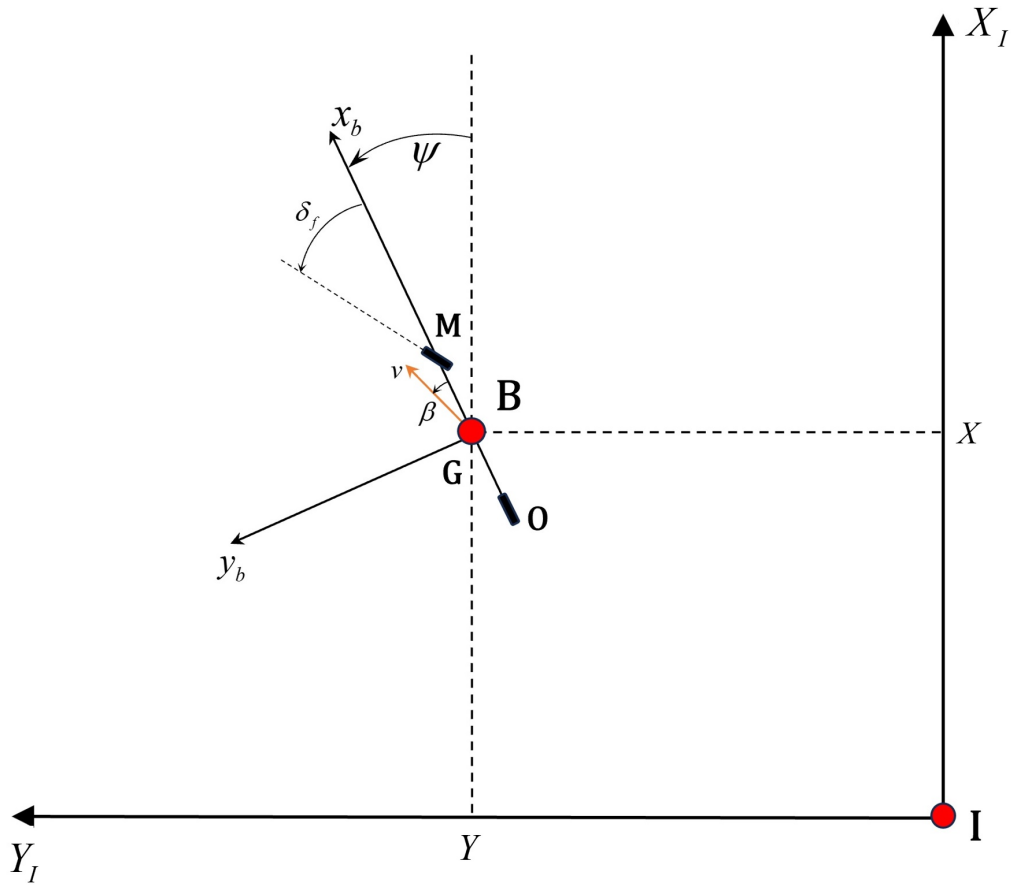


Figure 3.2: Inertial and body reference frames

$$v^B = R_I^B v^I$$
$$v^I = (R_I^B)^\top v^B$$

$$\text{where, } R_I^B = \begin{pmatrix} \cos(\psi) & \sin(\psi) \\ -\sin(\psi) & \cos(\psi) \end{pmatrix} \tag{3.2}$$

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

3.3 Kinematic model

Under the assumptions discussed before and under the following consideration, the vehicle behaviour can be described using this less complicated model.

- The slip angles between the road and the front or rear tyres are considered to be zero. This implies also that velocities vectors v_M and v_O are oriented along the direction of the front and rear wheels respectively. If we suppose that the vehicle is driving at a low speed $v < 5m/s$), since the lateral forces are small, this assumption is reasonable.

In this case, the motion of the vehicle is described independently from the factors influencing the motion itself, namely the forces acting on the system.

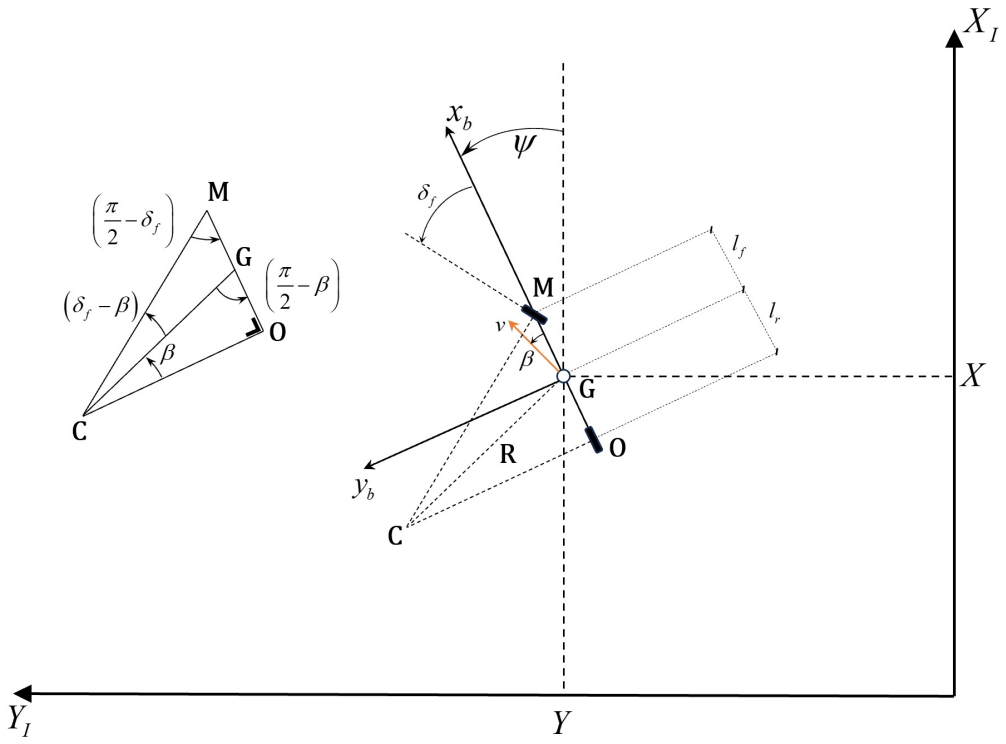


Figure 3.3: Vehicle representation and geometrical highlight

The rate of change of the vehicle's orientation $\dot{\psi}$, in steady-state condition, can be considered equal to the angular velocity of the vehicle itself. Indeed, if the sine-rule is applied to the triangle MOC, assuming that the radius of the path trajectory varies slowly as the vehicle is moving at a low speed $\dot{\psi}$ is equal to the angular velocity of the vehicle itself:

$$\dot{\psi} = \omega = \frac{v}{R}. \tag{3.3}$$

Looking at the two triangles CGM and CGO, it is possible to make some geometrical considerations. Considering the former one:

$$\frac{\sin\left(\delta_f - \beta\right)}{l_f} = \frac{\sin\left(\frac{\pi}{2} - \delta_f\right)}{R} \tag{3.4}$$

Applying the same rule to the other one:

$$\frac{\sin(\beta)}{l_r} = \frac{\sin(\frac{\pi}{2})}{R} = \frac{1}{R} \tag{3.5}$$

Multiplying both sides of 3.4 by $\dfrac{l_f}{\cos(\delta_f)}$

$$\frac{\sin(\delta_f)\cos(\beta) - \cos(\delta_f)\sin(\beta)}{l_f} = \frac{\cos(\delta_f)}{R} \Rightarrow \tan(\delta_f)\cos(\beta) - \sin(\beta) = \frac{l_f}{R} \tag{3.6}$$

Multiplying both sides of 3.5 by l_f

$$\sin(\beta) = \frac{l_r}{R} \tag{3.7}$$

Combining together 3.6 and 3.7 it is possible to obtain:

$$\cos(\beta)\tan(\delta_f) = \frac{l_f + l_r}{R} \tag{3.8}$$

Finally, substituting 3.8 in 3.3 the yaw rate can be expressed as function of model's input variables, v and δ as follows:

$$\dot{\psi} = \frac{v\cos(\beta)}{l_f + l_r}\left(\tan(\delta_f)\right). \tag{3.9}$$

Equation 3.9, is one of the four equations that constitute the complete set of the kinematic model. Considering now that we want to express the velocity of the vehicle, known in the body frame, in the inertial reference frame. As said before, body reference is at every time instant rotated of an angle $\psi(t)$ with respect to the fixed reference one. So, given the velocity vector decomposed in components in the body reference frame as:

$$v = \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} v \cdot \cos(\beta) \\ v \cdot \sin(\beta) \end{bmatrix} \tag{3.10}$$

As explained before, two reference frames are related by geometrical transformation such that:

$$V^{(I)} = \begin{bmatrix} V_X \\ V_Y \end{bmatrix} = R_B^I v^{(B)} = \begin{pmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{pmatrix} \cdot \begin{bmatrix} v \cdot \cos(\beta) \\ v \cdot \sin(\beta) \end{bmatrix} \tag{3.11}$$

Developing equation 3.11, one obtain that:

$$V_X = v \cdot \cos(\psi) \cdot \cos(\beta) - v \cdot \sin(\psi) \cdot \sin(\beta) = v \cdot \cos(\psi + \beta) \tag{3.12}$$

$$V_Y = v \cdot \sin(\psi) \cdot \cos(\beta) + v \cdot \cos(\psi) \cdot \sin(\beta) = v \cdot \sin(\psi + \beta) \tag{3.13}$$

Finally, the body side slip angle as function of model inputs, can be obtained combining eq 3.6 multiplied by l_r and eq 3.7 multiplied by l_f:

$$\begin{aligned} l_r(\tan(\beta)\cos(\beta) - \sin(\beta)) &= \tfrac{l_r \cdot l_f}{R} \\ l_f \cdot \sin(\beta) &= \tfrac{l_r \cdot l_f}{R} \end{aligned} \tag{3.14}$$

From which it can be obtained that:

$$\beta = \tan^{-1}\left(\frac{l_r \tan(\delta_f)}{l_r + l_f}\right) \tag{3.15}$$

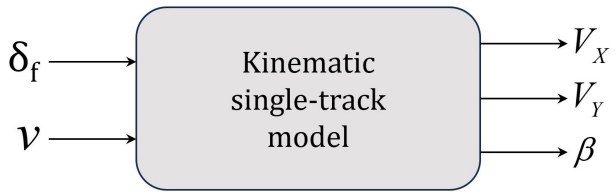


Figure 3.4: Block diagram of kinematic single-track model

Summarizing, the inputs of this model 3.9 are two, namely the front steering angle δ_f and the longitudinal velocity v. In a path tracking framework either both the input can be controlled or velocity can be provided to the plant in open-loop as it has been done in our simulations and in our experimental tests.

To conclude the chapter, the equations of the vehicle kinematic model are grouped in the following:

$$\begin{cases} \dot{\psi} = \frac{v\cos(\beta)}{l_f+l_r}\left(\tan(\delta_f)\right) \\ \beta = \tan^{-1}\left(\frac{l_r\tan(\delta_f)}{l_r+l_f}\right) \\ V_X = v\cos(\psi+\beta) \\ V_Y = v\sin(\psi+\beta) \end{cases} \tag{3.16}$$

3.4 Vehicle lateral analysis

In more challenging conditions the kinematic model does not approximate properly the vehicle's behaviour. At high vehicle speed, it can no more be assumed that the speed at each wheel is in the direction of the longitudinal axis of the wheel itself, so a different approach is required. This new model must take into account for the lateral dynamics of the body. In this framework we are considering the vehicle to be on a curve of radius R at constant speed, in steady state condition.

Before formulating the equation of motion, the description of the forces acting on the vehicle is required, specifically on the forces acting on the wheels. Through experimental observations, it can be stated that lateral tire forces are proportional to the slip-angle of the tire itself. Tire side slip angle denoted as α, represents the angle between the longitudinal axis of the wheel and the direction of the velocity vector of that wheel. For a steering wheel, it must be also taken into account of the steering angle. However, for non-steering wheels, this term is absent. It is possible to determine the side slip angles for both the front and rear wheels using the single-track model's, by means of a balance between the longitudinal and lateral direction of the velocities of the tires and the chassis.

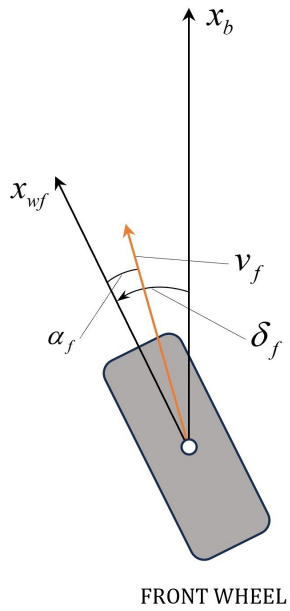


Figure 3.5: Front wheel side slip angle

The slip angle for the front wheel (figure 3.5) can be derived as follows:

$$\text{Lateral direction} \;\Rightarrow\; v_f \sin(\delta_f - \alpha_f) = v\sin(\beta) + l_f\dot{\psi} \tag{3.17}$$

$$\text{Longitudinal direction} \;\Rightarrow\; v_f \cos(\delta_f - \alpha_f) = v\cos(\beta) \tag{3.18}$$

The ratio between 3.17 and 3.18 gives:

$$\tan(\delta_f - \alpha_f) = \frac{l_f\dot{\psi} + v\sin\beta}{v\cos\beta} \tag{3.19}$$

Using small angle approximation such that $\beta \leq 10°$, $\sin(\beta) \cong \beta$, $\cos(\beta) = 1$, $\tan(\beta) = \beta$, equation 3.19 become:

$$\alpha_f = \delta_f - \frac{l_f\dot{\psi} + v\beta}{v} \tag{3.20}$$

The rear tire slip angle (figure 3.6) can be derived as follows:

$$\text{Lateral direction} \;\Rightarrow\; v_r \sin(\alpha_r) = -v\sin(\beta) + l_r\dot{\psi} \tag{3.21}$$

$$\text{Longitudinal direction} \;\Rightarrow\; v_r \cos(\alpha_r) = v\cos(\beta) \tag{3.22}$$

The ratio between 3.21 and 3.22 gives:

$$\tan(\alpha_r) = \frac{l_f\dot{\psi} - v\sin\beta}{v\cos\beta} \tag{3.23}$$

Using small angle approximation such that $\beta \leq 10°$, $\sin(\beta) \cong \beta$, $\cos(\beta) = 1$, $\tan(\beta) = \beta$, equation 3.23 become:

$$\alpha_r = \frac{l_r\dot{\psi} - v\beta}{v} \tag{3.24}$$

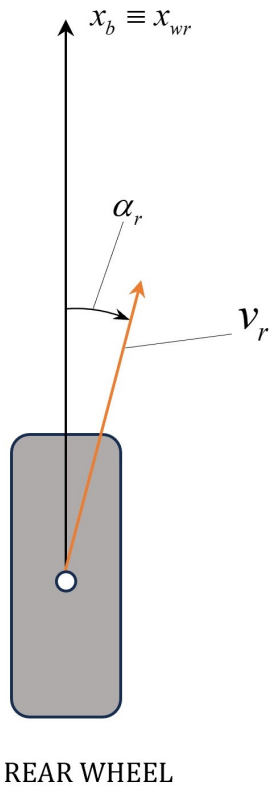


Figure 3.6: Rear wheel side slip angle

Given the formulation of the wheels side slip angles in equations 3.20 and 3.24, the forces generated by the wheel-road interaction that oppose to the centrifugal effect are highlighted in figure 3.7 and described below. Equation 3.25 for the front wheel and equation 3.26 for the rear wheel:

$$F_{yf} = C_{\alpha_f} \cdot \alpha_f = (C_{\alpha_{fl}} + C_{\alpha_{fr}}) \cdot \alpha_f \tag{3.25}$$

$$F_{yr} = C_{\alpha_r} \cdot \alpha_r = (C_{\alpha_{rl}} + C_{\alpha_{rr}}) \cdot \alpha_r \tag{3.26}$$

Where, C_{α_f} and C_{α_r} are the cornering stiffness of the front and rear wheel for the single-track model which are in turn the sum of the cornering stiffness of the two front wheels and of the two rear wheels.
These forces have a particular dependance from the side-slip angle as them are generated by the viscoelastic deformation of the tire. The magnitude of this deformation, as explained in [8] depends on the magnitude of the lateral velocity and on the time spent by the tread in the contact point. The first contribute is proportional to the wheel rotational velocity multiplied by the side slip angle whereas the second is inversely proportional to the rotational velocity of the wheel making so the overall contribute proportional to the side slip angle only. For small lateral velocities, hence for small slip angles, the lateral forces are proportional to the slip angle itself. As soon as the lateral velocitiy become higher, meaning the presence of a considerable tire deformation, the relation become higly non-linear. Figure 3.8 gives an idea of the explained relation.

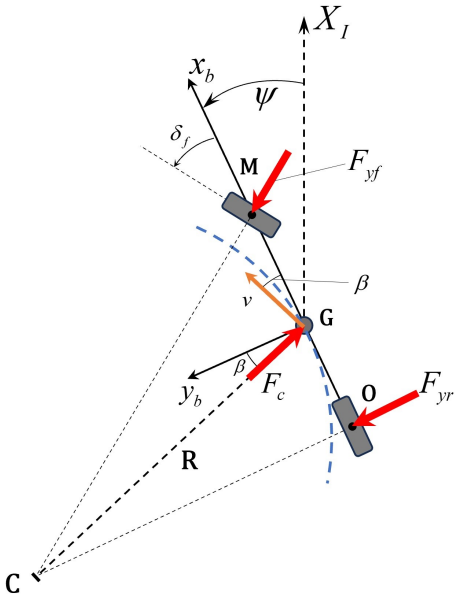


Figure 3.7: Single-track model cornering dynamics

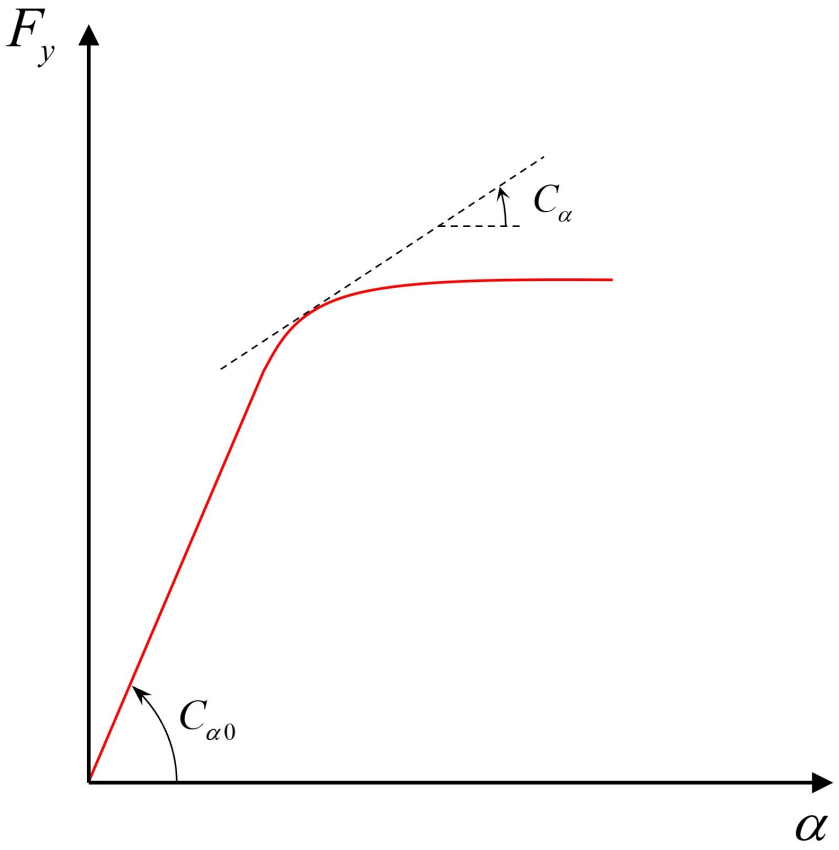


Figure 3.8: Lateral tire-road force as function of tire side-slip angle

The cornering stiffness C_α is the tangent to the curve for each point (α, F_y). For small α angles, it is possible to consider a constant cornering stiffness, denoted as $C_{\alpha 0}$. In the framework of this work, it can be assumed that the condition of small side slip angle is achieved.

3.5 Lateral dynamic equations

After considering the previous explanations it is possible to proceed to derive the equations of the dynamics. Referring to the figure 3.7, it is possible to write the equilibrium of forces along the y axis of the body reference frame:

$$m \cdot a_y = F_c \cos\beta = F_{yr} + F_{yf}\cos(\delta_f) \tag{3.27}$$

In the following, the aim is to present the derivation of the formula of the lateral acceleration of the vehicle ([7]). Starting from the velocity of the vehicle in the vehicle's reference frame $v^b = \begin{bmatrix} v\cos(\beta) \\ v\sin(\beta) \\ 0 \end{bmatrix}$, the acceleration of the body in this reference frame is expressed as:

$$\begin{aligned} a^b = \tfrac{dv^b}{dt} + \omega^b \wedge v^b = \begin{bmatrix} -v\sin(\beta)\cdot\dot\beta \\ v\cos(\beta)\cdot\dot\beta \\ 0 \end{bmatrix} + \begin{pmatrix} 0 \\ 0 \\ \dot\psi \end{pmatrix} \wedge \begin{bmatrix} v\cos(\beta) \\ v\sin(\beta) \\ 0 \end{bmatrix} = \\ = \begin{bmatrix} -v\sin(\beta)\cdot\dot\beta \\ v\cos(\beta)\cdot\dot\beta \\ 0 \end{bmatrix} + \begin{bmatrix} -v\sin(\beta)\cdot\dot\psi \\ v\cos(\beta)\cdot\dot\psi \\ 0 \end{bmatrix} = \begin{bmatrix} -v\sin(\beta)\cdot(\dot\beta+\dot\psi) \\ v\cos(\beta)\cdot(\dot\beta+\dot\psi) \\ 0 \end{bmatrix} \end{aligned} \tag{3.28}$$

The acceleration of the vehicle, having assumed constant longitudinal velocity, has only the normal component which magnitude is given by:

$$|a^b| = {a_n}^b = v\cdot(\dot\beta+\dot\psi) \tag{3.29}$$

The acceleration component perpendicular to the longitudinal direction of the vehicle under the assumption of small side slip angle ($\cos\beta \cong 1$, such that $v_x = v\cdot\cos(\beta) \cong v$) is the following:

$$a_y = v\cos\beta(\dot\beta+\dot\psi) \cong v(\dot\beta+\dot\psi) \tag{3.30}$$

From the balance of the forces along y-direction (refer to figure 3.7) it is possible to write the equation for the lateral dynamics:

$$mv\cdot(\dot\beta+\dot\psi) = F_c\cos\beta = F_{yr} + F_{yf}\cos(\delta_f) \tag{3.31}$$

Substituting expressions 3.20, 3.25, 3.24, 3.26 in the previous equation:

$$\begin{aligned} mv\cdot(\dot\beta+\dot\psi) = C_{\alpha r}\alpha_r + C_{\alpha f}\alpha_f\cos(\delta_f) = \\ = C_{\alpha r}\left(\tfrac{l_r\dot\psi}{v} - \beta\right) + C_{\alpha f}\cos(\delta_f)\left(\delta_f - \tfrac{l_f\dot\psi}{v} - \beta\right) \end{aligned} \tag{3.32}$$

Under the same assumptions of small front steering angles (such that $\cos\delta_f \cong 1$), the previous equations can be written as:

$$mv\dot\beta + \frac{\dot\psi}{v}\left(mv^2 - C_{\alpha r}l_r + C_{\alpha f}l_f\right) + \beta\left(C_{\alpha f} - C_{\alpha r}\right) = C_{\alpha f}\delta_f \tag{3.33}$$

That is the equation for the vehicle lateral dynamics.
Based on the balance of the moments around the z-axis of the vehicle's centre of mass it is possible to derive that:

$$I_z\ddot{\psi} = l_f F_{yf} - l_r F_{yr} \tag{3.34}$$

Substituting equations 3.20, 3.25, 3.24, 3.26 in the previous equation:

$$I_z\ddot{\psi} = l_f F_{yf} - l_r F_{yr} = l_f C_{\alpha f}\left(\delta_f - \frac{l_f\dot{\psi}}{v} - \beta\right) - l_r C_{\alpha r}\left(\frac{l_r\dot{\psi}}{v} - \beta\right) \tag{3.35}$$

Which can be rewritten as:

$$I_z\ddot{\psi} + \frac{\dot{\psi}}{v}\left(\frac{C_{\alpha f}{l_f}^2}{v} + \frac{C_{\alpha r}{l_r}^2}{v}\right) + \beta\left(C_{\alpha f}l_f - C_{\alpha r}l_r\right) = C_{\alpha f}l_f\delta_f \tag{3.36}$$

That is the equation for the vehicle yaw dynamics.

In conclusion the equations (3.33, 3.36) for the dynamics of the vehicle within the assumptions of the single-track model are grouped below:

$$\begin{cases} mv\dot{\beta} + \frac{\dot{\psi}}{v}\left(mv^2 - C_{\alpha r}l_r + C_{\alpha f}l_f\right) + \beta\left(C_{\alpha f} - C_{\alpha r}\right) = C_{\alpha f}\delta_f \\ I_z\ddot{\psi} + \frac{\dot{\psi}}{v}\left(\frac{C_{\alpha f}{l_f}^2}{v} + \frac{C_{\alpha r}{l_r}^2}{v}\right) + \beta\left(C_{\alpha f}l_f - C_{\alpha r}l_r\right) = C_{\alpha f}l_f\delta_f \end{cases} \tag{3.37}$$

As done before a scheme of the dynamic single-track model is presented below. Inputs are front steering angle and velocity, that are provided in closed and open loop respectively. As outputs we have the states of the model which are the yaw-rate $\dot{\psi}$ and the side-slip angle β.

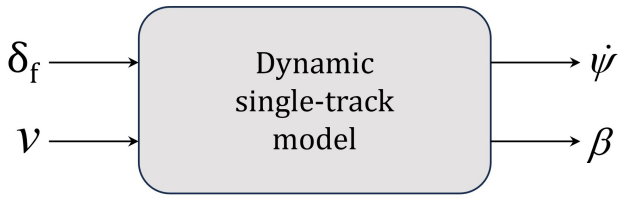


Figure 3.9: Block diagram of dynamic single-track model

3.6 Equations for the longitudinal model

The equations for the longitudinal model are derived leveraging the equations of a DC machine. The first equation is derived from Kirchoff law while the second one of the driving torque is obtained from the system dynamic equilibrium.

$$\begin{cases} V_a = R_a i_a + L_a \frac{di_a}{dt} + E \\ \\ C_m - C_r = J \frac{d\omega}{dt} + B\omega \end{cases} \tag{3.38}$$

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. We are considering steady-state analysis so this term can be neglected;
- E is the back electro-motive-force;
- C_m is the driving torque provided by the motor;
- C_r is the load torque due to actions opposing the motion;
- $J \frac{d\omega}{dt}$ is the product between motor inertia and motor angular acceleration; item $B\omega$ is the product between viscous friction and motor angular speed.

For a DC motor it is possible to express the driving torque and the back e.m.f as follows:

$$\begin{cases} C_m = K_m \Phi i_a \\ E = K_e \Phi \omega \end{cases} \tag{3.39}$$

where,

- K_m is a constant that depends on the specific characteristics of the machine;
- Φ is the excitation flux, a constant for permanent magnet machines;
- ω is the motor angular speed;
- K_e is again a motor constant that depends on specific characteristichs of the device.

Putting together 3.38, 3.39, considering that $K_t = K_m \Phi$ and $K_v = K_e \Phi$ it is possible to obtain the following:

$$\begin{cases} i_a = \frac{V_a - K_v \omega}{R_a} \\ K_t i_a - C_r = J \frac{d\omega}{dt} + B\omega \end{cases} \tag{3.40}$$

Substituting the expression of armature current i_a in the second equation:

$$K_t \frac{V_a - K_v \omega}{R_a} - C_r = J \frac{d\omega}{dt} + B\omega \tag{3.41}$$

Rewriting the equation as explicit funciton of angular acceleration:

$$\dot{\omega} = \frac{1}{J}(\frac{K_t}{R_a}(V_a - K_v\omega) - B\omega - C_r) \tag{3.42}$$

Equation 3.42 is referred to the electric motor, however under the assumptions of no longitudinal wheels slip and small vehicle side-slip angle, this relation can be linked to the wheel speed through the transmission ratio τ:

$$v_{wh} = \frac{1}{\tau} \cdot \omega r_{wh} \tag{3.43}$$

Equation 3.42 will be used in section 5.3 and can be rewritten as:

$$\dot{\omega} = \frac{K_t}{J \cdot R_a}V_a + (\frac{K_t K_v}{JR_a} - B)\omega - \frac{C_r}{J} \tag{3.44}$$

or, equivalently:

$$\dot{\omega} = P_1 V_a - P_2\omega - P_3 \tag{3.45}$$

Chapter 4

Trajectory generation and derivation of state-space representation

In this chapter we want to describe the metodology used to generate the trajectory used as reference for path tracking problem. This part plays a crucial role since the curvature that derives from the created trajectory is used for the states generation whithin the control environment. In this work, five trajectories are presented:

- U-shaped trajectory;
- S-shaped trajectory;
- Trajectory for obstacle avoidance, as example of severe lane-change manoeuvre.
- Circular-shaped trajectory;
- Eight-shaped trajectory.

Notice that in the design process, the reference environment taken into consideration was the laboratory test track, visible in figure 4.1.

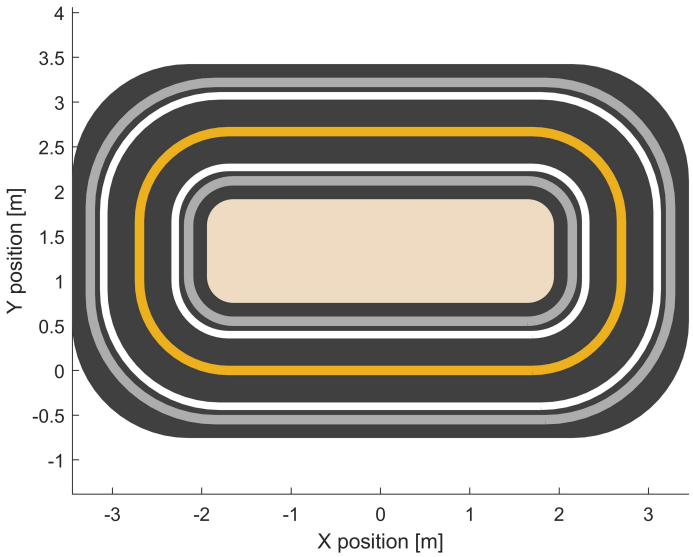


Figure 4.1: Test track used in the experimental setup.

4.1 U-shaped trajectory

The procedure begins with geometrical considerations: from (0,0) position we designed a path that consists of two straight horizontal segments connected by a semicircular arc. All the measures, unless otherwise specified, are provided in meters.

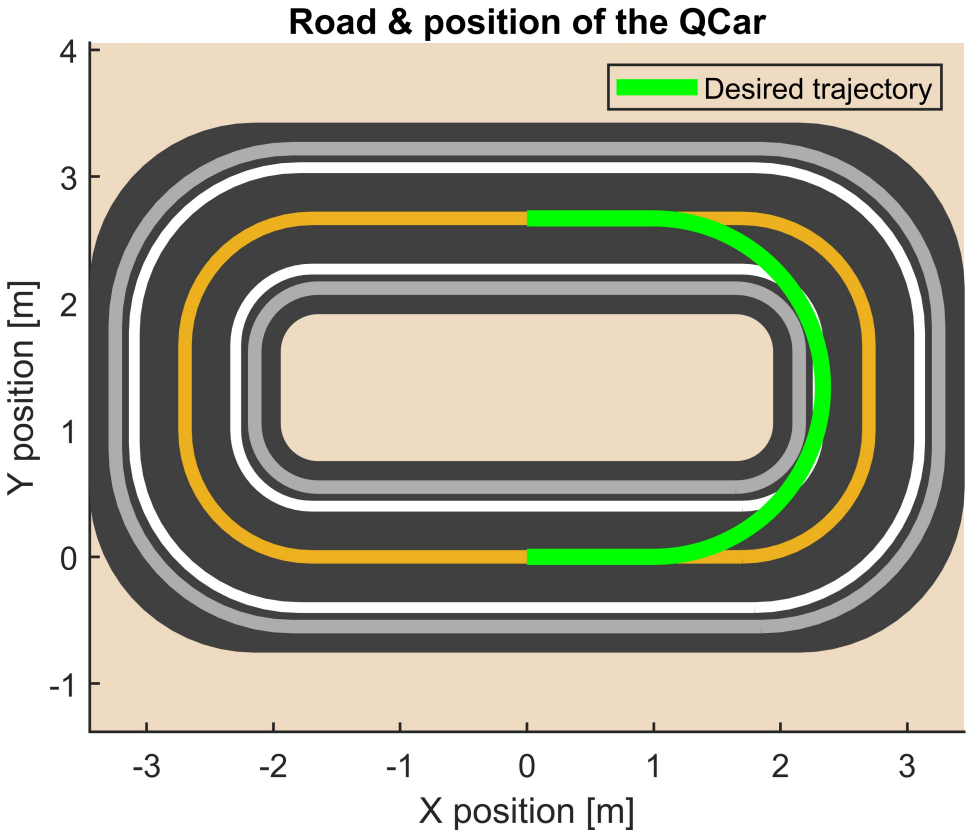


Figure 4.2: U-shaped trajectory drawn on the experimental track

The elements which constitute the path in figure 4.2 are detailed below:

- Straight segment 1 m long starting at (X,Y) = (0,0), ending at (X,Y) = (1, 0);
- Semicircumference with radius r = 1.335 m starting at (X,Y) = (1,0), ending at (X,Y) = (1, 2.67);
- straight segment 1m long starting at (X,Y)=(1, 2.67), ending at (X,Y) = (0, 2.67).

After these geometrical elements coordinates have been defined, they must be combined together to produce a first reference vector. It is crucial to emphasise the requirement for consecutive vectors not to overlap in order to prevent issues with the following procedure. This first part can be appreciated in the extract of code below:

Listing 4.1: Geometrical definition of the entities used to construct the trajectory

```
%  Initial straight segment
y_in = zeros(1, 1000);                  % straight line at y = 0
x_in = linspace(0,0.999, 1000);         % straight line from x=0 to x=1
```

```
% Semicircumference
r = 1.335;                                  % radius
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);                 % straight line at y = 2.67
x_fin = linspace(0.999, 0, 1000);           % straight line from x=1 to x=0

x_f = [x_in, x, x_fin]';                    % final coordinates
y_f = [y_in, y, y_fin]';                    % final coordinates

xRef = x_f;
yRef = y_f;

refpos = [xRef, yRef];  % vector containing the column vectors of x and
                        % y coordinates for each point of the trajectory
```

The procedure goes on with the interpolation over a defined number of points and smoothing of the reference trajectory as reported in the following code extract which allows to obtain the reference coordinates X_{Ref}, Y_{Ref} and the reference heading angle ψ_{Ref} that are reported in figure 4.3.

Listing 4.2: Geometrical definition of the entities used to construct the trajectory

```
%% Distance computation
% pdist command is used to compute the euclidean distance between each
% couple of points (xRef, yRef). Squareform command creates a symmetric
% and square matrix, making the data provided by pdist readable, so
% that Z(i,j) denotes the distance between the i and j objects in the
% original data

distancematrix = squareform(pdist(refpos));
distancesteps = zeros(length(refpos)-1,1);

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

totalDistance = sum(distancesteps);             %Total distance
distbp = cumsum([0; distancesteps]);            %Cumulative distance
s = linspace(0, totalDistance, 300);            %curvilinear abscissa

% It is possible to regulate the sampling rate for the curvilinear
% abcissa acting on N parameter of linspace command. In the following,
% the interp1 command will interpolate to find the xRef2 and yRef2
% values of the function y = f(x), considering y to be xRef and yRef,
% x to be distbp, at the query points s. Successively the interpolated
% values are smoothed with the corresponding command.

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);
```

The last step is to calculate the curvature for each point of the trajectory. Curvature is the reciprocal of the curvature radius that represents the radius of circular arc which best approximates the curve at that point.
The radius of curvature of the curve given in parametric coordinates as (x(s), y(s)), is given by Equation 4.1 ([9]):

$$\rho(s) = \frac{1}{R} = \frac{x'y'' - x''y'}{(x'^2 + y'^2)^{3/2}} \tag{4.1}$$

Where:

$$x' = \frac{dx}{ds} \tag{4.2}$$

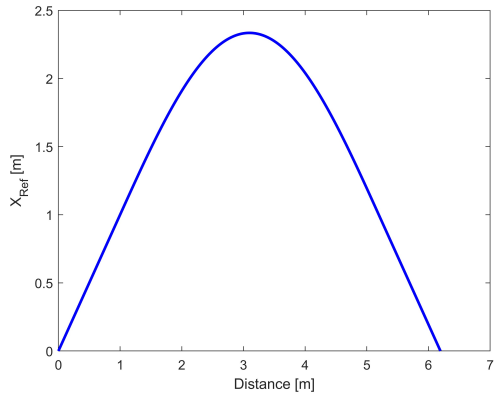
$$x'' = \frac{dx'}{ds} \tag{4.3}$$

The implementation of this formula is reported in the code extract below.

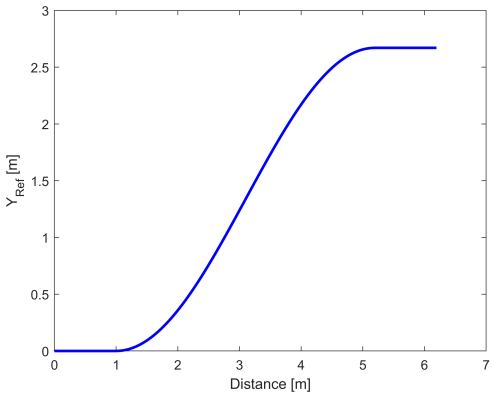
Listing 4.3: Computation of curvature value for each point of the trajectory

```
% 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
```

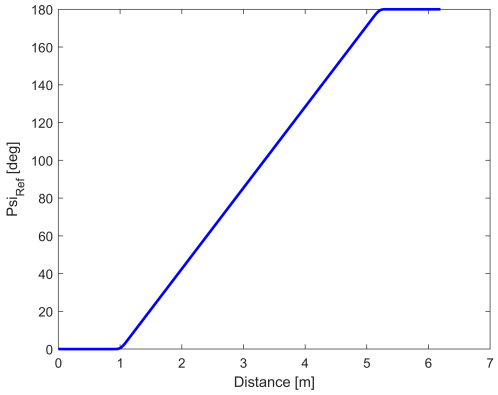
In the following figures the reference variables for this trajectory are presented.



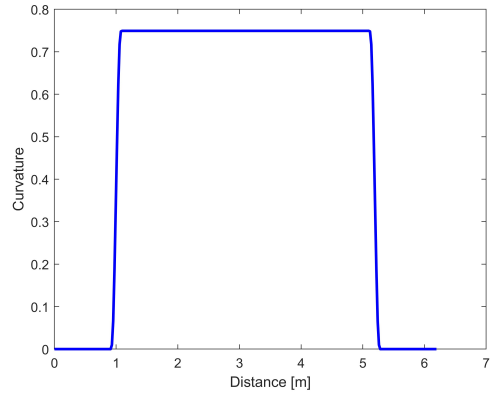
(a) X reference coordinate for U-shaped trajectory



(b) Y reference coordinate for U-shaped trajectory



(c) Reference heading angle ψ_{Ref} for U-shaped trajectory



(d) Reference curvature ρ_{Ref} for U-shaped trajectory

Figure 4.3

4.2 S-shaped trajectory

With some additional geometric considerations, the process used to construct this second trajectory is identical to the one previously discussed. The reference trajectory begins at (0,0) and consists of a straight horizontal segment, three consecutive semicircumferences and a final vertical straight segment.
The entire reference trajectory, represented in figure 4.4 is made up of these elements:

- straight segment 1m long starting at (X,Y) = (0,0), ending at (X,Y) = (1.334, 0);
- a quarter of circumference with radius r = 1.335m starting at (X,Y) = (1.335,0), ending at (X,Y) = (0, 1.335);
- semicircumference with radius r = 1.335m starting at (X,Y) = (0, 1.335), ending at (X,Y) = (-2.67, 1,335);
- semicircumference with radius r = 1.335m starting at (X,Y) = (0, 1.335), ending at (X,Y) = (-2.335, 1,335);
- straight segment 1m long starting at (X,Y)=(-2.67, 1.335), ending at (X,Y) = (-2.67, 2.335).

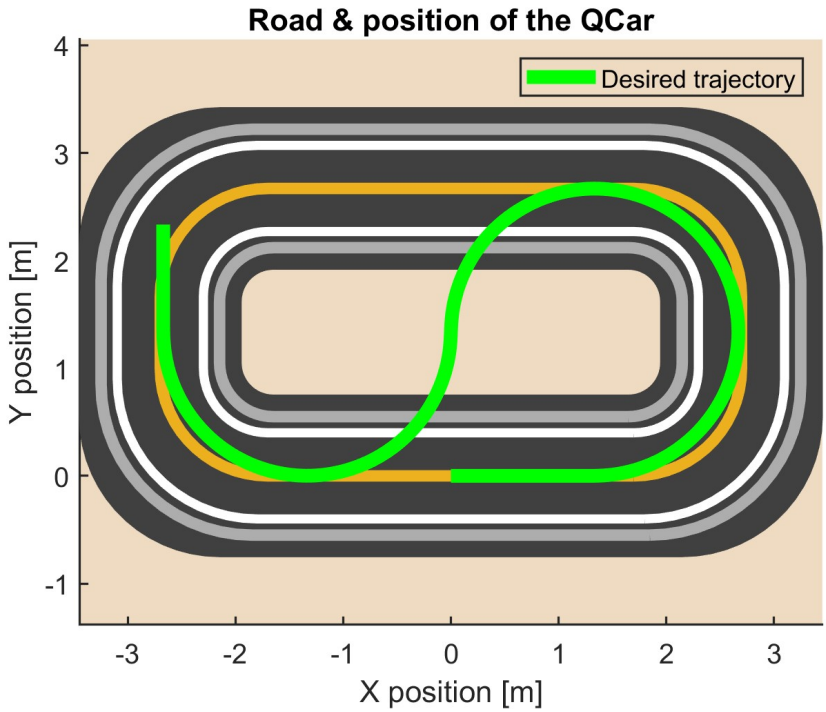


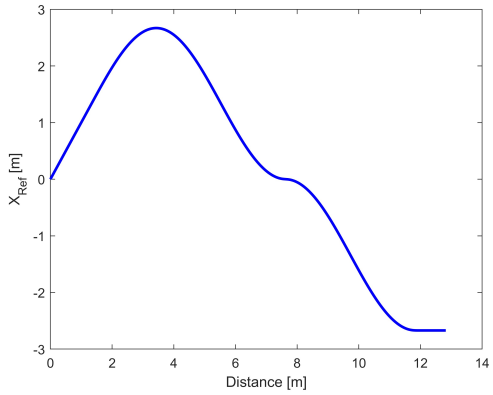
Figure 4.4: S-shaped trajectory drawn on the experimental track

As previously done, an extract of the code required to define and combine all the mentioned entities is presented below. This code creates the reference trajectory vector, which is then used in the smoothing operation and to determine the reference coordinates, the reference heading angle and the point-to-point curvature. These entities are reported in figure 4.5.

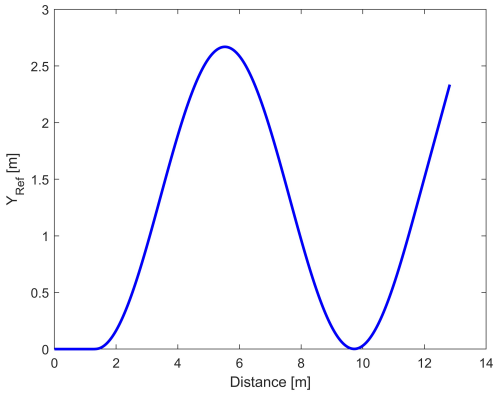
Listing 4.4: Geometrical definition of the entities used to construct the trajectory

```
% Initial straight segment
r = 1.335;   %radius
theta0 = linspace(-pi/2, 0, 1000);   % angles from 0 to pi
y_in = zeros(1, 1000);               % straight line at y = 0
x_in = linspace(0, 1.334, 1000);   % straight line from x=0 to x=1.334

% First 1/4 of circumference
x0 = r*cos(theta0) + 1.335 ;         % x coordinates
y0 = r*sin(theta0) + r;              % y coordinates

% 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 straight line
y_fin = linspace(r+0.001, r+1, 1000);   % straight line from
                                        % y = 1.336 to y = 2.335
x_fin = -2*r*ones(1, 1000)- 0.001;      % straight line at x=-2.67


x_f = [x_in, x0, x1, x2, x_fin]';       % final coordinates
y_f = [y_in, y0, y1, y2, y_fin]';       % final coordinates

xRef = x_f;
yRef = y_f;

refpos = [xRef, yRef];
```

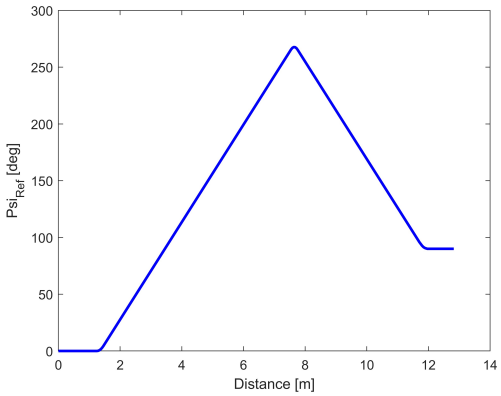
In the following figures the reference variables for this trajectory are presented.



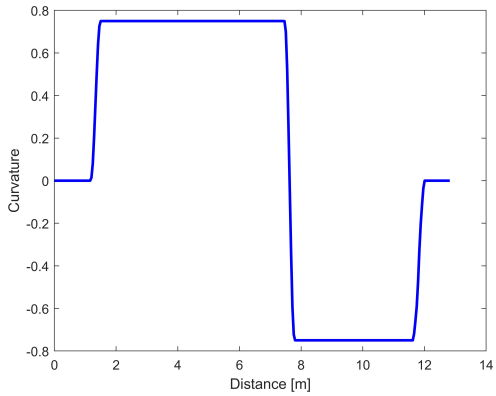
(a) X reference coordinate for U-shaped trajectory



(b) Y reference coordinate for U-shaped trajectory



(c) Reference heading angle ψ_{Ref} for U-shaped trajectory



(d) Reference curvature ρ_{Ref} for U-shaped trajectory

Figure 4.5

4.3 Trajectory for obstacle-avoidance manouevure

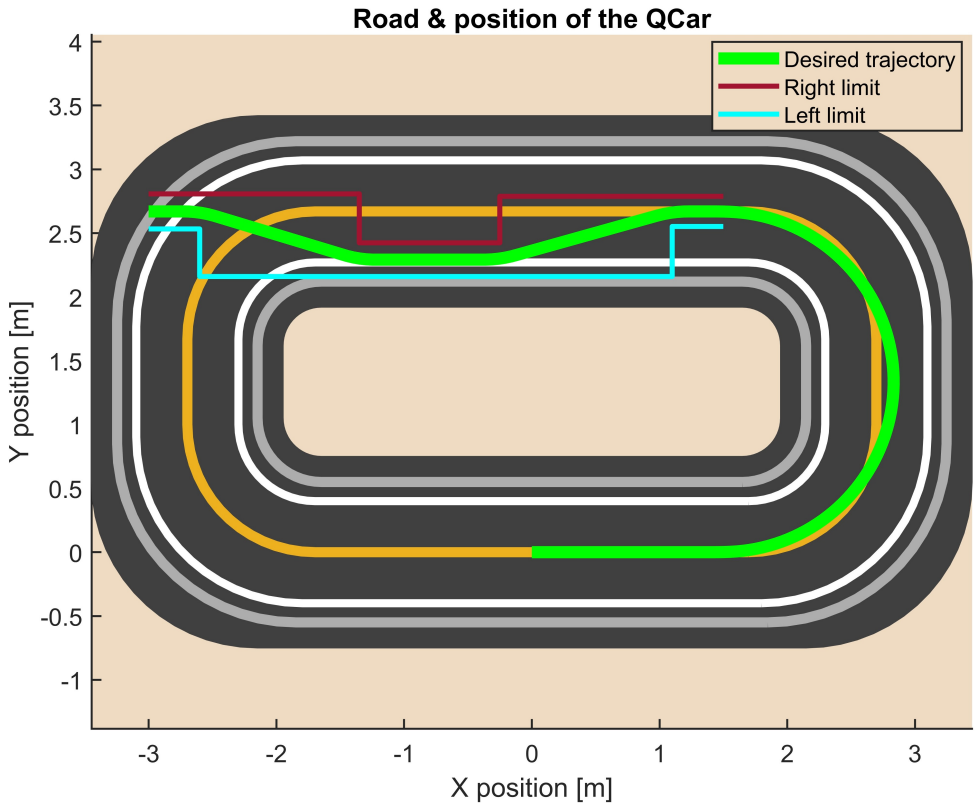


Figure 4.6: Obstacle-avoidance trajectory drawn on the experimental track

In order to design the obstacle avoidance section of the complete trajectory reported in figure 4.6, the normative [10] was consulted. However, some adjustments were required owing to the dimension of the test vehicle and the constraints of the test track.

- The vehicle being considered is in a 1:10 scale, therefore the measures of the track are scaled by a factor of 10;
- Vehicle width is equal to 0.192 m;
- The initial and final sectors of the track were shortened from 1.2 m to 0.4 m and from 1.2 m to 0.12 m, respectively.

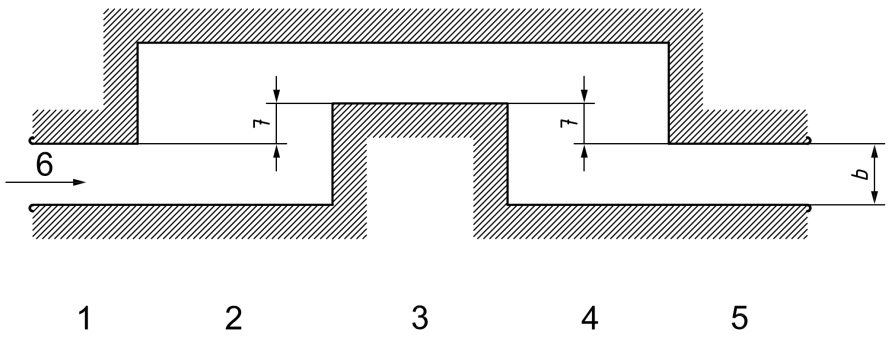


Figure 4.7: Obstacle avoidance track with designation of sections from ISO regulation.

Section	**Lenght**	**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 4.1: Obstacle avoidance track dimension in **meters**

Finally, the lane change maneuvre section which schematic is reported in figure 4.7, was combined with the remaining portions of trajectory listed in the bullet list below, such that the QCar starting point was at the halfway point of the track lower straight segment, where better lidar behaviour is obtained:

- Straight segment 1 m long starting at (X,Y) = (0,0), ending at (X,Y) = (1, 0);
- Semicircumference with radius r = 1.335 m starting at (X,Y) = (1,0), ending at (X,Y) = (1, 2.67).

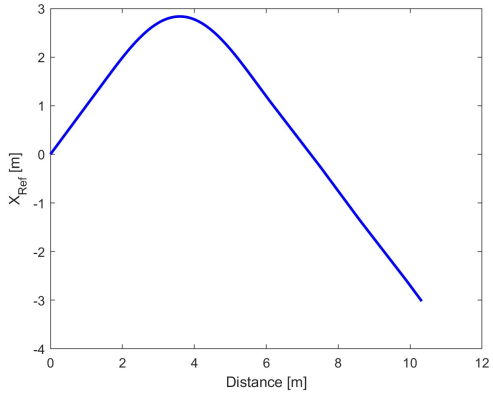
Finally, the code extract where the trajectory is created is reported below.

Listing 4.5: Computation of curvature value for each point of the trajectory

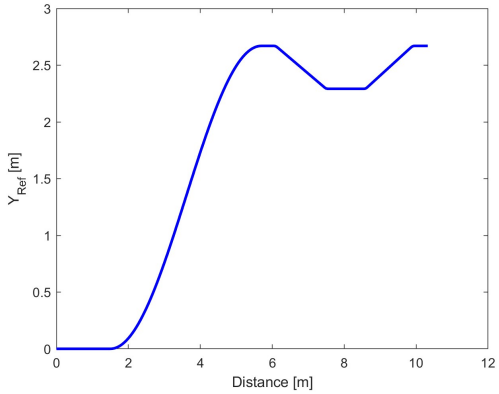
```
%% Initial straight segment
interpPoints = 1000;
y_in = zeros(1, 1000);                          % straight line at y = 0
x_in = linspace(0,1.499, interpPoints);         % straight line from x=0 to
                                                % x=1.499
% Semicircumference
r = 1.335;                                          %radius
theta = linspace(-pi/2, pi/2, interpPoints);
x = r*cos(theta) + 1.5 ;                            % x coordinates
y = r*sin(theta) + r;                               % y coordinates

%% Obstacle avoidance section
s_init = 0;
lane_offset = 0.1;

% Test limit definition
vehicle_width = 0.192;
x_init = x(end) - 0.001;
x_1 = x_init - 0.4;
x_2 = x_1 - 1.35;
x_3 = x_2 - 1.1;
x_4 = x_3 - 1.25;
x_5 = x_4 - 0.3;
x_end = x_5 - 0.12;

% Right limit
reference_path_tmp.x_right = [x_init; x_init-10^-10; x_1 ; x_1-10^-10;
                              x_2; x_2-10^-10; x_3; x_3-10^-10; x_4;
                              x_4-10^-10; x_5; x_end];

y_right_1 = y(end) + (1.1*vehicle_width+0.025)/2;
```

```
y_right_2 = y_right_1;
y_right_3 = y(end) - lane_offset - ((vehicle_width+0.1)/2);
y_right_4 = y(end) + (0.3)/2;  % (1.3*vehicle_width+0.025) but not less
                               % than 0.3m
y_right_5 = y_right_4;
reference_path_tmp.y_right = [y_right_1;y_right_1;y_right_1;y_right_2;
                              y_right_2; y_right_3;y_right_3;y_right_4;
                              y_right_4;y_right_5; y_right_5;y_right_5];

% Left limit
reference_path_tmp.x_left = [x_init; x_init-10^-10; x_1 ; x_1-10^-10;
                             x_2; x_2-10^-10; x_3; x_3-10^-10; x_4;
                             x_4-10^-10; x_5; x_end];

y_left_1 = y(end) - (1.1*vehicle_width+0.025)/2;
y_left_2 = y_left_1 - lane_offset - (vehicle_width+0.1);
y_left_3 = y_left_2;
y_left_4 = y_left_2;
y_left_5 = y(end) - (0.3)/2;  % (1.3*vehicle_width+0.025) but not less
                              % than 0.3m

reference_path_tmp.y_left=[y_left_1;y_left_1;y_left_1;y_left_2;y_left_2;
                           y_left_3;y_left_3;y_left_4;y_left_4;y_left_5;
                           y_left_5;y_left_5];

% Mid line
reference_path_tmp.x_mid = reference_path_tmp.x_left;
y_mid_1 = mean([y_left_1 y_right_1]);
y_mid_2 = mean([y_left_2 y_right_2]);
y_mid_3 = mean([y_left_3 y_right_3]);
y_mid_4 = mean([y_left_4 y_right_4]);
y_mid_5 = mean([y_left_5 y_right_5]);
reference_path_tmp.y_mid = [y_mid_1;y_mid_1;y_mid_1;y_mid_1;y_mid_3;
                            y_mid_3;y_mid_3;y_mid_3;y_mid_5;y_mid_5;
                            y_mid_5;y_mid_5];

%% Calculate reference vectors with the same length
reference_path_tmp.x_ref=reference_path_tmp.x_mid;
reference_path_tmp.y_ref=reference_path_tmp.y_mid;

% Interpolate the original vectors
x_interp = interp1(1:numel(reference_path_tmp.x_ref),...
          reference_path_tmp.x_ref,...
          linspace(1, numel(reference_path_tmp.x_ref), interpPoints));

y_interp = interp1(1:numel(reference_path_tmp.y_ref),...
          reference_path_tmp.y_ref,...
          linspace(1, numel(reference_path_tmp.y_ref), interpPoints));

x_f = [x_in, x, x_interp]';                % final coordinates
y_f = [y_in, y, y_interp]';                % final coordinates
xRef = x_f;
yRef = y_f;

refpos = [xRef, yRef];
```

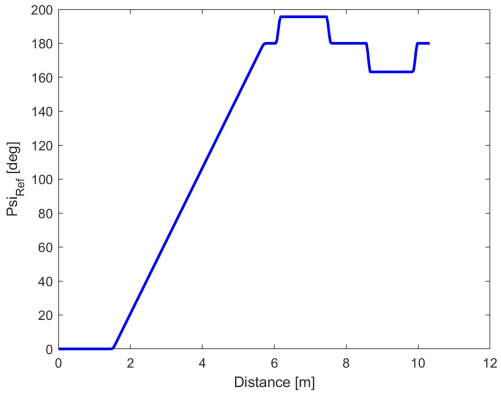
In the following figures the reference variables for this trajectory are presented.



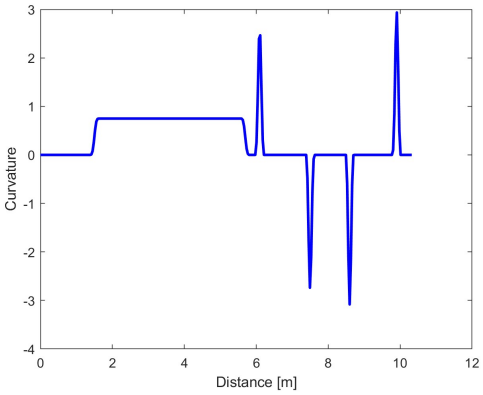
(a) X reference coordinate for Obstacle-avoidance trajectory



(b) Y reference coordinate for Obstacle-avoidance trajectory



(c) Reference heading angle ψ_{Ref} for Obstacle-avoidance trajectory



(d) Reference curvature ρ_{Ref} for Obstacle-avoidance trajectory

Figure 4.8

4.4 Circular trajectory

This section to present the geometrical considerations used to create this path. The reference trajectory begins at (0,0) and consists of an initial straight segment followed by a complete circumference, ending with a final straight segment.
The entire reference trajectory, represented in figure 4.9 is made up of these elements:

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

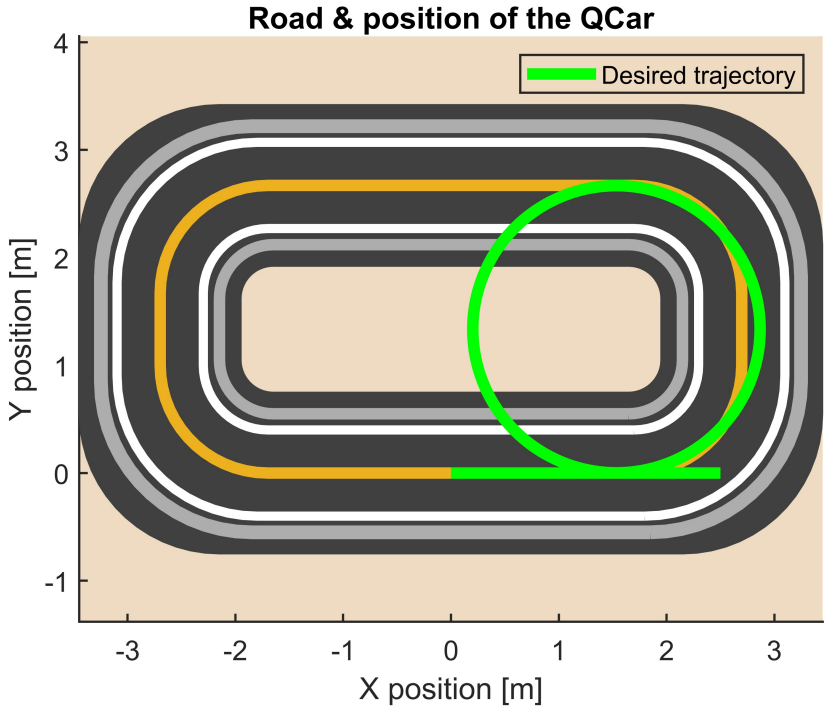


Figure 4.9: Circular trajectory drawn on the experimental track

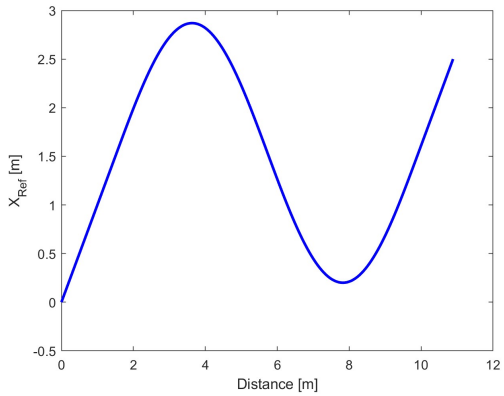
As previously done, an extract of the code required to define and combine all the mentioned entities is presented below. This code creates the reference trajectory vector, which is then used for the smoothing operation and to determine the point-to-point curvature.

Listing 4.6: Geometrical definition of the entities used to construct the trajectory

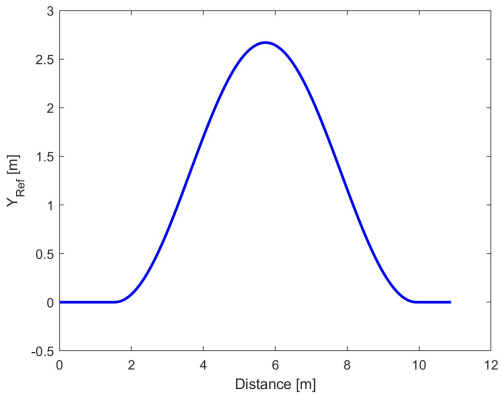
```
%% Initial straight segment
y_in = zeros(1, 1000);                    % straight line at y=0
x_in = linspace(0,r+0.199, 1000);         % straight line from
                                          % x=0 to x=1.534

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

```
% Second circumference (as the first one)
% Third circumference  (as the first one)

% Final straight segment
y_fin = zeros(1, 1000);                          % straight line at y=0
x_fin = linspace(x(end)+0.001, 2.5, 1000);  % straight line from
                                                  x=1.536 to x=2.5

% Final coordinates
x_f = [x_in, x, x_fin]';                    % final coordinates
y_f = [y_in, y, y_fin]';                    % final coordinates

xRef = x_f;
yRef = y_f;

refpos =[xRef, yRef]; % vector containing the column vectors of x and y
                      % coordinates for each point of the trajectory
```

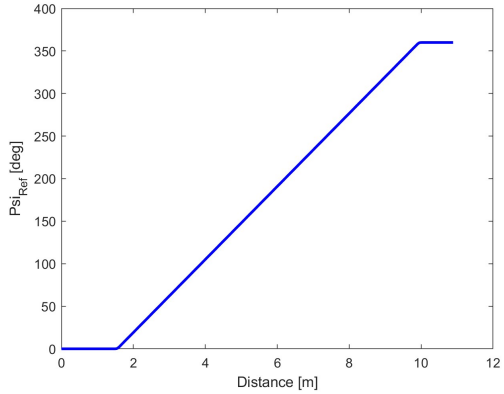
In the following figures the reference variables for this trajectory are presented.



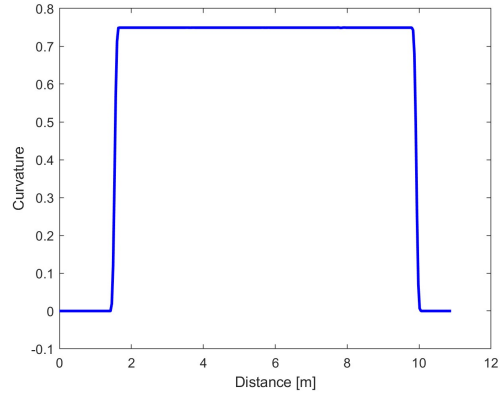
(a) X reference coordinate for Circular trajectory



(b) Y reference coordinate for Circular trajectory



(c) Reference heading angle ψ_{Ref} for Circular trajectory



(d) Reference curvature ρ_{Ref} for Circular trajectory

Figure 4.10

4.5 Eight-shaped trajectory

This section to present the geometrical considerations used to construct this path. The reference trajectory begins at (0,0) and consists of an initial straight segment followed by a complete circumference followed by a final straight segment.
The entire reference trajectory, represented in figure 4.9 is made up of these elements:

- straight segment starting at (X,Y) = (0,0), ending at (X,Y) = (1.335, 0);
- quarter of circumference with radius r = 1.335m starting at (X,Y) = (1.335, 0), ending at (X,Y) = (2.67, 1.335), with center in (X,Y) = (1.335, 1.335);
- semicircumference with radius r = 1.335m starting at (X,Y) = (2.67, 1.335), ending at (X,Y) = (0, 1.335), with center in (X,Y) = (1.335, 1.335);
- semicircumference with radius r = 1.335m starting at (X,Y) = (0, 1.335) and ending at (X,Y) = (-2.67, 1.335), with center in (X,Y) = (-1.335, 1.335);
- semicircumference with radius r = 1.335m starting at (X,Y) = (-2.67, 1.335) and ending at (X,Y) = (1.335, 0), with center in (X,Y) = (-1.335, 1.335);
- quarter of circumference with radius r = 1.335m starting at (X,Y) = (0, 1.335) and ending at (X,Y) = (1.335, 0), with center in (X,Y) = (1.335, 1.335);
- final straight segment starting at (X,Y)=(1.335, 0), ending at (X,Y) = (2.5, 0).

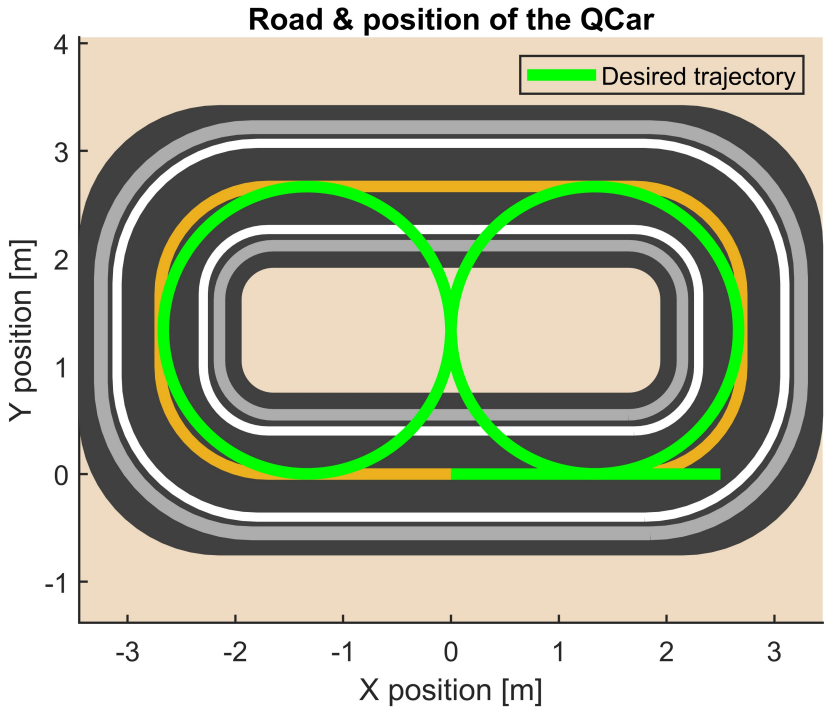


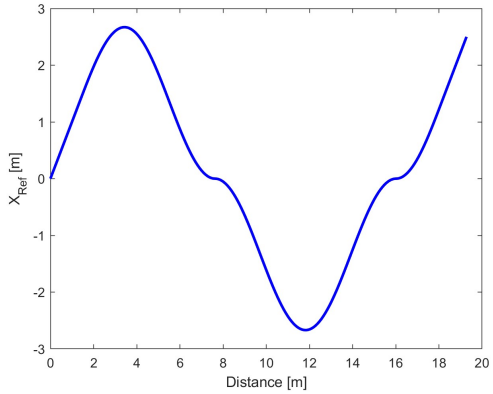
Figure 4.11: Eight-shaped trajectory drawn on the experimental track

As previously done, an extract of the code required to define and combine all the mentioned entities is presented below. This code creates the reference trajectory vector, which is then used to for the smoothing operation and to determine the point-to-point curvature.

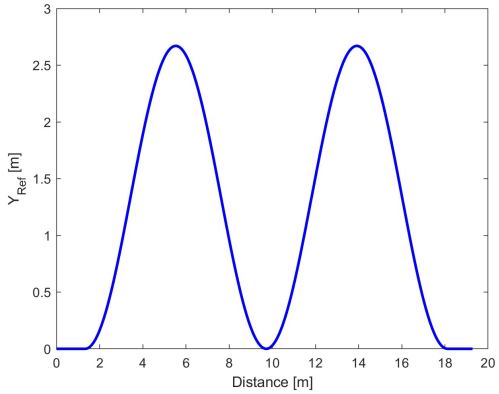
Listing 4.7: Geometrical definition of the entities used to construct the trajectory

```
%%  Initial straight segment
y_in = zeros(1, 1000);           % straight line at y=0
x_in = linspace(0, 1.349, 1000); % straight line from
                                 % x=0 to x=1.349

% first 1/4 of circumference
r = 1.335;                       %radius
theta0 = linspace(-pi/2, 0, 1000);
x0 = r*cos(theta0) + r ;         % x coordinates
y0 = r*sin(theta0) + r;          % y coordinates

% second semicirfumference
theta1 = linspace(0.001, pi, 1000);
x1 = r*cos(theta1) + r;          % x coordinates
y1 = r*sin(theta1) + r;          % y coordinates

% third semicircumference
theta2 = linspace(0.001, pi, 1000);
x2 = r*cos(theta2) - r;          % x coordinates
y2 = -r*sin(theta2) + r;         % y coordinates

% fourth semicircumference
theta3 = linspace(pi+0.001, 0.001, 1000);
x3 = r*cos(theta3) - r;          % x coordinates
y3 = r*sin(theta3) + r;          % y coordinates

% fifth quarter of semicircumference
theta4 = linspace(pi, pi/2, 1000);
x4 = r*cos(theta4) + r;          % x coordinates
y4 = -r*sin(theta4) + r;         % y coordinates


% final vertical straight line
y_fin = zeros(1, 1000);              % straight line at y=0
x_fin = linspace(r+0.001, 2.5, 1000); % straight line from
                                     % x=1.336 to x=25

x_f = [x_in, x0, x1, x2, x3, x4, x_fin]'; % final coordinates

y_f = [y_in, y0, y1, y2, y3, y4, y_fin]'; % final coordinates

xRef = x_f;
yRef = y_f;

refpos = [xRef, yRef];
```

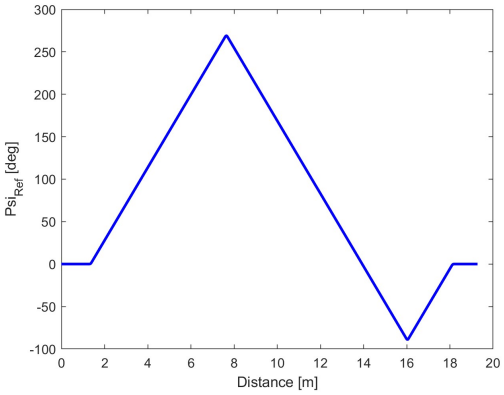
In the following figures the reference variables for this trajectory are presented.



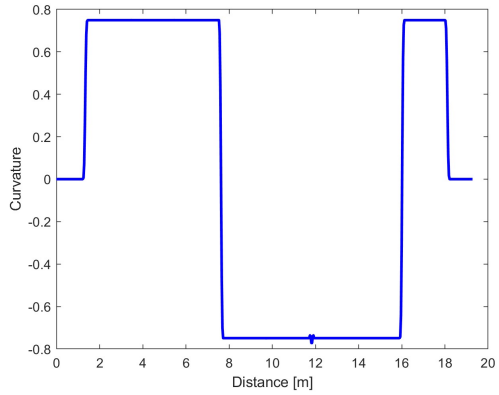
(a) X reference coordinate for Eight-shaped trajectory



(b) Y reference coordinate for Eight-shaped trajectory



(c) Reference heading angle ψ_{Ref} for Eight-shaped trajectory



(d) Reference curvature ρ_{Ref} for Eight-shaped trajectory

Figure 4.12

4.6 Derivation of tracking errors and vehicle travelled distance for path tracking control

The trajectory generation architecture discussed in the previous sections, provides the reference path in terms of travelled distance s_{path}, reference curvature ρ_{ref}, reference heading 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 4.13. If the vehicle was travelling ideally on the reference path, position, curvature and yaw angle of the car would be coincident with the reference ones. However in the experimental setup the vehicle deviates from the reference path and the method to deal with this issue is presented in the following. The idea is to keep trace of this mismatch between reference and actual path thanks to two indicators that are the lateral displacement error e_y and the heading angle error e_ψ. The first one takes into account the distance between the vehicle center of gravity and the reference path line, whereas the second one takes into account the orientation error of the vehicle with respect to the road.

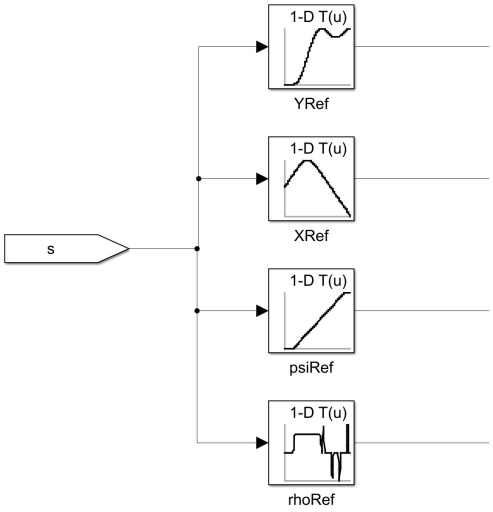


Figure 4.13: Look-Up-Tables used to provide the reference variables to the control architecture, 's' is the vehicle travelled distance.

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

$$d = e_y = (Y - Y_{ref})\cos(\psi_{ref}) - (X - X_{ref})\sin(\psi_{ref}) \tag{4.4}$$

This formula can be used both for cases in which the vehicle has only a lateral mismatch both in situations characterized also by longitudinal mismatch with respect to the desired point on the reference path at a certain travelled distance s as the lateral mismatch is the only thing that matters in lateral control.

For what concerns the heading angle error it can be simply calculated as difference between current heading angle of the vehicle and reference heading angle for the vehicle at the travelled distance s.

$$e_\psi = (\psi - \psi_{ref}) \tag{4.5}$$

These errors described above are implemented in Simulink through a subsystem which output are provided to the control architecture.

Now it is needed to introduce the way the travelled distance is calculated. For each discrete time instant a segment ds_1 is calculated considering the Euclidean distance between two consecutive points $(X, X_{previous}), (Y, Y_{previous})$ as reported in equation 4.6. This segment is the distance travelled by the vehicle on its longitudinal direction and can be related to the space travelled on the reference trajectory ds with this formula:

$$ds_1 = \sqrt{(X - X_{ref})^2 + (Y - Y_{ref})^2} \tag{4.6}$$

$$ds = \frac{ds_1}{1 - \frac{e1}{\rho}} \tag{4.7}$$

All the ds segments are summed in a cumulative way to compute the total distance travelled, namely s up to the end of the trajectory as odometric reference. This variable is then fed to the look-up-tables in order to obtain all the reference variables.

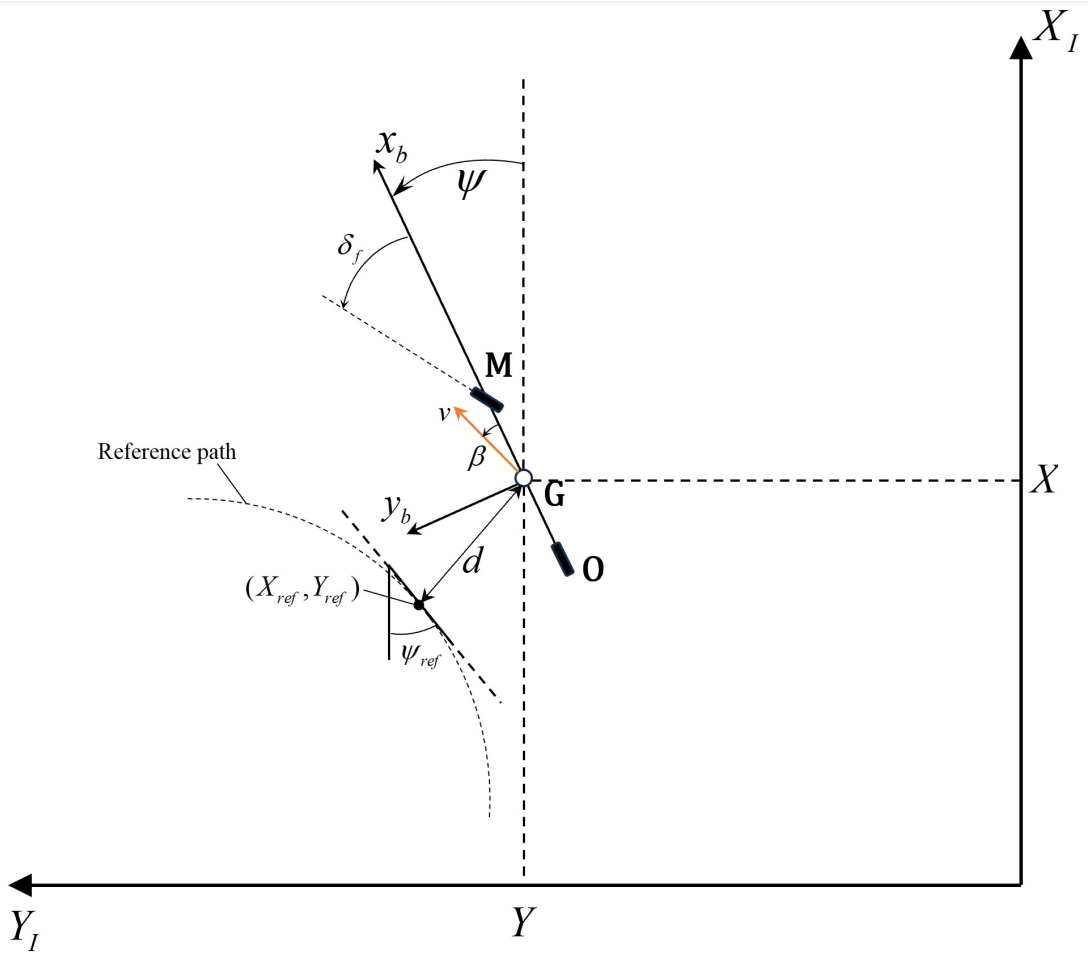


Figure 4.14: Definition of variables used for calculation of distance travelled along the path

4.7 Derivation of dynamic state-space representation

In this chapter the aim is to expose the procedure to derive the state-space representation of the plant in terms of errors with respect to the road, lateral velocity and yaw rate. As vehicle model, the dynamic one is considered, namely the one developed in chapter 3.5. This will be re-expressed in terms of two error variables, lateral error e_y and heading angle error e_ψ that have been already introduced in section 4.6.
Considering a vehicle travelling with constant longitudinal speed V_x being in steady state cornering condition, it is possible to define, as reported in [12]:

$$\begin{cases} \ddot{e}_y = \ddot{y} + v_x(\dot{\psi} - \dot{\psi_{des}}) \\ e_\psi = \psi - \psi_{des} \end{cases} \tag{4.8}$$

From the assumption of constant longitudinal velocity, which lead to a Linear-Time-Invariant model, it is possible to rewrite:

$$\begin{cases} \dot{e}_y = \dot{y} + v_x(\psi - \psi_{des}) \\ e_\psi = \psi - \psi_{des} \end{cases} \tag{4.9}$$

Substituting equations 3.25 and 3.26 into the following two equations (and noting that that the first equation is equivalent to 3.31, considering $v \cdot \dot{\beta} = \ddot{y}$ and small steering angle δ_f) :

$$\begin{cases} m(\ddot{y} + \dot{\psi}v_x) = F_{yf} + F_{yr} \\ I_z\ddot{\psi} = l_f F_{yf} - l_r F_{yr} \end{cases} \tag{4.10}$$

This lead to the following:

$$\begin{cases} m(\ddot{y} + \dot{\psi}) = C_{\alpha f}(\delta - (\frac{\dot{y}+l_f\dot{\psi}}{v_x})) + C_{\alpha r}(\frac{l_r\dot{\psi}-\dot{y}}{v_x}) \\ I_z\ddot{\psi} = l_f C_{\alpha f}(\delta + (\frac{-\dot{y}-l_f\dot{\psi}}{v_x})) - l_r C_{\alpha r}(\frac{l_r\dot{\psi}-\dot{y}}{v_x}) \end{cases}$$
$$\begin{cases} \ddot{y} = -(\frac{C_{\alpha f}+C_{\alpha r}}{mv_x})\dot{y} - (\frac{C_{\alpha f}l_f - C_{\alpha r}l_r}{mv_x} + v_x)\dot{\psi} + \frac{C_{\alpha f}}{m}\delta \\ \ddot{\psi} = (\frac{C_{\alpha r}l_r - C_{\alpha f}l_f}{I_z v_x})\dot{y} - (\frac{C_{\alpha r}l_r^2 + C_{\alpha f}l_f^2}{I_z v_x})\dot{\psi} + \frac{C_{\alpha f}l_f}{I_z}\delta \end{cases} \tag{4.11}$$

Combining all together, considering that $\psi_{des} = v_x \cdot \rho$, the four equations can be written in the following state-space form:

$$\dot{x} = Ax + B_1\delta + B_2\rho \tag{4.12}$$

where, state vector is $x = \begin{bmatrix} \dot{y} & \dot{\psi} & e_y & e_\psi \end{bmatrix}$ and matrices A, B_1, B_2 are defined as follows:

$$A = \begin{bmatrix} -(\frac{C_{\alpha f}+C_{\alpha r}}{mv_x}) & -(\frac{C_{\alpha f}l_f - C_{\alpha r}l_r}{mv_x} + v_x) & 0 & 0 \\ (\frac{C_{\alpha r}l_r - C_{\alpha f}l_f}{I_z v_x}) & -(\frac{C_{\alpha r}l_r^2 + C_{\alpha f}l_f^2}{I_z v_x}) & 0 & 0 \\ 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.13}$$

$$B_1 = \begin{bmatrix} \frac{C_{\alpha f}}{m} \\ \frac{C_{\alpha l_f}}{I_z} \\ 0 \\ 0 \end{bmatrix} \tag{4.14}$$

$$B_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -v_x \end{bmatrix} \tag{4.15}$$

The description presented above represents the model employed to characterize the plant within the simulation environment, where the initial design of path tracking controllers has been performed. Moreover this state-space model is employed in the optimization routine environment that will be described in section 6.1.4.

Chapter 5

Kalman filters

The use of Kalman filters for estimating vehicle's state variables is a common technique especially for autonomous driving, navigation, and vehicle control applications, where this knowledge is essential. This task is complicated by some factors that are related to real world such as the vehicle's dynamics uncertainties, sensors noises, and measurements mistakes. In this thesis work, one of the aims is to provide examples of this technique.
The technique that Kalman filter algorithm uses to provide its estimation is composed by two steps: prediction and update. Based on the prior estimate and on the plant model dynamics, the filter predicts the state variables and an uncertainty covariance matrix (Q), that reflects the prediction's level of confidence, and is linked to the prediction itself.
In the update phase, the filter uses the measurements from the sensors to generate a correction of the previous stage prediction. By means of a weighted average, where the weights are based on the uncertainties of both the predicted state variables and the measurements (through covariance matrix R), the Kalman filter is able to provide a better estimation compared to the one provided by the model only. In this work, filters are employed to correct the vehicle's pose, velocity and accelerations respectively provided by lidar, encoder and IMU. Indeed, despite the localization system producing precise measurement of the vehicle's pose, sometimes happens that the measurements differ considerably from the actual pose producing outliers referred as "spikes" in the following.
In addition to the correction of the measurements, the filter is a way to have vehicle position data (X, Y, ψ, provided by the LiDAR at a frequency of 15 Hz) at sample frequency of 100 Hz, that is the one required by the path tracking control architecture.
In the following an explanation of the algorithm is provided.

5.1 Kalman Filters algorithm

Consider the discrete-time nonlinear system ([13]):

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

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 $P_0 = \mathcal{I}$.

5.2 Kinematic Kalman filter

The first Filter to be developed, makes use of the single-track kinematic model which, as previously explained, is valid in the case of uniformly accelerated motion. Considered this assumption, writing the equation of motion 5.1 and discretizing them, the model's equations are 5.2

$$\begin{cases} x(t) = x_0 + v_{x,0}t + \frac{1}{2}a_x t^2 \\ y(t) = y_0 + v_{y,0}t + \frac{1}{2}a_y t^2 \\ \dot{x}(t) = v_{x,0} + a_x t \\ \dot{y}(t) = v_{y,0} + a_y t \end{cases} \tag{5.1}$$

$$\begin{cases} X(k+1) = X(k) + V_x(k)T_s + a_X(k)\frac{T_s^2}{2} \\ Y(k+1) = Y(k) + V_y(k)T_s + a_Y(k)\frac{T_s^2}{2} \\ V_x(k+1) = V_x(k) + a_X(k)T_s \\ V_y(k+1) = V_y(k) + a_Y(k)T_s \\ \psi(k+1) = \psi(k) + \dot{\psi}(k)T_s \end{cases} \tag{5.2}$$

where T_s is the sampling time.

The longitudinal and lateral accelerations has to be 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, as shown in 5.3:

$$\begin{bmatrix} a_X \\ a_Y \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} a_x \\ a_y \end{bmatrix} \tag{5.3}$$

where $a_x \equiv a_{x,IMU}$, $a_y \equiv a_{y,IMU}$.

The linear time invariant system in **??** can be rewritten in the discrete time state-space form:

$$x(k+1) = Ax(k) + Bu(k) \tag{5.4}$$

where:

$$A = \begin{bmatrix} 1 & 0 & T_s & 0 & 0 \\ 0 & 1 & 0 & T_s & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} \frac{T_s^2}{2} & 0 & 0 \\ 0 & \frac{T_s^2}{2} & 0 \\ T_s & 0 & 0 \\ 0 & T_s & 0 \\ 0 & 0 & T_s \end{bmatrix}$$

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

The measurements are the vehicle pose (X, Y, ψ) provided by the lidar and vehicle longitudinal velocity v_x computed starting from the motor rotational velocity provided by the encoder. Instead, from the IMU, the accelerations, suitably rotated as indicated in 5.3, and the yaw rate are used as model input.

Under the assumption of small velocity and small vehicle side-slip angle, it can be written that:

$$v_x \simeq v = \sqrt{V_x^2 + V_y^2} \tag{5.5}$$

Therefore, the output equations are:

$$\begin{cases} y_1(k) = X(k) \\ y_2(k) = Y(k) \\ y_3(k) = \psi(k) \\ y_4(k) = v_x \simeq v \end{cases}$$

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, from the filtered state variables it is possible to derive the velocity components in the body reference frame and the side-slip angle:

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

The complete filtered state vector $\hat{x}_a = \left[\hat{X}\ \hat{Y}\ \hat{\psi}\ \hat{v}\ \hat{v}_x\ \hat{v}_y\ \hat{\beta}\right]^T$ is obtained.

A schematic overview of the described filter is presented below.

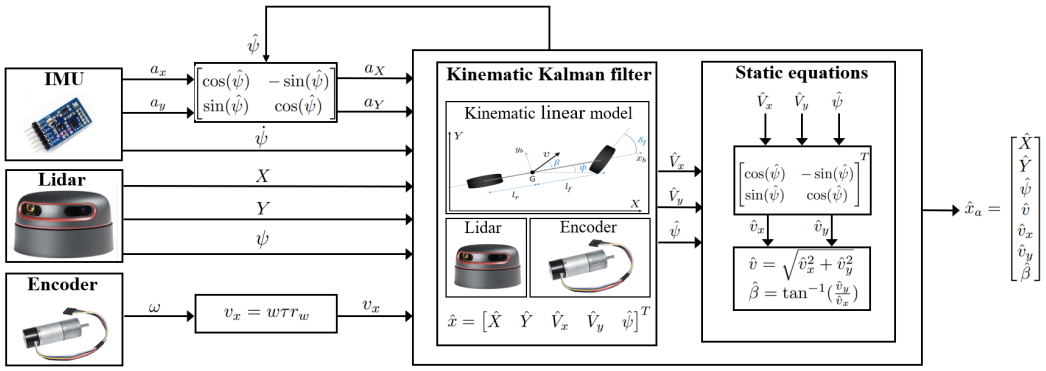


Figure 5.1: Kinematic Kalman filter schematic. Vehicle pose X, Y, vehicle heading angle ψ and vehicle velocity v_x are the measured quantities; the accelerations a_X, a_Y and yaw rate $\dot{\psi}$ provided by IMU are the model inputs.

5.3 Dynamic Extended Kalman Filter

The second filter developed is a non-linear Extended Kalman Filter, based on the single-track dynamic model described in chapter 3.5. This vehicle model is more complex than the one used for the kinematic filter and considers also the tyre-road interaction, providing better results in terms of estimation capabilities. For small tire slip angle and small front steering angle, the following equations describe the model lateral dynamics:

$$\begin{cases} \dot{\beta} = -\dot{\psi} + \frac{C_{\alpha f}}{mv}(\delta - \beta - \frac{l_f \dot{\psi}}{v}) + \frac{C_{\alpha r}}{mv}(-\beta + \frac{l_r \dot{\psi}}{v}) \\ \ddot{\psi} = \frac{l_f C_{\alpha f}}{I_z}(\delta - \beta - \frac{l_f \dot{\psi}}{v}) - \frac{l_r C_{\alpha r}}{I_z}(-\beta + \frac{l_r \dot{\psi}}{v}) \end{cases} \tag{5.6}$$

The time derivatives of the global coordinates i.e. the velocities expressed in the global reference frame can be written as (3.16):

$$\begin{cases} \dot{X} = v\cos(\beta + \psi) \\ \dot{Y} = v\sin(\beta + \psi) \\ \dot{\psi} = r \end{cases} \tag{5.7}$$

The differential equation that describes the electric motor dynamics (3.44) is:

$$\dot{\omega} = P_1 V_a - P_2 \omega - P_3 \tag{5.8}$$

Considering the transmission ratio τ between the motor and the wheel, the wheel radius r_w, neglecting the slip in the longitudinal direction it is possible to write that:

$$v_x = \omega \tau r_w \tag{5.9}$$

Finally:

$$\dot{v}_x = \tau r_w (P_1 V_a - \frac{P_2}{\tau r_w} v_x - P_3) \tag{5.10}$$

Putting together 5.6, 5.7 and 5.10 and discretizing, the complete model's equations are:

$$\begin{cases} X(k+1) = X(k) + (v(k)\cos(\beta(k) + \psi(k)))T_s \\ Y(k+1) = Y(k) + (v(k)\sin(\beta(k) + \psi(k)))T_s \\ \psi(k+1) = \psi(k) + \dot{\psi}(k)T_s \\ v(k+1) = v(k) + (\tau r_w (P_1 V_a(k) - \frac{P_2}{\tau r_w} v(k) - P_3))T_s \\ \beta(k+1) = \beta(k) + (-\dot{\psi}(k) + \frac{C_{\alpha f}}{mv(k)}(\delta(k) - \beta(k) - \frac{l_f \dot{\psi}(k)}{v(k)}) + \frac{C_{\alpha r}}{mv(k)}(-\beta(k) + \frac{l_r \dot{\psi}(k)}{v(k)}))T_s \\ \dot{\psi}(k+1) = \dot{\psi}(k) + (\frac{l_f C_{\alpha f}}{I_z}(\delta(k) - \beta(k) - \frac{l_f \dot{\psi}(k)}{v(k)}) - \frac{l_r C_{\alpha r}}{I_z}(-\beta(k) + \frac{l_r \dot{\psi}(k)}{v(k)}))T_s \end{cases} \tag{5.11}$$

The measurements are the vehicle pose provided by the LiDAR, the longitudinal velocity obtained from the encoder and the yaw rate provided by the IMU. The steering angle and the motor armature voltage are the inputs to the model.

$$\begin{cases} y_1(k) = X(k) \\ y_2(k) = Y(k) \\ y_3(k) = \psi(k) \\ y_4(k) = v_x(k) \simeq v \\ y_5(k) = \dot{\psi}(k) \end{cases} \tag{5.12}$$

The filtered state vector is: $\hat{x} = \left[\hat{X}\ \hat{Y}\ \hat{\psi}\ \hat{v}\ \hat{\beta}\ \hat{\dot{\psi}}\right]^T$

Additionally, considering these relations:

$$\begin{cases} \hat{v}_x = \hat{v}\cos(\hat{\beta}) \\ \hat{v}_y = \hat{v}\sin(\hat{\beta}) \end{cases} \tag{5.13}$$

it is possible to obtain the complete vector of filtered states $\hat{x}_a = \left[\hat{X}\ \hat{Y}\ \hat{\psi}\ \hat{v}\ \hat{v}_x\ \hat{v}_y\ \hat{\beta}\ \hat{\dot{\psi}}\right]^T$

A schematic of this dynamic extended kalman filter is presented in the figure below.

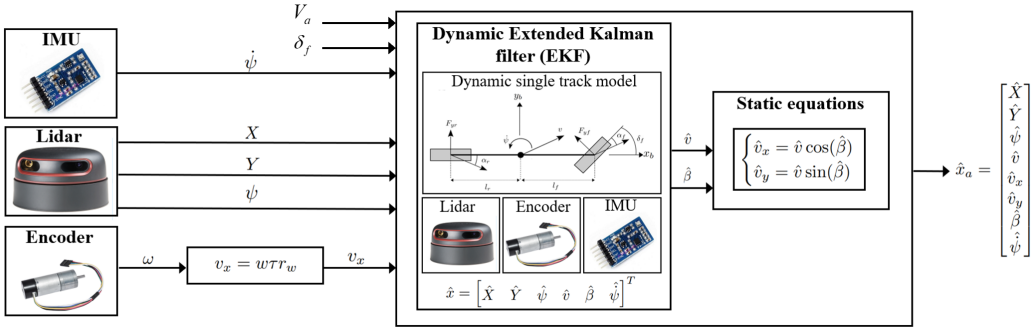


Figure 5.2: Dynamic Extended Kalman filter schematic. Vehicle pose X, Y, heading angle ψ, velocity v_x and yaw rate $\dot{\psi}$ are the measurements, whereas the front steering angle δ_f and the motor armature voltage V_a are the inputs to the model.

5.4 Simulation results

The filters have been designed at first in simulation where the QCar is emulated using the single-track dynamic model. After validation in this environment, the solutions have been implemented in the experimental setup.
In the simulation environment the variables that in the real case were provided by the sensors have been obtained from the vehicle model output, leading to an ideal situation where no model uncertainty is affecting the results and the informations about disturbances are completely available. In this way a comparison between the two ideally tuned filters has been realized.

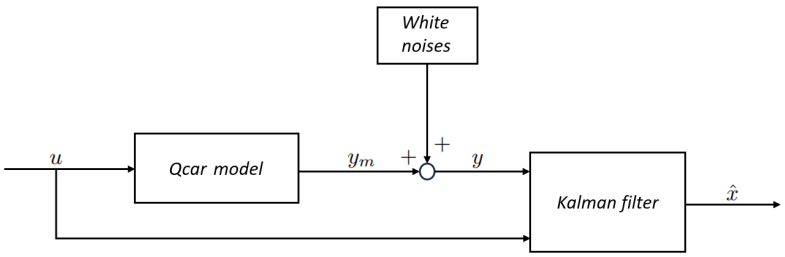


Figure 5.3: Architecture used in simulation. White noise is supposed to affect the measurements whereas ideality is considered for the process.

The output measurements are generated by summing white noises to the model 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 5.1: White noise variances

The filter is tested on a simulated manoeuvre performed in open-loop imposing constant longitudinal velocity and a steering angle with square-wave shape.
The quality of the results is evaluated by means of the root-mean-squared error calculated between the model output states (i.e. the ideal ones) and the filtered states that are provided by the filter.

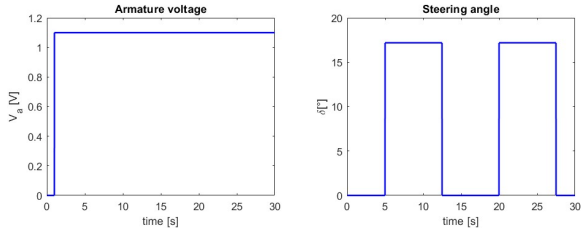


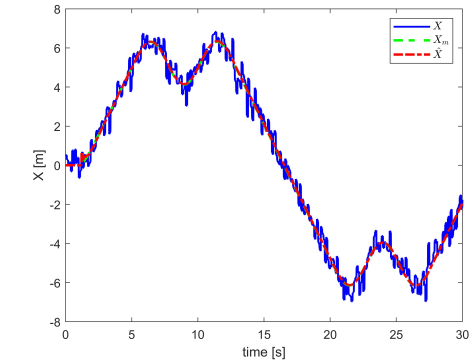
Figure 5.4: Inputs provided to the model for simulation:motor voltage on the left and steering angle command on the right.

5.4.1 Kinematic Kalman Filter

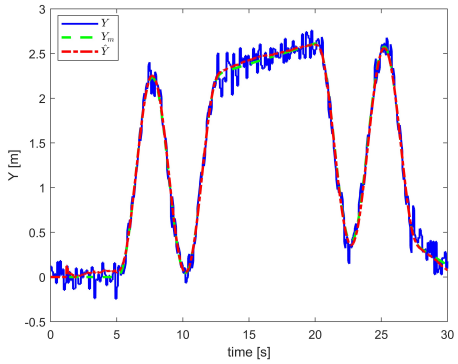
By using the covariance matrices Q and R presented in the following table, the simulation results of the KKF are shown below. In blue are reported the output variables affected by white noise, in green the model variables and in red the estimated ones provided by the filter.

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

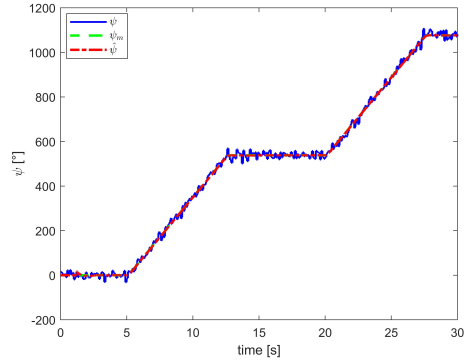
Table 5.2: Covariance matrices KKF



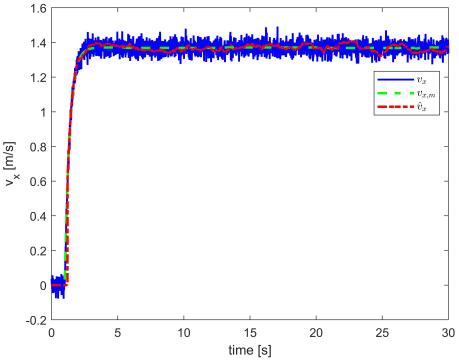
(a) X position



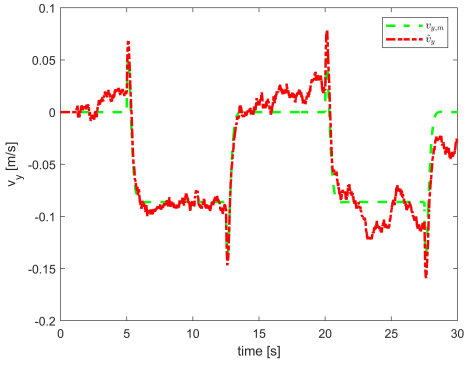
(b) Y position



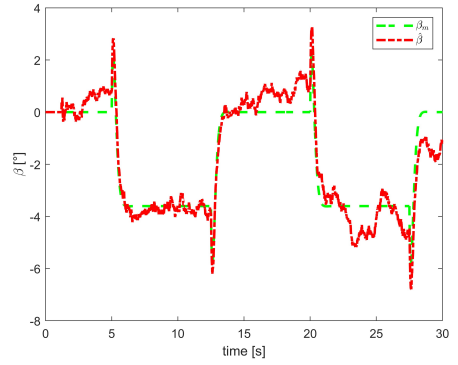
(c) Yaw angle



(d) Longitudinal velocity



(e) Lateral velocity



(f) Vehicle sideslip angle

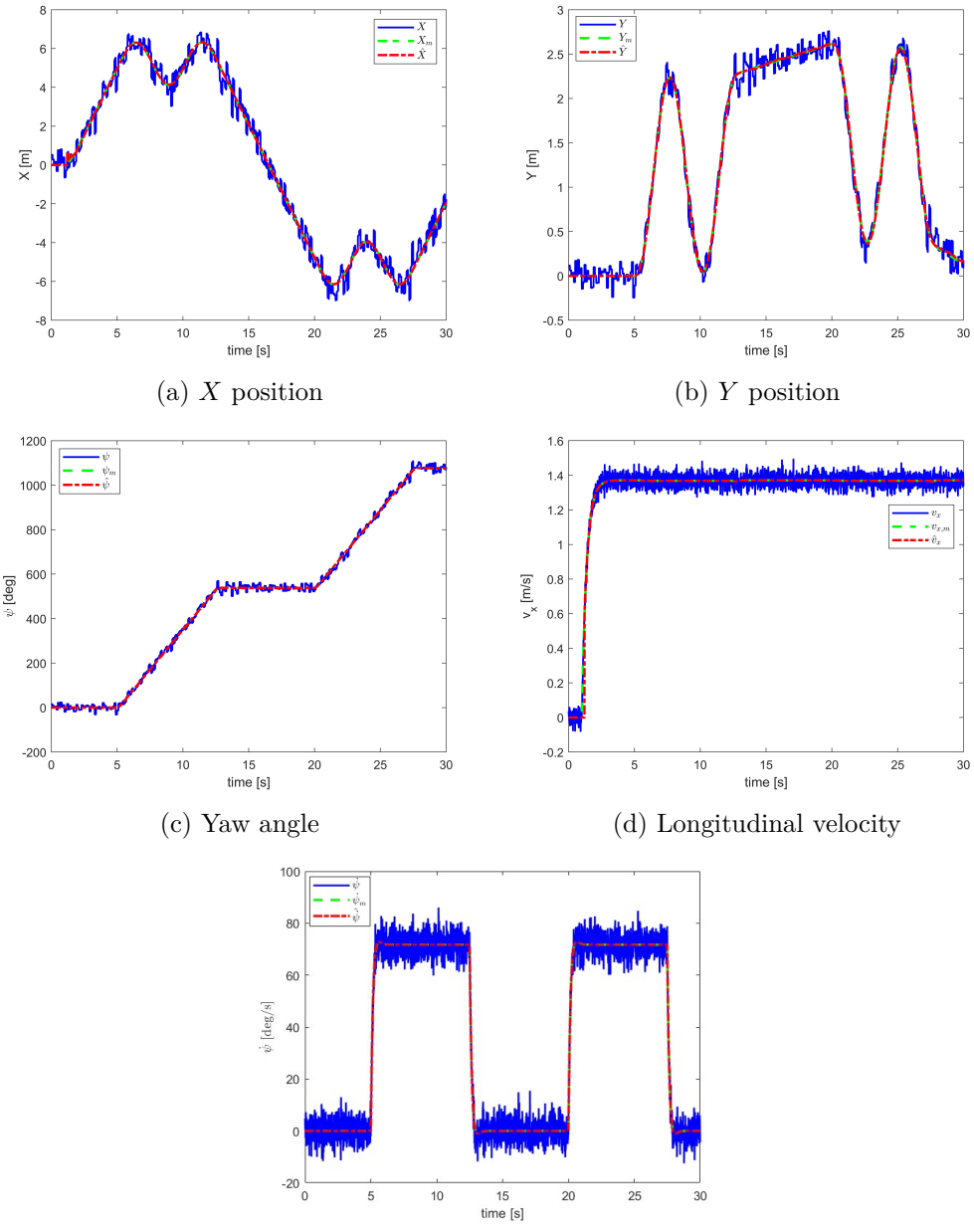
Figure 5.5: Kinematic Kalman filter results in simulation.

5.4.2 Dynamic Extended Kalman Filter

By using the covariance matrices Q and R presented in the following table, the simulation results of the second DEKF are shown below. In blue are reported the output variables affected by white noise, in green the model variables and in red the estimated ones provided by the filter.

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

Table 5.3: Covariance matrices DEKF β



(a) X position

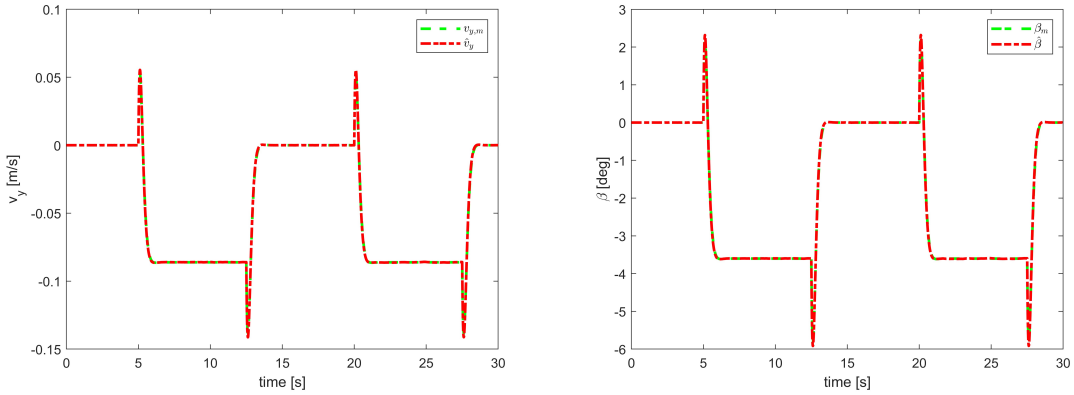
(b) Y position

(c) Yaw angle

(d) Longitudinal velocity

(e) Yaw rate

Figure 5.6: Dynamic Extended Kalman filter results in simulation (1/2).



(a) Lateral velocity

(b) Vehicle sideslip angle

Figure 5.7: Dynamic Extended Kalman filter results in simulation (2/2).

5.4.3 Comparison of simulation results

The Root-Mean-Squared-Error values between the filtered signals and the model state variables in simulation environment of the two filters that have been designed are reported in following table. As expected, the dynamic filter lead to better results with respect to the kinematic one.

	KKF	DEKF β	UoM
RMSE$(\hat{X}, X_m)$	0.0921	0.0503	m
RMSE$(\hat{Y}, Y_m)$	0.1120	0.0417	m
RMSE$(\hat{\psi}, \psi_m)$	0.0455	0.0316	rad
RMSE$(\hat{v}_x, v_{x,m})$	0.0337	0.0301	m/s
RMSE$(\hat{v}_y, v_{y,m})$	0.0199	0.0027	m/s
RMSE$(\hat{v}, v_m)$	0.0336	0.0302	m/s
RMSE$(\hat{\beta}, \beta_m)$	0.0204	0.0135	rad
RMSE$(\hat{\dot{\psi}}, \dot{\psi}_m)$	-	0.0193	rad/s

Table 5.4: Root-Mean-Squared-Error comparison for the Kinematic Kalman Filter and for the Dynamic Extended Kalman Filter.

5.5 Experimental results

As anticipated before, the designed filters are based on LiDAR measurements for the vehicle pose, the motor encoder for the longitudinal speed and the IMU for the accelerations and yaw rate. For the implementation of the filter, the relative Simulink block has been used as it has given the possibility to consider multi-rate system, where sensors provide data at different sample rate. In particular, the state is updated every 10 ms, the LiDAR provides data at a frequency of 15 Hz, encoder and IMU provide data at 10 ms. As already explained in the hardware chapter, the LiDAR registers data and computes the pose of the car basing on a comparison between real-time data (ranges and angles) and stored data obtained during the initialization step in which the sensor scans and creates a map of the surrounding environment with stationary car. Moreover, the point in which the LiDAR performs the initial scan is considered as the origin of the inertial reference frame.
Although the localization system provides quite precise measurements it happens sometime, due to external disturbances, that some of them diverge from the actual values of the vehicle pose. In the figures below it is possible to see an example of these outliers: the data provided by the LiDAR sensor are reported in blue and are compared with the respective reference coordinate which is reported in black.

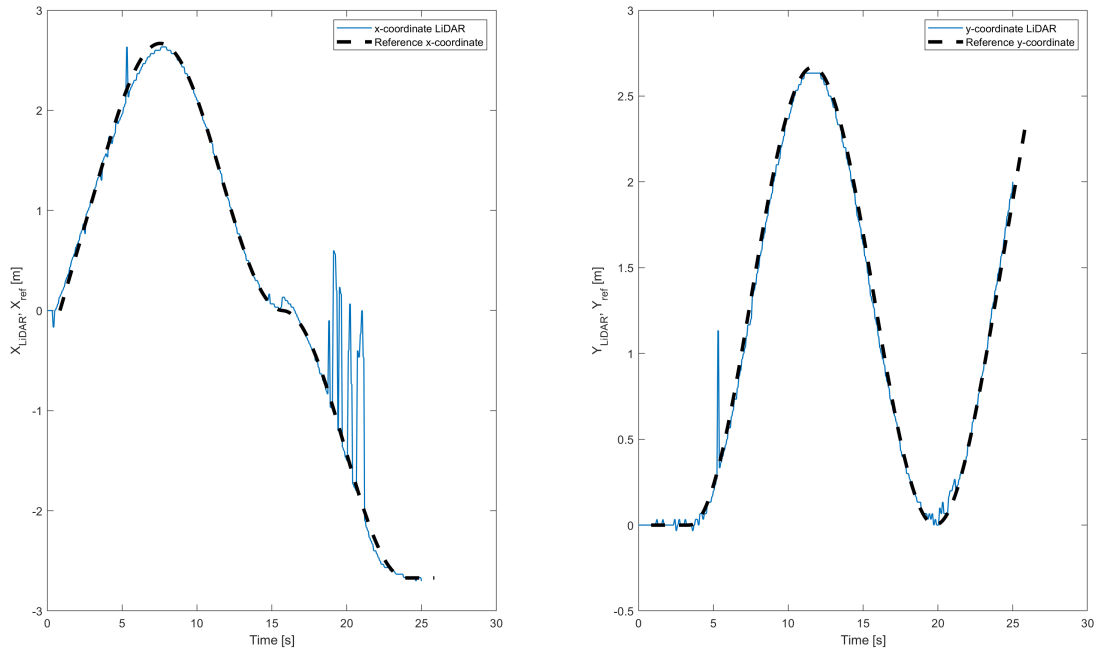


Figure 5.8: Pose data provided by the LiDAR when spikes are produced. In this picture some of them are visible in the red ellypse.

The LiDAR Simulink block designed by the manufacturing company provides a quality index of the measurement accuracy of the sensor named LiDAR score (figure 5.9) that can reach a maximum value of 200. An high value of the LiDAR score means accurate pose measurement provided by the sensor. In order to address the issue of bad measurements, the entries of the pose measurement noise's covariance matrix are time-varying and dependant on the LiDAR score value itself.
To give more importance to the model with respect to the measurements when outliers

are detected, increasing though the values of the LiDAR covariance matrix R, the LiDAR score is mapped into a sigmoid function (represented in figure 5.10) through a coefficient i that takes into account of the inverse proportionality between "bad measures" and LiDAR score and of the maximum value of this latter that is 200. In particular, the function is slightly different for x,y coordinate and for heading angle due to empirical reasons. All these considereations are summarized in equations 5.14.

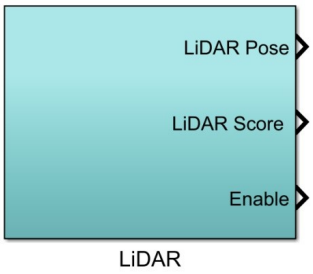


Figure 5.9: Simulink sub-system block for LiDAR localization.

$$\begin{cases} i = \frac{200}{|LiDarScore|} \\ R_{xx,0} = R_{yy,0} = 80 \cdot \tanh(\frac{i}{0.6} - 3) + 81 \\ R_{\psi\psi,0} = 9 \cdot \tanh(\frac{i}{0.7} - 3) + 9.3 \end{cases} \tag{5.14}$$

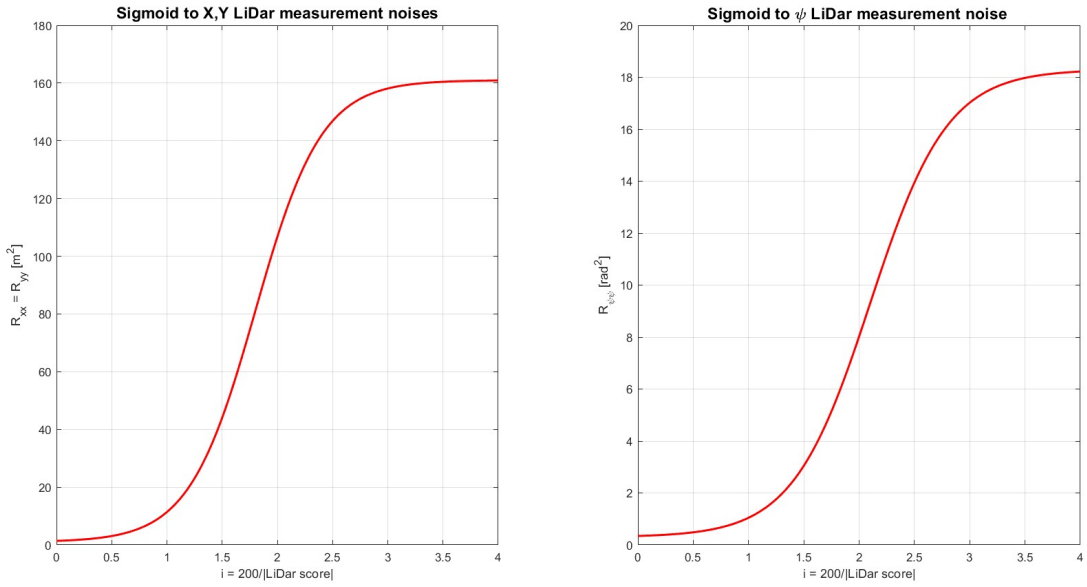


Figure 5.10: Sigmoid functions for LiDar measurement noises

With these approach, 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 prediction. However, in presence of spikes, the LiDar score does not penalize enough the wrong measurement and this leads to poor performances due to one or more values of $R_{xx}, R_{yy}, R_{\psi\psi}$ still too low.
In order to improve filters performance by further penalizing the spikes recognition, the respective entry of the LiDar measurement noise's covariance matrix, defined in **??**, is multiplied by a factor $\alpha_X, \alpha_Y, \alpha_\psi > 1$ when a spike (i.e. a very "bad" measurement) is

detected respectively in the X, Y and ψ measurement. The conditions that have to be satisfied to identify a spike are the following:

$$\begin{cases} |X(k) - X(k-1)| > 0.1 \rightarrow SPIKE_X \rightarrow R_{xx} = \alpha_X R_{xx,0} \\ or \\ |Y(k) - Y(k-1)| > 0.1 \rightarrow SPIKE_Y \rightarrow R_{yy} = \alpha_Y R_{yy,0} \\ or \\ |\psi(k) - \psi(k-1)| > 0.05 \rightarrow SPIKE_\psi \rightarrow R_{\psi\psi} = \alpha_\psi R_{\psi\psi,0} \end{cases} \tag{5.15}$$

The values of the multiplicative factors are presented in table 5.5.

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

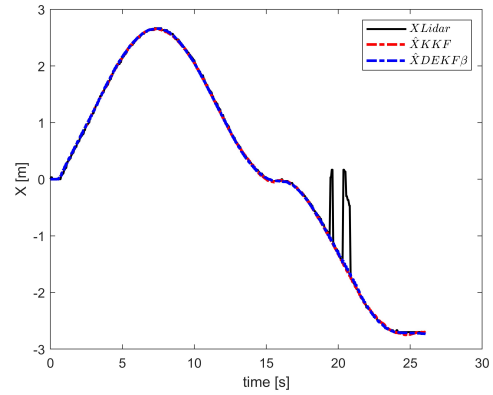
Table 5.5: Multiplicative factors that are used to further increase the LiDAR covariance matrix entries when very bad measurements (i.e. "spikes") are detected.

The values of covariances matrix entries for motor encoder R_{vx} and IMU $R_{\dot{\psi}}$ sensors, as well as the ones used for process disturbances Q are shown in table 5.6. The specific entries within the Q matrix are the result of a careful tuning performed during experimental tests to achieve the best performances possible for the individual filter.

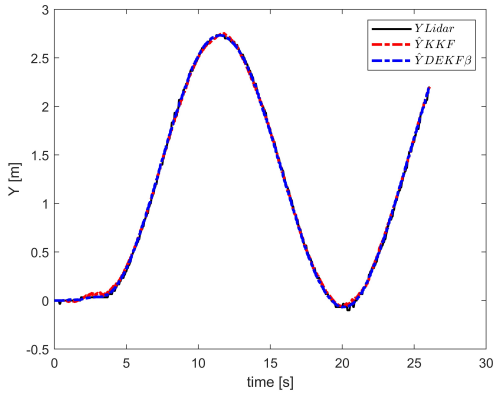
	KKF	DEKF
R_{v_x}	0.025	0.025
$R_{\dot{\psi}}$	-	0.04
Q	diag(1,0.6,100,60,80)	diag(0.05,0.05,0.0628,0.01,0.0001,0.1257)

Table 5.6: Covariance matrix entries for disturbances affecting encoder and IMU measurements as well as for the one affecting the process.

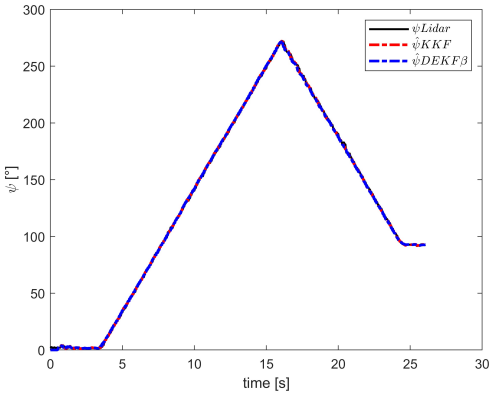
The experimental test whose results are shown below is performed in closed-loop imposing a S-trajectory tracking. With the numerical values presented above for matrices R and Q, the following experimental results are obtained. In the following section, the results of both Kinematic and Dynamic filters are presented with emphasis on the comparison of the two results rather than examining the results of each individual filter.
As it can be seen in Figure 5.11a some spikes are present (black line i.e. the data from LiDAR) and the performances of both filters are excellent in terms of outlier management and rejection.
For what concerns the vehicle side slip angle β, it can be seen in figure 5.12b that the estimation provided by the Dynamic EKF is better that the one provided by the Kinematic KF, since this latter one presents an excessive floating shape. The same can be said for the lateral velocity v_y. To conclude,



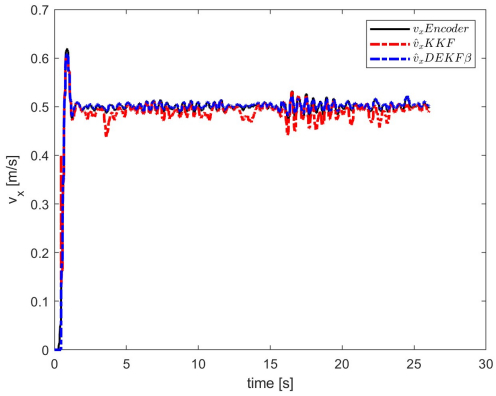
(a) X position



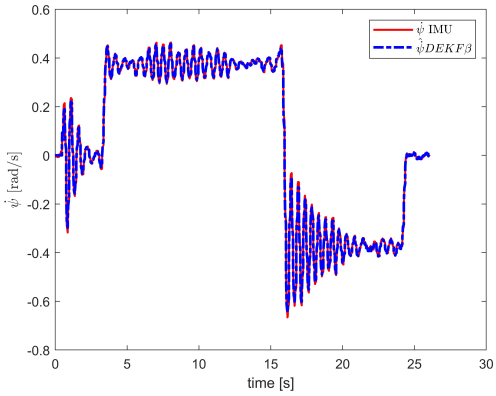
(b) Y position



(c) Yaw angle

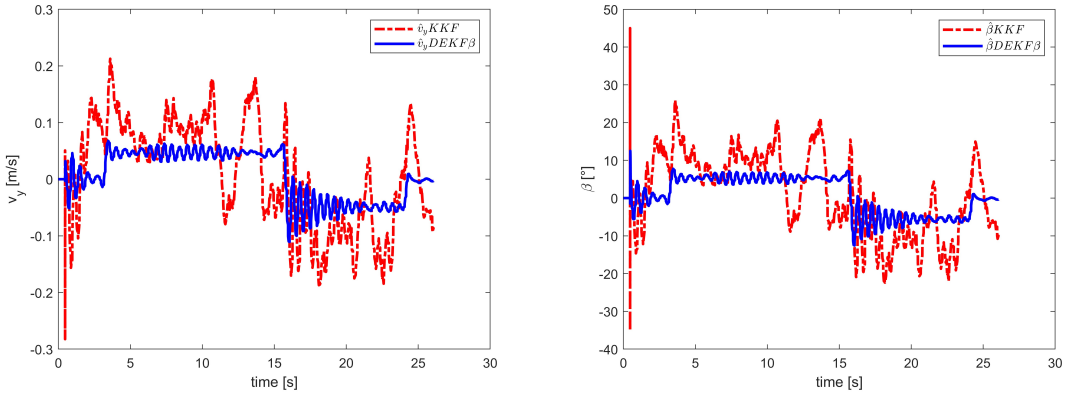


(d) Longitudinal velocity



(e) Yaw rate

Figure 5.11: Kinematic and Dynamic Kalman Filter experimental result comparison (1/2).



(a) Lateral velocity

(b) Vehicle sideslip angle

Figure 5.12: Kinematic and Dynamic Kalman Filter experimental result comparison (2/2).

To conclude, in light of these results, as predicted in simulation and verified experimentally, the dynamic filter performs better than the kinematic one, so the former will be used for the applications developed in the next chapters.

Chapter 6

Control architectures for Path-Tracking application

In this chapter the focus is about the design of the control algorithm for path tracking control, then applied to the scaled vehicle. At first a Proportional-Integral controller is designed, followed by the development of an LQR controller. After the design and integration with the vehicle plant done in simulation, experimental validation has been performed through tests in real scenario environment.

6.1 Proportional-Integral controller: introduction and architecture

The first controller developed for the path tracking application is a simple PI controller. A dynamic single-track model is used for the design process in simulation where the tuning was carried out using the $fmincon$ optimization routine. However this tuning approach proved to be inadequate when transitioning to the experimental setup. In the experimental phase, the controller gains were adjusted through a trial and error procedure, starting from the ones provided by the optimization process.

PI controller is a common control algorithm ([14]) widely used also in industries, due to its capabilities to give resilient performances across diverse operational conditions being at the same time simple from the architectural point of view. The control action is realized through the sum of the proportional and integral contribution which both multiply the error calculated as the difference between reference and feedback from the plant, called error term. The general control architecture can be appreciated in figure 8.18.

In our study case, the process variable $u(t)$ i.e. the variable of the plant to be controlled, is the steering angle $\delta(t)$. The third state of the plant, i.e. the lateral error e_y is considered for the feedback. To implement the control strategy, the tracking error $e(t)$ fed to the PI scheme is calculated as the difference between the current lateral error value and the set point $r(t)$ that in this case is zero. The reference road curvature $\rho(t)$ is considered as disturbance in the closed loop scheme.

The proportional component depends only on the error term and its correspondent gain

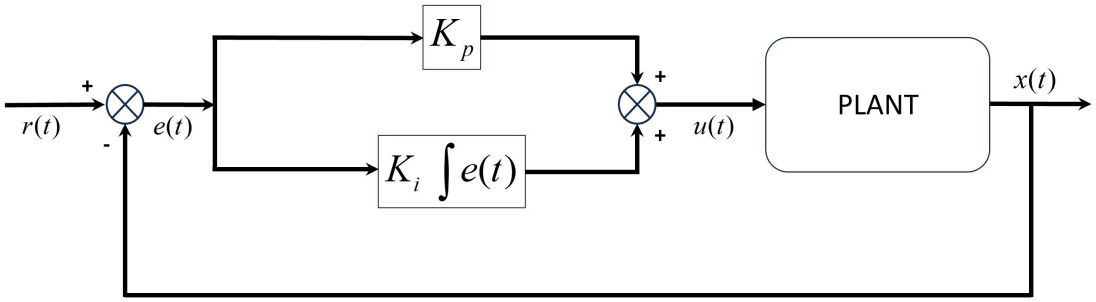


Figure 6.1: Generic closed-loop Proportional-Integral control scheme

establishes a linear relationship between the output response and the error signal. Typically, enhancing the proportional gain results in a swifter response from the control system. However an excessive high proportional gain can trigger oscillations in the process variable causing the system to become unstable oscillating out of control.
The integral component essentially takes into account the previous values of the tracking errors by summing them over time. The contribution is zero only if the error is zero and even a small tracking error will cause the integral term to increase slowly. The aim of this term is to drive the steady state tracking error to zero, which is the final difference between the process variable and the set point.

The main problem associated with the integral part is the windup phenomena: when a control system operates across diverse operational conditions, there's a possibility that the actuator could reach its limits. If this happens, the feedback loop is broken and the integral part will continue to integrate the error, leading to contribution that could potentially tend to infinity. Saturation is a nonlinearity that can be defined as follows:

$$\text{sat}(u) = \begin{cases} \text{u}_{\text{min}} & \text{if } u < \text{u}_{\text{min}} \\ u & \text{if } \text{u}_{\text{min}} \leq \text{u} \leq \text{u}_{\text{max}} \\ \text{u}_{\text{max}} & \text{if } u > \text{u}_{\text{max}} \end{cases} \tag{6.1}$$

The consequences of windup are related to performance deterioration consisting on large overshoots in the plant output and a long time is required to go steady-state. To address this issue it is possible to use an anti-windup technique in the integral part of the controller.

6.1.1 Back-calculation method

With this technique [15], when the output of the controller reaches saturation, a recalculation of the integral component is executed; this recalculated value is adjusted to align with the saturation limit. The adjustment is done considering another feedback branch which consists on the difference between the controller output and the actuator output. This signal is then multiplied with the back-calculation gain K_b and combined with the product between the tracking error and the integral gain K_I. This correction term is zero on the normal operation when the actuator does not saturate.
The K_b constant governs the speed at which the integrator resets. A general rule of thumb suggests suggests to choose a value of this coefficient equal to the integral one and then

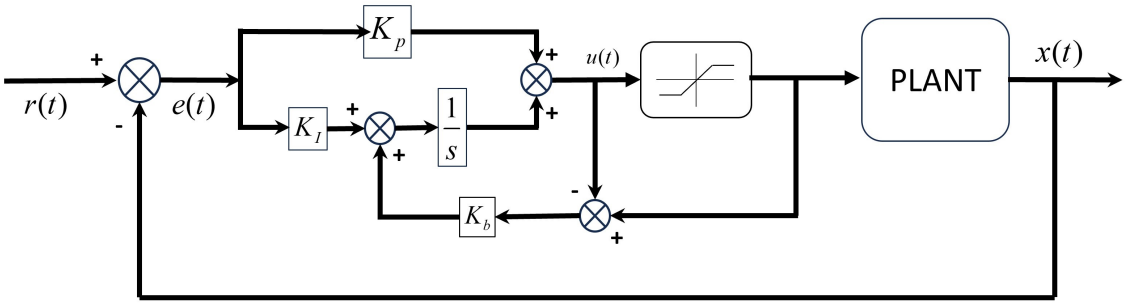


Figure 6.2: Closed-loop Proportional-Integral control scheme with back-calculation anti-windup technique.

eventually tune the former if better performances are needed.
With increasing values of K_b the saturation time is reduced, making the system slower in its response. Smaller values of K_b will provide a faster system response with the problem of increasing the over-shoot.
The architecture can be appreciated in figure 6.2.

6.1.2 Architecture for QCar control

In our study case, the process variable $u(t)$ i.e. the variable of the plant to be controlled, is the steering angle $\delta(t)$. The third state of the plant, i.e. the lateral error e_y is considered for the feedback. To implement the control strategy, the tracking error $e(t)$ fed to the PI scheme is calculated as the difference between the current lateral error value and the set point $r(t)$ that in this case is zero. The reference road curvature $\rho(t)$ is considered as disturbance in the closed loop scheme.
The command that is fed to the plant is composed by a feedback and a feedworward contributions. This architecture is similar to the one developed in [16]. The main advantage is that, introducing the feedforward contribution, the reliance on feedback term provided by controller is reduced allowing for smaller controller gains. Variations of this combinational architecture are widely used in this area, see also [12] and [17]. The proposed scheme can be appreciated in figure 6.3.

6.1.3 Controller tuning in simulation: Ziegler-Nichols tuning method

The tuning of the controlled has been performed at first in simulation, using $fmincon$ optimization algorithm. The guessing starting coefficients have been calculated using the empirical Ziegler-Nichols PI tuning method.

As reported in [18], this tuning technique has to be performed in closed-loop. Giving the system a step reference and observing its response starting with only the proportional gain K_P. Then increase this gain until the system response results in oscillations with constant amplitude. At this point register the value of proportional gain and the period of oscillation T_o. Having this two parameters it is possible to plug these values in the equations below to find the numerical value of the coefficients that according to this method give the best response.

The values that have been found with this tuning method are:

- $K_P = 25$;

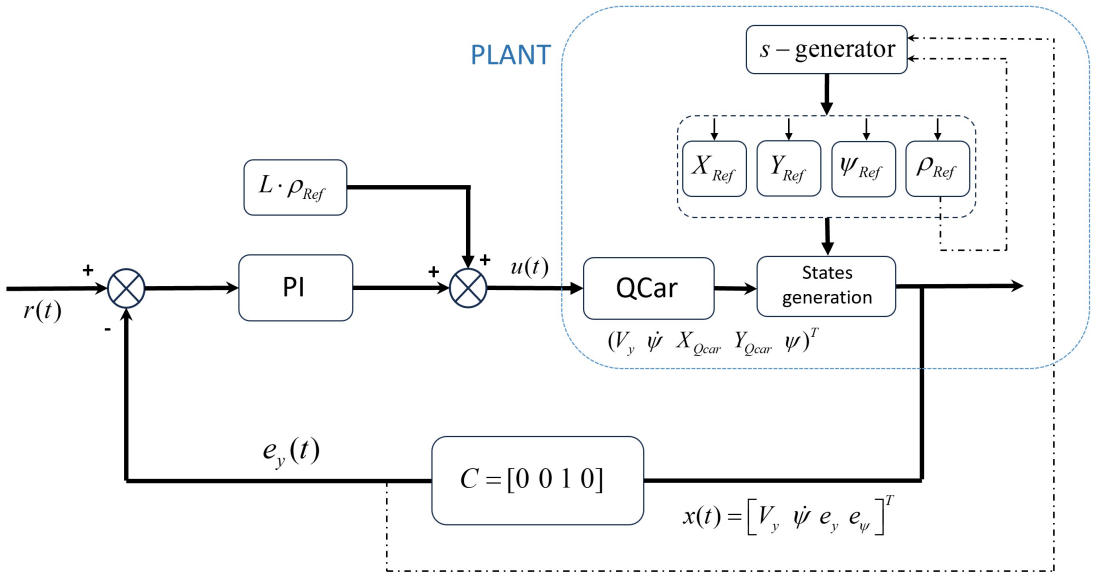


Figure 6.3: Complete control architecture: PI and kinematic contribution.

	Proportional gain K_P	**Integral gain** K_I
P	$K_P/2$	-
PI	$K_P/2.2$	$T_o/1.2$

Table 6.1: Closed-loop calculation of controller gains with empirical Ziegler-Nichols method.

- $T_o = 0.8324$;

Plugging these parameters into the equation of table 6.1 it is possible to obtain the numerical values of the coefficients that are then used as guessing poles in the optimization routine.

The values that have been found with this tuning method are:

- $K_P = 11{,}36$;
- $K_I = 0.694$;

The test has been performed with a step curvature input that is the set point for the control scheme. The speed considered for the tests is 1.0 m/s that is in the middle of the range at which the car is used in the simulation and experimental tests: 0.5:1.5 m/s.

6.1.4 *fmincon* optimization routine

The state equation of the plant is the following (derived in chapter 4.7):

$$\dot{x} = Ax + B_1\delta_f + B_2\rho \tag{6.2}$$

Where the input is the front wheel steering angle δ_f and the curvature ρ is the unactuated input to the system.

The vehicle dynamics is defined by the matrices reported in chapter 4.7 where

$$x = \begin{bmatrix} v_y \\ \dot{\psi} \\ e_y \\ e_\psi \end{bmatrix} \tag{6.3}$$

The design of the path tracking PI controller starts by defining the closed-loop dynamics. As already shown in figure 6.3 the control action can be expressed by:

$$\delta_f = K_P(v_x)e(t) + K_I(v_x)\int e(t) + L \cdot \rho \tag{6.4}$$

Where the tracking error $e(t)$:

$$e(t) = r(t) - y(t) = 0 - e_y(t) = -e_y(t) \tag{6.5}$$

Applying the abovementioned control input, and considering the augmented state:

$$\begin{aligned} x_I &= \int x_3 = \int e_y \\ \dot{x}_I &= x_3 = e_y \end{aligned} \tag{6.6}$$

the closed-loop dynamics results to be:

$$\dot{x} = Ax - B_1K_Px_3 - B_1K_Ix_I + B_1L\rho + B_2\rho \tag{6.7}$$

The system can be rewritten:

$$\tilde{A} = \begin{bmatrix} -\frac{C_{\alpha f}+C_{\alpha r}}{mv_x} & -\frac{l_fC_{\alpha f}-l_rC_{\alpha r}}{mv_x} - v_x & -\frac{C_{\alpha f}}{m}K_P & 0 & -\frac{C_{\alpha f}}{m}K_I \\ -\frac{l_fC_{\alpha f}-l_rC_{\alpha r}}{v_xI_z} & -\frac{l_f^2C_{\alpha f}-l_r^2C_{\alpha r}}{v_xI_z} & -\frac{l_fC_{\alpha f}}{I_z}K_P & 0 & -\frac{l_fC_{\alpha f}}{I_z}K_I \\ 1 & 0 & 0 & v_x & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{6.8}$$

$$\tilde{B}_1 = \begin{bmatrix} B_1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{C_{\alpha f}}{m} & \frac{l_fC_{\alpha f}}{I_z} & 0 & 0 & 0 \end{bmatrix}^T$$

$$\tilde{B}_2 = \begin{bmatrix} B_2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & v_x & 0 \end{bmatrix}^T$$

The closed loop dynamics can be expressed by:

$$\dot{\tilde{x}} = \tilde{A}\tilde{x} + (\tilde{B}_1L + \tilde{B}_2)\rho$$

$$\tilde{x} = \begin{bmatrix} v_y \\ \psi \\ e_y \\ e_\psi \\ \int e_y \end{bmatrix} \tag{6.9}$$

or equivalently:

$$\dot{\tilde{x}}_{cl} = \tilde{A}\tilde{x} + \tilde{B}_{cl}\rho \tag{6.10}$$

The gains of the controller are derived considering to minimize a positive cost function [19], [20] $J(K_P, K_I)$ such that:

- $J(K_P, K_I)$ has finite value for stable solutions of the system 6.7 to which is applied the command 6.4;
- Smaller values of $J(K_P, K_I)$ correspond to better tracking of the reference trajectory.

The final gains are the ones that satisfy the following equation:

$$J\left(K_P^{fin}, K_I^{fin}\right) = \min_{K_P, K_I} \{J(K_P, K_I)\}. \tag{6.11}$$

Where the function to be minimized is made by the weighted sum of the quadratic norm of three terms and is reported in equation 6.12:

- Settling-time error: $e_{t_s} = \left|t_{s,95\%} - t_{s,95\%,des}\right|$;
- Overshoot error: $e_{\hat{s}} = \hat{s} - \hat{s}_{des}$, where $\hat{s} = \frac{e_{y,\max}(i) - e_{y,ss}}{e_{y,ss}}$;
- Root mean squared error: $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(e_{y(i)} - e_{y(ss)}\right)^2}$;

$$J\left(K_P, K_I\right) = \alpha \cdot (e_{t_s})^2 + \beta \cdot (e_{\hat{s}})^2 + \gamma \cdot (RMSE)^2 \tag{6.12}$$

The minimization problem can be written so as:

$$\begin{aligned} & J\left(K_P^{fin}, K_I^{fin}\right) = \min_{K_P, K_I} \{J(K_P, K_I)\} \\ & J\left(K_P, K_I\right) = \alpha \cdot (e_{t_s})^2 + \beta \cdot (e_{\hat{s}})^2 + \gamma \cdot (RMSE)^2 \\ & \text{subject to:} \\ & K_P,\ K_I > 0 \end{aligned} \tag{6.13}$$

So the procedure will iterate starting from the guessing poles that have to be selected in a way such that to guarantee the stability of the controlled system (in our case this condition is satisfied as we used the Ziegler and Nichols method). Then stability is guaranteed at each iteration of the optimization process, since unstable solution will lead to higher values of the cost function. The optimization routine (figure 6.4) is numerically solved through $fmincon$ function in MatLab.

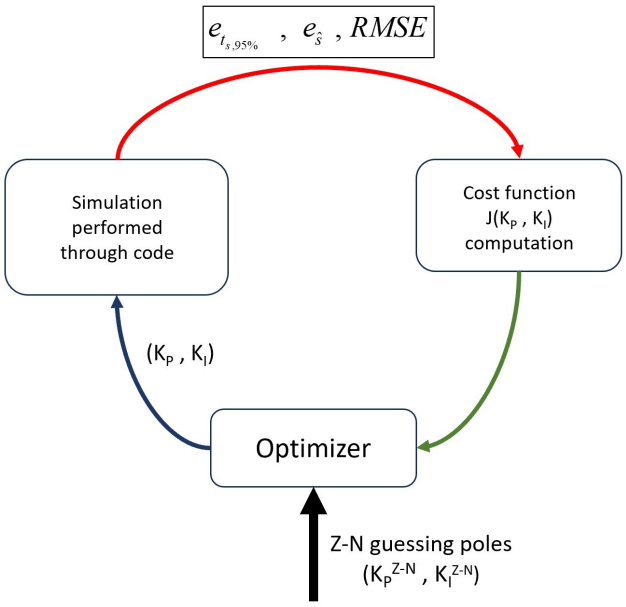


Figure 6.4: Procedure to numerically solve the optimization problem

6.2 Linear-Quadratic-Regulator controller

The second controller that has been designed is a Linear Quadratic Regulator (LQR). In this section the aim is to describe the design process ([21]).
For the dynamical system described by the dynamical equation obtained in 4.7

$$\dot{x} = Ax + B_1\delta_f + B_2\rho \tag{6.14}$$

the infinite horizon LQ problem is considered so that to stabilize the system with a time invariant controller. The infinite horizon LQR problem is formulated as follows:

$$\begin{aligned} &\min_{u(t)} J(u) = \min_{u(t)} \int_{t=0}^{t=\infty} [x^T(t)Qx(t) + u^T(t)Ru(t)]dt \\ &\text{subject to:} \\ &\dot{x(t)} = Ax + B_1\delta_f + B_2\rho \end{aligned} \tag{6.15}$$

The matrices Q and R are design parameters and will be chosen accordingly to the desired performances.
The optimal cost function $J(u^*)$ and the optimal input are:

$$\begin{aligned} u^*(t) &= \arg\min_{u(t)} J(u) \\ J^* &= J(u^*(t)) \end{aligned} \tag{6.16}$$

The system needs to be reachable to be stabilized by such an algorithm. Reachability (or controllability) referes to the property of the dynamic of a system to be modified from an initial state to a desired final state in a finite interval of time by applying a suitable control input.
Reachability is strictly related to the reachability matrix which is contructed from the state space matrices. If this matrix is full rank the system is reachable.

$$\mathcal{M}_{\mathcal{R}} = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix} \tag{6.17}$$

If:

$$\rho(\mathcal{M}_{\mathcal{R}}) = \rho(\begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix}) = n \tag{6.18}$$

where n is the system dimension, then the optimal solution to the LQR problem for the control input is:

$$u^*(t) = -R^{-1}B^TPx(t) = -K_xx(t). \tag{6.19}$$

P ($P^T > 0$) is the solution to the Algebraic Riccati Equation (ARE) that is reported here:

$$Q - PBR^-1B^TP^T + PA + A^TP = 0 \tag{6.20}$$

The control law for the static state feedback control architecture becomes:

$$u(t) = \delta_f(t) = -K_xx(t) \tag{6.21}$$

The test for reachability has been performed for each case. Fore the sake of completeness here it is reported the test for reachability at a speed of 0.5 m/s that is a value in the middle of the range used in the tests. Reachability matrix at the selected speed has the following numerical values:

$$\mathcal{M}_{\mathcal{R}} = 1 \cdot 10^5 * \begin{bmatrix} 0 & 0 & -0.0006 & 0.0119 \\ 0 & -0.0006 & 0.0119 & -0.2139 \\ 0 & 0.0003 & -0.0047 & 0.0727 \\ 0.0003 & -0.0047 & 0.0727 & -1.1236 \end{bmatrix}, \tag{6.22}$$

whose rank is 4.

Moreover the plant matrix A, for the same speed of 0.5 m/s (eq 6.23), has two eigenvalues at the origin with minimal polynomial multiplicity equal to 2 so the system is unstable.

$$A = \begin{bmatrix} 0 & 1.0000 & 0 & 0 \\ 0 & -15.0574 & 7.5287 & 0.0182 \\ 0 & 0 & 0 & 1.0000 \\ 0 & 1.1862 & -0.5931 & -15.6825 \end{bmatrix} \tag{6.23}$$

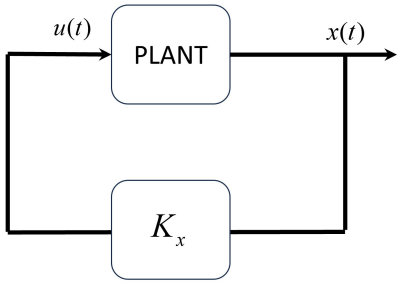


Figure 6.5: LQR control architecture with the feedback only

6.2.1 Feed-Forward contribution

Even if the the system is stabilized through the feedback (figure 6.6), because of the presence of $B_2\rho$ term, when the desired curvature is not zero, i.e. when the vehicle is moving on a turn, the tracking error will not converge to zero. To force this to happen a feedforward contribution is added. Zero lateral displacement error at steady state is imposed and through MATLAB symbolic toolbox the feedforward term is obtained. This term is dependant from the system matrices (A, B_1, B_2), from the feedback gain and from the curvature.
The procedure is detailed in the following lines. Considering the system 6.14, its transfer function can be computed using the Laplace transform so that:

$$G(s) = C(s\mathcal{I}_4 - A_{path})^{-1} B_{cl} \tag{6.24}$$

The third component of $G(s)$ is the tranfer function between input and lateral displacement error $G_3(s) = \frac{e_y(s)}{\rho(s)}$. The steady state value can be computed by means of the final value theorem, giving that:

$$|e_y(t)|_\infty = |\lim_{t\to\infty} e_y(t)| = |\lim_{s\to 0} s \cdot e_y(s)| = |\lim_{s\to 0} s \cdot G_3(s) \cdot \rho(s)| \tag{6.25}$$

Considering costant reference curvature, its laplace transform is: $\mathcal{L}(\rho) = \frac{\bar{\rho}}{s}$

$$|\lim_{s\to 0} s \cdot G_3(s) \cdot \rho(s)| = |\lim_{s\to 0} G_3(s) \cdot \bar{\rho}| = G_3(0) \cdot \bar{\rho} \tag{6.26}$$

To have:

$$|e_y(t)|_\infty = 0 \Rightarrow G_3(0) \cdot \bar{\rho} = 0 \tag{6.27}$$

The expression of feed-forward term is derived as function of δ_{ff} with the command $solve(G_1(0), \delta_{ff})$ and the result is quite long and complex so it can be resumed as:

$$\delta_{ff} = K_R \cdot \rho \tag{6.28}$$

The complete control law becomes:

$$u(t) = \delta_f(t) = -K_x x(t) + K_R \rho(t) \tag{6.29}$$

The closed loop dynamics becomes:

$$\dot{x} = (A - B_1 K_x)x + (B_1 K_R + B_2)\rho \tag{6.30}$$

The design has been performed at first in simulation then, once sure about the proper working of the control architecture the algorithm has been deployed on the QCar. The tuning of the controller has been performed through trial and error procedure. Due to the discrepancy between expected results from simulation, and obtained ones in experimental tests, the experimental tuning has been readjusted in order to achieve the best performances.

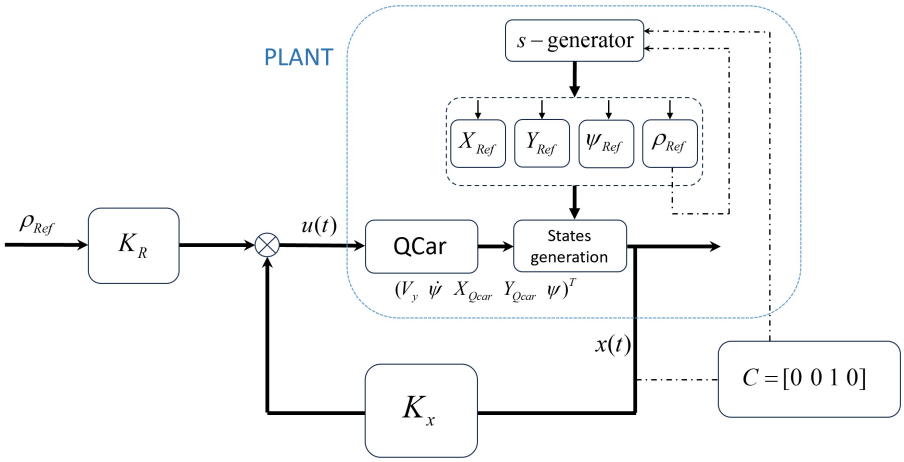


Figure 6.6: LQR control architecture with the feedback and feedforward contributions.

Chapter 7

Simulation Results

This chapter to report the results of developed and simulated path tracking control architectures. The simulation primarily focuses on low-speed scenarios. For the most challenging trajectories that are the one for obstacle avoidance maneuvre and the eight-shaped trajectory, are also provided figures for simulations conducted at higher speeds to illustrate the expected performance trends. Despite simulations have been performed on all trajectories for several velocities, not all are reported here for the sake of brevity.
In particular, in the proposed plots are reported:

- Comparison between reference and actual trajectories;
- Lateral displacement error e_y;
- Heading angle error e_ψ;
- Control action: steering angle δ_f.

It is worth to say that the trajectory for obstacle avoidance manoeuvre, that is dynamically more demanding for the vehicle, has been designed to be executed on the laboratory track, starting from the middle point of the lower straightaway segment. To ensure an equal evaluation of the maneuver performance and to avoid any error carried over from the previous U-shaped segment, the vehicle speed has been kept at a low value (0.5 m/s) and only just after the vehicle finishes the semicircumference (after the vehicle travelled a distance of 5.694 m), the speed is increased up to the target one. The implementation of this concept is illustrated in the Simulink block shown in figure 7.1.

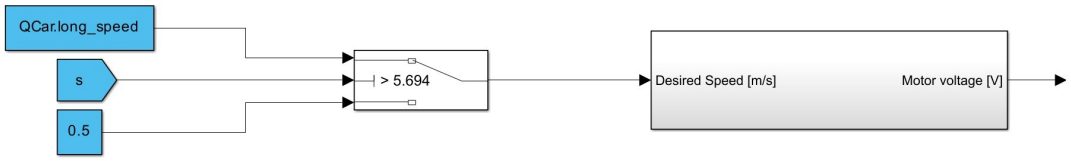


Figure 7.1: Block for speed variation used for the obstacle-avoidance trajectory.

7.1 Proportional-Integral controller

This section to present the results obtained with Proportional-Integral control architecture. As it can be seen in figures below, at low speed results are good enough both in terms of lateral error, that is within the millimeter range, both in the control action signal. Moreover, it can be appreciated how the controller manages to bring the lateral displacement error to zero after a transient that can be observed each time there is a change in the reference curvature.

7.1.1 Circular-shaped trajectory

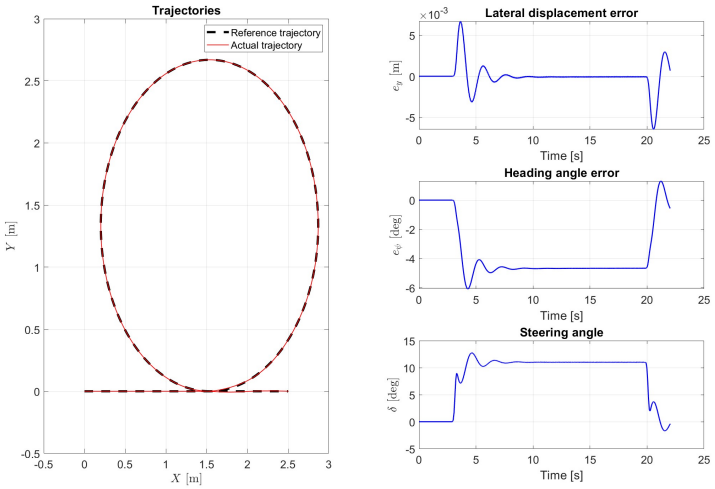


Figure 7.2: Simulation results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

7.1.2 U-shaped trajectory

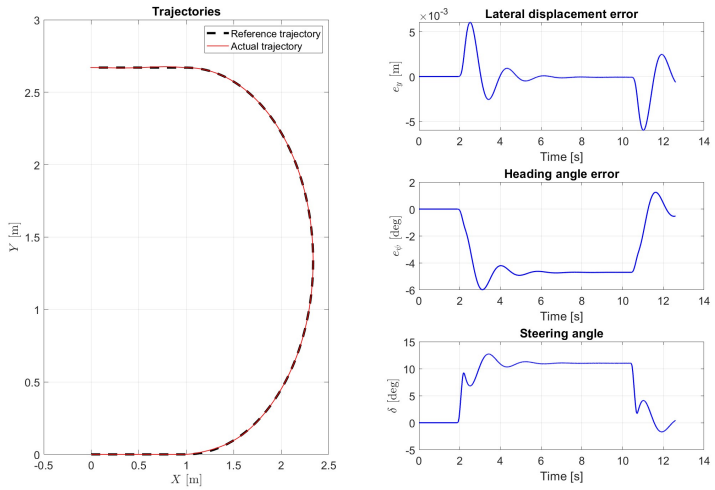


Figure 7.3: Simulation results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

7.1.3 S-shaped trajectory

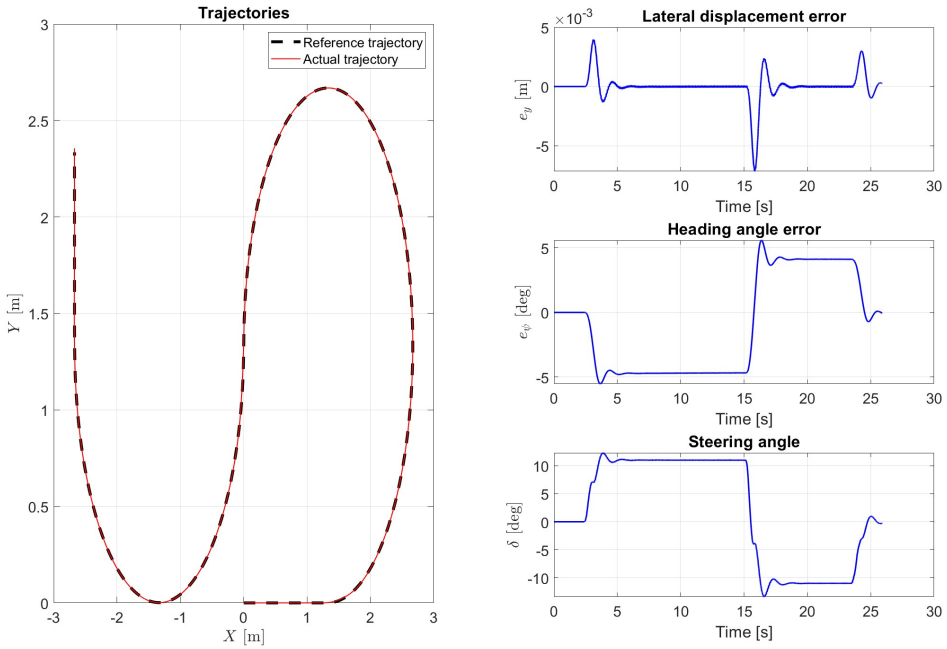


Figure 7.4: Simulation results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

7.1.4 Eight-shaped trajectory

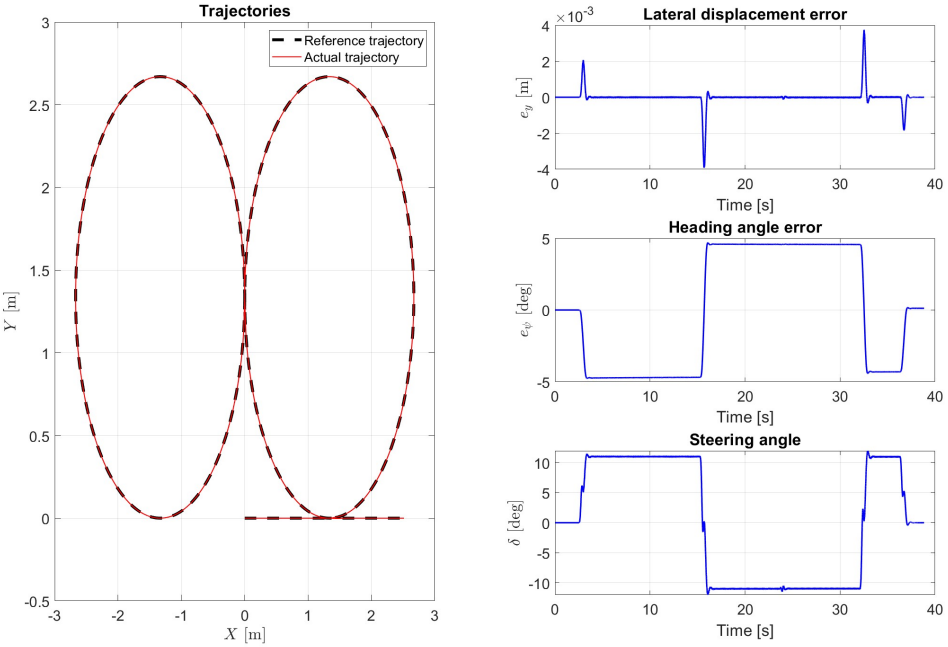


Figure 7.5: Simulation results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

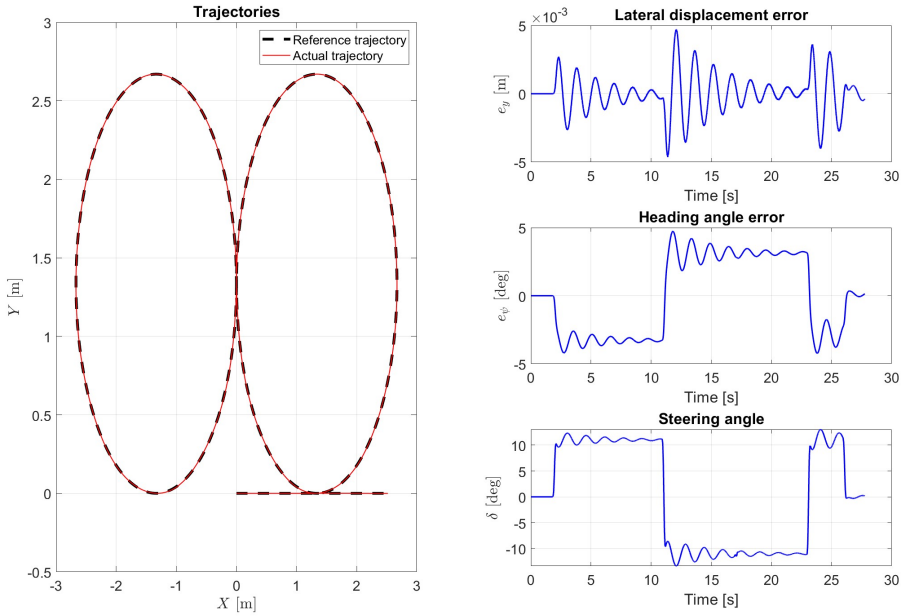


Figure 7.6: Simulation results at 0.7 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

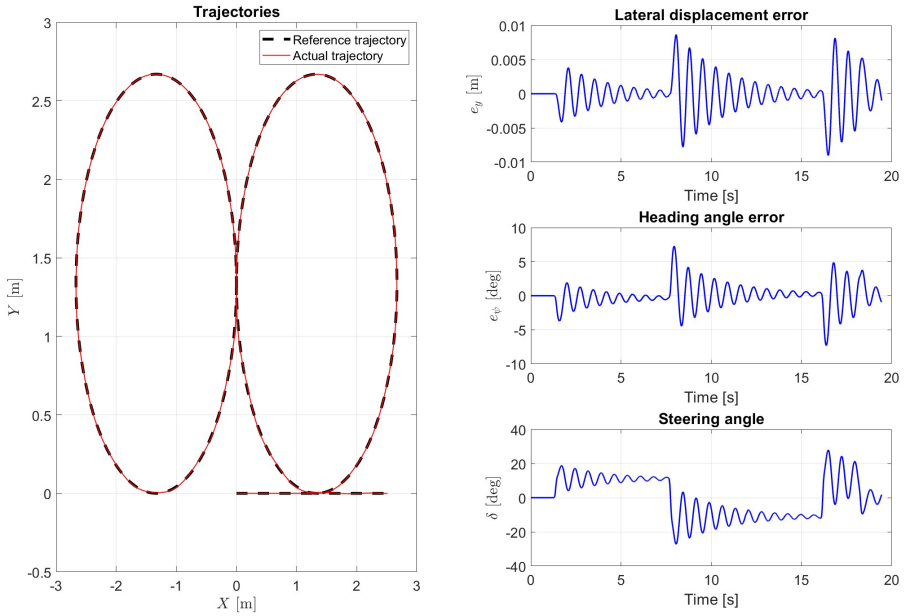


Figure 7.7: Simulation results at 1.0 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

As velocity of the vehicle increases, the control problem becomes more challenging. This leads to an increase in the lateral displacement error and a deterioration in control performances, as the oscillatory behaviour increases despite the increase in integral contribution. This is one of the drawbacks of this baseline controller.

7.1.5 Obstacle-avoidance trajectory

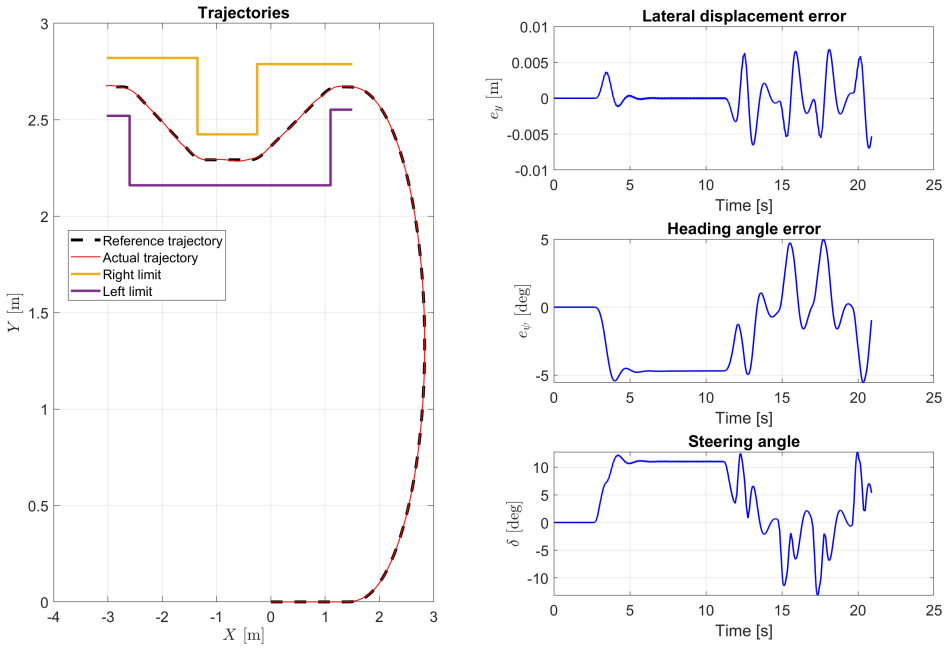


Figure 7.8: Simulation results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

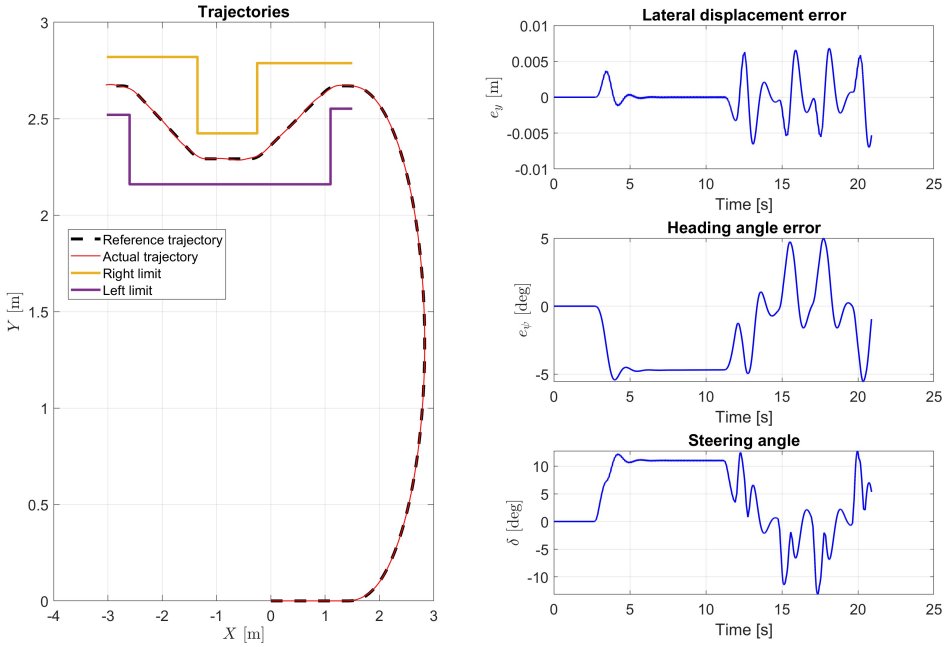


Figure 7.9: Simulation results at 0.7 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

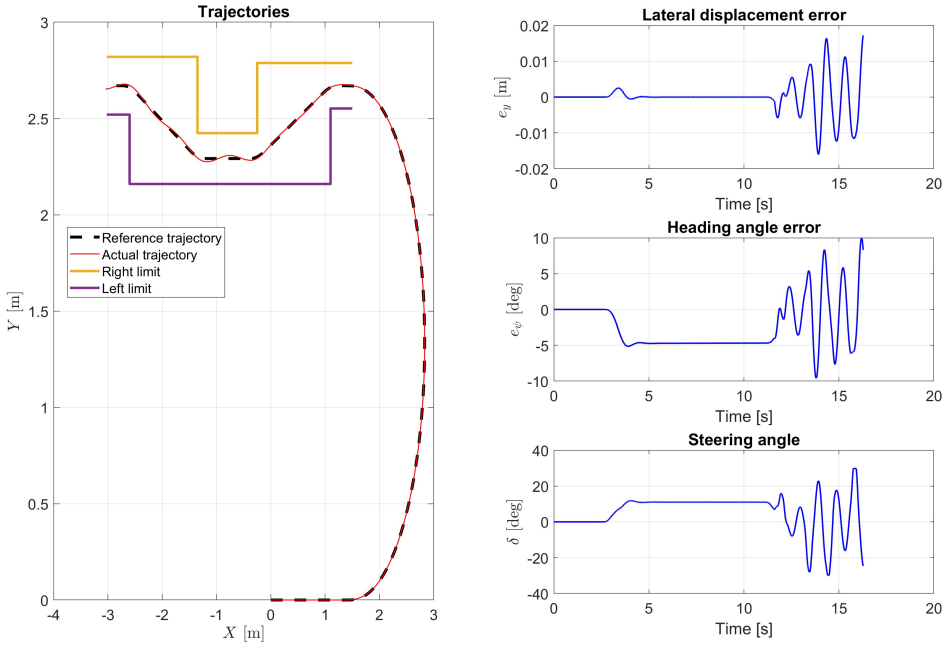


Figure 7.10: Simulation results at 1.0 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

As for the eight-shaped trajectory, also here we can observe a deterioration of performances as the speed increases. The tuning of the controller remains the same across all trajectories which can account for the larger lateral errors experienced in this more challenging trajectory. Furthermore, since the controller tuning is speed-dependent but performed offline, it results in being undertuned in the initial section due to the varying speed profile after the initial U-shaped section, as previously explained.

7.2 LQR controller

This section present the simulation results obtained with LQR controller. As it can be noticed, The LQR controller demonstrates superior performance, compared to the previous controller. For simpler trajectories such as S-shaped and U-shaped paths, the lateral displacement errors are on the order of magnitude of 10^4 meters. Additionally, the control action shows significantly reduced oscillations, even at higher speeds, particularly on the eight-shaped and obstacle-avoidance trajectories. This represents a significant advantage of this controller and is consistent with the outcomes observed during experimental tests.

7.2.1 Circular-shaped trajectory

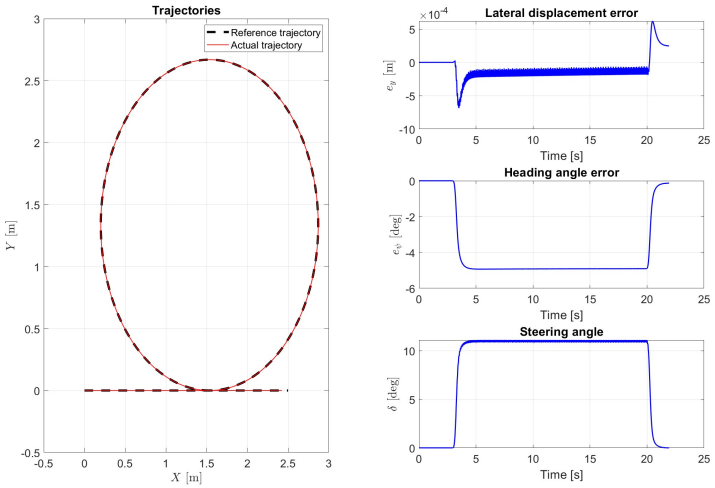


Figure 7.11: Simulation results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

7.2.2 U-shaped trajectory

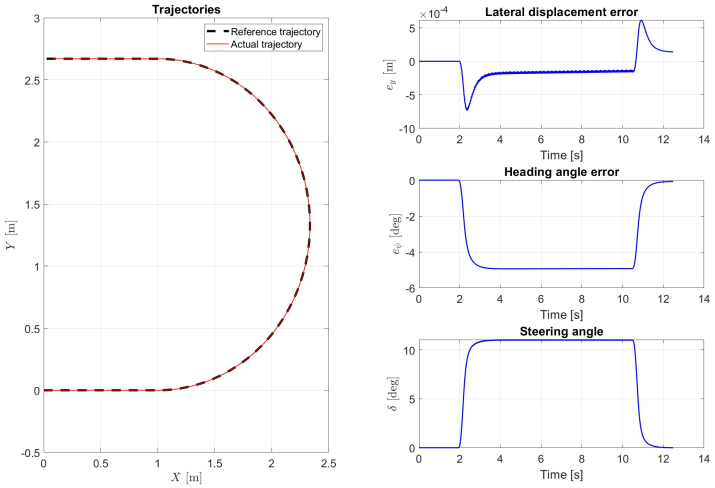


Figure 7.12: Simulation results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

7.2.3 S-shaped trajectory

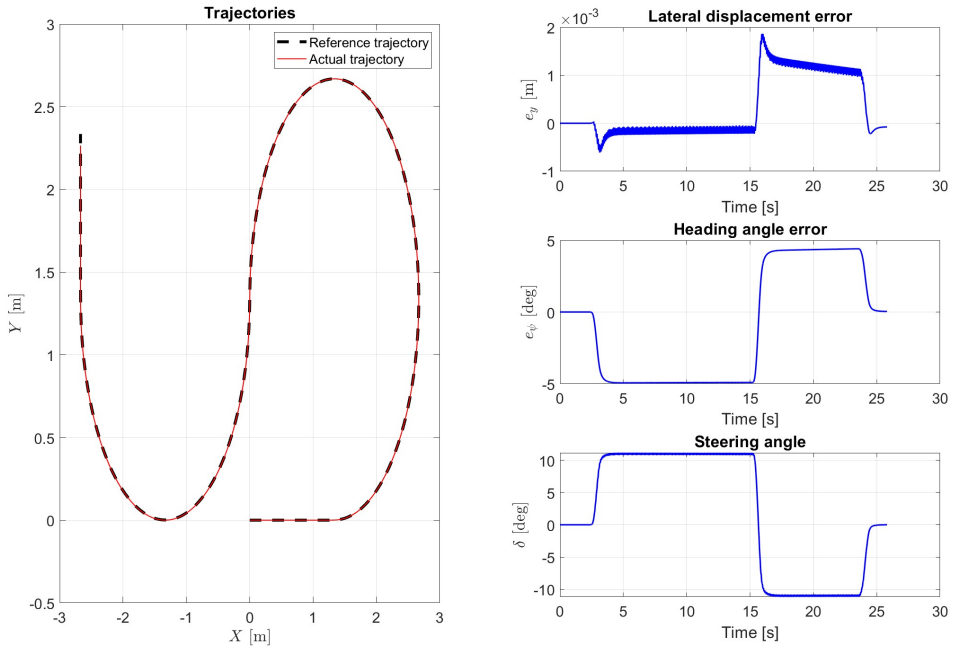


Figure 7.13: Simulation results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

7.2.4 Eight-shaped trajectory

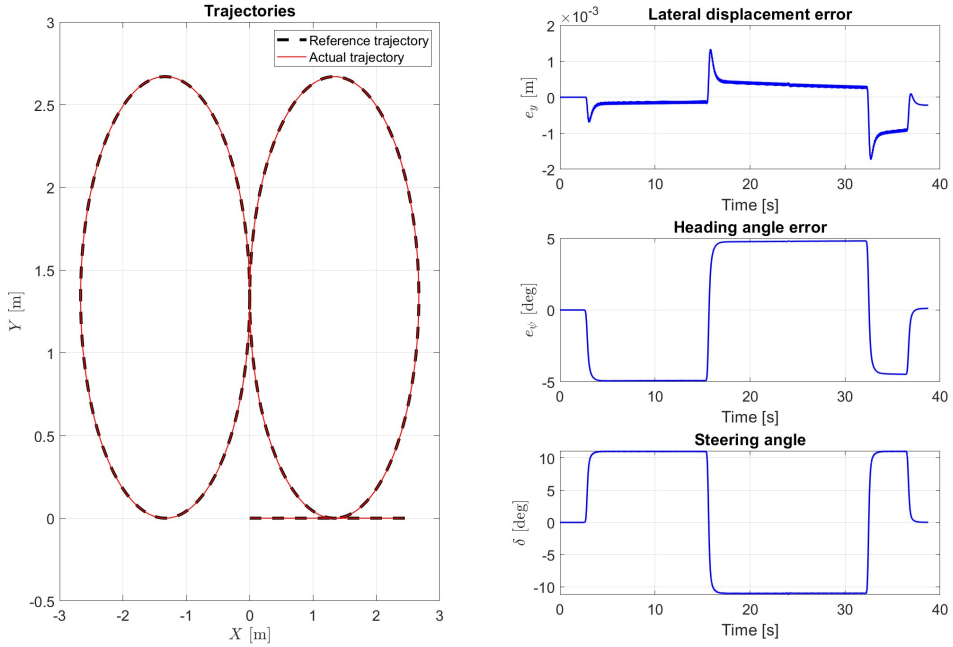


Figure 7.14: Simulation results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

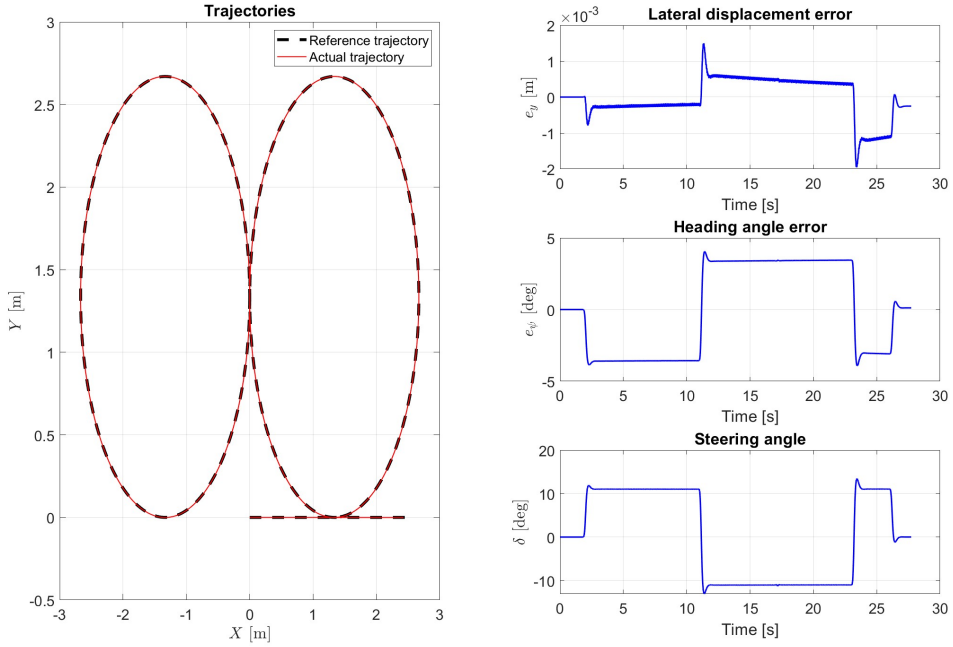


Figure 7.15: Simulation results at 0.7 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

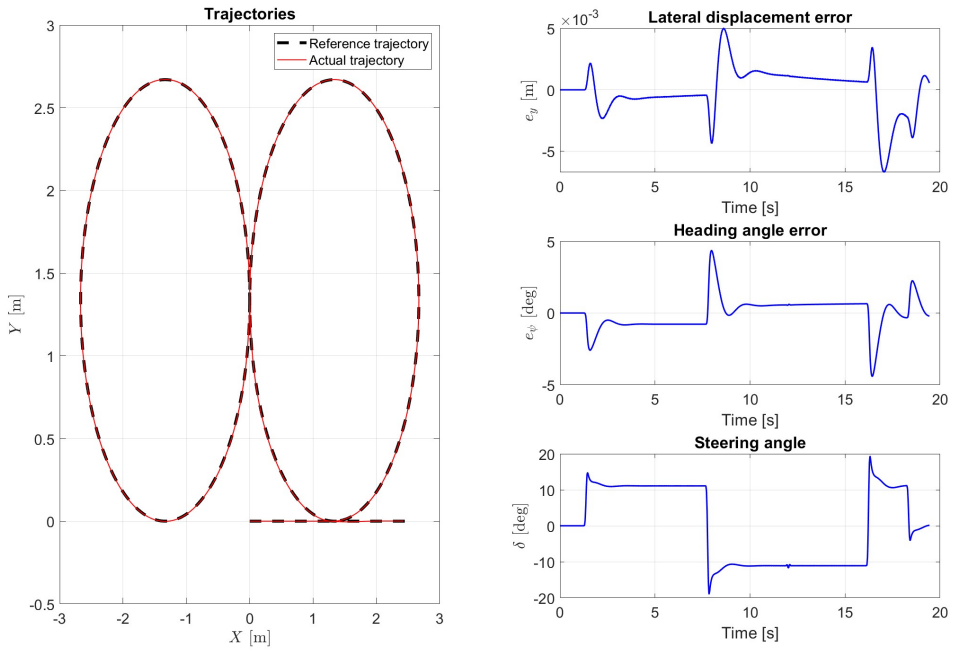


Figure 7.16: Simulation results at 1.0 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

7.2.5 Obstacle-avoidance trajectory

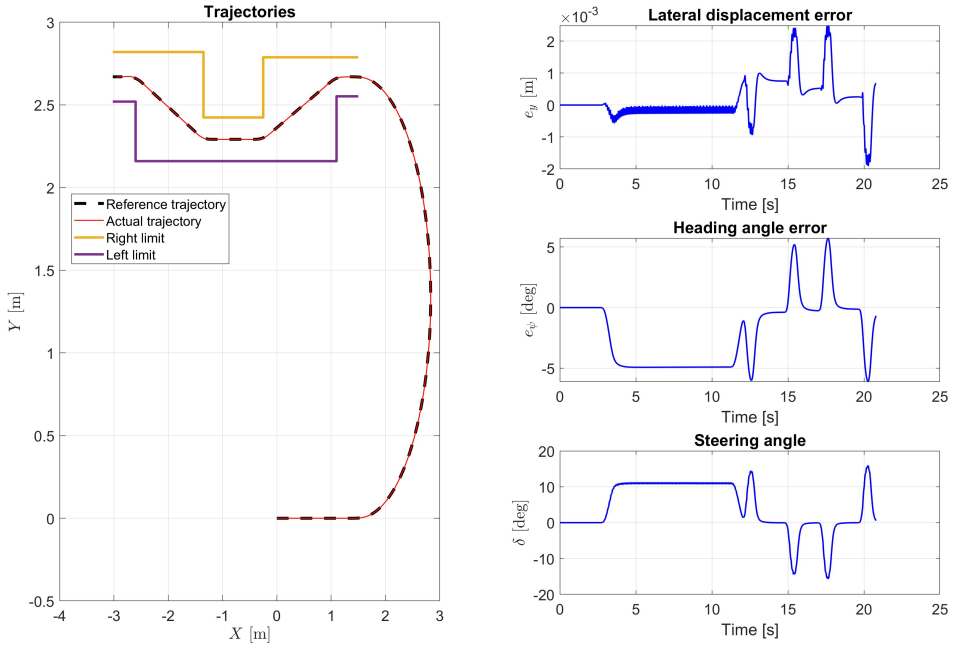


Figure 7.17: Simulation results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

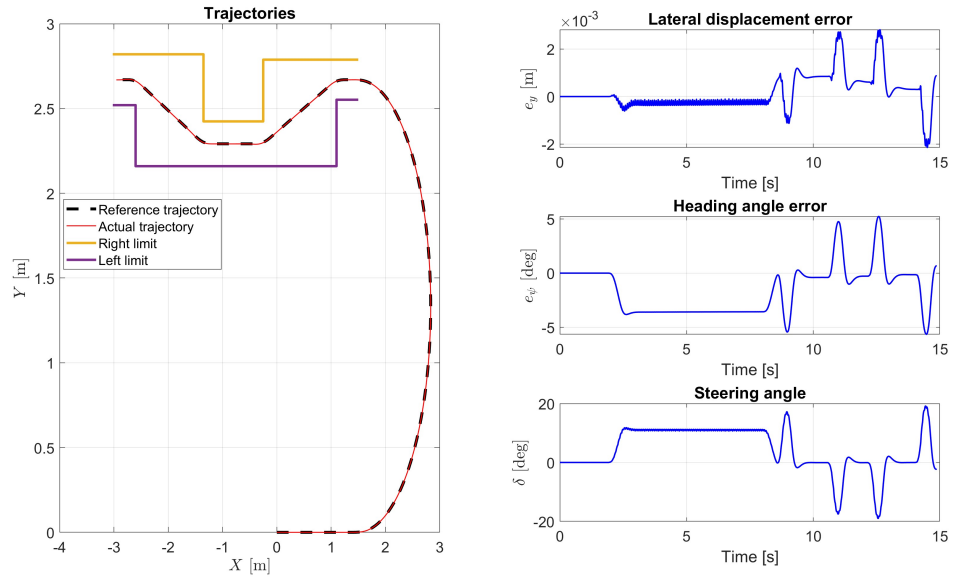


Figure 7.18: Simulation results at 0.7 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

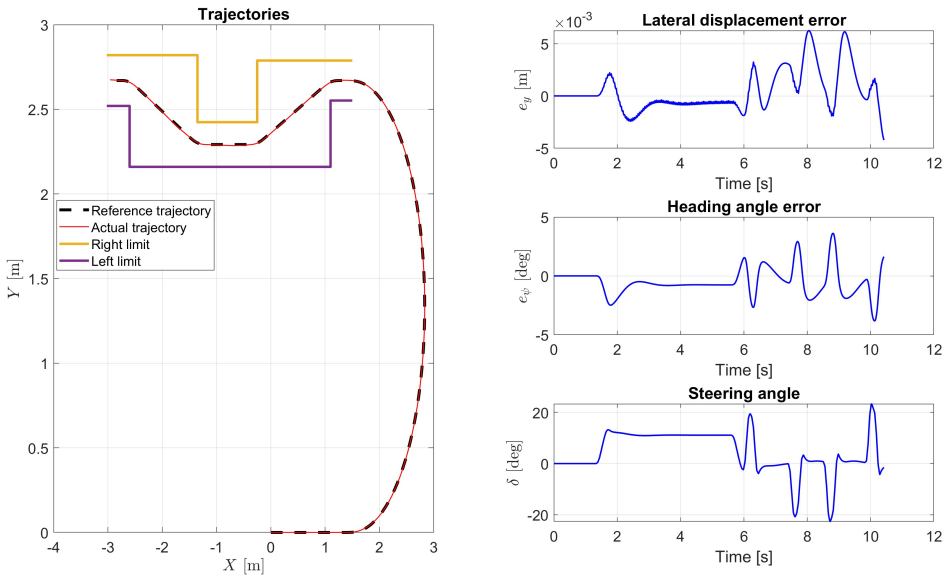


Figure 7.19: Simulation results at 1.0 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

For complex trajectories, simulation results show that the lateral error increases with increasing velocity. However, in comparison to the PI controller, the LQR controller exhibits less oscillatory control action.

Chapter 8

Experimental Results

In this chapter the aim is to present the results for each controller individually in a low-speed scenario for all five trajectories. For the sake of brevity, there will be included plots at higher speeds only for the most demanding trajectories. Considered though that the experimental tests have been conducted for all the velocities listed here [0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.2, 1.5] m/s, all the results will be presented in a more compact format using Key-Performance-Indicators (KPIs), with istogram-bar plots in the following section.
One general observation is about the disparity between real-world signals and the simulated ones. The former present a pronounced oscillatory behaviour which is accentuated in the PI control scheme and slightly reduced when using the LQR control architecture. In the first case, these oscillations are a results of the trade-off made during the experimental tuning, where adjustments were necessary to adapt coefficients from the optimization routine. This trade-off was intended to achieve smaller lateral displacement error with the drawback of having more oscillatory control action.

8.1 Proportional-Integral controller

8.1.1 Circular-shaped trajectory

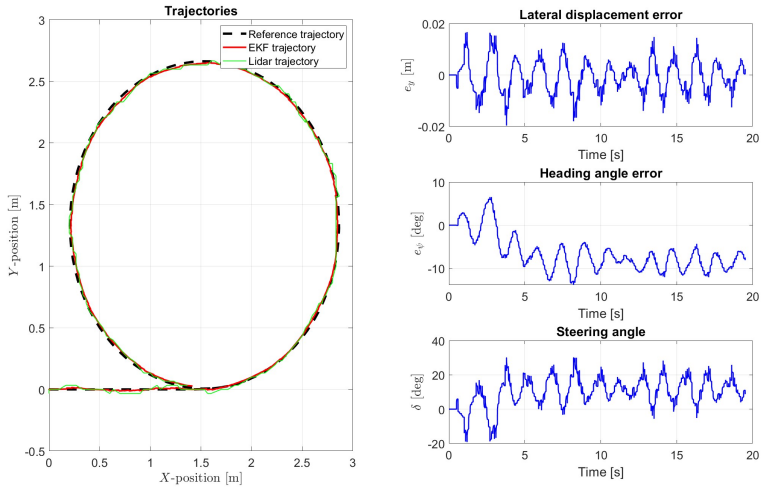


Figure 8.1: Experimental results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

8.1.2 U-shaped trajectory

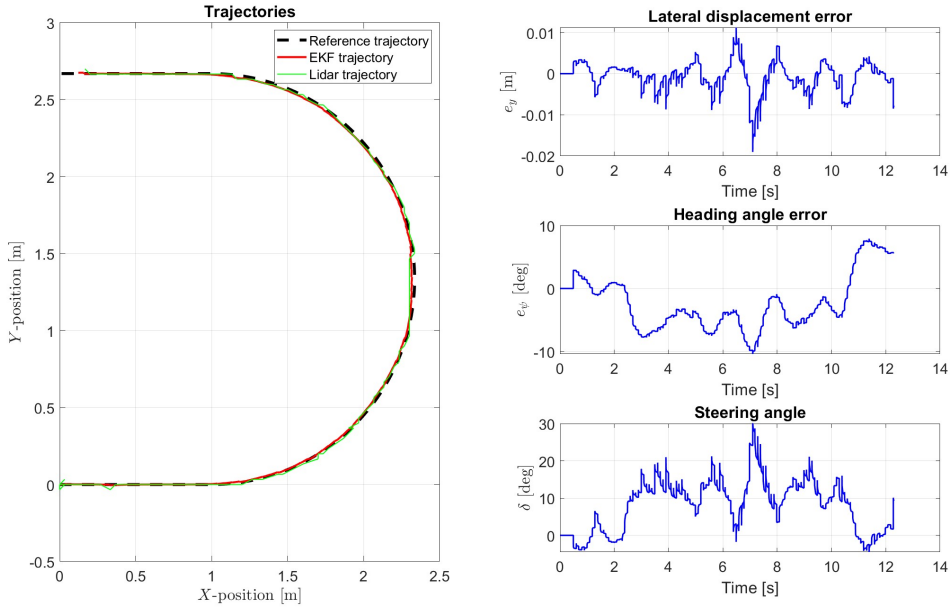


Figure 8.2: Experimental results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

8.1.3 S-shaped trajectory

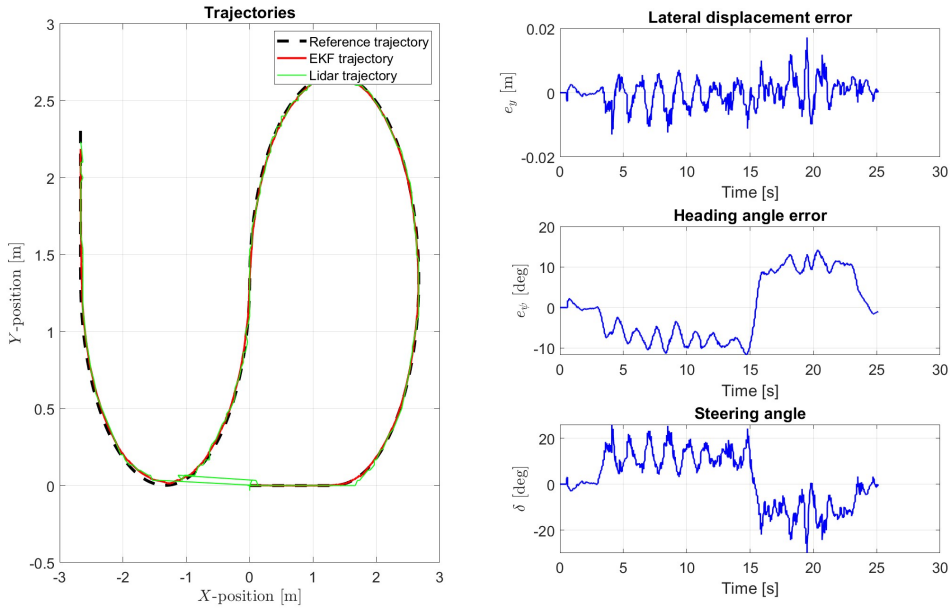


Figure 8.3: Experimental results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

8.1.4 Eight-shaped trajectory

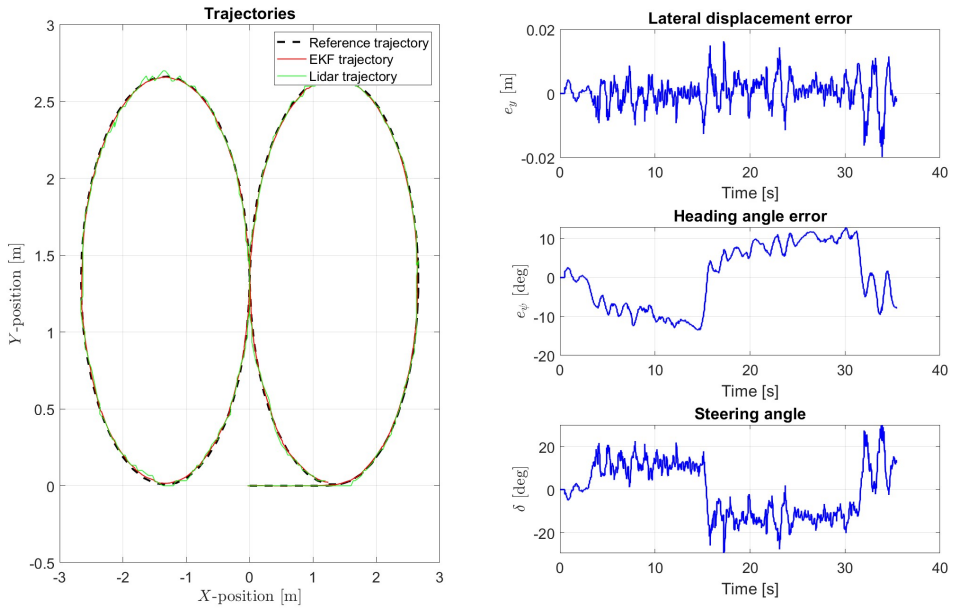


Figure 8.4: Experimental results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

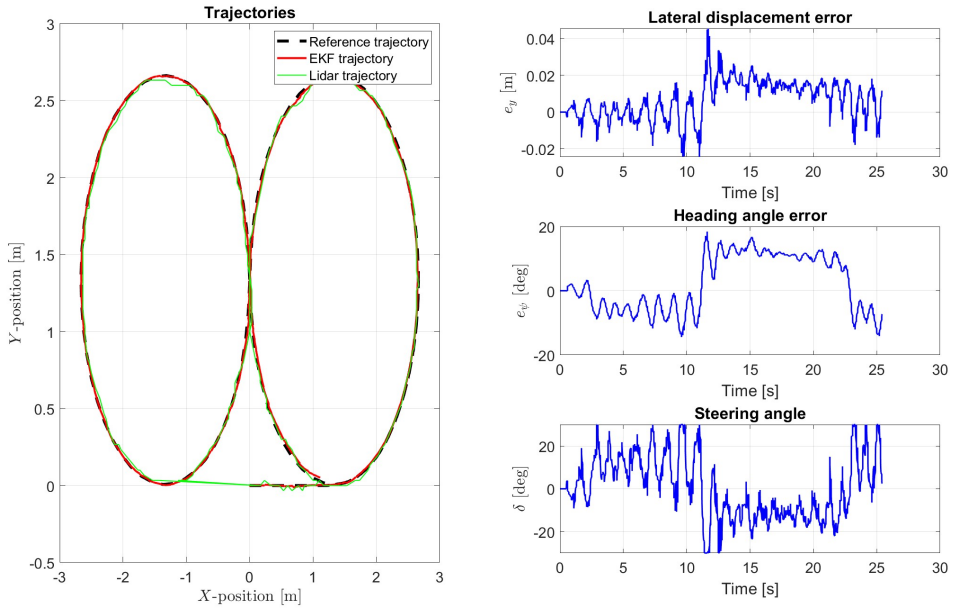


Figure 8.5: Experimental results at 0.7 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

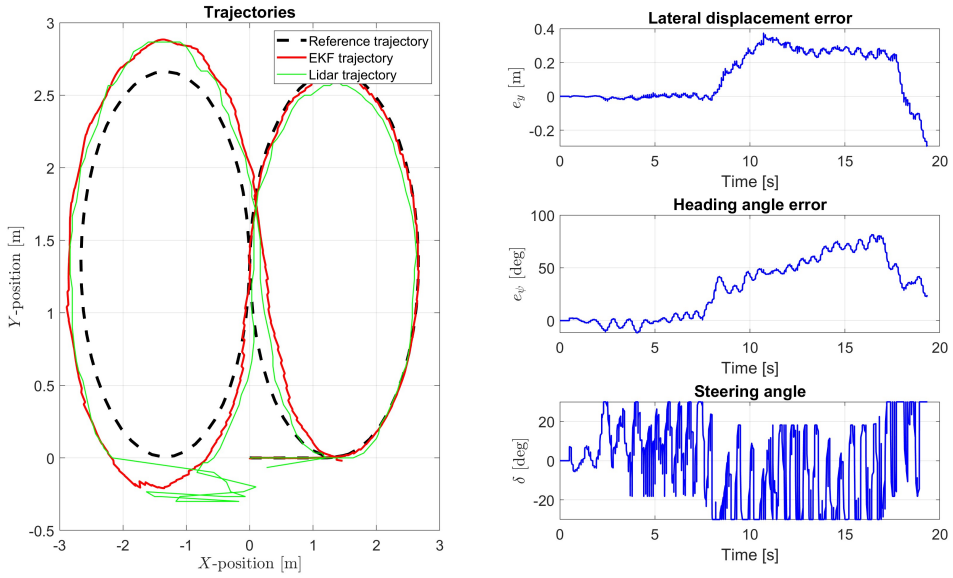


Figure 8.6: Experimental results at 1 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

Figure 8.6 shows that the PI controller, in the case of eight-shaped trajectory travelled at 1 m/s, faced some difficulty in tracking, with noticeable lateral errors and a highly oscillating control action. Meanwhile, Figure 8.15 demonstrates the superior performance of the LQR control scheme for the same scenario.

8.1.5 Obstacle-avoidance trajectory

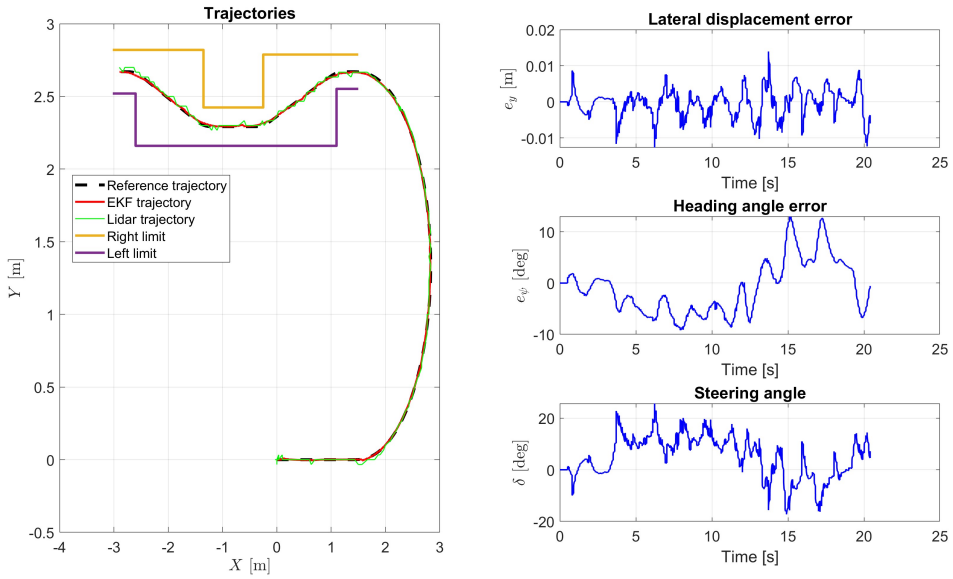


Figure 8.7: Experimental results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

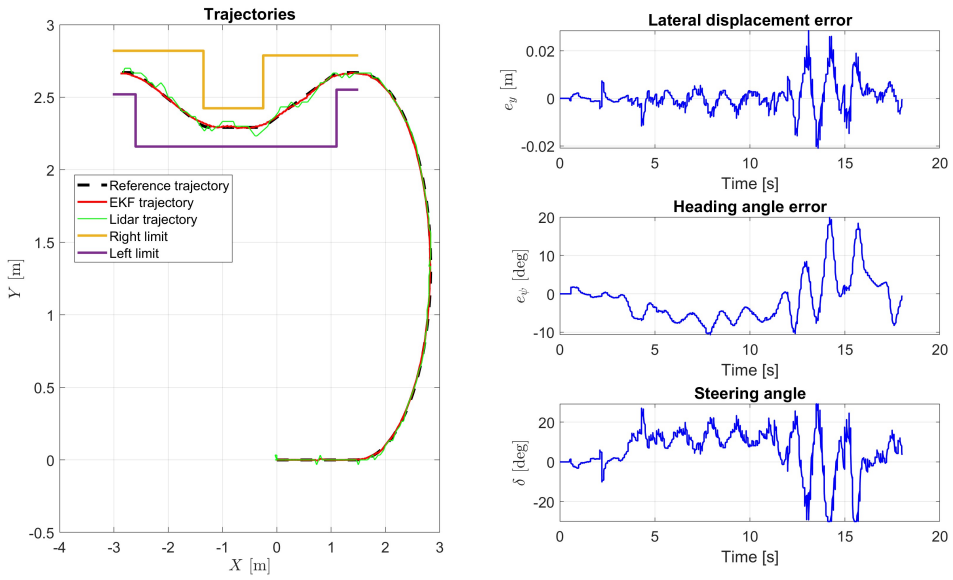


Figure 8.8: Experimental results at 0.7 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

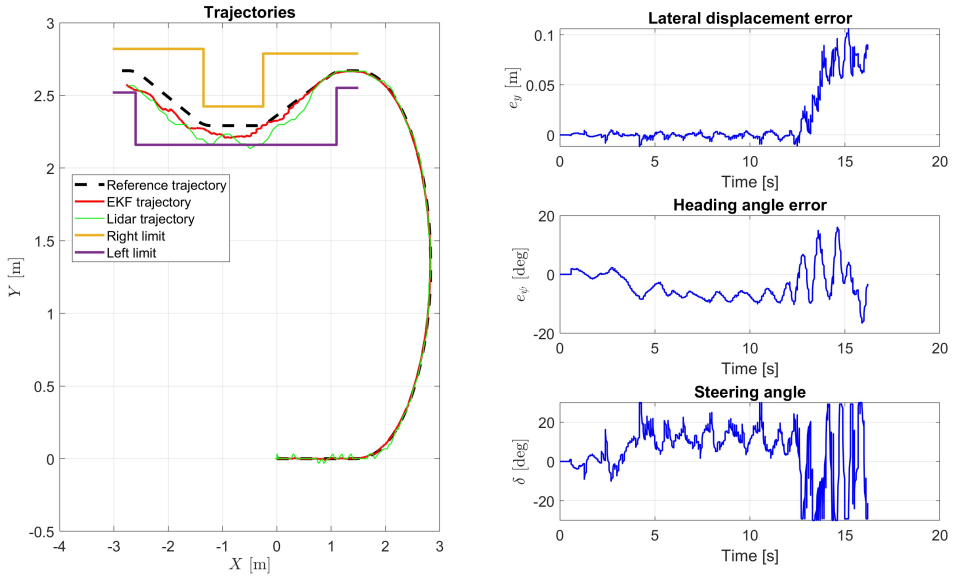


Figure 8.9: Experimental results at 1 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

In this section it was chosen to include all the figures from the experimental tests to show the cases in which the control system was able to make the car perform the complete maneuver. Notably, at all velocities less or equal than 0.9 m/s the vehicle passed the test without any collision with obstacles, with a maximum lateral error of 5.10 cm, as shown in figure **??**.

8.2 LQR controller

8.2.1 Circular-shaped trajectory

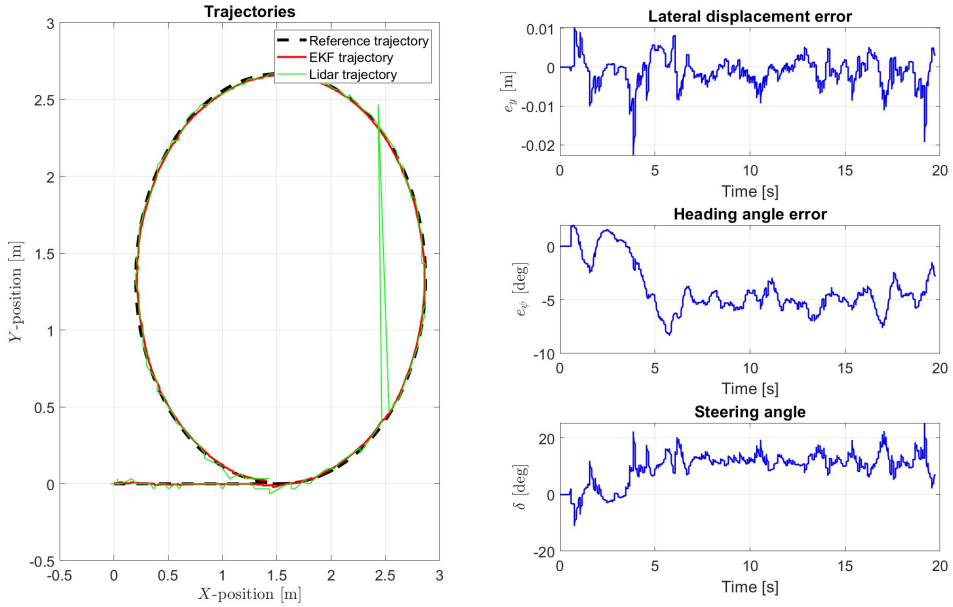


Figure 8.10: Experimental results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

8.2.2 U-shaped trajectory

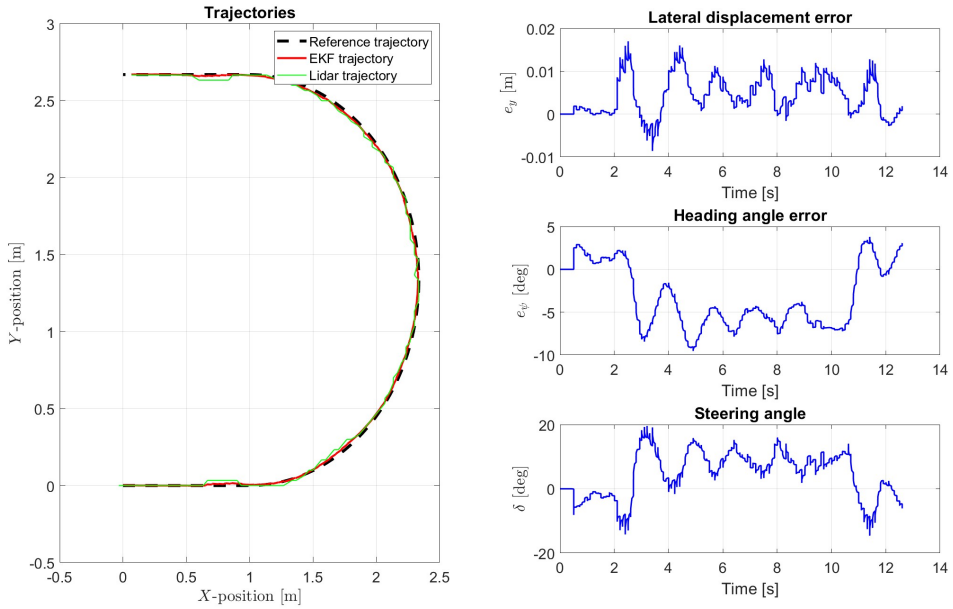


Figure 8.11: Experimental results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

8.2.3 S-shaped trajectory

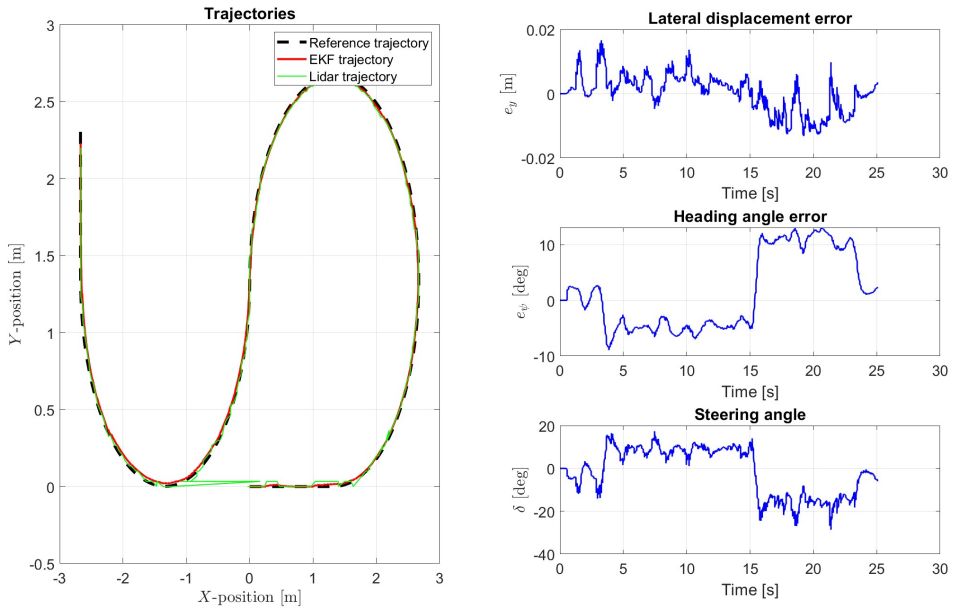


Figure 8.12: Experimental results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

8.2.4 Eight-shaped trajectory

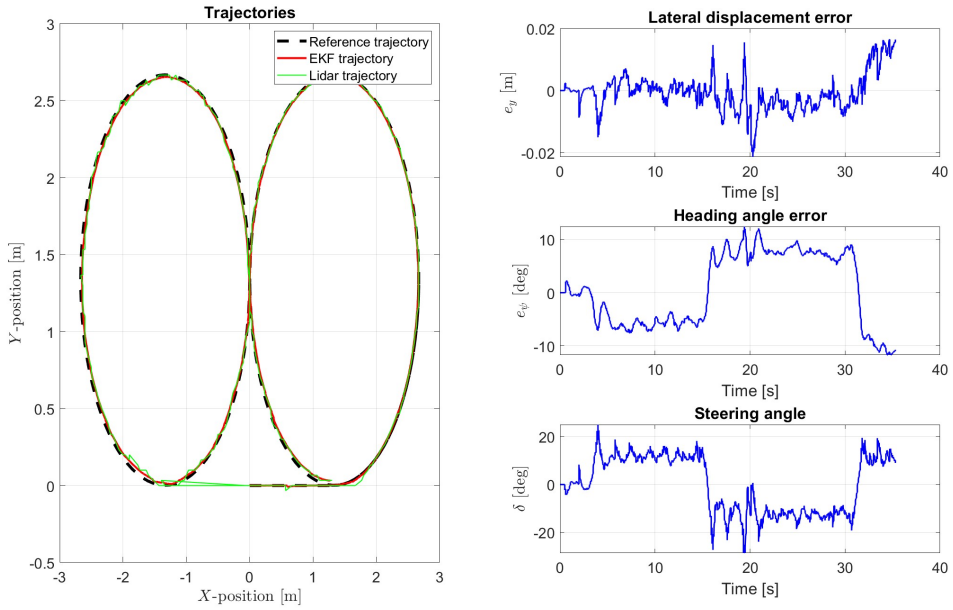


Figure 8.13: Experimental results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

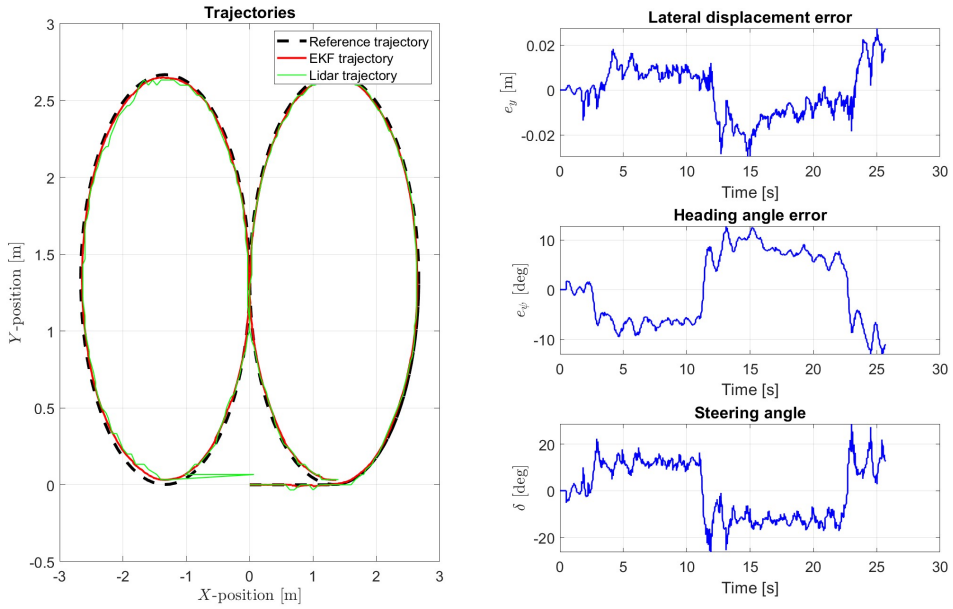


Figure 8.14: Experimental results at 0.7 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

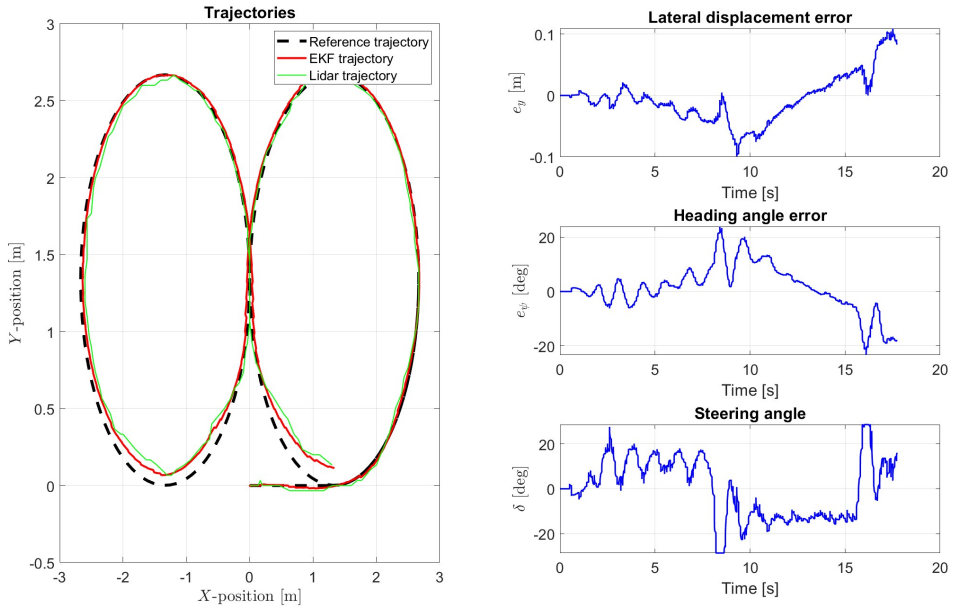


Figure 8.15: Experimental results at 1 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

8.2.5 Obstacle-avoidance trajectory

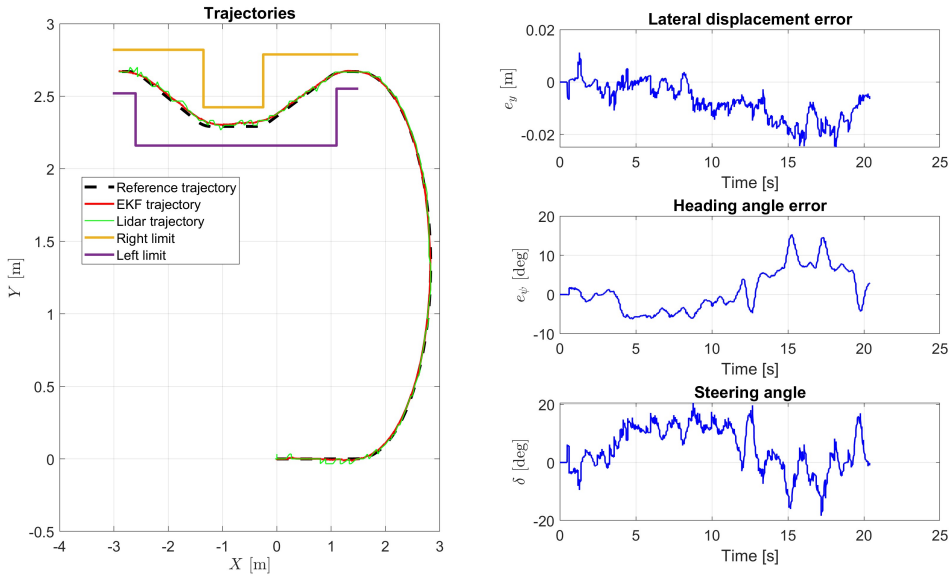


Figure 8.16: Experimental results at 0.5 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

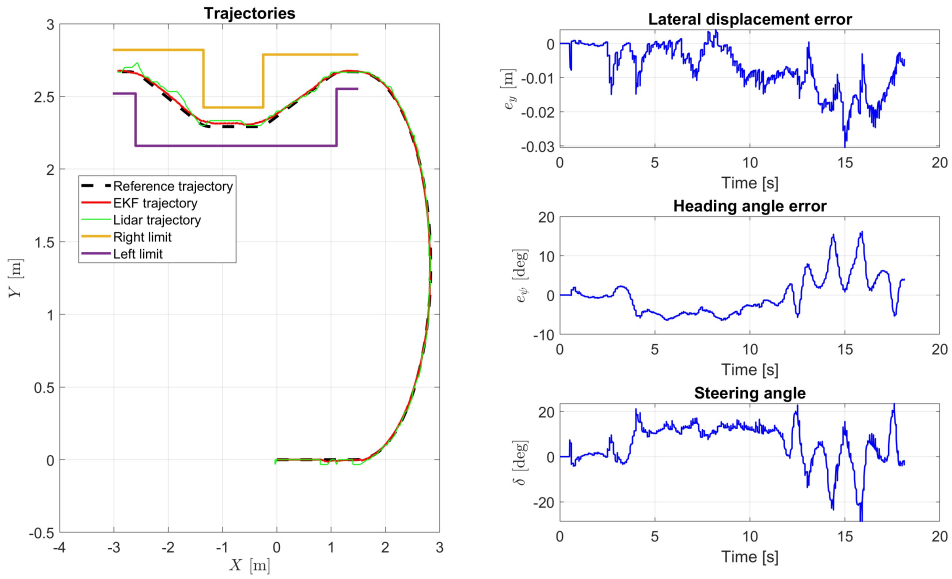


Figure 8.17: Experimental results at 0.7 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

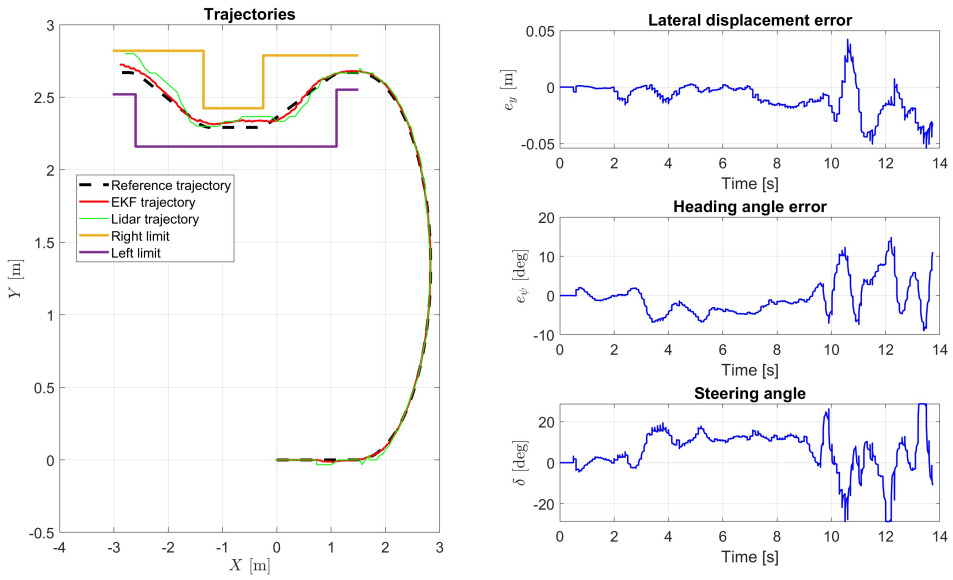


Figure 8.18: Experimental results at 1 m/s: reference and actual trajectories, lateral displacement and heading angle errors, control action.

As done for the PI control scheme, also here are presented plots at several velocities to put evidence on the manoeuvre execution limit. In this case, the manouvre is completed up to a speed of 1.0 m/s.

8.3 Comparison between experimental results

In this section the aim is to present a direct comparison between two control techniques within the experimental tests conducted on the five trajectory at several velocities. Performances are evaluated in a coincise way through six Key-Performance-Indicators (KPI):

- 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 (IACA);
- Integral of the absolute value of the variations of the control action (IAVCA).

The analytical expression of the KPIs listed before is given below:

$$MAX_{e_y} = \max(e_y) \tag{8.1}$$

$$RMS_{e_y} = \sqrt{\frac{1}{t_{fin} - t_{in}} \int_{t_{in}}^{t_{fin}} (e_y)^2 dt} \tag{8.2}$$

$$MAX_{e_\psi} = \max(e_\psi) \tag{8.3}$$

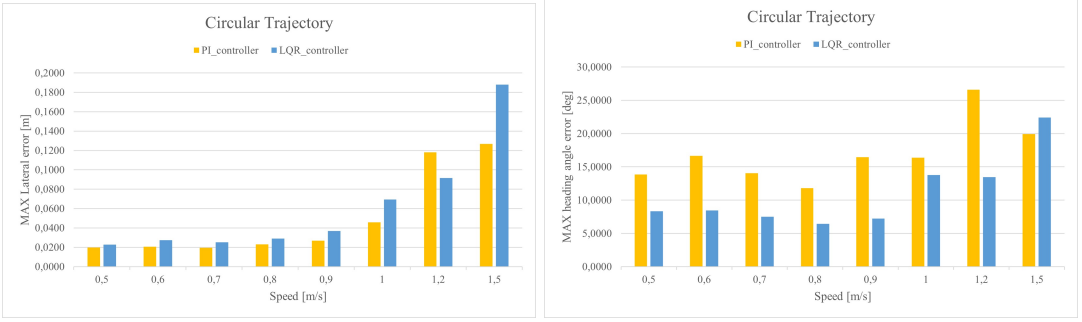
$$RMS_{e_\psi} = \sqrt{\frac{1}{t_{fin} - t_{in}} \int_{t_{in}}^{t_{fin}} (e_\psi)^2 dt} \tag{8.4}$$

$$IACA_\delta = \frac{1}{t_{fin} - t_{in}} \int_{t_{in}}^{t_{fin}} |\delta| \, dt \tag{8.5}$$

$$IAVCA_\delta = \frac{1}{t_{fin} - t_{in}} \int_{t_{in}}^{t_{fin}} |\Delta\delta| \, dt \tag{8.6}$$

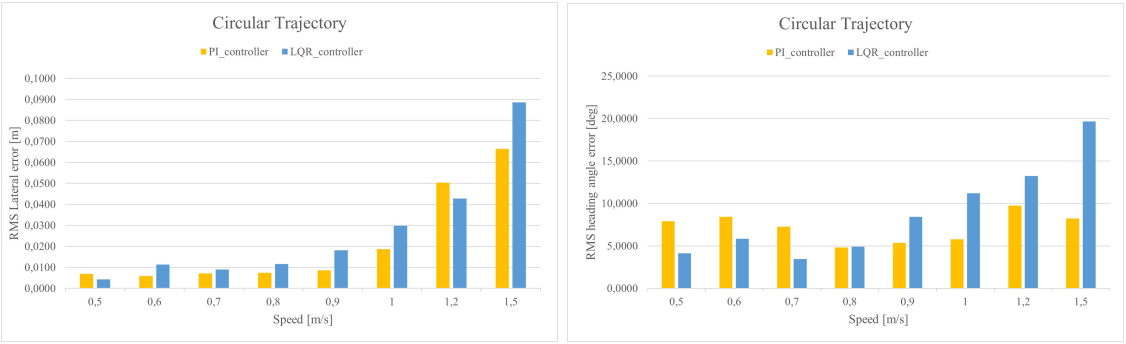
8.3.1 Circular-shaped trajectory

Figure 8.19: Maximum errors



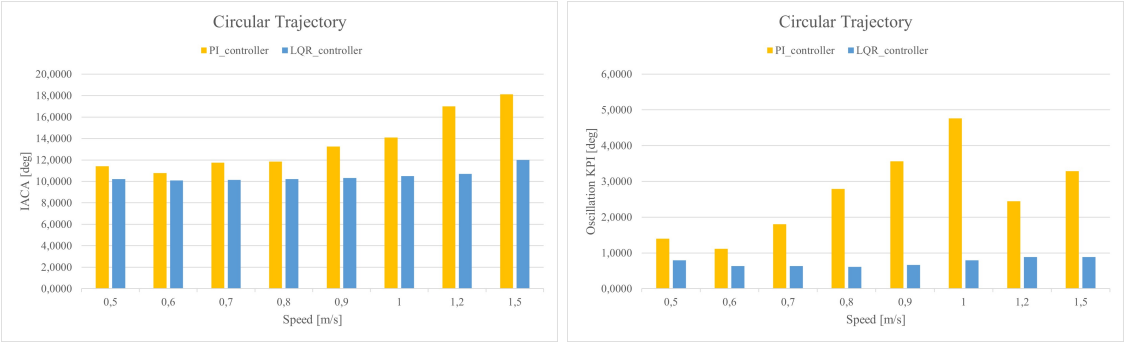
(a) Maximum lateral error (b) Maximum heading angle error

Figure 8.20: Root-Mean-Squared-errors



(a) Root-mean-squared lateral error (b) Root-mean-squared heading angle error

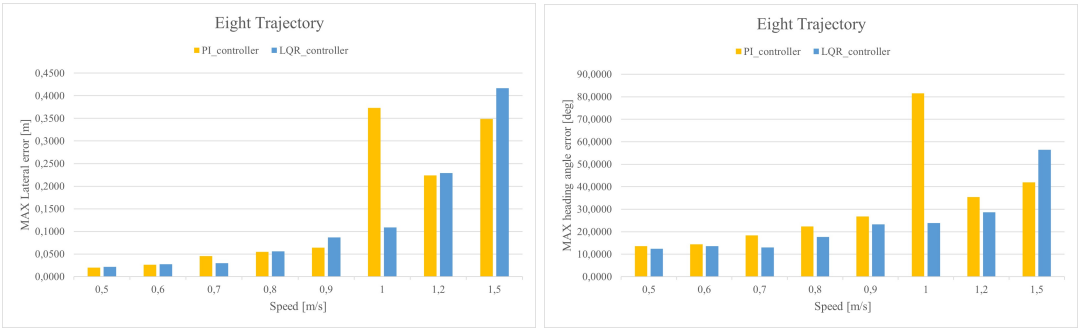
Figure 8.21: Control action Key-Performance-Indicators



(a) IACA Key-Performance-Indicator (b) Oscillation Key-Performance-Indicator

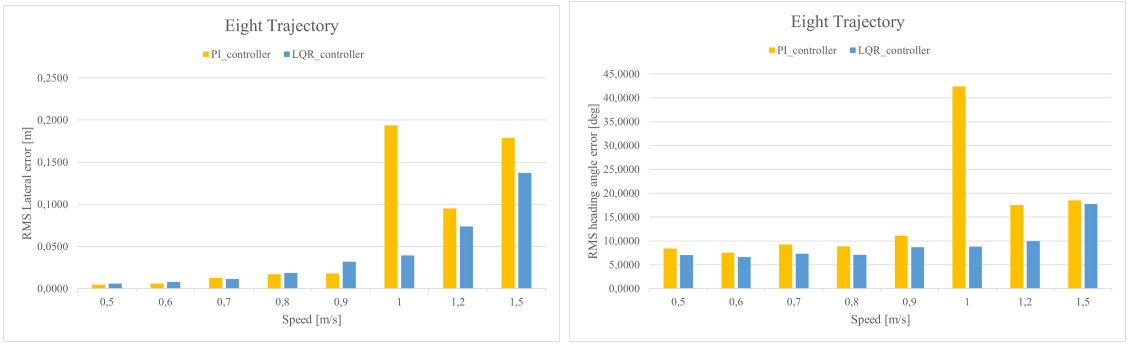
8.3.2 Eight-shaped trajectory

Figure 8.22: Maximum errors



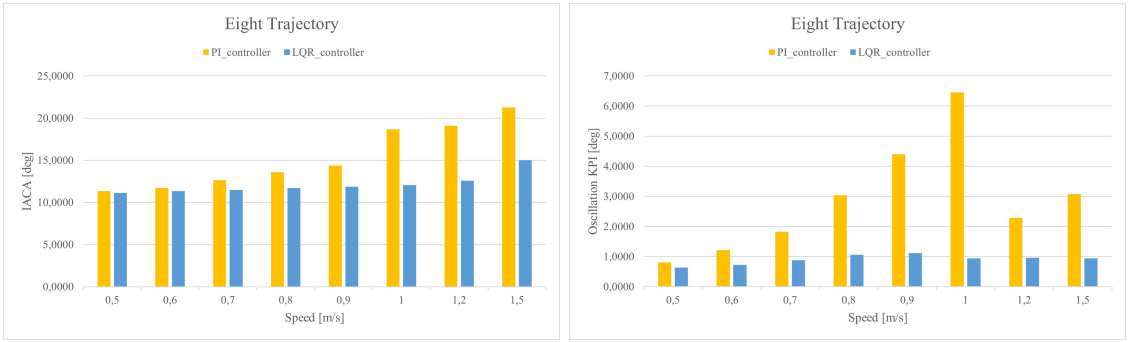
(a) Maximum lateral error (b) Maximum heading angle error

Figure 8.23: Root-Mean-Squared-errors



(a) Root-mean-squared lateral error (b) Root-mean-squared heading angle error

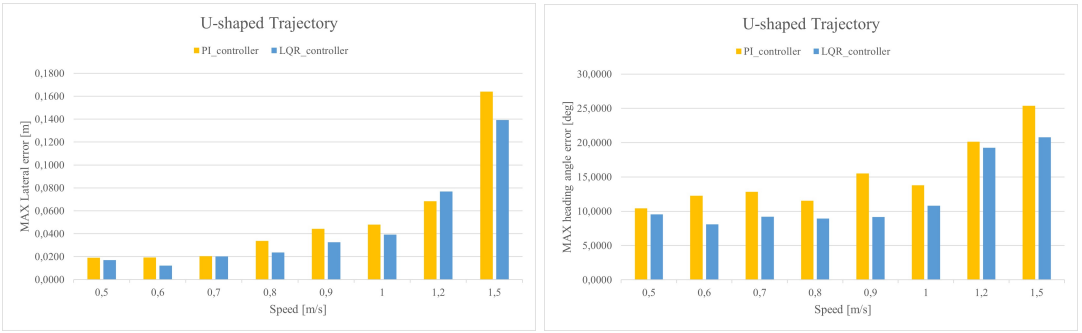
Figure 8.24: Control action Key-Performance-Indicators



(a) IACA Key-Performance-Indicator (b) Oscillation Key-Performance-Indicator

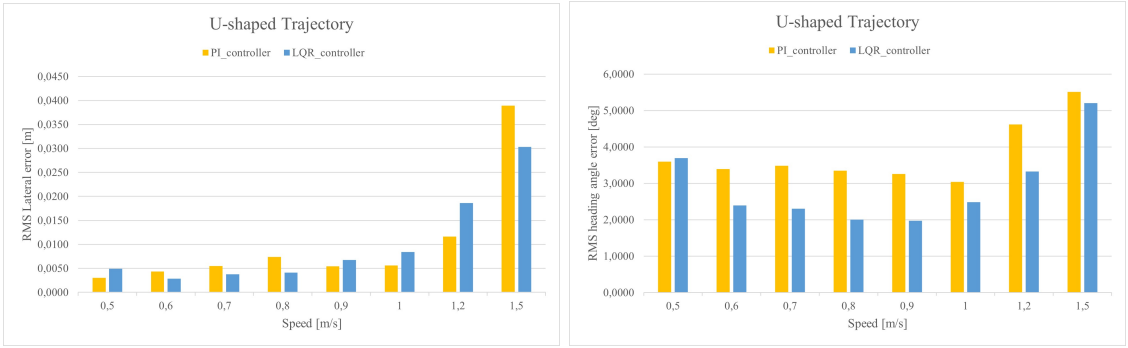
8.3.3 U-shaped trajectory

Figure 8.25: Maximum errors



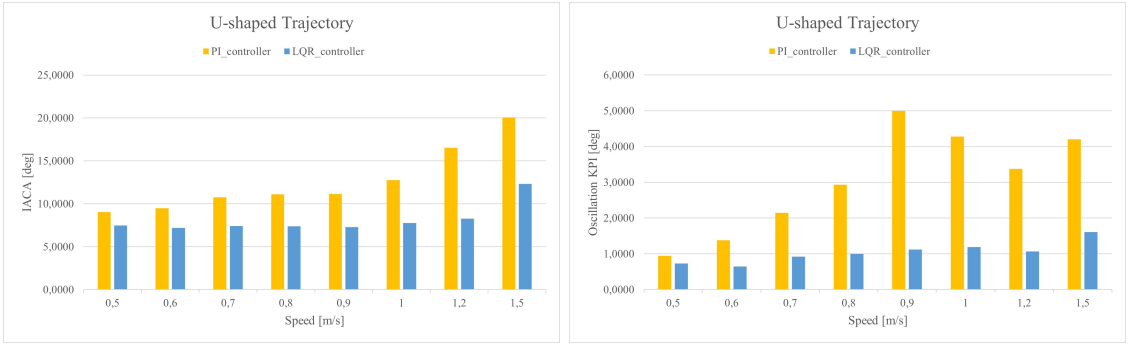
(a) Maximum lateral error (b) Maximum heading angle error

Figure 8.26: Root-Mean-Squared-errors



(a) Root-mean-squared lateral error (b) Root-mean-squared heading angle error

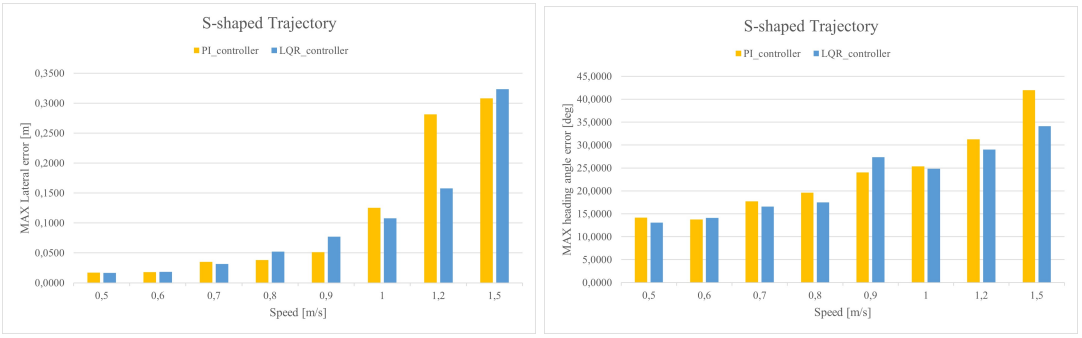
Figure 8.27: Control action Key-Performance-Indicators



(a) IACA Key-Performance-Indicator (b) Oscillation Key-Performance-Indicator

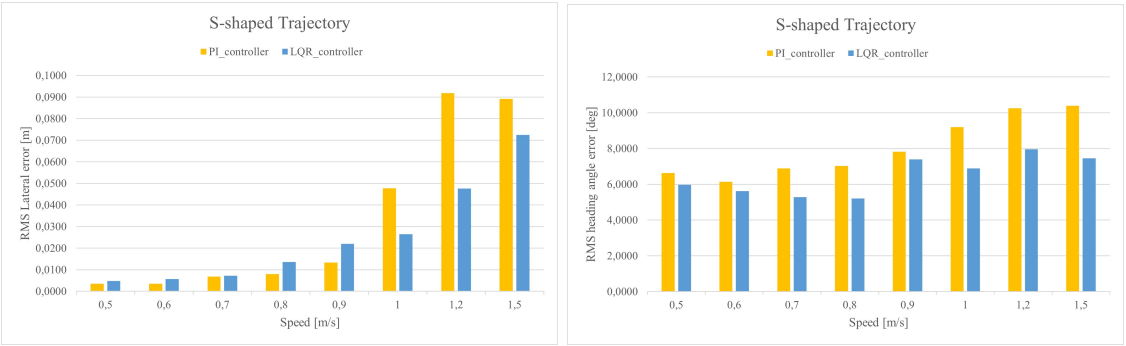
8.3.4 S-shaped trajectory

Figure 8.28: Maximum errors



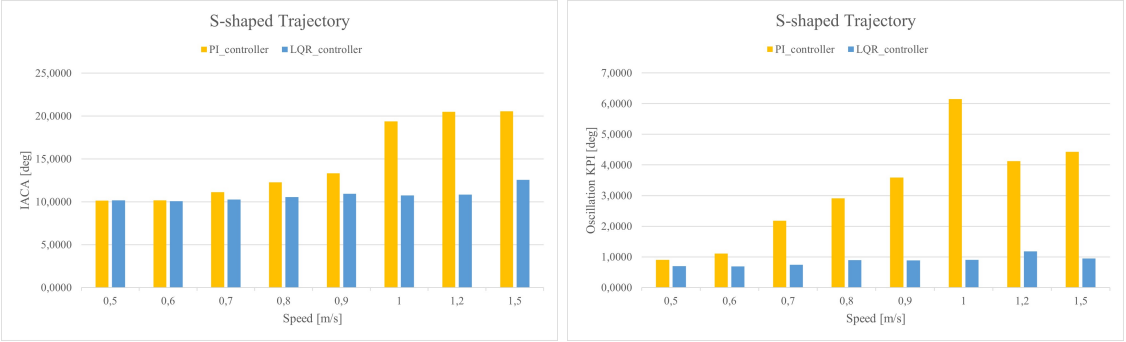
(a) Maximum lateral error (b) Maximum heading angle error

Figure 8.29: Root-Mean-Squared-errors



(a) Root-mean-squared lateral error (b) Root-mean-squared heading angle error

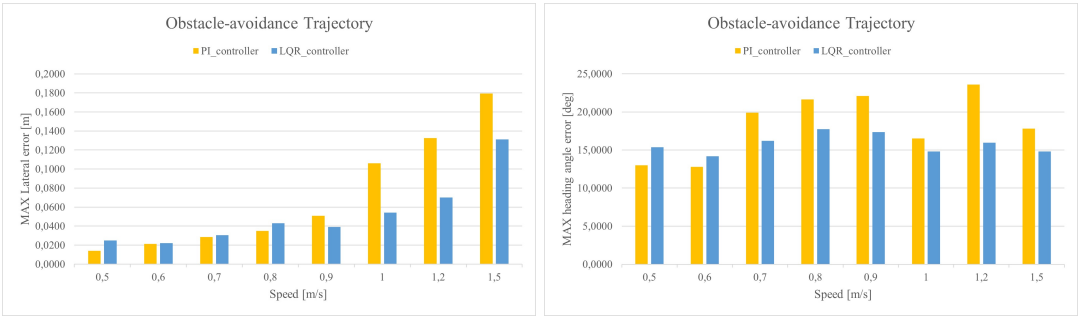
Figure 8.30: Control action Key-Performance-Indicators



(a) IACA Key-Performance-Indicator (b) Oscillation Key-Performance-Indicator

8.3.5 Obstacle-avoidance trajectory

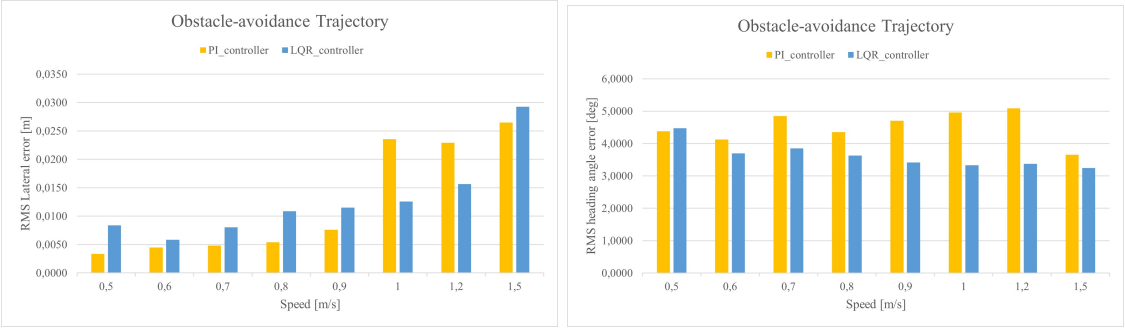
Figure 8.31: Maximum errors



(a) Maximum lateral error

(b) Maximum heading angle error

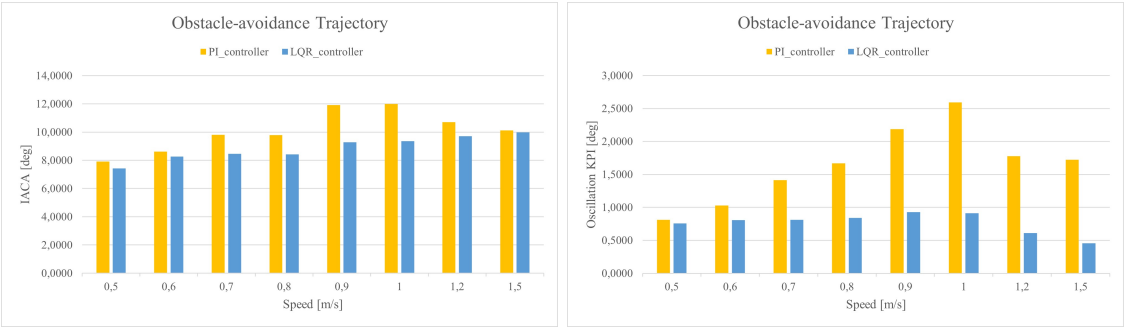
Figure 8.32: Root-Mean-Squared-errors



(a) Root-mean-squared lateral error

(b) Root-mean-squared heading angle error

Figure 8.33: Control action Key-Performance-Indicators



(a) IACA Key-Performance-Indicator

(b) Oscillation Key-Performance-Indicator

8.4 Conclusions and future works

To conclude, some considerations about the comparison: in simulation it could be appreciated that the performaces of both controllers worsens with increasing speed and this is what was also obtained in the experimental tests. Although in simulation the LQR controller gave better performances in terms of maximum lateral error on all trajectories than

the PI controller, this was not always the case in the experiments: for simpler trajectory such as the U-shaped one the LQR controller performs better while for more complex trajectories the PI achieves smaller lateral errors. This can be explained by considering that the gains of the LQR controller are obtained through optimal placement, which in turn is based on dynamic parameters of the system, inherited from previous works and probably not very accurate; when the controller has to deal with a more complex trajectory, the dynamic effects have a greater influence on the behaviour of the vehicle leading to these variations in performance between the two controllers.
On the other hand it is important to consider that these good performances obtained with the PI controller are the result of a trade-off between lateral error and the quality of control action. In this trade-off, greater importance has been given on minimizing lateral error, resulting in slightly oscillatory steering signal. This trade-off is reflected in the IACA key performance indicator which is significantly higher for the PI controller.
If we consider the heading angle error, the LQR control resulted in a smaller error than that obtained with the PI control. This difference in performance can be attributed to the specific feedback signals used in each control scheme. In the PI control scheme the feedback only concerns the lateral displacement error, so the heading angle one is not directly considered. On the other hand the LQR controller operates within the state space representation where the heading angle error is one of the state variables, so it actively works to minimize both of them.

As a future development, to enhance this study, system identification techniques with Neural-Network approach will be implemented, in order to have a more realistic simulation model. This could be useful for the development of more complex control strategies and scenarios like vehicle platooning.

Bibliography

[1] D. Watzenig and M. Horn, *Automated driving: safer and more efficient future driving*. Springer, 2016.

[2] S. Sunil, S. Mozaffari, R. Singh, B. Shahrrava, and S. Alirezaee, "Feature-based occupancy map-merging for collaborative slam," *Sensors*, vol. 23, no. 6, p. 3114, 2023.

[3] J. Hu, Y. Zhang, and S. Rakheja, "Adaptive lane change trajectory planning scheme for autonomous vehicles under various road frictions and vehicle speeds," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 2, pp. 1252–1265, 2022.

[4] H. Ke, S. Mozaffari, S. Alirezaee, and M. Saif, "Cooperative adaptive cruise control using vehicle-to-vehicle communication and deep learning," in *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2022, pp. 435–440.

[5] ——, "Cooperative adaptive cruise control using vehicle-to-vehicle communication and deep learning," in *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2022, pp. 435–440.

[6] J. Hu, Y. Zhang, and S. Rakheja, "Adaptive trajectory tracking for car-like vehicles with input constraints," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 3, pp. 2801–2810, 2021.

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

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

[9] "Curvature," University of Cambridge, 2016, accessed on July 14, 2023. [Online]. Available: https://undergroundmathematics.org/glossary/curvature

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

[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] J. Guldner, H.-S. Tan, and S. Patwardhan, "Analysis of automatic steering control for highway vehicles with look-down lateral reference systems," *Vehicle System Dynamics*, vol. 26, no. 4, pp. 243–269, 1996.

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

[14] "The pid controller & theory explained," 2023, accessed on August 25, 2023. [Online]. Available: https://www.ni.com/en/shop/labview/pid-theory-explained.html

[15] "Anti-windup techniques." University of Trento, Prof. Alberto Bemporad, 2011, accessed on August 25, 2023. [Online]. Available: http://cse.lab.imtlucca.it/~bemporad/teaching/ac/pdf/AC2-09-AntiWindup.pdf

[16] H. Inoue, H. Mouri, H. Sato, A. Asaoka, and S. Ueda, "Technologies of nissan's automated highway system (ahs) test vehicle," 1996.

[17] R. Isermann, M. Schorn, and U. Stählin, "Anticollision system proreta with automatic braking and steering," *Vehicle System Dynamics*, vol. 46, no. S1, pp. 683–694, 2008.

[18] LibreTexts, "Pid tuning via classical methods," Year of Access, accessed on 20, July 2023. [Online]. Available: https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Chemical_Process_Dynamics_and_Controls_(Woolf)/09%3A_Proportional-Integral-Derivative_(PID)_Control/9.03%3A_PID_Tuning_via_Classical_Methods

[19] A. Buonomano, U. Montanaro, A. Palombo, S. Santini *et al.*, "Indoor air temperature control in buildings via an optimal tuned pi strategy," *Int. J. Eng. Innov. Technol*, vol. 4, no. 4, pp. 77–85, 2014.

[20] S. Dixit, U. Montanaro, M. Dianati, A. Mouzakitis, and S. Fallah, "Integral mrac with bounded switching gain for vehicle lateral tracking," *IEEE transactions on control systems technology*, vol. 29, no. 5, pp. 1936–1951, 2020.

[21] M Canale, "Automatic control, lecture notes." politecnico di Torino.