POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Modeling and Simulating ACC and LKA with Advanced Control Strategies

Supervisors

Prof. Stefano Alberto MALAN

Dott. Davide FAVERATO

Dott. Angelo BORNEO

Candidate

Giulia ROTILI

Academic Year 2023/2024

July 2024

Acknowledgements

I wish to extend my gratitude to MCA for providing me with this opportunity for professional growth. In particular, I would like to express my appreciation to my co-authors, Davide Faverato and Angelo Borneo, for their support, patience and assistance throughout these months.
Special thanks are due to my thesis advisor, Professor Stefano Malan, for his dedication, expertise, and unfailing availability to offer valuable guidance and feedback.

Abstract

The focus of this thesis project is to model and test autonomous driving systems, with particular emphasis on Adaptive Cruise Control (ACC) and Lane Keeping Assistant (LKA), using the MATLAB and Simulink environment. The vehicle of interest, referred to as the ego, and the vehicle in front of it, referred to as the lead, are both subjected to a dynamic model based on the single track model, which allows their dynamic behaviour during driving to be represented in an accurate and simplified manner. The simulations involve various working situations, in which the ego vehicle, based on suitable variables, activates the previously designed ACC or LKA control via an integrated control logic. This approach allows the effectiveness of such systems to be assessed in realistic scenarios and their performance improved in order to enhance the safety and efficiency of autonomous driving. Furthermore, the flexibility of the model also allows it to be adapted to the characteristics of the vehicle of interest, such as the Rosmaster X3, which has significant mechanical and dynamic behaviour compared to a conventional vehicle.

Contents

Chapter 1

State of Art

Intelligent Transport Systems (ITS) represent a key research area in the automotive and transport sector; they use advanced communication technologies and intelligent control methods to improve traffic safety and reduce the incidence of accidents [1]. One of the most significant developments concerns the integration of Advanced Driver Assistance Systems (ADAS), which integrate high and new technologies and, as a result, they use semi-autonomous or fully autonomous interventions in longitudinal and lateral directions to assist the driver.

1.1 Story

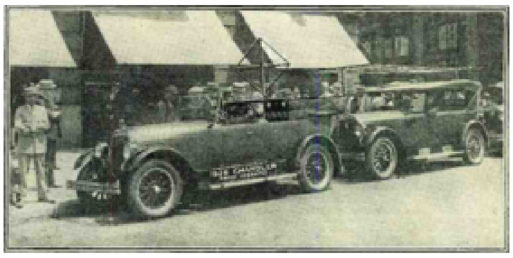
Self-driving cars have their roots in the 1920s, the first demonstration of autonomous driving took place in 1925 thanks to Houdina Radio Control. The vehicle, called the Linrrican Wonder [Fig. 1.1a], involved an antenna that was connected to the various control elements, it drove around the streets of New York but it was controlled by a remote user.
Afterwards, people started to think about the possibility of creating driverless cars and taxis with the aim of freeing American cities from the ever-increasing traffic. Studies in this field were carried out especially after the Second World War, when the General Motors presented the Firebird III, which had a cloche instead of a steering wheel and was already equipped with cruise control and sensors for autonomous driving on the road [Fig. 1.1b] .
In 1986, the tests of what we can consider the first real self-driving vehicle with human passengers on board took place in Pittsburgh. It was called NavLab and it was based on a Chevrolet van equipped with several computers that, in those days, had to be placed on board to monitor in real time the processes and operations that the vehicle performed

autonomously [Fig. 1.1c].

Since the 1990s, developments in the field of autonomous driving have become more consistent, with successful experiments carried out in Italy. In 1998, a remarkable project was ARGO starring a Lancia Thema that drove 94% of the way in complete autonomy along the famous Mille Miglia route. [Fig. 1.1d].

In the 21st century, it has been possible to reach unimaginable levels of automation thanks to the rise of technology and the development of both software and hardware such as cameras, radar, navigation systems and many others. A lot of companies are trying to capture the autonomous driving scenario, such as Tesla and Google.

One of the pioneers in the field of electric cars, Elon Musk, has developed a control system: the Autopilot. It is made up of a neural network, which simulates the behaviour of the human brain for basic logic, and it is able to process images and data collected via cameras and sensors.


(a) Linrrican Wonder

(b) Firebird III

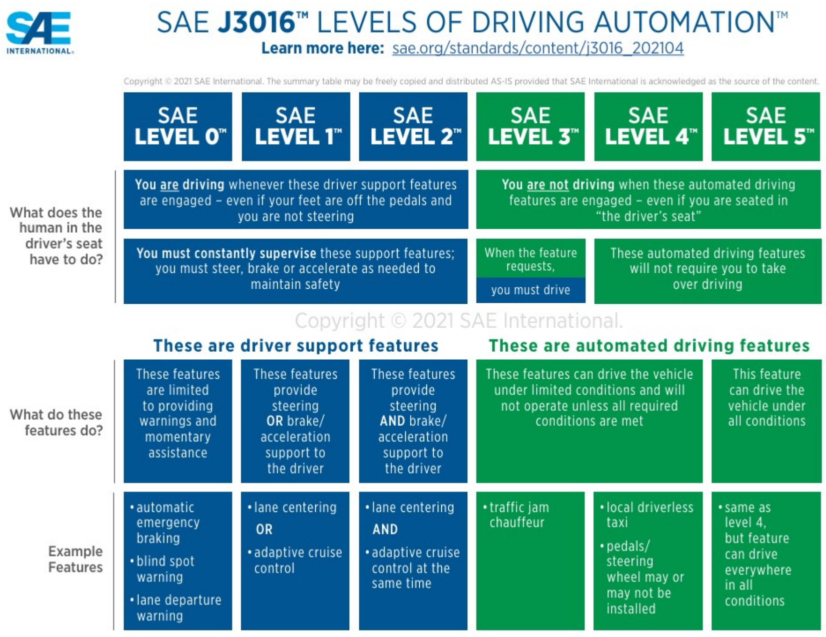
(c) NavLab

(d) ARGO

Figure 1.1: Models of autonomous driving in time

1.2 Autonomous Driving Levels

It is important to define the levels of autonomous driving according to the Society of Automotive Engineers (SAE), which is the standardisation body for the automotive industry. It ranks 6 levels for autonomous driving as shown in the figure below [Fig. 1.2] [2].


SAE INTERNATIONAL

SAE **J3016**™ LEVELS OF DRIVING AUTOMATION™

Learn more here: sae.org/standards/content/j3016_202104

	SAE LEVEL 0™	SAE LEVEL 1™	SAE LEVEL 2™	SAE LEVEL 3™	SAE LEVEL 4™	SAE LEVEL 5™
What does the human in the driver's seat have to do?	**You are driving** whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering			**You are not driving** when these automated driving features are engaged – even if you are seated in "the driver's seat"		
	You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety			When the feature requests, you must drive	These automated driving features will not require you to take over driving	
	These are driver support features			**These are automated driving features**		
What do these features do?	These features are limited to providing warnings and momentary assistance	These features provide steering **OR** brake/ acceleration support to the driver	These features provide steering **AND** brake/ acceleration support to the driver	These features can drive the vehicle under limited conditions and will not operate unless all required conditions are met		This feature can drive the vehicle under all conditions
Example Features	• automatic emergency braking • blind spot warning • lane departure warning	• lane centering **OR** • adaptive cruise control	• lane centering **AND** • adaptive cruise control at the same time	• traffic jam chauffeur	• local driverless taxi • pedals/ steering wheel may or may not be installed	• same as level 4, but feature can drive everywhere in all conditions

Figure 1.2: Automation levels [2]

- Level 0 - No Automation: The operator does the driving without any support from electronic components.

- Level 1 - Driver Assistance: In this case the driver is supported by electronic systems, such as lane centering or adaptive cruise control, the vehicle keeps a safe distance from the car ahead, but the driver takes care of other aspects of driving such as steering and braking.

- Level 2 - Partial Driving Automation: through the use of controllers, the vehicle is able to act autonomously on acceleration, braking and lane centring but the driver can take control of the vehicle at any time. The systems, in this level, are those belonging to the ADAS category. An example are Tesla's Autopilot and Cadillac Super cruise system. [3]

- Level 3 - Conditional Driving Automation: The vehicle has the needed information of the surrounding environment and it is able to manage ordinary driving conditions by deciding on acceleration, braking and direction but it is not completely unaffected by human intervention, that occurs when there are critical situations.

- Level 4 - High Driving Automation: The vehicle is able to handle any eventuality but the driver is still allowed to act, as in the case of bad weather.

- Level 5 - Full Driving Autonomous: The vehicle takes care of the path to be followed in complete autonomy and in any environmental and traffic conditions. These autonomous cars will be able to determine the best route, adapting speed, braking and direction to any situation, managing the various complexities without the need for human intervention (which will only be necessary to set the desired destination).

Currently, the level of autonomous driving allowed in Italy is level 2: Partial Driving Automation. A very important aspect to consider has to deal with the ethical sphere, and for this reason we often hear about the trolley dilemma and we ask ourselves how autonomous cars would behave if they were faced with a choice of equal criticality [Fig. 1.3].

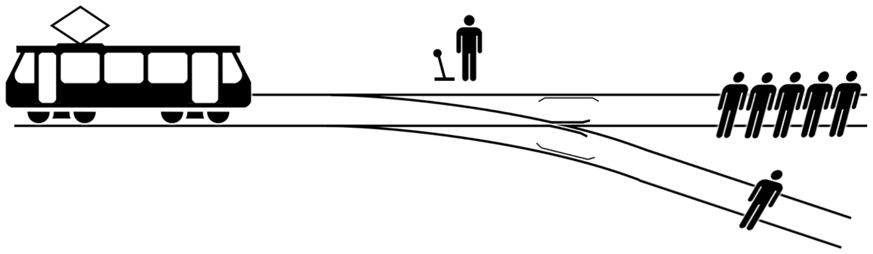


Figure 1.3: Trolley problem

1.3 Advanced Driver Assistance Systems

In order to ensure proper working and to be able to travel in complete independence, vehicles must have a number of components that can replace a driver's ability to drive in any aspect: Perception, Decision, Control and Actuation [Fig. 1.4].

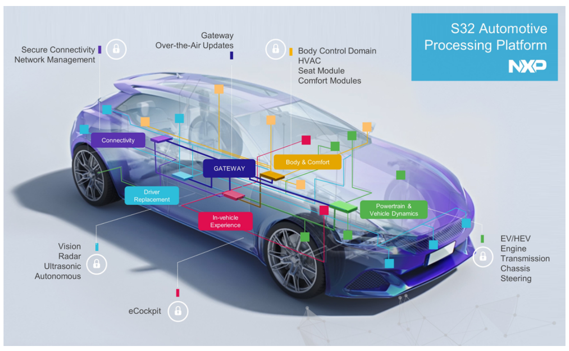


Figure 1.4: Autonomous vehicle architecture

1.3.1 Hardware components

The essential components available in Autonomous Vehicles (AVs) are:

CAMERAS: they are used in order to have a 360 view around the vehicle.

VIDEO CAMERAS: these allow us to have a real-vision of what is happening in order to help the driver during the most complex manoeuvres.

GPS: it allows us to have the position of the vehicle, more or less precise depending on the signal, so it must be combined with other sensors.

RADAR: it means Radio Detection and Ranging. By using electromagnetic waves it detects and determines the position, size and speed of surrounding objects, both fixed and mobile. Its operation exploits the wavelength, therefore it is unaffected by atmospheric conditions. [4]

LiDAR: it is the Light Detection and Ranging. Thanks to a laser pulse, it allows us to determine the reference position of objects, unlike Radar, it works in the ultraviolet field; therefore, it allows to have more information even on elements whose size is equal to that of the wavelength used, but its range of use is only a few metres. LiDAR provides 3D shapes and their information and can operate in any lighting condition [Fig. 1.5]. [5]

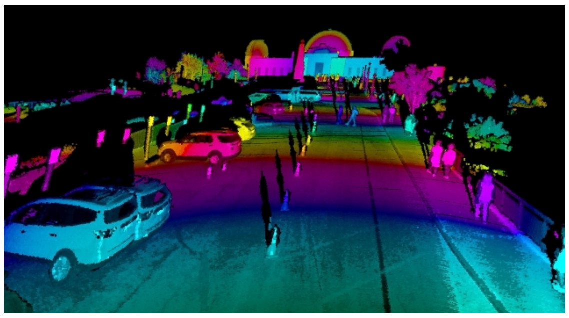


Figure 1.5: LiDAR vision

Sensors and Ultrasonics: they use high-frequency sound waves to detect objects. Their limitation is their range and accuracy; therefore, they are used as an assistance in parking manoeuvres.

1.3.2 ADAS technologies

The data obtained from the hardware components (input) are collected and processed into functions by the vehicle (output) via on-board processors. The combination of hardware and software is referred to as Advanced Driver Assistance Systems (ADAS), involving a series of technologies that are able to guarantee greater safety, both active and passive, and comfort for the driver and passengers. Some of the most innovative technologies are reported according to their area of application.

SPEED AND DISTANCE CONTROL:

Adaptive Cruise Control (ACC): it is a feature embedded in the vehicle that can automatically maintain the speed of the vehicle constantly in order to guarantee a safe distance. With this feature, pressing the gas pedal by the driver is no longer needed during driving when cruise control has been activated. The acceleration of the vehicle can be automatically regulated by cruise control when there is a change in the motion of the vehicle caused by external forces [6], [Fig. 1.6].

Forward Collision Warning (FCW): it warns a driver before the collision is going to occur between two cars. FCW system helps in preventing the collision between two cars while moving. FCW can also avoid a collision at the rear end of the car that is moving in front of our model car. While the distance between two vehicles becomes less than a predefined threshold, a warning sound is generated to notify the driver about the danger which would lead to a collision between two vehicles. This warning includes an audible signal [7].

Automatic Emergency Braking (AEB): it detects and maintains a safe distance from the collision zone in the absence of prompt action by the driver [8].

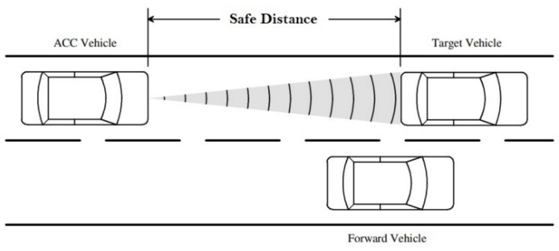


Figure 1.6: ACC application case

LANE KEEPING:

Lane Keeping Systems: The Lane Departure Warning system (LWD) warns the driver, but does not physically act on the vehicle [9]. Whereas, the most advanced ADAS system in this field is Lane Keeping Assistance (LKA); it normally estimates the position of the vehicle relative to the road by using a camera to track the road markings; it can interact with the driver, to prevent unintentional lane departure, using acoustic, visual or haptic feedback on the steering wheel [10].

DRIVING MONITORING SYSTEM:

The Driver Monitoring System (DMS) is addressing the driver's fatigue problem, which is responsible for serious number of road accidents. The driver is recorded with some camera equipment and computer vision algorithms are applied in order to detect where the driver is looking and if he/she pays attention to the road [11].

OTHER TECHNOLOGIES:

Traffic Sign Recognition (TSR): it recognizes and displays road signs (such as speed limits) on the vehicle display.

Night Vision: it uses infrared cameras to enhance night visibility, detecting pedestrians, animals, and other obstacles not visible to the naked eye.

Blind Spot Detection (BSD): it alerts the driver to vehicles in the blind spots.

Highway Assist: it combines ACC and LKA logic in order to keep the vehicle on track and at a safe speed and distance from other vehicles in situations such as highways.

V2X Communication Systems: it allows the vehicle of interest to communicate with other vehicles (V2V), sharing its information such as speed, position and direction, and with the infrastructure (V2I) in order to have continuous updates on road conditions for better traffic management.

In the following work, adaptive cruise control and lane keeping assistance will be analysed and integrated to simulate autonomous driving in different scenarios.

Chapter 2

Vehicle dynamics

In order to carry out a study of the controls that are used in AVs, it is important to study the dynamics of the vehicle. Thus, the dynamic model employed by the controllers involved will be presented in this chapter; it is based on the *single track model*, whose dynamics are already implemented in the Vechicle Dynamics blockset of Matlab [12]. It is important to emphasise that the relationships in this chapter have been derived based on the various sources. [13], [14], [15]

2.1 Model description

Vehicle motion is usually described in terms of speed: forward, longitudinal, lateral, roll, pitch and yaw in the vehicle fixed co-ordinate system. The vehicle can be divided into 2 main parts: *unsprung mass*, the wheels and the part of suspension and braking system directly connected, and the *sprung mass*, all the rest. The SAE has introduced standard co-ordinates and notations to describe the dynamics of the vehicle.
As shown below [Fig. 2.1], there are: the fixed reference system *(X,Y,Z)*, the vehicle reference system *(x,y,z)* and the angles: roll (ϑ), pitch (φ) and yaw (ψ).

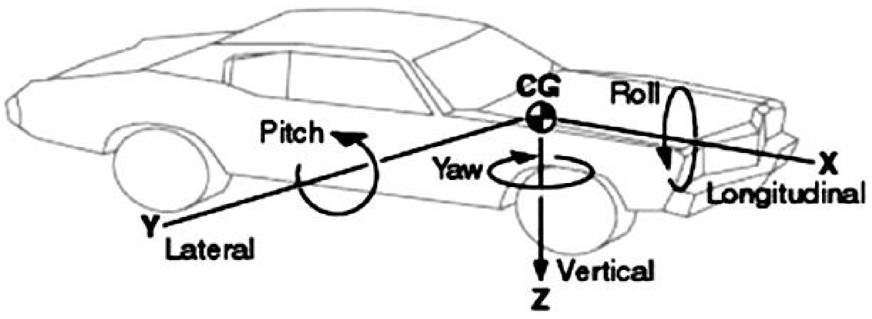


Figure 2.1: Coordinates and angles in SAE system

An accurate and simple model is essential to design an effective controller and state estimator. Many different vehicle models have been proposed but some of them are too complicate to use, because of the computational burden or the effects of the parameters change [16].
Therefore, we proceed to modelling by means of a single track model, also known as a bicycle model; this approach allows us to analyse both the lateral and longitudinal dynamics of the vehicle in a simplified manner [Fig. 2.2]. The bicycle model kinematic simplifies the vehicle to a two-wheel model, focusing on geometric relationships and motion without considering forces or mass. It is mainly used for route planning and control in low-speed scenarios where dynamic effects are minimal.

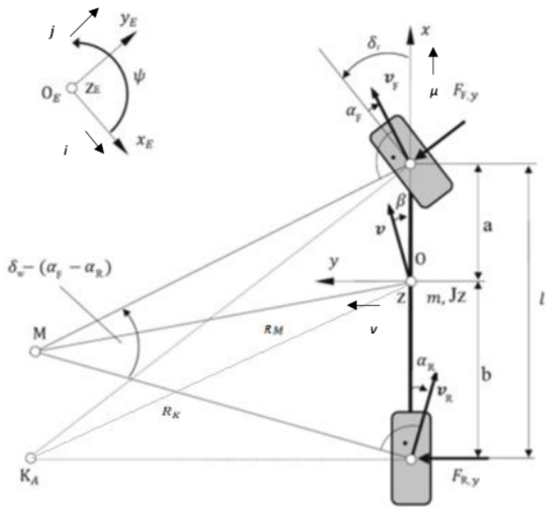


Figure 2.2: Bicycle model

For the model under analysis, a symmetry w.r.t. the longitudinal axis is assumed, so that the axles can be represented by a single virtual wheel. Furthermore, the vehicle is studied as a rigid body of mass m concentrated in O, its Centre of Gravity (CoG). The distances between the CoG and the front and rear wheels, referred to as a and b respectively, are known. The angle of rotation around the z-axis of the wheel-reference with respect to the vehicle-reference is referred to as the steering angle.In addition, a front steering angle δ_F and a similar one for the rear axle δ_R (represented as null in the figure) is introduced.
Having defined a reference system *(x, y, z; O)* attached to the chassis with its origin at the CoG, whose versors are ($\boldsymbol{\mu}$,$\boldsymbol{\nu}$,$\boldsymbol{\lambda}$). As illustrated in the figure [Fig. 2.2], the x-axis is assumed coincident with the longitudinal direction of the vehicle and directed forward, the z-axis orthogonal to the road and directed upwards, and the y-axis perpendicular to the other two and directed to the left. In addition, a fixed reference system *(X, Y, Z; O_E)* is necessary to determine the position of the vehicle in absolute terms, which is fundamental for trajectory tracking in a control logic later on.

The rotation matrix [eq. 2.1] that allows any vector to be rotated from the vehicle system *(x, y, z)* to the inertial system *(X, Y,Z)*.

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.1}$$

Where θ represents the rotation angle around the vertical axis.

Thus, based on the components of the translation velocity of the vehicle CoG in *(x,y,z)*, it is possible to calculate its velocities components w.r.t. the fixed reference system [eq. 2.2].

$$\begin{cases} \dot{X}_G = u\cos\psi - v\sin\psi \\ \dot{Y}_G = v\cos\psi + u\sin\psi \end{cases} \tag{2.2}$$

Where: u and v are the longitudinal and lateral components of the velocity w.r.t. the mobile reference system and ψ is the yaw angle.

By integrating, the position of the CoG in the system *(X, Y,Z)* are derived, allowing to compute the trajectory.
In purely kinematic conditions with K_A [Fig. 2.2], as the centre of rotation, and for a trajectory of radius R_K, it is possible to define the steering angle δ_0. In the case of R_K being much greater than the vehicle wheelbase, we will have the relationship:

$$\delta_0 \approx \frac{l}{R_K} \tag{2.3}$$

In the case of dynamic steering, side slip angles are developed by moving the centre of rotation from K_A to M. Using the Sine Theorem and geometric considerations, the following relationship is derived:

$$\delta_F - \delta_0 = \alpha_F - \alpha_R \tag{2.4}$$

Where α_F and α_R are the slip angles of the tyres, respectively front and rear.

Therefore, for very low speeds, it can be assumed that the side slip angles of the two axles are very small and similar, the dynamic steering angle coincides with the static steering angle:

$$\delta_F \approx \frac{l}{R_K} \tag{2.5}$$

In order to obtain the dynamic model equations presented in the next section, the following assumptions must be taken into account:

- Constant vehicle speed $\boldsymbol{V}$.
- Vehicle body side slip angle β and tyre side slip angle α are small enough to handle the case linearly.
- The steering angles δ are small.

2.2 Dynamic Equations

Given the above assumptions, the following dynamic equations are derived along the longitudinal ($\boldsymbol{\mu}$) and lateral ($\boldsymbol{\nu}$) directions and around the vertical axis of the vehicle through the CoG:

$$\begin{cases} ma_x = F_{F,x}\cos(\delta_F) + F_{R,x}\cos(\delta_R) - F_{F,y}\sin(\delta_F) - F_{R,y}\sin(\delta_R) \\ ma_y = F_{F,y}\cos(\delta_F) + F_{R,y}\cos(\delta_R) + F_{F,x}\sin(\delta_F) + F_{R,x}\sin(\delta_R) \\ I_z\ddot{\psi} = a[F_{F,y}\cos(\delta_F) + F_{F,x}\sin(\delta_F)] - b[F_{R,y}\cos(\delta_R) + F_{R,x}\sin(\delta_R)] \end{cases} \tag{2.6}$$

2.2.1 Inertial terms

To get the acceleration of the vehicle, we have to consider the velocity in the reference system associated with it:

$$\boldsymbol{V} = u\boldsymbol{\mu} + v\boldsymbol{\nu} \tag{2.7}$$

By deriving:

$$\frac{d\boldsymbol{V}}{dt} = \dot{u}\boldsymbol{\mu} + u\frac{d\boldsymbol{\mu}}{dt} + \dot{v}\boldsymbol{\nu} + \frac{d\boldsymbol{\nu}}{dt} = (\dot{u} - \dot{\psi}v)\boldsymbol{\mu} + (\dot{v} + \dot{\psi}u)\boldsymbol{\nu} \tag{2.8}$$

Thus, the acceleration in the reference system of the vehicle is:

$$\boldsymbol{a} = a_x\boldsymbol{\mu} + a_y\boldsymbol{\nu} = (\dot{u} - \dot{\psi}v)\boldsymbol{\mu} + (\dot{v} + \dot{\psi}u)\boldsymbol{\nu} \tag{2.9}$$

Recalling that the trajectory of the CoG is always tangent to the velocity $\boldsymbol{V}$, the acceleration can be split into centripetal a_n and tangential a_t components; so:

$$\boldsymbol{a_G} = a_t\boldsymbol{t} + a_n\boldsymbol{n} \tag{2.10}$$

Where: $\boldsymbol{t}$ and $\boldsymbol{n}$ are the versors, rispectively, parallel and orthogonal to the velocity of the centre of gravity $\boldsymbol{V}$:

$$\boldsymbol{t} = \cos(\beta)\boldsymbol{\mu} + \sin(\beta)\boldsymbol{\nu} \tag{2.11}$$

$$\boldsymbol{n} = -\sin(\beta)\boldsymbol{\mu} + \cos(\beta)\boldsymbol{\nu} \tag{2.12}$$

The acceleration is obtained considering small values of the side slip angles of the vehicle $\beta = v/u$:

$$a_t = \boldsymbol{a}_G \cdot \boldsymbol{t} = a_x \cos(\beta) + a_y \sin(\beta) = \frac{\dot{u}v + \dot{v}u}{\sqrt{u^2 + v^2}} \tag{2.13}$$

$$a_n = \boldsymbol{a}_G \cdot \boldsymbol{n} = -a_x \sin(\beta) + a_y \cos(\beta) = \frac{\dot{\psi}(u^2 + v^2) - \dot{u}v + \dot{v}u}{\sqrt{u^2 + v^2}} \tag{2.14}$$

Now, the general expression of the curvature radius R_G can be derived:

$$R_G = \frac{\boldsymbol{V}^2}{a_n} = \frac{V}{\dot{\psi} - \frac{\dot{V}\beta + \dot{\beta}V}{u}} \tag{2.15}$$

The previous relation can be approximated as follow [eq. 2.16], recalling the assumptions done:

$$R_G \approx \frac{V}{\dot{\psi} + \dot{\beta}} \tag{2.16}$$

2.3 Congruency Equations

The goal is getting the side slip angles as function of the fundamental parameters of the vehicle below :

- Steering angle: δ.
- vehicle slip angle: β.
- Yaw rate: $\dot{\psi}$.
- Velocity components: u and v.

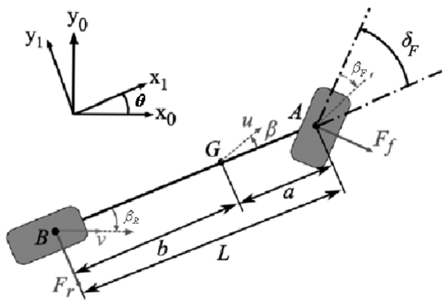


Figure 2.3: Bicycle model: CoG and Front Wheel relationship

In order to obtain what needed, it is necessary to evaluate the speed at which the centres of the two wheels move; so, we recall the fundamental equation of the kinematics of rigid bodies [eq. 2.17]:

$$\boldsymbol{V}_i = \boldsymbol{V}_G + \dot{\boldsymbol{\psi}} \times \boldsymbol{r}_i \tag{2.17}$$

where, i =F,R (front and rear).

The above equation allows to compute the absolute velocities of the two wheels, $\boldsymbol{V}_F$ and $\boldsymbol{V}_R$, considering the absolute velocity of the CoG,$\boldsymbol{V}_G$, the yaw rate, $\dot{\psi}$, and the respective radii,$\boldsymbol{r}_F$ and $\boldsymbol{r}_R$ [2.18 and 2.19], joining the centres of wheels to the CoG.

$$\boldsymbol{r}_F = \begin{bmatrix} a \\ 0 \end{bmatrix} \tag{2.18}$$

$$\boldsymbol{r}_R = \begin{bmatrix} -b \\ 0 \end{bmatrix} \tag{2.19}$$

Thus, the following velocity components for each wheel are obtained:

$$\begin{cases} u_F = u \\ v_F = v + a\dot{\psi} \\ u_R = u \\ v_R = v - b\dot{\psi} \end{cases} \tag{2.20}$$

It is possible to compute the moduli of the velocities and angles β_i:

$$\begin{cases} |\boldsymbol{V}_F| = \sqrt{(u_F)^2 + (v_F)^2} \\ |\boldsymbol{V}_R| = \sqrt{(u_R)^2 + (v_R)^2} \end{cases} \tag{2.21}$$

$$\begin{cases} \beta_F = \arctan \frac{v_F}{u_F} = \arctan \frac{v + a\dot{\psi}}{u} \\ \\ \beta_R = \arctan \frac{v_R}{u_R} = \arctan \frac{v - b\dot{\psi}}{u} \end{cases} \tag{2.22}$$

The slip angles α_i can be obtained as follow and they can be simplified since β_i are considered small:

$$\begin{cases} \alpha_F = \delta_F - \beta_F = \delta_F - \beta - \frac{a}{u}\dot{\psi} \\ \\ \alpha_R = \delta_R - \beta_R = \delta_R - \beta + \frac{b}{u}\dot{\psi} \end{cases} \tag{2.23}$$

2.4 Single-Track Model

Replacing the equation 2.9 in the equation 2.6 and basing on the assumptions discussed in the previous sections, we get:

$$\begin{cases} m(\dot{u} - \dot{\psi}v) = F_{F,x} + F_{R,x} - F_{F,y}\delta_F - F_{R,y}\delta_R \\ m(\dot{v} - \dot{\psi}u) = F_{F,y} + F_{R,y} - F_{F,x}\delta_F + F_{R,x}\delta_R \\ I_z\ddot{\psi} = a[F_{F,y} + F_{F,x}\delta_F] - b[F_{R,y} + F_{R,x}\delta_R] \end{cases} \tag{2.24}$$

To linearise the model, it is possible to use an approximation for the lateral forces:

$$F_{i,y} = C_i\alpha_i \tag{2.25}$$

where C is the cornering stiffness of the i-th axle.

Recalling the lateral equilibrium and the rotational one around the vertical axis, for very small β velues: $\cos(\beta) \approx 1$ and $\sin(\beta) \approx \beta$.
So:

$$\begin{cases} u = V\cos(\beta) \approx V \\ v = V\sin(\beta) \approx V\beta \end{cases} \tag{2.26}$$

For V constant, the derivative of the lateral component is $\dot{v} = V\dot{\beta}$.
Rewriting the previous system and considering the relationships obtained, we get:

$$\begin{cases} m(V\dot{\beta} - \dot{\psi}V) = -(C_F + C_R)\beta - (\frac{C_F a - C_R b}{V})\dot{\psi} + C_F\delta_F + C_R\delta_R \\ \\ I_z\ddot{\psi} = (-C_F a + C_R b)\beta - (\frac{C_F a^2 + C_R b^2}{V})\dot{\psi} + aC_F\delta_F + bC_R\delta_R \end{cases} \tag{2.27}$$

where V is the longitudinal component of the velocity.

The obtained equation can be rearranged as a set of differential equations of the first order and it can be rewritten in a state-space format:

$$\begin{cases} \dot{\boldsymbol{x}}(\boldsymbol{t}) = \boldsymbol{A}\boldsymbol{x}(\boldsymbol{t}) + \boldsymbol{B}\boldsymbol{u}(\boldsymbol{t}) \\ \boldsymbol{y}(\boldsymbol{t}) = \boldsymbol{C}\boldsymbol{x}(\boldsymbol{t}) + \boldsymbol{D}\boldsymbol{u}(\boldsymbol{t}) \end{cases} \tag{2.28}$$

where: the state vector,$\boldsymbol{x}$ and the input vector,$\boldsymbol{u}(\boldsymbol{t})$ are defined as follow:

$$\boldsymbol{x}(\boldsymbol{t}) = \begin{bmatrix} \beta \\ \dot{\psi} \end{bmatrix} \tag{2.29}$$

$$\boldsymbol{u}(\boldsymbol{t}) = \begin{bmatrix} \delta_F \\ \delta_R \end{bmatrix} \tag{2.30}$$

The output vector $\boldsymbol{y}$ is:

$$\boldsymbol{y}(\boldsymbol{t}) = \begin{bmatrix} \beta \\ \dot{\psi} \\ \rho \\ \alpha_F \\ \alpha_R \end{bmatrix} \tag{2.31}$$

ρ is the curvature of the road: $\frac{1}{R}$.

Rearranging the equations, the matrices are:

$$\boldsymbol{A} = \begin{bmatrix} -\frac{C_F+C_R}{m} & -\frac{aC_F-bC_R}{mV} - V \\ -\frac{aC_F-bC_R}{I_z} & -\frac{a^2C_F+b^2C_R}{I_Z V} \end{bmatrix} \tag{2.32}$$

$$\boldsymbol{B} = \begin{bmatrix} \frac{C_F}{m} & \frac{C_R}{m} \\ \frac{aC_F}{I_Z} & \frac{bC_R}{I_z} \end{bmatrix} \tag{2.33}$$

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -\frac{C_F+C_R}{mV^2} & \frac{-aC_F+bC_R}{mV^3} \\ -1 & -\frac{a}{V} \\ -1 & \frac{b}{V} \end{bmatrix} \tag{2.34}$$

$$\boldsymbol{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{C_F}{mV^2} & \frac{C_R}{mV^2} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.35}$$

From the State-Space equations, the stability of the system can be analysed by means of the calculation of eigenvalues, which will not be carried out in this thesis project.
However, it is important to point out that in the literature this problem is present as an augmented model and it is used for the linear control strategy [17].

2.5 MATLAB and Simulink model

MATLAB and Simulink are the development environments for modelling the bicycle model previously explained; in Simulink there is a block, shown in the figure 2.4, that allows us to obtain the information necessary for vehicle dynamics.

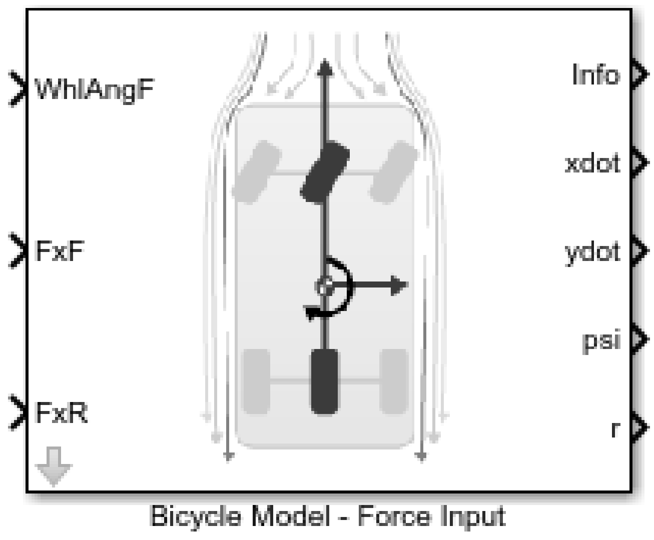


Figure 2.4: Simulink block of bicycle model [12]

The inputs of the Simulink block are the steering angle and the forces, which are equally divided between the two wheels.
The outputs of the plant are:

- The position and the velocity of the CoG in the fixed-frame $X, \dot{X}, Y, \dot{Y}$.
- The velocity components in the mobile-frame $\dot{x}, \dot{y}$.
- The yaw angle ψ.
- The yaw rate r.

Chapter 3

MATLAB and Simulink

This Chapter will focus on the development of the models for the two technologies analysed in MATLAB and Simulink enviroments.
The set up of the vehicle and its controllers were done customising the libraries and blocks on MathWorks. [12]

3.1 Adaptive Cruise Control set up

For the development of ACC, whose purpose is to maintain a safe distance from the vehicle in front and not to exceed a cruising speed, three macro-blocks were used as represented by the figure below [Fig. 3.1].

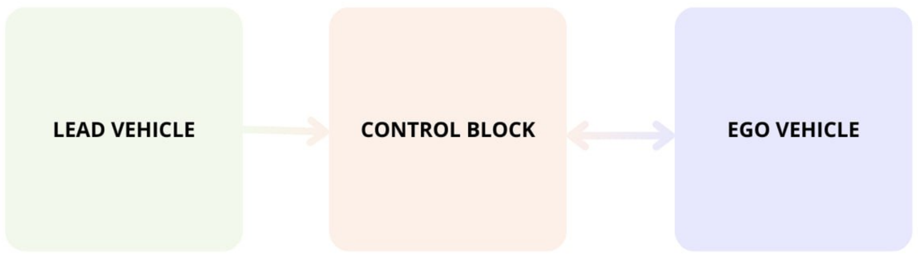


Figure 3.1: Schematic block for Simulink set up

Starting from the left [Fig. 3.1], we have:

- *Lead vehicle*: it represents the car ahead to the ego car. It is modelled using a single track model and it is used to give all the information to controller simulating the sensors, such as position and velocity.
- *Control block*: it acts on the ego car acceleration in order to keep a safe distance between the two vehicles and and to allow the ego car to reach a previously established longitudinal velocity.
- *Ego vehicle*: it is modelled as a single track, whose acceleration is directly regulated by the controller thanks to a feedback signals given by its longitudinal velocity.

3.1.1 ACC: Lead vehicle

The lead vehicle is modelled using *bicycle model* block of the Automated Driving Toolbox in Mathworks [12].

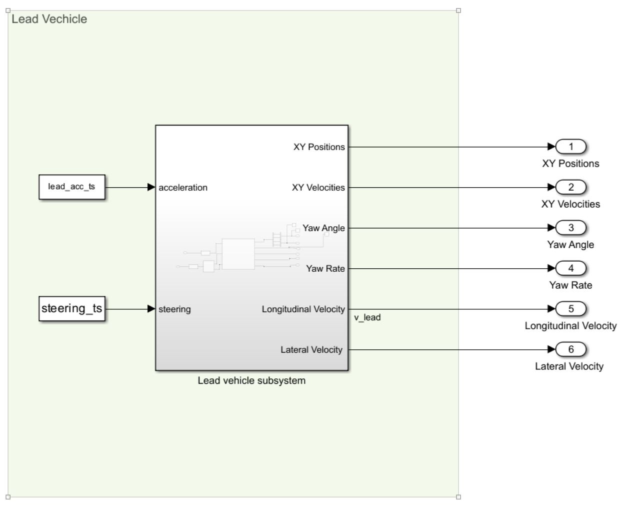


Figure 3.2: Lead Vehicle Subsystem

As shown in Figure 3.2, the inputs, computed within the MATLAB environment, are:

1. the steering angle $[rad]$;
2. the acceleration $[m/s^2]$;

The acceleration values have a sinusoidal behaviour with an amplitude equal to 0.6 m/s^2, chosen in order to have a smoother behaviour, while the steering angle equation is written as function of the trajectory.
The steering angle and the acceleration must have a *timeseries* format for the Simulink block, so they vary in function of the time. The total time employed within the simulation is 180 s and a step time equal to 2 s.
The outputs of lead vehicle block are:

- the position;
- the velocity components obtained mathematically;
- the longitudinal and lateral velocity considering the model and its characteristics;
- the yaw rate;
- the yaw angle.

The information is in the fixed reference frame.
The *bicycle block* is implemented considering the following parameters [Tab. 3.1]:

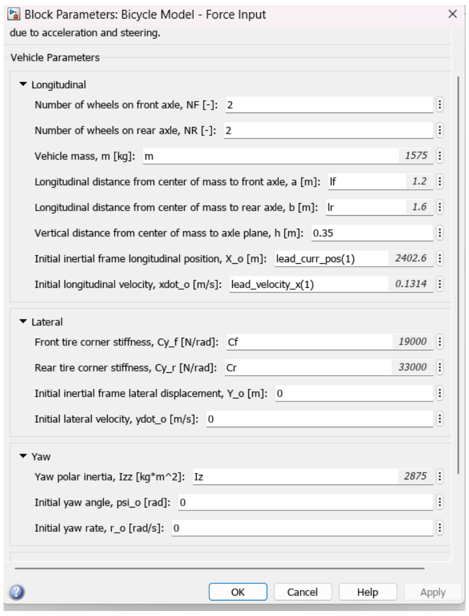
Parameter	**Value**	**Description**	**SI**
m	1575	Vehicle mass	kg
I_z	2875	Yaw polar inertia	m^2 kg
l_f	1.2	Longitudinal distance from center of mass to front axle	m
l_r	1.6	Longitudinal distance from center of mass to rear axle	m
C_f	19000	Front tire cornering stiffness	N/rad
C_r	33000	Rear tire cornering stiffness	N/rad
τ	0.5	Longitudinal time constant	N/A

Table 3.1: Vehicle parameters

These values are used for the ego car model analysed in the subsection 3.1.2.

In addition, further information regarding the initial position and the initial velocity, for each of the two components, the initial yaw angle and yaw rate, are set in the Simulink block [Fig. 4.1].

The Simulink block takes into account the parameters about the aerodynamic of the vehicle and the environment as shown in the figure below 3.4.



Block Parameters: Bicycle Model - Force Input

due to acceleration and steering.

Vehicle Parameters

Longitudinal

Number of wheels on front axle, NF [-]: 2

Number of wheels on rear axle, NR [-]: 2

Vehicle mass, m [kg]: m *1575*

Longitudinal distance from center of mass to front axle, a [m]: lf *1.2*

Longitudinal distance from center of mass to rear axle, b [m]: lr *1.6*

Vertical distance from center of mass to axle plane, h [m]: 0.35

Initial inertial frame longitudinal position, X_o [m]: lead_curr_pos(1) *2402.6*

Initial longitudinal velocity, xdot_o [m/s]: lead_velocity_x(1) *0.1314*

Lateral

Front tire corner stiffness, Cy_f [N/rad]: Cf *19000*

Rear tire corner stiffness, Cy_r [N/rad]: Cr *33000*

Initial inertial frame lateral displacement, Y_o [m]: 0

Initial lateral velocity, ydot_o [m/s]: 0

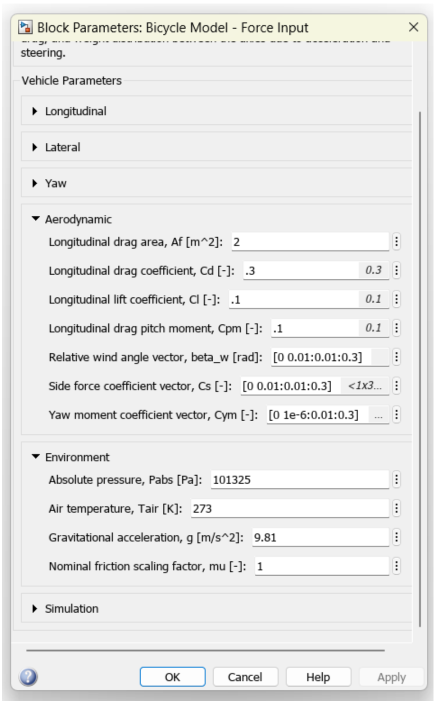
Yaw

Yaw polar inertia, Izz [kg*m^2]: Iz *2875*

Initial yaw angle, psi_o [rad]: 0

Initial yaw rate, r_o [rad/s]: 0

Figure 3.3: Set up parameters



Block Parameters: Bicycle Model - Force Input

steering.

Vehicle Parameters

Longitudinal

Lateral

Yaw

Aerodynamic

Longitudinal drag area, Af [m^2]: 2

Longitudinal drag coefficient, Cd [-]: .3 *0.3*

Longitudinal lift coefficient, Cl [-]: .1 *0.1*

Longitudinal drag pitch moment, Cpm [-]: .1 *0.1*

Relative wind angle vector, beta_w [rad]: [0 0.01:0.01:0.3]

Side force coefficient vector, Cs [-]: [0 0.01:0.01:0.3] *<1x3...*

Yaw moment coefficient vector, Cym [-]: [0 1e-6:0.01:0.3] *...*

Environment

Absolute pressure, Pabs [Pa]: 101325

Air temperature, Tair [K]: 273

Gravitational acceleration, g [m/s^2]: 9.81

Nominal friction scaling factor, mu [-]: 1

Simulation

Figure 3.4: Aerodynamics and Environment information for the bicycle model block

The inputs of the *bicycle block* [Fig. 2.4] are the steering angle, described in chapter 2, and the front and rear forces. Those forces are obtained manipulating the acceleration vector. Indeed, the variable *lead_acc_ts* is used to get the two forces, through the use of a transfer function, multiplied by the mass m and equally divided on the two axles [Fig. 3.5].

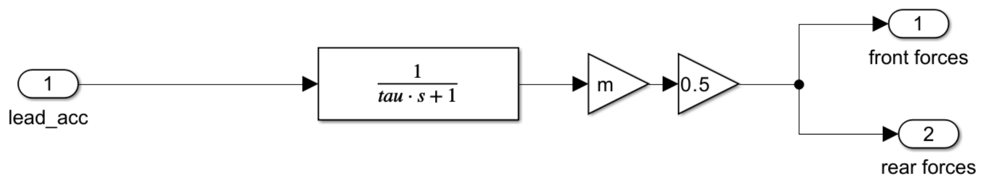


Figure 3.5: How the forces are implemented within the Simulink environment

3.1.2 ACC: Ego vehicle

The ego vehicle is the one of our interest [Fig. 3.6].

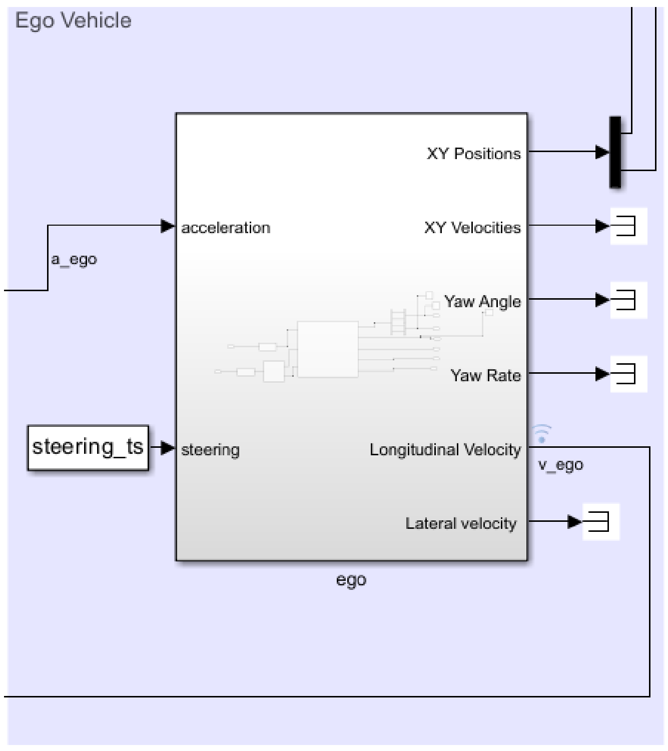


Figure 3.6: Ego Vehicle Subsystem

The inputs of the subsystem are the steering angle, implemented as explained in 3.1.1, and the acceleration, shown as *a_ego* in Fig. 3.6, is computed by the controllers, in this way the value changes considering the safe distance and the set velocity.
The outputs are the ones of the *Lead Vehicle Subsystem*, with the longitudinal velocity serving as a feedback signal for the controller block, which will be analysed in the next section.

In the *Ego Vehicle Subsystem*, the *bicycle model* is consistently used. Its parameters, listed in Table 3.1 and illustrated in Figure 3.4, represent the most common vehicle parameters in use today.

3.1.3 ACC: Control block

The control block aims to adjust the behaviour of ego car, considering the information obtained from the sensors such as the relative distance and the velocity of the lead vehicle in order to act on the acceleration of ego car.

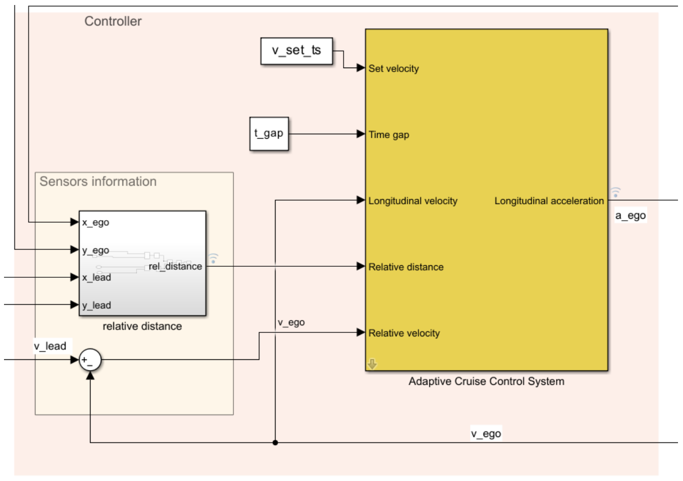


Figure 3.7: Controller block with MPC controller

Fig. 3.7 shows the two main blocks:

- Sensors Information block;
- Adaptive Cruise Control System block.

The first block simulates the sensor data that the ego car would typically gather. In our case, this information is generated by simulating the behavior of the lead vehicle using MATLAB code and a Simulink block. This allows us to calculate the relative distance and velocity between the two vehicles. The relative distance block also accounts for a possible curved trajectory when determining the distance.

The second main block [Fig. 3.7] is the core of the ACC. Its inputs are:

- *Set Velocity*: it represents the maximum speed of the ego vehicle and it cannot be exceed. It is implemented as a *timeseries*;
- *Time gap*: period of time the ego vehicle waits before leaving after the start of the simulation;
- *Longitudinal Velocity*:The feedback signal is obtained from the ego vehicle model;
- *Relative Distance*: is the difference between the positions of the lead and ego vehicle;
- *Relative Velocity*: is the difference between the velocities of the lead and ego vehicle.

The adaptive cruise controller offers two variants: a classical design (default) and an MPC-based design. Both variants adhere to the same fundamental principles. The ACC vehicle (ego vehicle) estimates the relative distance and velocity to the lead car. The ACC ensures that the ego vehicle travels at a driver-set velocity while maintaining a safe distance from the lead car. The first method of controlling the acceleration of the ego vehicle is implemented as follows:

1. Calculation of the safety distance $D_{\text{safe}} = D_{\text{default}} + \text{Time\_gap} \times V_{\text{ego}}$;
2. Calculation of the distance error $x_{\text{error}} = D_{\text{safe}} - D_{\text{relative}}$.

The latter serves as a discriminating factor in the evaluation of acceleration: if $x_{\text{error}} > 0$, the acceleration increases up to a maximum of $3\,\text{m/s}^2$ to reach V_{set} under safe conditions. If $x_{\text{error}} < 0$, the vehicle decelerates to achieve the minimum D_{safe}.These design principles are achieved through the Min and Switch blocks.

The second type of controller is a Model Predictive Controller (MPC). MPC is a discrete-time, multi-variable control architecture that operates as follows: at each control interval, the MPC uses an internal model to predict future behavior of the system. Based on these predictions, the controller computes the optimal control actions to be taken. The logic behind the Model Predictive Controller (MPC) is as follows:

Model predictive control solves an optimization problem, specifically, a Quadratic Program (QP), at each control interval. The solution determines the Manipulated Variables (MVs) to be used in the plant until the next control interval.

This QP problem includes the following features:

- **Objective (cost) function**: A scalar, nonnegative measure of controller performance to be minimized.
- **Constraints**: Conditions the solution must satisfy, such as physical bounds on MVs and plant output variables.
- **Decision**: The MV adjustments that minimize the cost function while satisfying the constraints.

In our case, the MPC is implemented using MathWorks tools [12]. The underlying optimization problem is formulated to track the driver-set velocity while adhering to a constraint. This constraint ensures that the relative distance is always greater than the safe distance.

The Model Predictive Controller (MPC) for Adaptive Cruise Control (ACC) is designed to ensure safe and efficient vehicle operation by enforcing a set of constraints. These constraints can be formulated as follows:

$$\begin{aligned} &\min_{u} \quad |V_{\text{ego}} - V_{\text{set}}|^2 \\ &\text{subject to} \quad D_{\text{safe}} - D_{\text{relative}} > 0 \\ &-3 \leq u \leq 2 \end{aligned} \tag{3.1}$$

Where u is the ego acceleration. To configure the Adaptive Cruise Control System block, the linear model for ACC design, G, is used. This model, G, is derived from the vehicle dynamics. Summarizing the analysis conducted in Chapter 2, the matrices derived from the study of vehicle dynamics are as follows:

$$A = \begin{bmatrix} -\frac{2C_f+2C_r}{mv_{0,\text{ego}}} & 0 & -v_{0,\text{ego}} - \frac{2C_f l_f - 2C_r l_r}{mv_{0,\text{ego}}} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ -\frac{2C_f l_f - 2C_r l_r}{I_z v_{0,\text{ego}}} & 0 & -\frac{2C_f l_f^2 + 2C_r l_r^2}{I_z v_{0,\text{ego}}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -\frac{1}{\tau} \end{bmatrix} \tag{3.2}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{\tau} \end{bmatrix} \tag{3.3}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.4}$$

These matrices enable us to define the Transfer Function (TF) of the ego vehicle plant in the MATLAB environment.

$$G(s) = \frac{2}{s^2 + 2s} \tag{3.5}$$

It is computed using the MATLAB command `ss2tf`.

For both controllers used, the inputs are:

- *velocity set*
- T_{gap}
- *Velocity*
- *relative distance*
- *relative velocity*

The output of the controller block is the acceleration, which is adjusted based on the speed information provided by the ego vehicle block.

3.2 Lane Keeping Assistant set up

The LKA (Lane Keeping Assist) system ensures that the ego vehicle travels along the centerline of the road lanes by adjusting its front steering angle. The goal of lane keeping control is to minimize both the lateral deviation and the relative yaw angle, keeping them as close to zero as possible.

Unlike Adaptive Cruise Control, the presence of a lead vehicle block is unnecessary for Lane Keeping Assist (LKA) since its objective is to maintain the lane regardless of the vehicles ahead of the ego vehicle; this means that only the control block and the ego vehicle block of the figure 3.1 are present.

As far as the control of lateral dynamics is concerned, the ego vehicle model was implemented as previously described in section 3.1.2. In this case, the ego parameters involved in controlling lane keeping are:

1. lateral velocity;
2. yaw angle;
3. steering angle;
4. longitudinal velocity.

3.2.1 LKA: Control block

There are two crucial components in LKA control, see Figure 3.8: Sensors Dynamics and MPC controller.

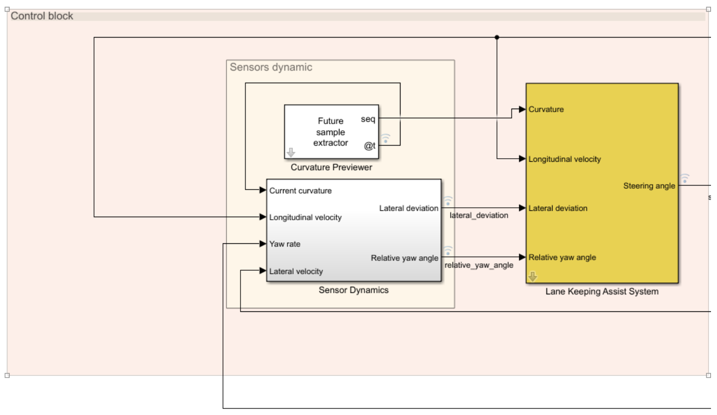


Figure 3.8: LKA control block [12]

Sensors Dynamics: The sensors dynamic block enables the simulation of a curved trajectory and extrapolates information typically acquired from vehicle sensors. This simulation provides essential input data required for subsequent control actions. Within this block the function `getCurvature` [12] is designed to calculate the curvature of a desired trajectory for the Lane Keeping Assist (LKA) system. Given the longitudinal velocity (`v0_ego`) and a time vector (`time`), the function performs the following steps:

1. Computes the desired X position (`Xref`) as a function of the longitudinal velocity and time.
2. Determines the desired Y position (`Yref`) using a piecewise hyperbolic tangent function, which simulates a realistic trajectory with lane changes or curved paths.
3. Calculates the curvature of the desired trajectory using the gradients of `Xref` and `Yref`. This involves computing the first gradient (`DX`) and second gradient (`D2Y`) of the X and Y positions.
4. Stores the computed curvature values in a structure `md`, which includes the time vector and curvature values. This structure serves as the input for the LKA system.

The purpose of this function is to provide a smooth and realistic curvature profile, ensuring that the vehicle follows a safe and predictable path.

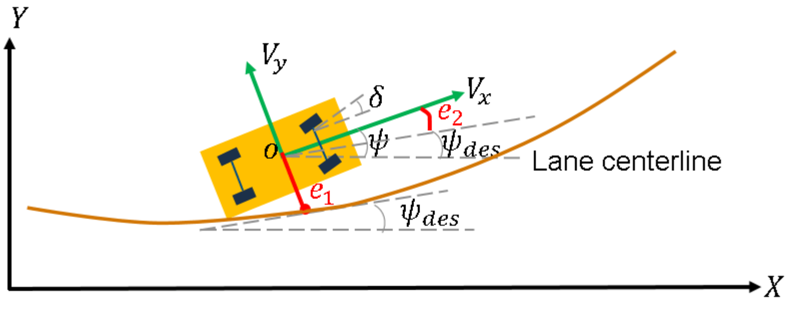


Figure 3.9: It is a graphical representation of the control logic employed in the LKA (Lane Keeping Assist) system. [12]

The second part of the sensors dynamic block aims to minimize the lateral deviation and relative yaw angle. To achieve this, the errors, defined as e_1 and e_2, as illustrated in Figure 3.9, are calculated as follow[1]:

[1]all this information is gathered on MathWorks

$$\dot{e}_1 = V_x e_2 + V_y$$
$$\dot{e}_2 = \dot{\psi} - V_x \rho$$

where:

- ρ is the curvature;
- $\dot{\psi}$ is the yaw rate;
- V_y is the lateral velocity;
- V_x is the longitudinal velocity;

MPC Controller: The MPC (Model Predictive Controller) utilizes the information obtained from the sensors dynamic block to regulate the lateral dynamics of the vehicle. It processes the simulated sensor data to make real-time decisions, ensuring the vehicle maintains its desired trajectory within a lane.

The Adaptive Model Predictive Control (AMPC) for Lane Keeping Assist (LKA) system is designed to enhance vehicle safety by maintaining the vehicle within its lane. The adaptive nature of this controller allows it to dynamically adjust to varying driving conditions and uncertainties, ensuring robust performance. Below are the main features of this advanced control system.

The Adaptive MPC continuously updates the vehicle model in real-time based on current driving conditions. This feature ensures that the control actions are always based on the most accurate representation of the vehicle dynamics, leading to improved performance and safety.

The controller is designed to handle various constraints effectively. These include:

- Steering angle limitations
- Road boundaries
- Vehicle dynamic constraints

By managing these constraints, the system ensures safe and feasible control actions under all circumstances. Adaptive MPC optimizes future control actions over a prediction horizon. This predictive capability allows the controller to anticipate and counteract potential deviations from the lane, providing smooth and stable lane-keeping performance. The adaptive nature of the controller enhances its robustness against model uncertainties and external disturbances. By continuously adjusting to the current state of the vehicle and its environment, the controller maintains high performance even in the presence of unexpected changes. Key components include:

- A state-space vehicle model capturing essential lateral dynamics.
- An MPC controller designed using MATLAB toolbox [18].
- An adaptive mechanism to update model parameters dynamically.
- Integration into Simulink for real-time simulation and validation.

The Adaptive MPC for LKA system combines the strengths of predictive control with the flexibility of adaptive mechanisms, resulting in a robust, efficient, and reliable solution for lane-keeping assist.

Chapter 4

Simulation results

This section reports the results obtained from simulations performed in the MATLAB/Simulink environment.

4.1 ACC: linear trajectory

Considering a linear trajectory [Figure 4.1] developing along the X-axis, with respect to the fixed reference system, as well as the longitudinal direction of the analyzed vehicle model, simulations were carried out for each controller: a classical controller and an MPC controller.

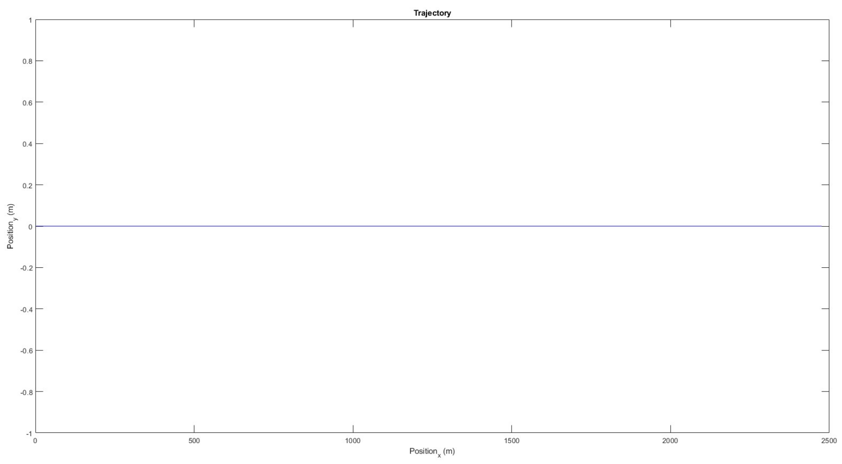


Figure 4.1: Vehicle trajectory

4.1.1 Classical Controller

Figure 4.2 presents the data characterizing the adaptive cruise control system, including:

- Relative distance
- Longitudinal speed of the ego vehicle
- Longitudinal acceleration of the ego vehicle

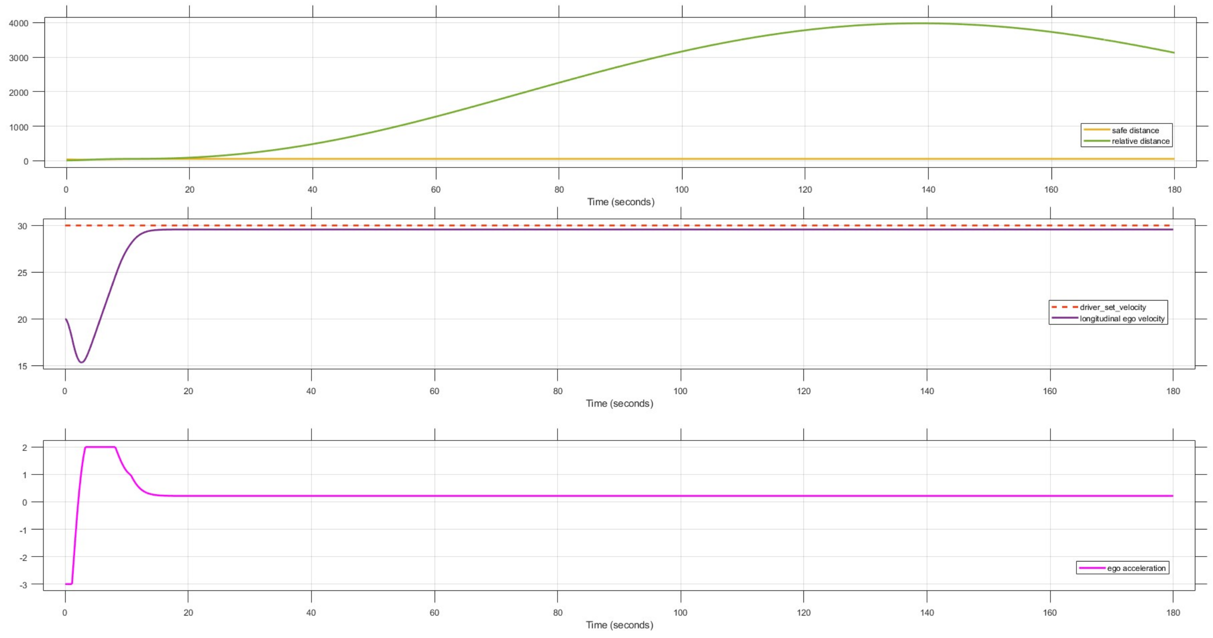


Figure 4.2: Classical controller simulations results

In all three graphs, the x-axis represents the simulation time, while the y-axis shows the following variables (in the order of the graphs):

- Distance in meters (m);
- Speed in meters per second (m/s);
- Acceleration in meters per second squared (m/s^2).

In order to better analyse the behaviour of the controller and the vehicle under test, it is essential to study the first few seconds of the simulation. As shown in Figure 4.3, the behaviour complies with the desired outcomes towards the end of the simulation, which are:

- A relative distance greater than the safety distance
- Reaching a cruising speed, which in our case was set at 30 m/s

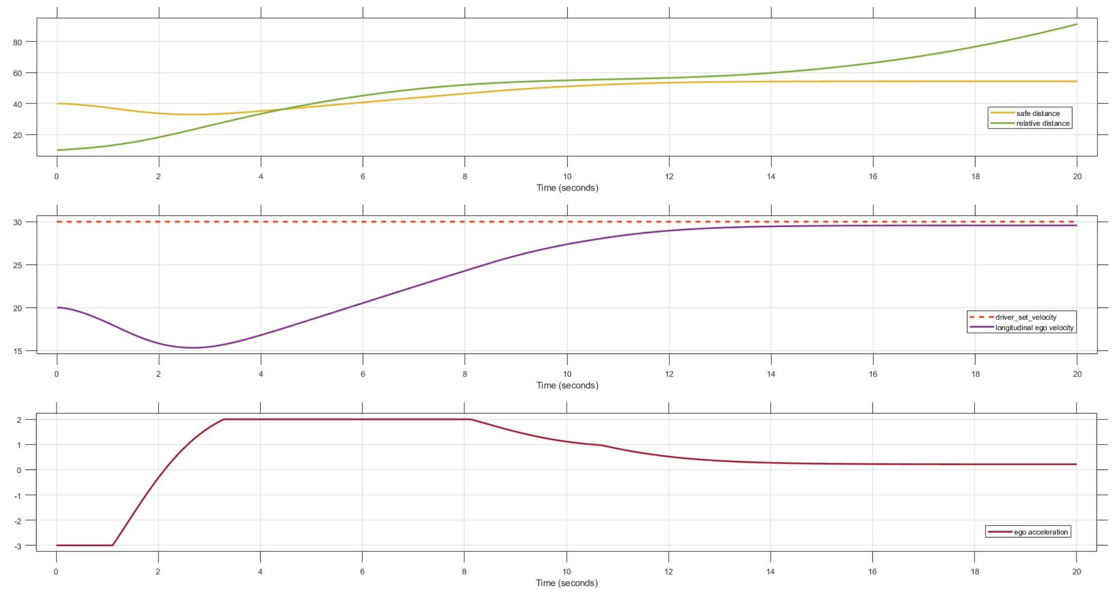


Figure 4.3: Analysis of the first 20 seconds of the simulation

Initial conditions are as follows: the lead vehicle starts from position $x = 10$ m with a longitudinal velocity of 0 m/s, while the ego vehicle starts at position $x = 0$ m with a time gap of 1.5 s and a longitudinal velocity of 20 m/s. Therefore, once the simulation begins, we observe a relative distance between the vehicles of approximately 10 m, which is less than the safety distance. The safety distance is calculated based on the longitudinal speed of the ego vehicle, which can vary moment by moment due to the acceleration output from the controller. Therefore, the controller immediately acts on the acceleration, reaching a maximum deceleration value of -3 m/s^2, chosen based on the achievable acceleration values of typical passenger vehicles. Consequently, there is a decrease in speed, reaching a minimum of 15.5 m/s, while the relative distance increases.

There is a slight delay between the deceleration and the actual increase in relative distance, as well as the change in speed, which is used as feedback to the controller for calculating the safe distance. Increasing the relative distance triggers an acceleration impulse, reaching a value of 2 m/s^2, resulting in an increase in the ego vehicle speed until it reaches the set speed.

Finally, once the safe distance is surpassed, the deceleration gradually decreases to a value of 0.2 m/s^2, which allows maintaining the cruising speed. The ACC applies a slight acceleration to compensate for forces that would naturally slow the vehicle, such as air resistance or wheel rolling resistance. In this case, the acceleration does not increase the speed, but keeps the speed constant by counteracting these forces.

4.1.2 MPC Controller

While maintaining the initial conditions, the simulation was conducted employing an MPC controller.

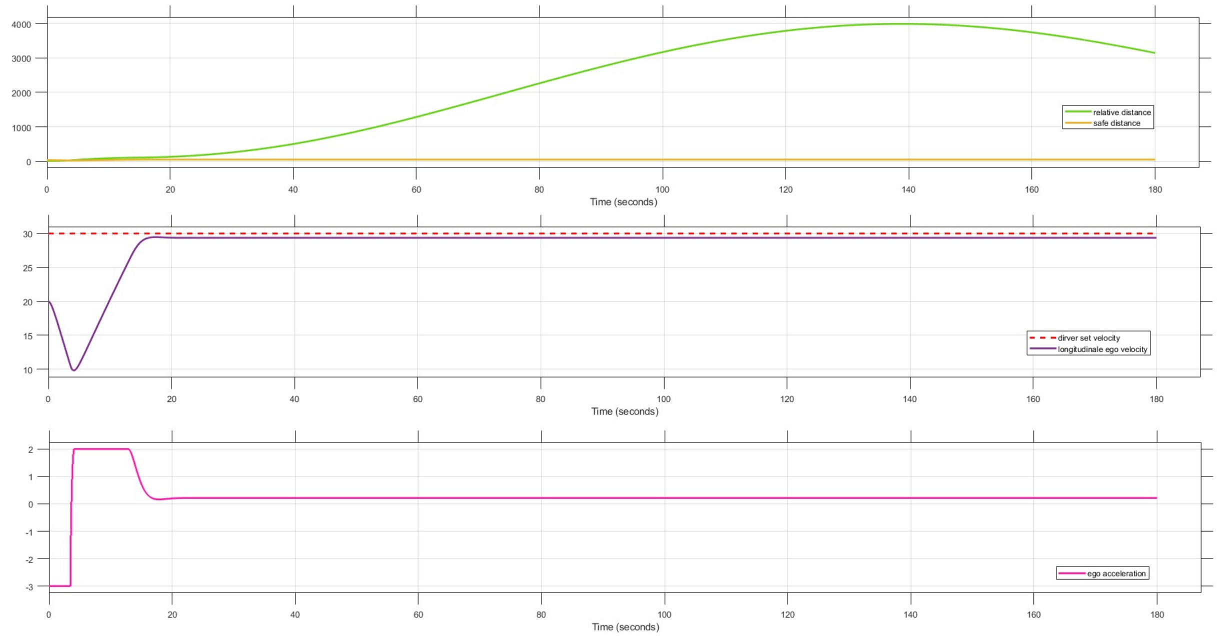


Figure 4.4: MPC controller simulations results

In all three graphs [Fig. 4.4], the x-axis represents the simulation time, while the y-axis shows the following variables (in the order of the graphs):

- Distance in meters (m);
- Speed in meters per second (m/s);
- Acceleration in meters per second squared (m/s^2).

To enhance the analysis of the components under investigation, it is advantageous to focus on the events occurring within the initial 20 seconds of the simulation.

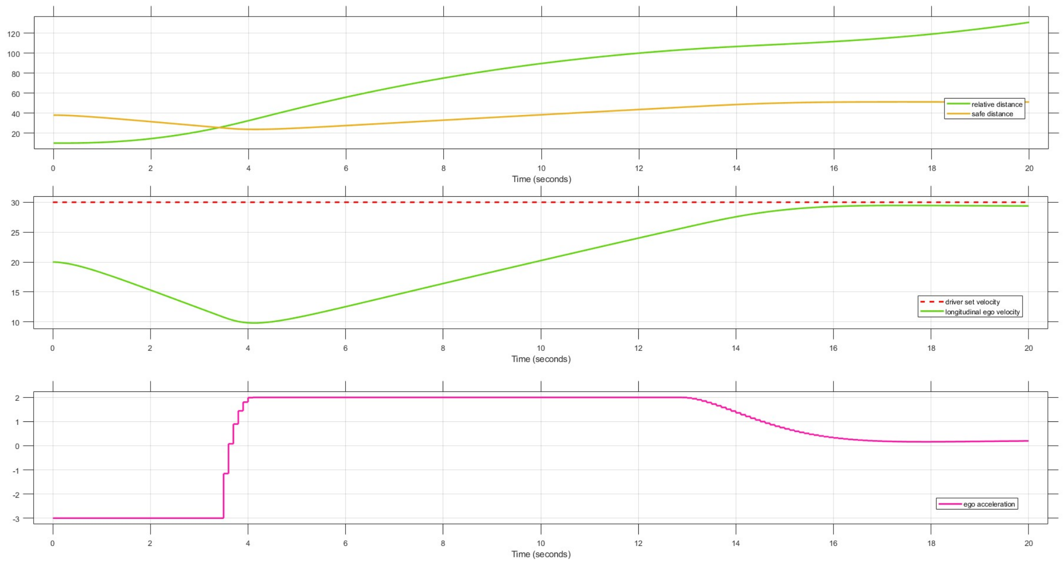


Figure 4.5: Analysis of the first 20 seconds of the simulation

Therefore, the controller immediately acts on the acceleration, reaching a maximum deceleration value of -3 m/s^2, chosen based on the achievable acceleration values of typical passenger vehicles. Consequently, there is a decrease in speed, reaching a minimum of 9.8 m/s, while the relative distance increases.

There is a slight delay between the deceleration and the actual increase in relative distance, as well as the change in speed, which is used as feedback to the controller for calculating the safe distance. Increasing the relative distance triggers an acceleration impulse, reaching a value of 2 m/s^2, resulting in an increase in the ego vehicle speed until it reaches the set speed.

Finally, once the safe distance is surpassed, the deceleration gradually decreases to a value of 0.2 m/s^2, which allows maintaining the cruising speed.

4.1.3 Comparison of two controllers

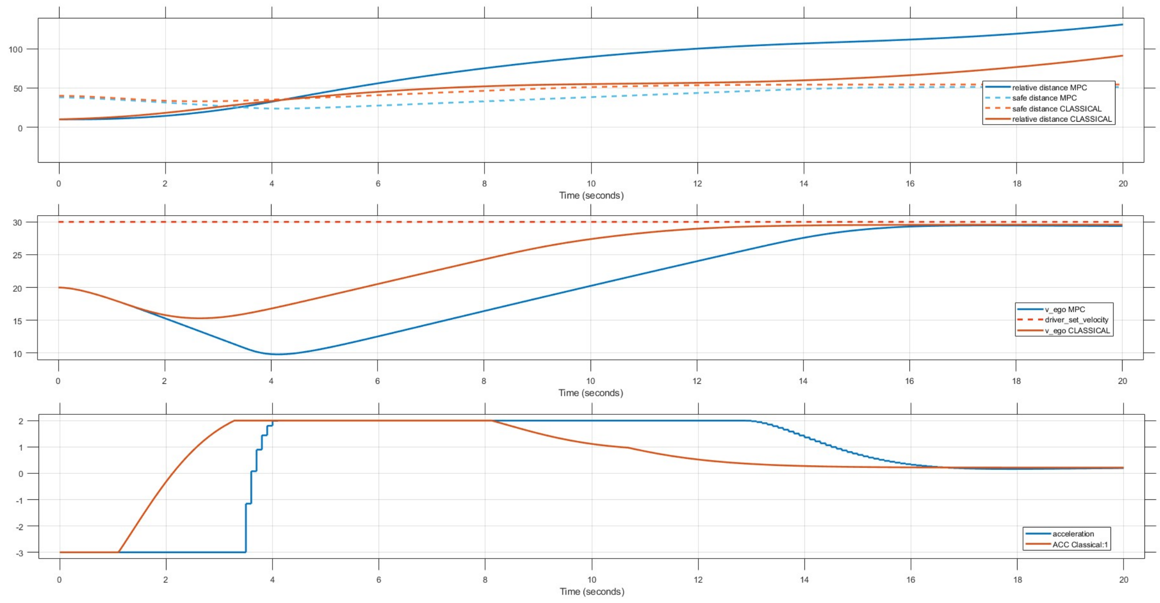


Figure 4.6: Comparison of the controllers in the first 20 seconds of simulation

In Figure 4.6, the simulation results comparing the responses of the two controllers are presented. The results from the classical controller are shown in orange, while those from the MPC controller are depicted in blue.

In both cases, the acceleration value is -3 m/s^2 as the relative distance is less than the safe distance. In the case of the MPC, the deceleration value is maintained longer compared to the classical controller; indeed, the safe distance is reached 1 second earlier than with the other controller. The model response to MPC control is more rapid.

Regarding reaching cruising speed, this occurs more slowly with MPC because deceleration is maintained longer. This deliberate deceleration strategy by the MPC aims to ensure a smoother transition to the target speed, avoiding overshoot and ensuring stability in the control process. Hence, the minimum speed attained is lower compared to the classical controller case.

Additionally, there are noticeable "steps" in the acceleration impulse to reach its maximum value, attributed to MPC capability to provide better overall system optimisation and a more adaptive response to varying conditions. This characteristic can also lead to more pronounced variations in control output compared to a classical controller, which may be perceived as smoother and more immediate in acceleration regulation.

4.2 ACC linear trajectory and curve

This paragraph simulates a scenario where the trajectory is initially linear, followed by a curved path with a radius of 28 m [Figure 4.7].

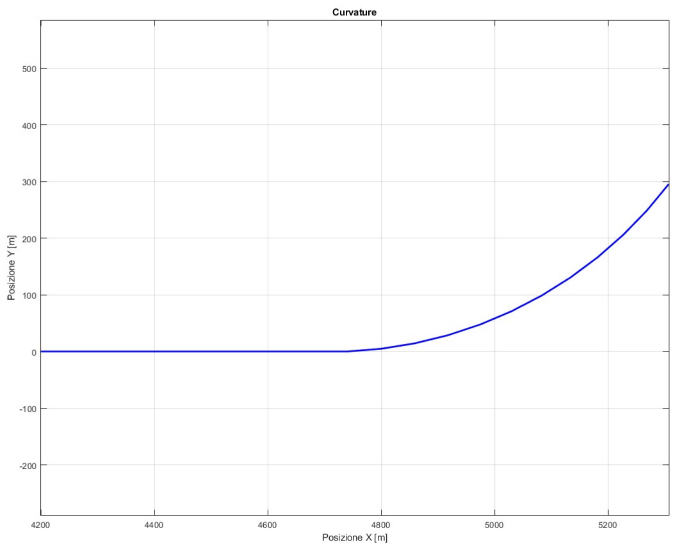


Figure 4.7: Linear trajectory followed by a curved path

Adjustments are made regarding:

- Ego vehicle speed
- Relative distance

When analysing behaviour in curves, it is crucial to consider the entire velocity vector, including both lateral and longitudinal components. Unlike linear trajectories, where distance can be simply calculated as a difference of coordinates along the X-axis, curved paths require accounting for variations along the Y-axis as well. To address this, a computational block was implemented to consider both axes. The simulations involved the only MPC controller.

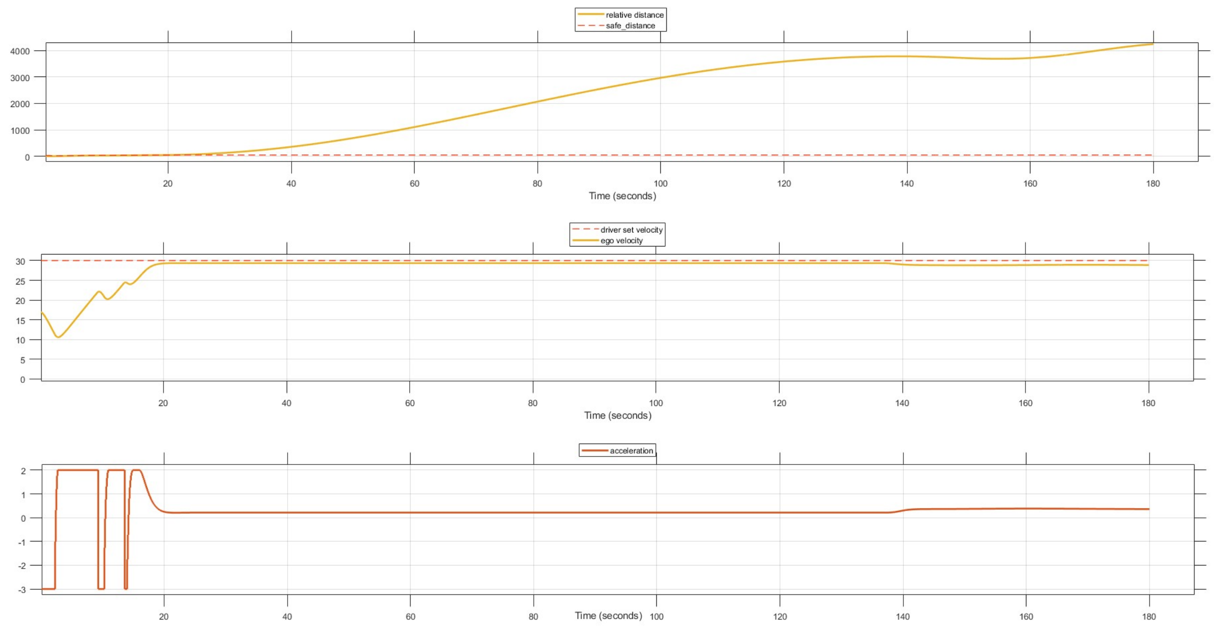


Figure 4.8: Simulation results with a curved path - MPC

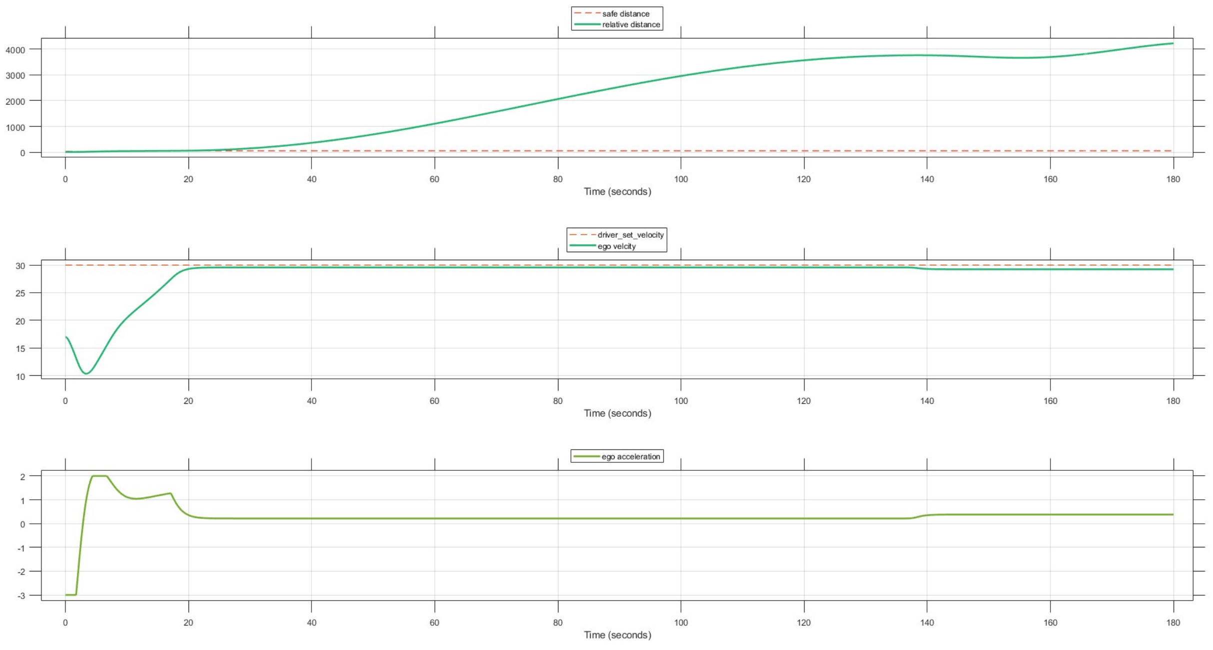


Figure 4.9: Simulation results with a curved path - Classical Controller

Additionally, the steering angle values were changed from 0 rad to 0.04 rad in the final 20 seconds of the simulation. Despite these computational modifications, the results from previous simulations remained consistent. The ego vehicle successfully maintained a safe distance from the vehicle ahead and reached cruising speed. This outcome highlights the effectiveness of the individual controllers, which primarily influence the initial seconds of the simulation before stabilizing to achieve the expected results [Figure 4.8, 4.9].

4.3 LKA results

A similar study was carried out to analyse the road holding in case of curvature of the ego vehicle.[Figure 4.10 and 4.11]

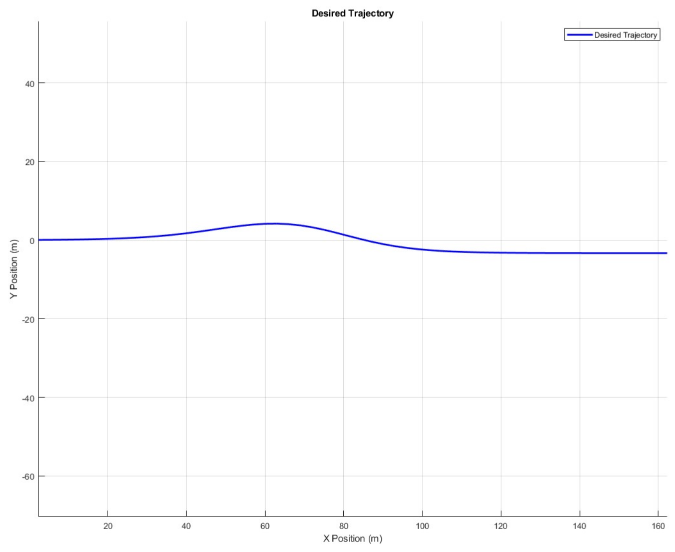


Figure 4.10: LKA: zoom in of desired trajectory

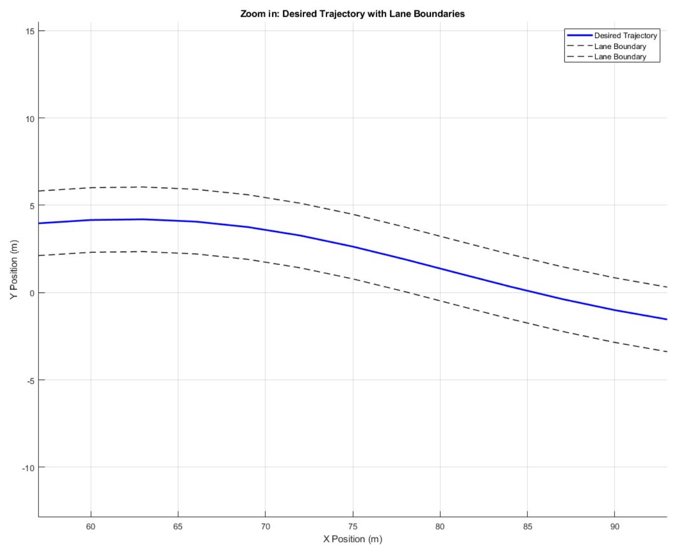


Figure 4.11: LKA: trajectory with lane boundaries

The simulations resulted in the following:

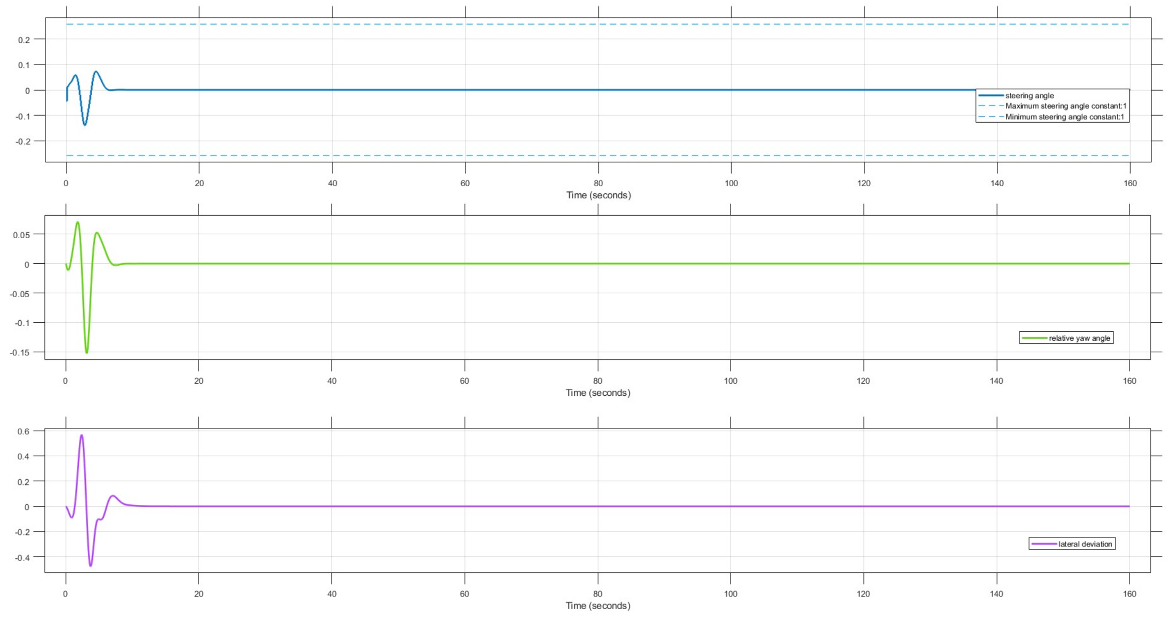


Figure 4.12: LKA simulation results

In Figure 4.12, the x-axis represents time, while the y-axis shows, respectively:

- Steering angle in radians;
- Yaw angle in radians;
- Lateral deviation in metres.

As can be seen from the overall trend, it is indeed possible to minimize the errors discussed in Chapter 3. To better understand the dynamics, it is beneficial to focus on the first 20 seconds of the simulations, as these are crucial for achieving the desired outcomes.

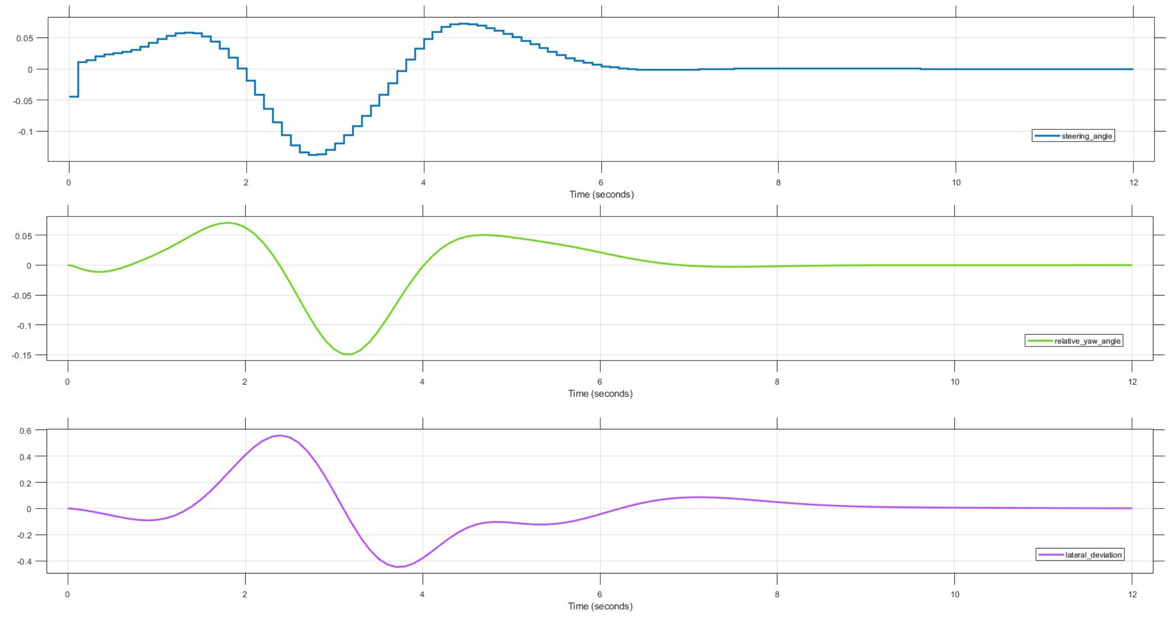


Figure 4.13: LKA first 20 seconds of simulation

The relationship between Steering Angle and Yaw Angle is pivotal: adjusting the steering angle alters the yaw angle, aligning the vehicle in the desired direction. This alignment directly influences the vehicle lateral deviation from the lane center. Specifically, changes in the yaw angle determine the vehicle trajectory, influencing its lateral position within the lane. Moreover, adjustments in the steering angle play a crucial role in minimizing lateral deviation, aiding in keeping the vehicle centered within its lane.

The Adaptive Model Predictive Control (MPC) applied to a Lane Keeping Assistant (LKA) system uses predictive models and adaptive algorithms to maintain a vehicle within its lane, enhancing safety and comfort. Key variables in this system include the steering angle, yaw angle, and lateral deviation.

- **Steering Angle** (δ): The angle at which the front wheels are turned relative to the vehicle's direction. It is the primary control variable used by the MPC to correct the vehicle trajectory.
- **Yaw Angle** (ψ): The angle between the vehicle direction and the lane longitudinal axis, indicating the vehicle rotation around its vertical axis. It helps determine if the vehicle is

turning appropriately to stay in the lane.

- **Lateral Deviation (e)**: The distance between the vehicle center and the lane center. The MPC minimizes this to keep the vehicle centered in the lane.

The objective of the adaptive MPC is to minimize the errors relative to the yaw angle and lateral deviation, ensuring that both $e(t)$ and $\psi(t)$ approach zero. This results in the vehicle maintaining the desired lane position and orientation, achieving optimal lane keeping. The MPC controller predicts the future states of the vehicle based on these variables and optimally adjusts the steering angle to minimize lateral deviation and correct the heading, ensuring the vehicle remains centered in its lane while adapting to changes in the vehicle dynamics and the driving environment.

Chapter 5

ROSMASTER X3 Robot

In this chapter, the behaviour of a ROSMASTER X3 will be analysed by adapting models previously developed for a generic vehicle.

5.1 ROSMASTER X3 characteristics

The ROSMASTER X3 is a state-of-the-art mobile robot engineered for research, development, and educational purposes in the field of robotics. This robot boasts a comprehensive array of technical features and functionalities, rendering it a versatile and powerful instrument for investigating various robotics applications. The robot parameters are given in the Table 5.1.

Parameter	**Value**	**Description**	**SI**
m	6	Vehicle mass	kg
I_z	0.0434	Moment of inertia with respect to yaw axis	kg·m^2
l_f	0.08	Longitudinal distance from c.g. to front wheels	m
l_r	0.08	Longitudinal distance from c.g. to rear wheels	m
C_f	5000	Roll stiffness of front wheels	N/rad
C_r	5000	Roll stiffness of rear wheels	N/rad
τ	0.1	Longitudinal time constant	s

Table 5.1: robot parameters

These parameters are incorporated into the Simulink 'bicycle model' block, enabling adaptation of the model to the specific characteristics of the robot under investigation.

When designing and implementing control systems such as Adaptive Cruise Control (ACC) and Lane Keeping Assist (LKA) for the ROSMASTER X3 Robot, it is crucial to consider its physical limitations. The robot has a maximum speed of approximately 2.5 m/s, which is significantly lower than the speeds achieved by modern cars. Additionally, its acceleration capacity is limited, with a maximum acceleration of 1.5 m/s^2. These constraints must be taken into account to ensure that the control systems are optimized for the robot performance capabilities, ensuring safe and effective operation within its specific limits.

5.2 Adaptive Cruise Control

The control of longitudinal dynamics was addressed by considering a linear trajectory, as illustrated in Figure 5.1. Simulations employed two types of controllers: the classical controller and the Model Predictive Control (MPC) controller.

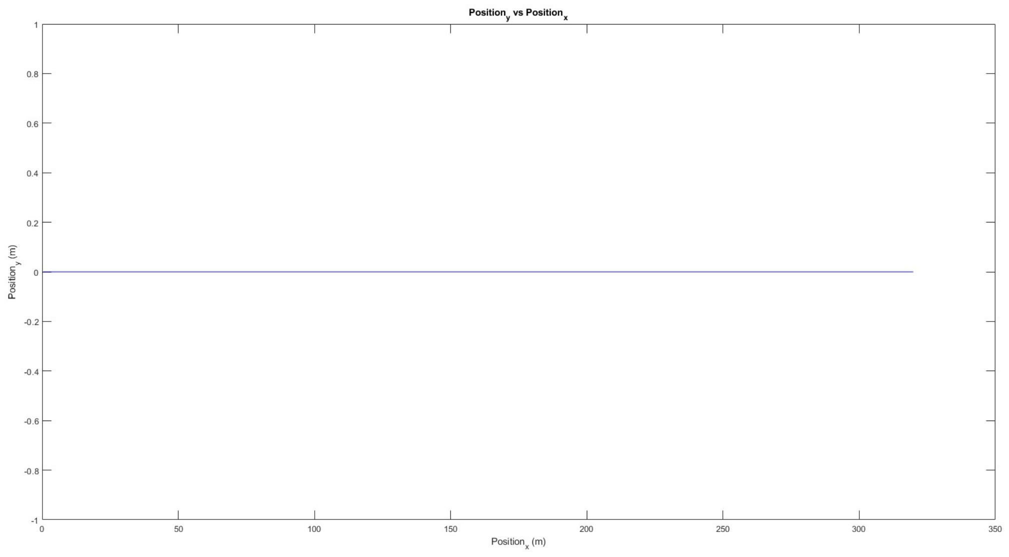


Figure 5.1: linear trajectory

The initial conditions for the lead vehicle and the ego vehicle are different: the lead vehicle starts with an initial velocity of 0 m/s and is positioned at $X = 2$ m in the fixed reference system, while the ego vehicle starts at $X = 0$ m with a velocity of 1.1 m/s.

For this analysis, a cruising speed of 1.2 m/s was chosen, and the acceleration range spans from [-0.3; 0.3] m/s^2. These values were selected based on the specific characteristics of the robot.

5.2.1 Classical controller

Considering the observations above mentioned, the simulation results with the classical controller are as follows [Figure 5.2]:

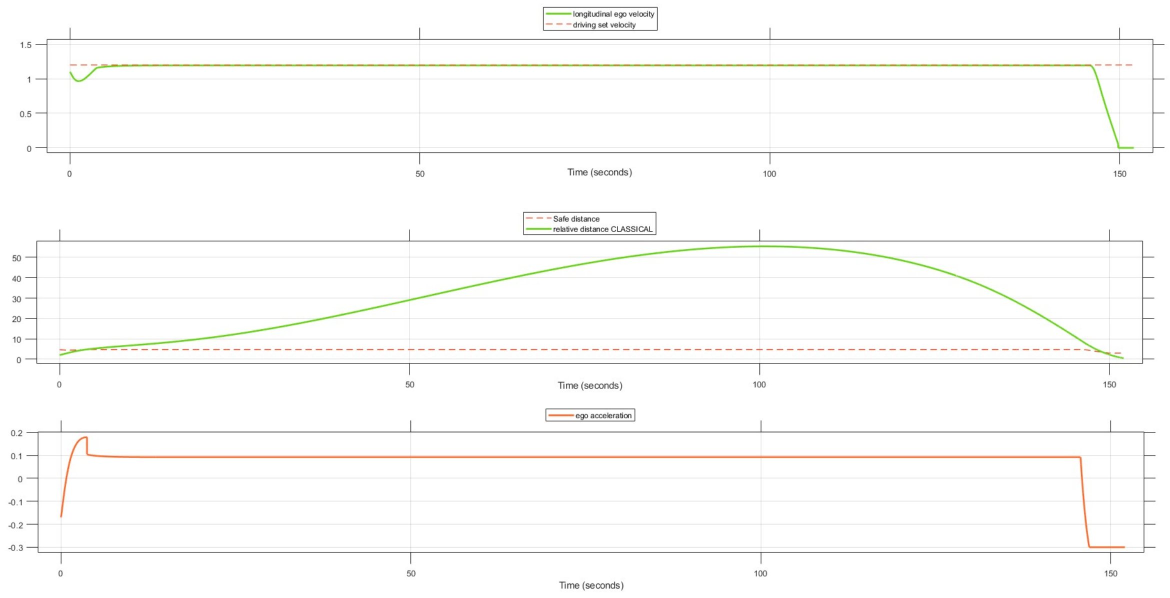


Figure 5.2: ACC simulation results for ROSMASTER X3

In all three graphs [Figure 5.2], the x-axis represents the simulation time, while the y-axis shows the following variables (in the order of the graphs):

- longitudinal ego velocity in meters per second (m/s);
- relative distance in meter (m);
- ego acceleration in meters per second squared (m/s^2).

Overall, the ego vehicle demonstrates capability in meeting requirements by maintaining a relative distance above the prescribed limit while achieving cruising speed. However, it is crucial to analyze both the initial and final phases of the simulation.

During the first 15 seconds of the simulation, the following scenario unfolds [Figure 5.3]:

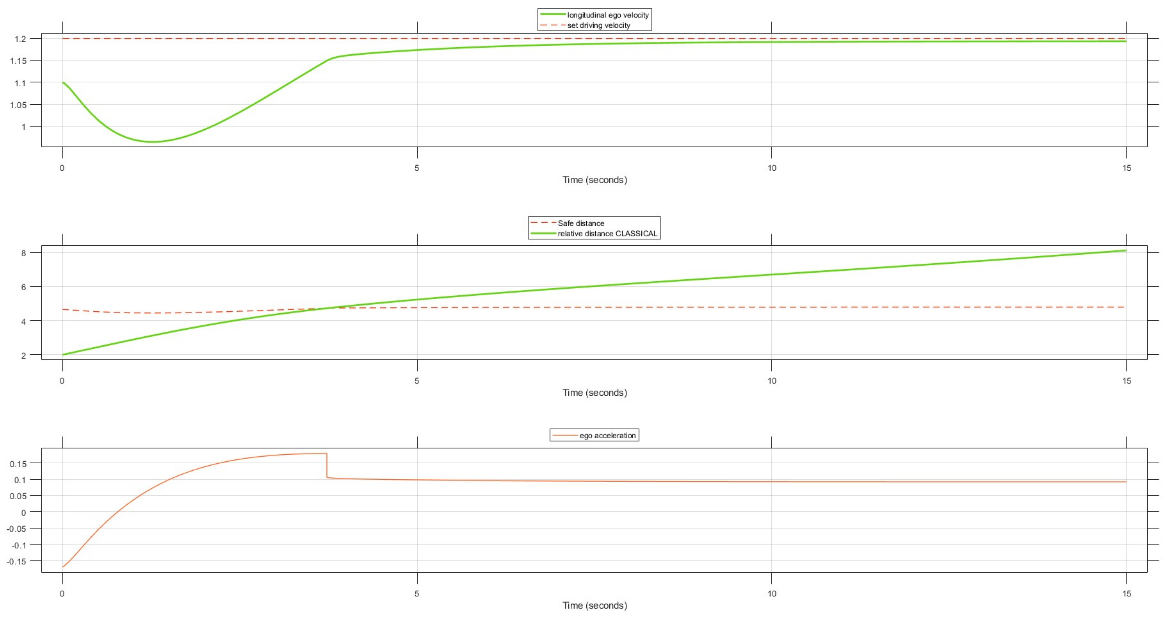


Figure 5.3: First 15 seconds of simulation

The control of longitudinal dynamics is managed through a sequence of phases. Initially, the ego vehicle starts with its relative distance d_{rel} to the lead vehicle below the safe distance d_{safe}, while maintaining a constant longitudinal velocity V_{ego}. In the deceleration phase, the Adaptive Cruise Control (ACC) applies negative acceleration a_{decel} to reduce V_{ego} and increase d_{rel}. Subsequently, V_{ego} gradually decreases as a_{decel} takes effect. Upon d_{rel} is close to d_{safe}, ACC transitions to zero acceleration $a_{\text{transition}}$, smoothly reducing a_{decel} until V_{ego} stabilizes in the transition phase. In the acceleration phase, ACC applies positive acceleration a_{accel} to increase V_{ego} towards the set cruising speed, gradually accelerating until reaching the desired velocity. During steady-state operation, ACC maintains V_{ego} at cruising speed by making small adjustments to a_{accel} or a_{decel} as necessary. The system continuously monitors and adjusts based on real-time feedback. When the ego vehicle reaches cruising speed, the positive acceleration compensates for ongoing energy losses and resistances that occur during steady-state driving. This includes overcoming air resistance, internal friction and mechanical losses. In essence, the positive acceleration ensures the vehicle maintains a stable and controlled speed by continuously providing the necessary energy to counteract these forces, thereby sustaining smooth operation at the desired velocity.

Final part of simulations [Figure 5.4];

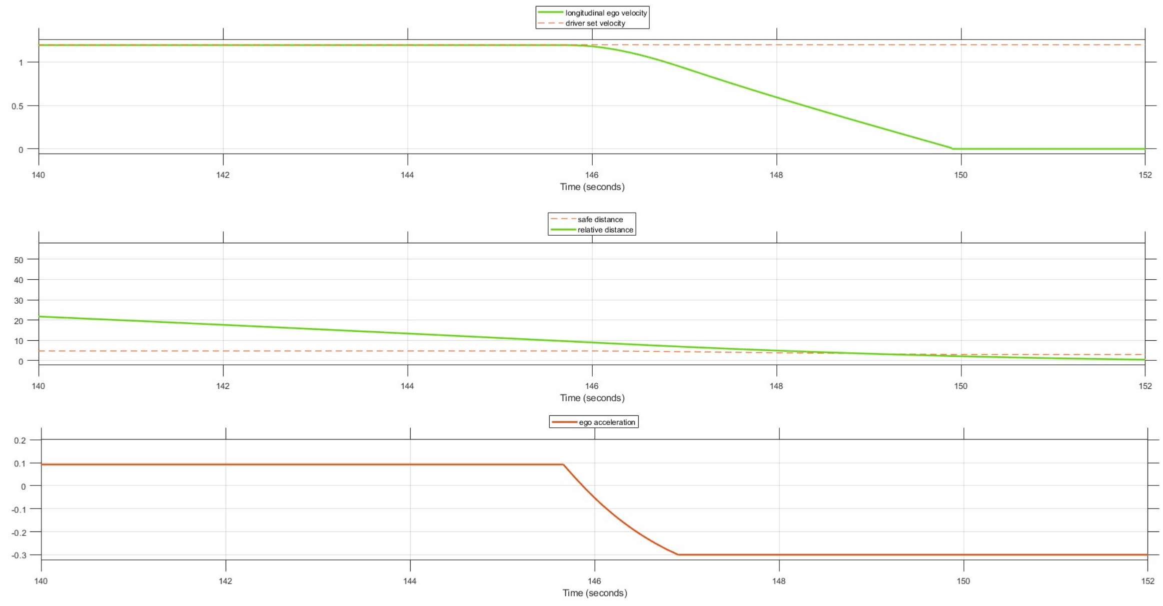


Figure 5.4: Final seconds of simulation

In the final phase of the simulation, it is evident that the relative distance decreases due to the actions of the preceding vehicle. Consequently, the control system responds by reducing the acceleration to the maximum allowed for deceleration, resulting in a significant reduction in the vehicle speed. This sudden decrease in acceleration is necessary to maintain the prescribed safety distance from the preceding vehicle, which may have suddenly reduced its speed. Despite the acceleration decrease, if the maximum deceleration set by the system is not sufficient to counteract the reduction in speed of the preceding vehicle, the relative distance may continue to decrease. This phenomenon can occur if the preceding vehicle brakes more rapidly than the ACC system can adjust, causing the vehicle under ACC to reduce its speed until it reaches a value close to 0 m/s.

5.2.2 MPC controller

Referring to what is defined in 5.2, in terms of trajectory and initial conditions, the Figure 5.5 reports the results obtained using an MPC controller. In all three graphs [Figure 5.5], the x-axis

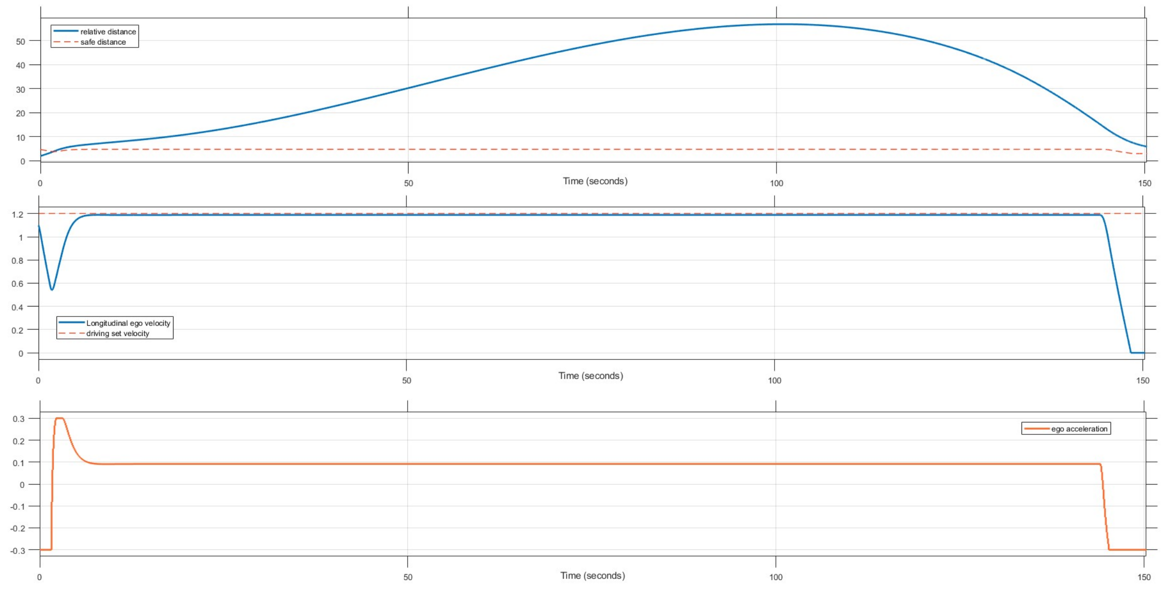


Figure 5.5: MPC simulation results for ROSMASTER X3

represents the simulation time, while the y-axis shows the following variables (in the order of the graphs):

- relative distance in meter (m)
- longitudinal ego velocity in meters per second (m/s);;
- ego acceleration in meters per second squared (m/s^2).

Regarding the classical controller, it is beneficial to analyse its performance during the initial and final phases of the simulation, which are notably influenced by the preceding vehicle behaviour.

In the initial seconds of the simulation, the following conditions are observed, Figure 5.6:

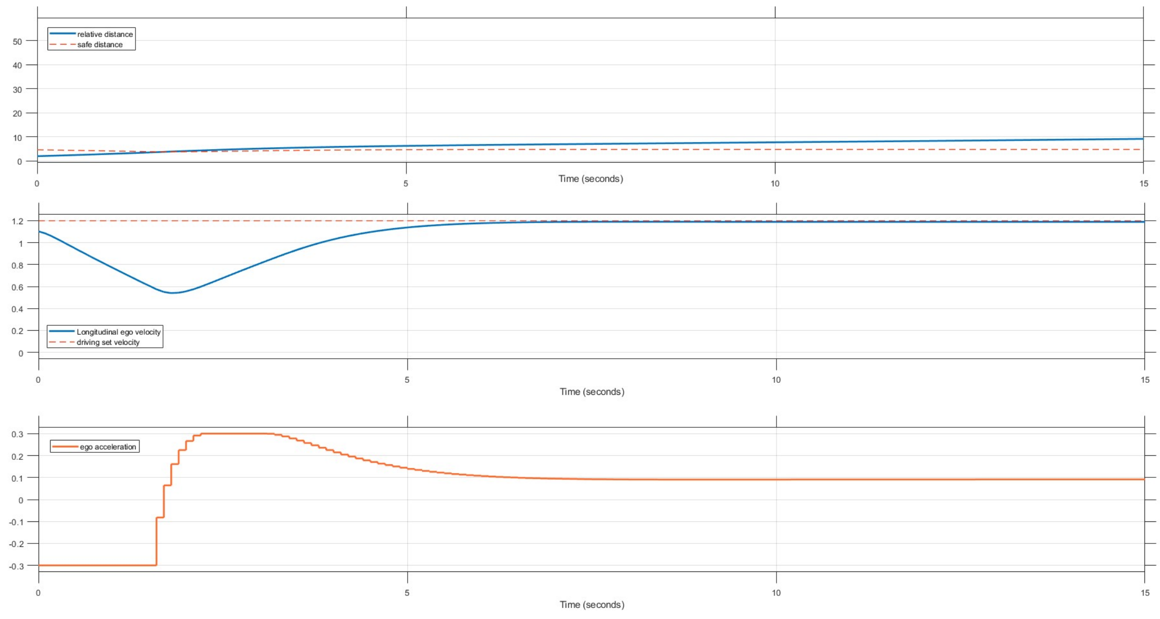


Figure 5.6: Initial part of simulation with MPC

Therefore, the controller promptly adjusts the acceleration, achieving a maximum deceleration of -0.3 m/s^2, selected based on the robot physical capabilities. This results in a reduction of speed to a minimum of 0.55 m/s, while concurrently increasing the relative distance.

A slight delay exists between the onset of deceleration and the actual increase in relative distance, as well as in the speed change, which provides feedback to the controller for calculating the safe distance. As the relative distance increases, an acceleration impulse of 0.3 m/s^2 is activated, causing an increase in the ego vehicle velocity until it matches the set speed.

Subsequently, once the safe distance is exceeded, deceleration gradually diminishes to 0.1 m/s^2, thereby ensuring the maintenance of the cruising speed; the acceleration value is greater than 0 in order provide enough energy to proceed at constant speed compensating the losses.

In the final part of the simulation, the following situation occurs [Figure 5.7]:

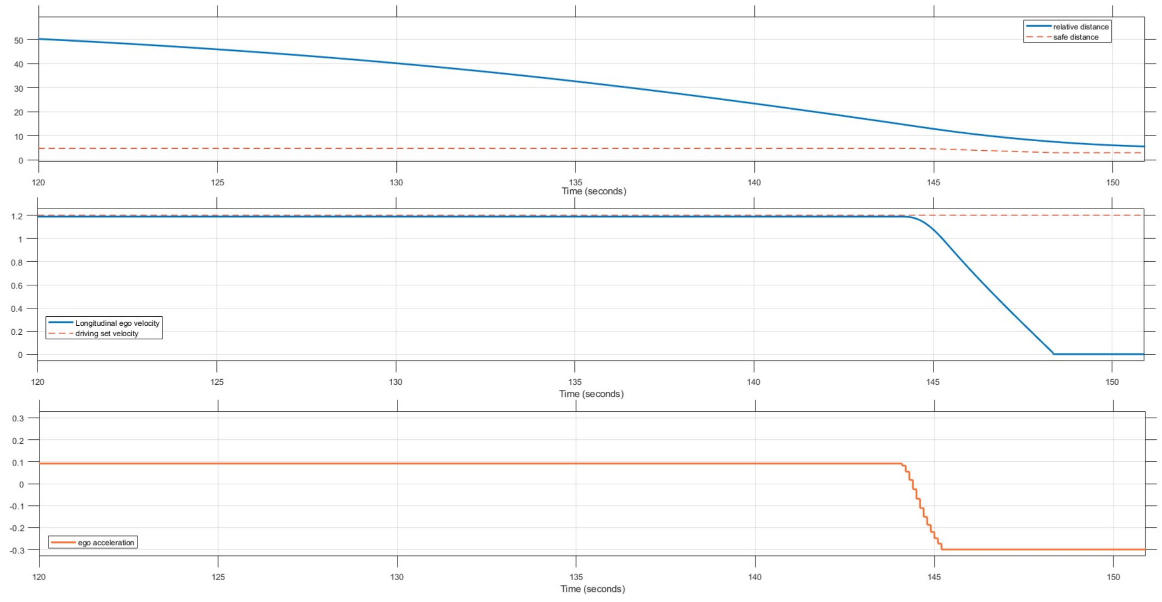


Figure 5.7: Final part of simulation with MPC

In the final phase of the simulation, it becomes apparent that the relative distance diminishes due to the actions of the preceding vehicle. Consequently, the control system responds by decreasing the acceleration to the maximum permissible for deceleration, resulting in a notable reduction in the vehicle speed. This abrupt acceleration surge is essential to uphold the mandated safety distance from the preceding vehicle, which might have abruptly reduced its speed. Moreover, throughout this phase, the relative distance remains consistently close to the safe distance, indicating stable control system performance under varying traffic conditions.

5.2.3 Comparison of the results

Comparing the results obtained by the two controllers, in the initial [Figure 5.8] and final phase [Figure 5.9] of the simulation, we observe:

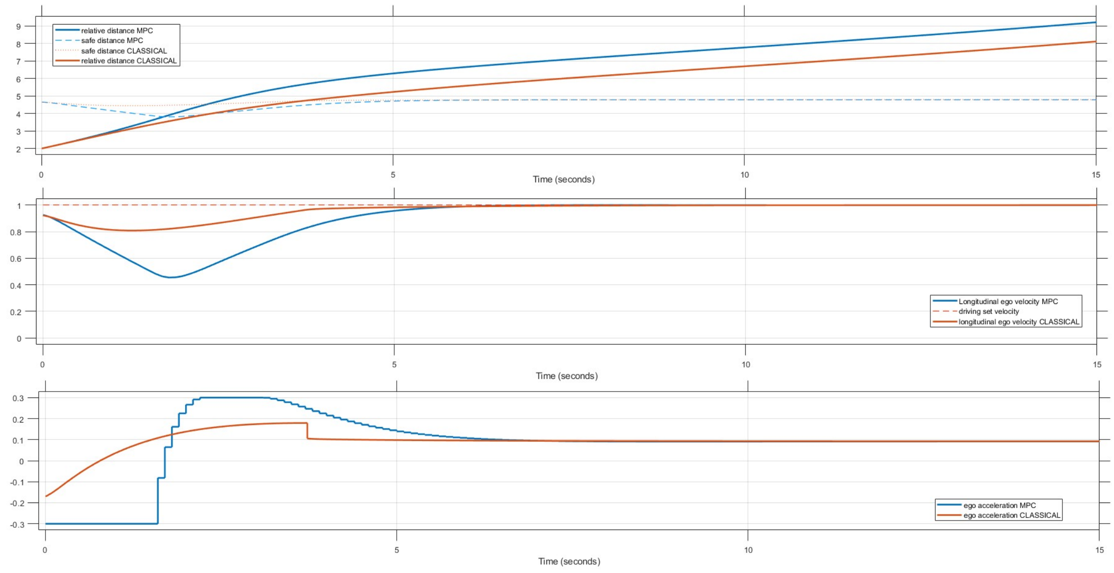


Figure 5.8: Comparison of the initial phase of simulation

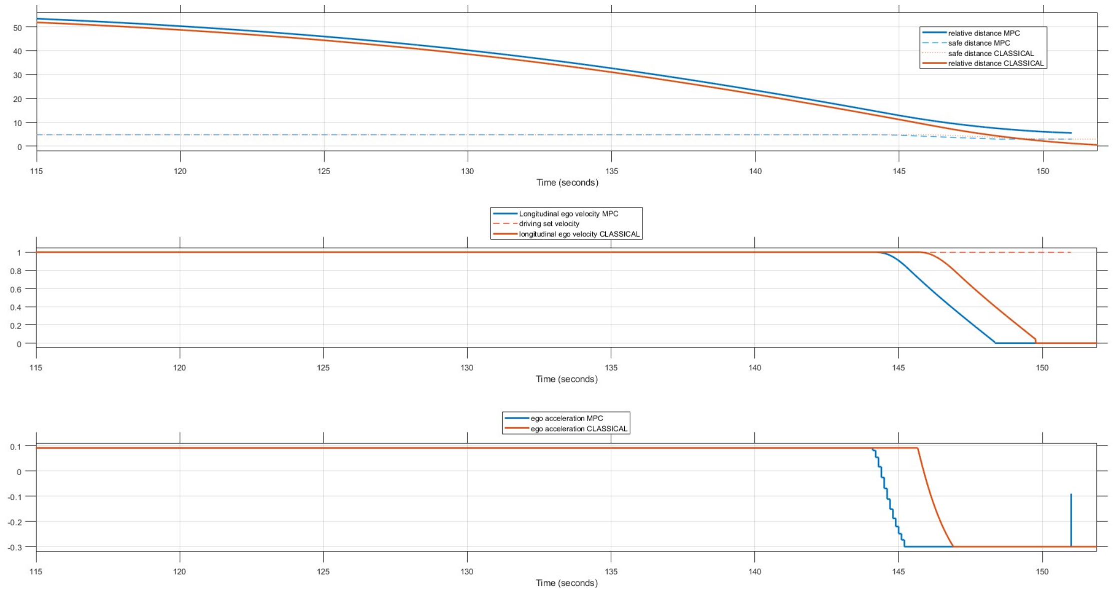


Figure 5.9: Comparison of the final phase of simulation

In Figures 5.8 and 5.9, the results obtained by the classical controller are represented in orange, while those of the MPC are shown in blue. As noted in 4.1.3, during the initial phase [Figures 5.8], it is observed that the MPC achieves cruise speed more gradually due to prolonged deceleration. This strategy ensures a smoother transition to the desired speed, avoiding overshoot and ensuring stability in the control process. Consequently, the minimum speed attained is lower compared to the classical controller.

Furthermore, noticeable steps in acceleration to reach its maximum value are evident, attributed to the MPC ability to optimize the system more effectively overall and provide a more adaptive response to changing conditions.

In the final part of the simulation [Figures 5.9], as the relative distance decreases, it is observed that the acceleration controlled by the MPC decreases earlier compared to that of the classical controller. This is because the MPC is based on predicting future system dynamics to anticipate control needs, thereby adjusting its response in advance of upcoming events to optimize the system long-term behaviour.

5.3 LKA

A similar study was carried out to analyse the road holding in the case of the ego vehicle following a curve [Figure 5.10, Figure 5.11].

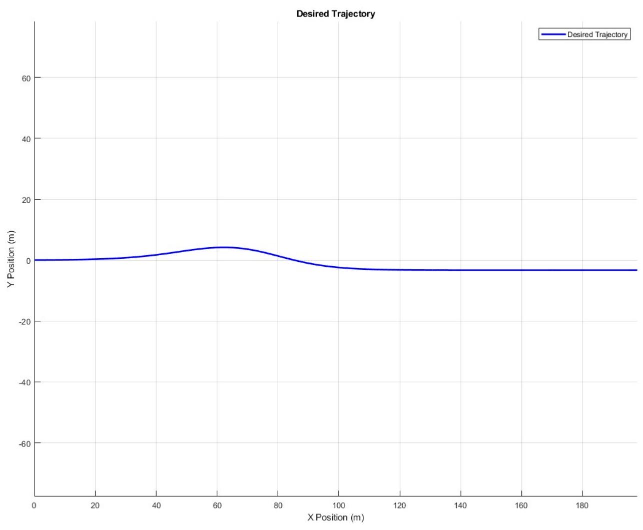


Figure 5.10: Desired trajectory

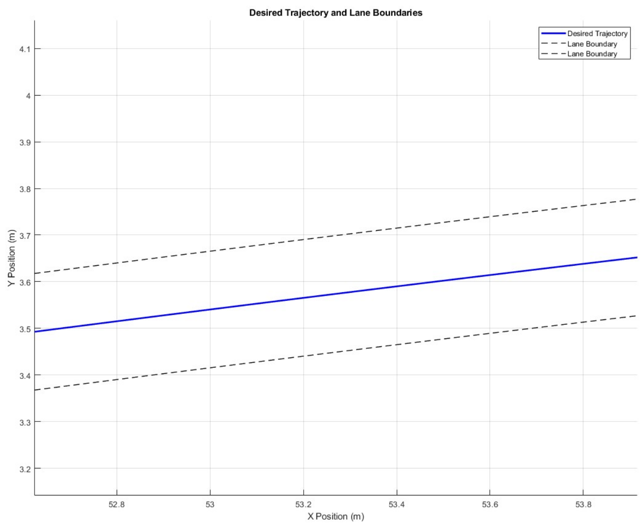


Figure 5.11: Desired trajectory with boundaries

Figure 5.12 shows the simulation results.

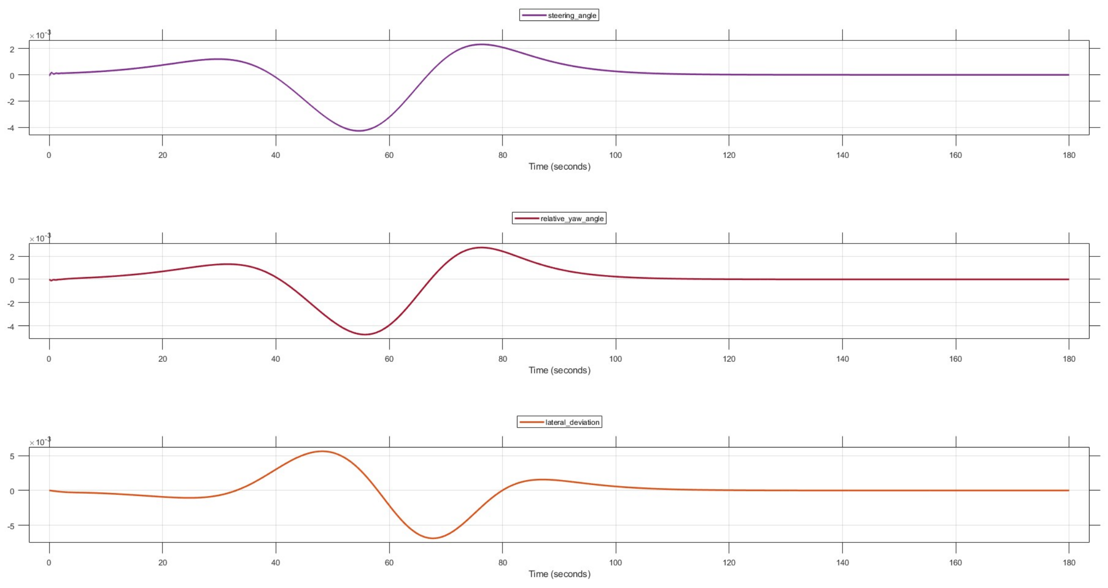


Figure 5.12: LKA simulation results

In Figure 5.12, the x-axis represents time, while the y-axis shows, respectively:

- Steering angle in radians;
- Yaw angle in radians;
- Lateral deviation in metres.

The relationship between the steering angle and the yaw angle is pivotal: adjusting the steering angle alters the yaw angle, aligning the vehicle in the desired direction. This alignment directly influences the vehicle lateral deviation from the lane centre. Specifically, changes in the yaw angle determine the vehicle trajectory, influencing its lateral position within the lane. Moreover, adjustments in the steering angle play a crucial role in minimising lateral deviation, aiding in keeping the vehicle centred within its lane.

Acting on the steering angle to control the vehicle, considering the yaw angle and lateral deviation, ensure the vehicle stays optimally and safely within its lane. This alignment ensures that the vehicle heading, or its orientation relative to the lane, is maintained accurately, promoting safe and effective lane keeping. The objective of the adaptive MPC is to minimise the errors related to the yaw angle and lateral deviation, ensuring that both $e(t)$ and $\psi(t)$ approach zero. This results in the vehicle maintaining the desired lane position and orientation, achieving optimal lane keeping.

Chapter 6

Conclusions and future developments

The conducted study yielded highly satisfactory results in terms of both Adaptive Cruise Control (ACC) and Lane Keeping Assistant (LKA). From the implementation of the ACC, it emerged that the two controllers have different timings in acting on the controlled system. The model employing Model Predictive Control (MPC) proved to be the most efficient due to its ability to make decisions considering a predictive horizon, optimising control variables to minimise the cost function while adhering to system constraints.

With regard to the Lane Keeping Assistant, improvements can be achieved by integrating logic that permits overtaking while maintaining a lateral safety distance.

Furthermore, the two models can be integrated through more complex control logic that enables the simultaneous action of these technologies, resulting in a single principal model composed of two subsystems: one for the control of longitudinal dynamics and one for the control of lateral dynamics.

Considering the detailed results obtained from the simulations using the specific data of the ROSMASTER X3 for the Adaptive Cruise Control, we conclude that the outcomes achieved with the Classical Controller are deemed acceptable in comparison to those with the MPC. This preference is underpinned by several factors, including the ease of implementation, rapid response, optimisation of limited computational resources, and operational robustness. Such a selection proves particularly advantageous within the context of code generation for subsequent deployment on the dedicated hardware of the robot, given the constraints posed by limited memory capacities.

Similarly, the Lane Keeping Assistant utilises an MPC, necessitating increased memory allocation on the card for its optimal functionality. It is crucial to highlight that the simulations conducted stand to benefit significantly from the integration of the advanced sensors available on the ROSMASTER X3, alongside the incorporation of the computer vision component. This integrated approach not only enhances the overall performance of the system but also augments its capability to adapt effectively to the dynamic conditions of the operational environment.

Bibliography

[1] Sijie Wei, Peter E. Pfeffer, and Johannes Edelmann. State of the art: Ongoing research in assessment methods for lane keeping assistance systems. *IEEE Transactions on Intelligent Vehicles*, pages 1–28, 2023.

[2] SAE International. Sae international, 2021.

[3] Synopsis.

[4] Tao Wang, Jingmin Xin, and Nanning Zheng. *A Method Integrating Human Visual Attention and Consciousness of Radar and Vision Fusion for Autonomous Vehicle Navigation.* 2011 IEEE Fourth International Conference on Space Mission Challenges for Information Technology, 2011.

[5] Shripad Bhatlawande, Swati Shilaskar, and Amol Dhanawade. Lidar based detection of small vehicles. In *2022 3rd International Conference for Emerging Technology (INCET)*, pages 1–5, 2022.

[6] Joko Slamet Saputro, M. Shafiy Widi Kresno Wibisono, Joko Hariyono, Miftahul Anwar, and Warindi. Design of an adaptive cruise control system using pid control method on electric vehicle prototypes. In *2023 International Conference on Advanced Mechatronics, Intelligent Manufacture and Industrial Automation (ICAMIMIA)*, pages 599–605, 2023.

[7] Shivam Kumar, Vivek Shaw, Janapriyo Maitra, and Raja Karmakar. Fcw: A forward collision warning system using convolutional neural network. In *2020 International Conference on Electrical and Electronics Engineering (ICE3)*, pages 1–5, 2020.

[8] Naga Praveen Babu Mannam and P. Rajalakshmi. Determination of adas aeb car to car and car to pedestrian scenarios for autonomous vehicles. In *2022 IEEE Global Conference on Computing, Power and Communication Technologies (GlobConPT)*, pages 1–7, 2022.

[9] Palepu Sai Sri Harsha, Cherishma Reddy Sriyapu Reddy, Tirumalasetty Nikhil Venkata Sai, and Sreeja Kochuvila. Vehicle detection and lane keep assist system for self-driving cars. pages 1–5, 2023.

[10] Diomidis I. Katzourakis, Nenad Lazic, Claes Olsson, and Mathias R. Lidberg. Driver steering override for lane-keeping aid using computer-aided engineering. *IEEE/ASME Transactions on Mechatronics*, 20(4):1543–1552, 2015.

[11] Aleksandra Simić, Ognjen Kocić, Milan Z. Bjelica, and Milena Milošević. Driver monitoring algorithm for advanced driver assistance systems. pages 1–4, 2016.

[12] The MathWorks Inc. Automated driving toolbox (r2023b), 2023.

[13] U. Kiencke and L. Nielsen. Automotive control systems: For engine, driveline, and vehicle. *Springer*, 2005.

[14] Stefano Malan. Materiale didattico corso di automotive control systems. 2022.

[15] A. Galip Ulsoy, Huei Peng, and Melih Çakmakci. Automotive control systems. *Cambridge University Press*, 2012.

[16] Gang Liu, Hongbin Ren, Sizhong Chen, and Wenzhu Wang. The 3-dof bicycle model with the simplified piecewise linear tire model. pages 3530–3534, 2013.

[17] Huei Peng and Masayoshi Tomizuka. Lateral control of front-wheel-steering rubber-tire vehicles. *Institute of Transportation Studies, UC Berkeley, Institute of Transportation Studies, Research Reports, Working Papers, Proceedings*, 01 1990.

[18] The MathWorks Inc. Model predictive control toolbox matlab r2023b, 2023.