POLITECNICO DI TORINO

Master's Degree in Mechatronic Egineering

Master's Degree Thesis

Design, Simulation and Testing of a linear MPC for lateral dynamic control of a Formula Student Driverless prototype

Supervisors

Prof. Nicola AMATI

Prof. Massimo CANALE

PhD Luis M. CASTELLANOS MOLINA

Candidate

Alessandro DI ROSIO

December 2023

Summary

The automotive world has passed through a large number of evolutionary steps in history, the latter being the development and implementation of autonomous driving capabilities in passenger vehicles, with the objective of making travelling by car more efficient and secure for people.
With this awareness, Formula Student, that is one of the most important University student competitions of engineering, introduced the Driverless category in 2017. The aim is to give the opportunity to autonomous driving to develop even faster inside universities, making future engineers able to confront this new technological challenge.
This thesis work presents the design and real-time deployment of a model predictive controller (MPC) for vehicle dynamics control in the Formula Student Driverless prototype of Politecnico di Torino. The scope is to propose a simple yet effective approach that can effectively control the yaw dynamics to follow a reference trajectory of a closed loop circuit.
To properly design any controller, the system dynamics have to be deeply analysed first. For this reason, a proper simulation model is developed in MATLAB® Simulink, composed by the mathematical forward model of the vehicle, validated with experimental data, and the low level controller (LLC) already deployed on the real prototype.
The controller algorithm is written in C++ language to enhance the code efficiency and runtime, allowing a software in the loop (SiL) testing procedure, making straight forward the real time implementation (RTI) on the embedded hardware of the prototype. An hardware in the loop (HiL) validation is performed, allowing a full-scale testing and tuning of the developed controller in its final configuration.
Finally, results of the MPC performance on the HiL tests bench are presented and future developments are addressed.

Table of Contents

List of Tables

List of Figures

Acronyms

SCD
Squadra Corse Driverless PoliTO

RRT
rapid-exploring random tree

MPC
model predictive controller

SAE
Society of Automotive Engineers

FSAE
Formula SAE

ADAS
advanced driver assistance systems

CV
internal combustion vehicle

EV
electric vehicle

DV
driverless vehicle

ACU
autonomous control unit

ECU

electronic control unit

LLC

low level controller

ROS

Robotic Operating System

ROS2

Robotic Operating System 2

μC

micro-controller

RF

reference frame

CoG

center of gravity

CoP

center of pressure

TV

torque vectoring

IMU

inertial measurement unit

LiDAR

light detection and ranging

LTI

linear time invariant

LTV

linear time variant

LPV

linear parameter varying

OP

optimization problem

QP

quadratic program

SLAM

simultaneous localization and mapping

RTI

real time implementation

SiL

software in the loop

HiL

hardware in the loop

Chapter 1

Introduction

1.1 Formula SAE

The Society of Automotive Engineers (SAE) is a non-profit educational and scientific organization, which is committed to advancing mobility technology for the benefit of humanity. With a membership exceeding 130'000 engineers and scientists, SAE focuses on generating technical knowledge across various self-propelled vehicles. The organization shares this wealth of information through its meetings, publications, technical papers, magazines, standards, reports, professional development programs, and electronic databases. Among its different commitments, in 1981 the first Formula SAE (FSAE) event was organised by the SAE organization, a university competition aimed to make engineering students apply their knowledge in the automotive field to design and build racing prototypes.
Other than making young engineers improve their skills with practical experience while still studying, the FSAE aims to be a research competition where universities are spurred to invest in the automotive research, speeding up the innovation process in the engineering field. For this reason the FSAE has always followed the most actual trends in the automotive field, expanding and updating the competition: starting from having only the internal combustion vehicle (CV) category, the Hybrid Vehicles class was introduced in 2007, followed by the electric vehicle (EV) class in 2010, and concluding with the most recent driverless vehicle (DV) category, introduced in 2017. Focusing on this newest class of vehicles, the SAE organization introduced the concept of advanced driver assistance systems (ADAS) as a transformative paradigm in automotive technology, aiming to enhance vehicle safety, efficiency, and overall driving experience. This includes the introduction of cutting-edge sensors, like cameras, radar, and other advanced technologies to provide real-time data and feedback. Common ADAS functionalities include adaptive cruise

control, lane departure warning and assistance, automatic emergency braking, blind-spot detection, parking assistance, and collision avoidance systems. By actively monitoring the vehicle's surroundings and analyzing potential risks, ADAS alert drivers to potential hazards and, in some cases, intervenes autonomously to prevent or mitigate collisions. The integration of ADAS into modern vehicles reflects a commitment to improve road safety, reducing accidents, and enhancing overall transportation efficiency. As technology continues to evolve, ADAS is expected to play a pivotal role in the development of autonomous vehicles, paving the way for a future where driving is not only safer but also more intelligent and connected. Five classes of ADAS capabilities have been introduced by SAE to describe the level of automation and intervention requested to the driver, as shown in Figure 1.1.

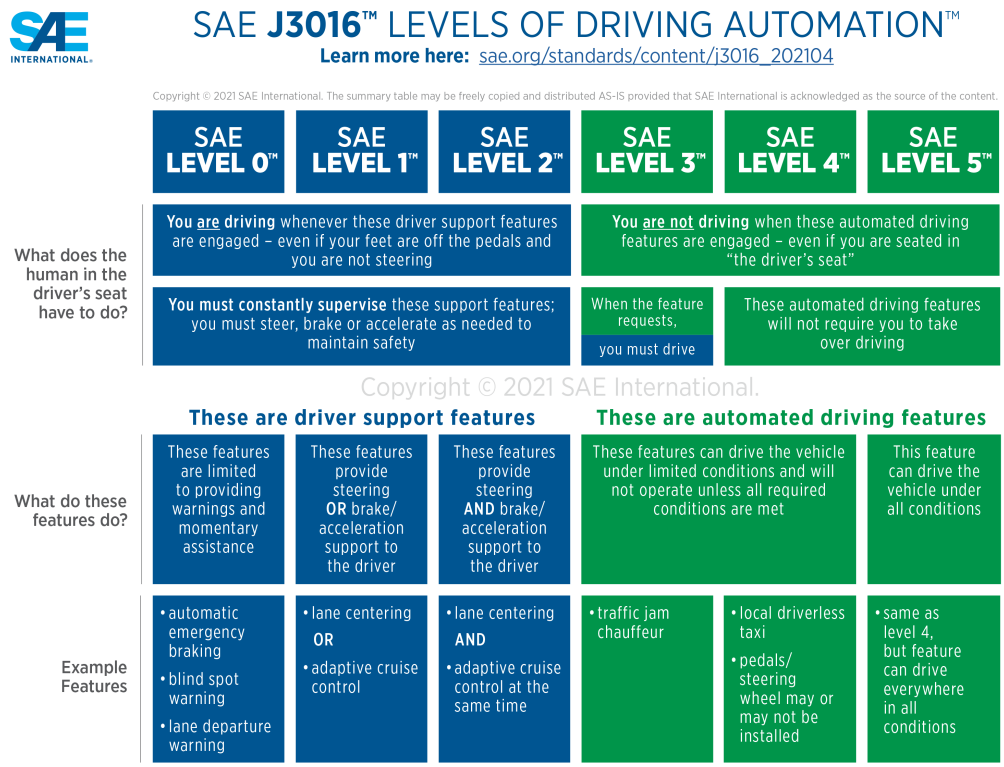

Figure 1.1: SAE levels of Driving Automation.

As of today, most advanced vehicle companies in the automated driving capabilities achieved a SAE Level 3 of automation, which are already deployed in road vehicles. The FSAE competition, in particlar with the DV category, aims to develop and test automated driving capabilities of higher levels, ensuring high safety standards. During the competition the Driverless prototypes run without

any driver on board, along four dynamic events with different characteristics. In general, the circuit is made by cones of different shapes and colours as in figure 1.2, which must be recognized by the car first, which then moves accordingly in the fastest time possible.

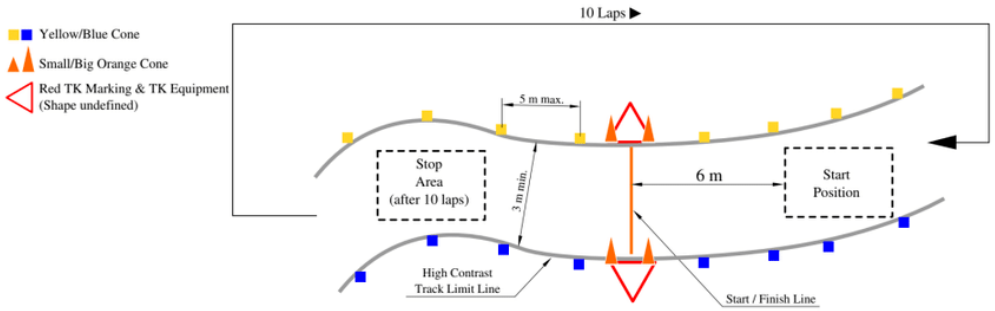

Figure 1.2: FSAE DV track cones layout.

Being an engineering design competition, a FSAE competition holds two different type of events: static and dynamic. In the static events, namely Design Event, Cost Event and Business Plan Event, each team presents the overall season project to a judges committee, that validate the team organization and improvements, as well as its capability to confront with typical industrial challenges, such as the costs management for the prototype realisation and the capability to derive a business plan idea from the designed prototype. In the dynamic events, where performances achieved are measured directly on the track, teams are called to let the car run through four different disciplines:

- Acceleration: straight driving run for 75 meters, after which the car should come to a safe stop.
- Skidpad: 8-shaped track with standard dimensions, which is run two time in each circle
- Autocross: single run of an unknown track
- Trackdrive: 10 runs of an unknown track

Most points are awarded for the Trackdrive event, since it is the most complicated and proves each team capabilities to optimize the lap time through innovative control algorithms.

1.2 Squadra Corse Driverless Team

1.2.1 The Team

Squadra Corse Driverless PoliTO, in short SCD, is a university student team of Politecnico di Torino, born in April 2021 with the goal to research, develop and test Autonomous Driving solutions. The team uses a FSAE electric racing prototype, realised by the fellow student team Squadra Corse PoliTO, to implement and test all the developed solutions, both hardware and software. The SCD team, together with the Center for Automotive Research and Sustainable mobility, CARS@PoliTO, and the LIM laboratory of Politecnico di Torion, developed and installed the Autonomous Steering System and the Autonomous Braking System, which ensured the missing actuation capabilities of the base prototype. Different other researches about state estimation, environmental perception and path planning were carried on together with the research group, which laid the basis of the new born student team. Since the team establishment, all the different subsystems have been integrated on the base prototype, and the missing parts where developed, up until in May 2022 the first SCD prototype made its first meters in fully autonomous mode. The first season ended with a third place overall in the Driverless category of the Formula SAE event held in Varano de' Melegari, Italy, but the driverless system of the SCD prototype was still in its first stages and needed a lot of upgrades.
During the second season of the Team, huge steps forward have been made, both for the hardware and the softwares. On the prototype, a completely redesigned braking system has been developed and tested, which ensured higher safety standards, while a brand new set of sensors and algorithms have been added: a 64 channels Ouster LiDAR, together with 2 ALVIUM Allied Vision cameras, allowed to greatly improve the precision of cones identification and positioning, and also allowed the development of high level algorithms such as SLAM ([1]), Ground Filtering etc. This rapid development of the software package allowed the Driverless package provided by the team to achieve high levels of performance and safety, also giving the opportunity to start researching more complicated solutions. The full equipped prototype has been given the name of VaLentina, which is a female italian name and, if read as two separate words as "Va Lentina", translates to "She goes slowly", a fun way to describe the first very slow meters made in autonomous mode.

1.2.2 The prototype: VaLentina

The prototype which is used by the Squadra Corse Driverless PoliTO team is a Formula SAE electric prototype developed by the fellow team Squadra Corse in 2019, which won that season's edition of the Formula SAE Italy event. The

Figure 1.3: Squadra Corse Driverless and VaLentina at Formula Student East 2023, event held at the Hungaroring F1 circuit.

prototype is equipped with 4 in-wheel electric motors, each one capable of delivering up to 35kW of power and 21Nm of torque. Each motor has a self-developed three-stage planetary gear transmission with fixed velocity reduction ratio of 14.92, which allows each wheel to deliver to the ground up to 313 Nm of torque for each wheel. The motors are driven by their manufacturer inverters, each one controlled independently, and the energy is provided by a self-developed High Voltage battery pack of about 500V and 2.5kWh of storable energy. The total output power of the battery is limited by Formula SAE regulations to 80kW, which also limits the actual power delivered by the motors, but a well-designed torque vectoring (TV) is able to exploit all the 80kW optimally.
To guarantee autonomous capabilities, the steering actuator controls the full steering range of the system, and together with the braking actuator, also providing emergency braking functionalities, the VaLentina prototype has the full control over lateral and longitudinal dynamic.
The environmental perception is made by acquiring data from a 64 channels Ouster LiDAR and 2 ALVIUM Allied Vision cameras, which through a custom developed sensor fusion algorithm gives the prototype a 80° horizontal field of view with a cones identification confidence distance of 25 meters, due to the small shape of the cones to be identified.

The Odometry information, which derives from the integration of a Fuzzy logic

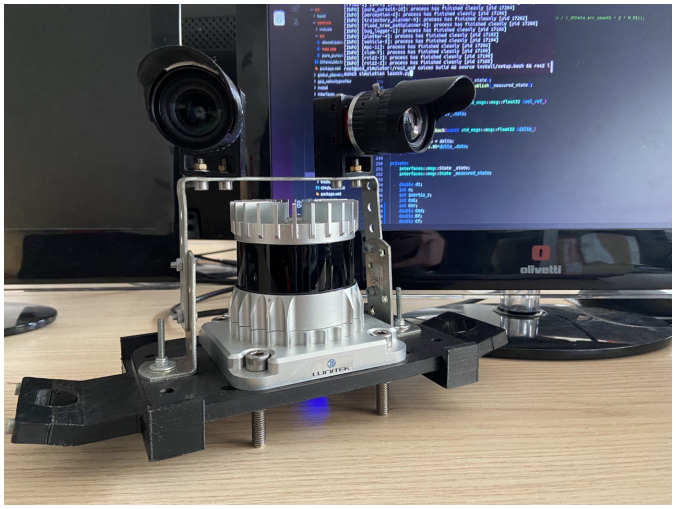

Figure 1.4: Lidar and Camera system of the VaLentina prototype.

algorithm for velocity estimation information ([2]) and an EKF for pose estimation (both using data coming from sensors such as IMU, wheel encoders and steering encoders), is fused with a self developed SLAM algorithm that provides optimized state measures.
The high level stack of softwares, starting from the environmental perception, through SLAM and high level control, runs on a NVIDIA Jetson AGX Orin computer, which is referred s the onboard autonomous control unit (ACU). The low level controller (LLC), such as the steering control, braking control and torque vectoring (TV) runs on a dSPACE MicroAutoBox III Real Time hardware, commonly referred as the prototype electronic control unit (ECU), which also implements CANbus communication with all the boards and subsystem of the prototype. The two Control Units communicate via CANbus protocol too using a serial-to-CAN converter Kvaser Leaf Light v2.

1.2.3 Use of the MPC algorithm

The VaLentina prototype developed by the team, after two season of development, has now become a test bench for autonomous driving solutions testing, since the Control Units installed are highly flexible for prototyping. In particular, the ACU runs all the different algorithms in a ROS2 workspace, which is composed by nodes communicating with each other using the native publisher-subscriber protocols of ROS2. This means that all the informations and data can be made available to be elaborated from each individual node, and the results published as topics made available to other nodes.

For the Formula SAE Driverless events described in 1.1, the track is marked by small yellow and blue cones, that are recognized by the sensors. When the circuit is not known, the VaLentina prototype runs a simple steering controller, which tries to follow the local path generated by a self developed rapid-exploring random tree (RRT) algorithm ([3]), which allowed the prototype to move freely in an unknown circuit since its first meters. In Driverless events like Trackdrive, where only the first lap is unknown, it is possible to develop optimised control strategies starting from the second lap on to achieve better performances. One of the most used control algorithms, both in FSAE and in industrial applications, is without doubt the model predictive controller (MPC), that has proved to be well suited for the objective of achieving performances over a known circuit, since it can account for system constraints directly in the computation of the optimal control input.
This thesis work has the objective to investigate the MPC as a solution to the problem of path tracking, with particular focus on its real time implementation, trying to achieve the best trade-off between computational effort and performance of the controller. The MPC algorithm is included in the full autonomous system as represented in Figure 1.5. In order to make the controller easy to implement on the real prototype, both software in the loop (SiL) and hardware in the loop (HiL) validation is performed.

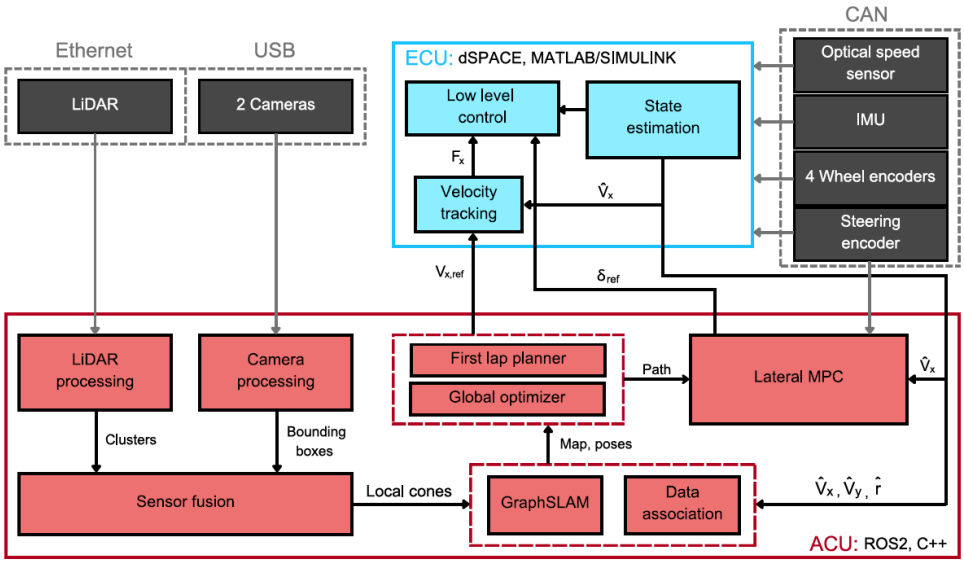

Figure 1.5: Autonomous System overview including MPC.

Chapter 2

Vehicle Modeling

The first step in designing any controller algorithm is to study the dynamic of the system to be controlled and build a proper representation of it, called plant, in a simulation environment. The objective is to represent the most accurately possible the system to be controlled, that can be later used to evaluate stability and performance of the designed controller. This step is also fundamental in order to properly study open loop stability of the system, evaluate limits and constraints, and uncertainties that can affect the real system.

2.1 Reference frames and transformations

The system to be controlled is a 4WD electric prototype race-car based on the Formula SAE regulations, equipped with four in-wheel electric motors that can be controlled individually allowing torque vectoring implementation. Moreover it's equipped with steering and braking actuators to allow it to move in full autonomous mode without any driver action. In order to fully describe the dynamics of the system, a total of 6 reference frame (RF) representations are needed: the fixed inertial frame RF^0, a mobile reference frame RF^1 that is integral with the car and four reference systems RF^w that are integral with each wheel.
Starting with RF^1, it is based on the ISO 8855-2011 standard where its origin follows the center of gravity (CoG) of the vehicle, the x-axis is called longitudinal axis and is directed along the centerline of the vehicle, the z-axis is perpendicular to the ground when the car is steady and pointing upward, and the y-axis describes a right-handed orthogonal reference system with the two axis described above.
The fixed reference frame RF^0 follows the same standard as RF^1, but is placed in the starting position of the vehicle and does not move.
Finally, the four reference systems $RF_i^w, i = fl, fr, rl, rr$ have their origin placed in

each wheel's contact point at the ground, defined by the ISO 8855-2011 standard, represented in Figure 2.1.

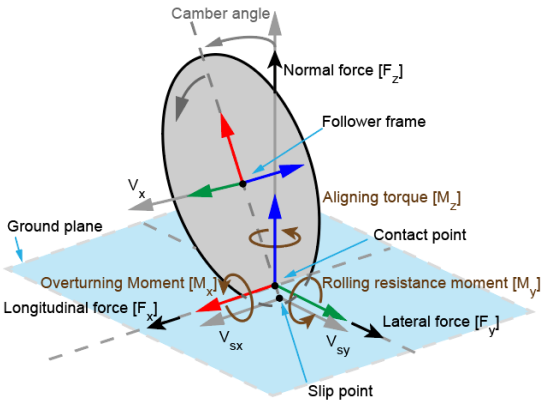

Figure 2.1: R_w reference frame following ISO sign convention.

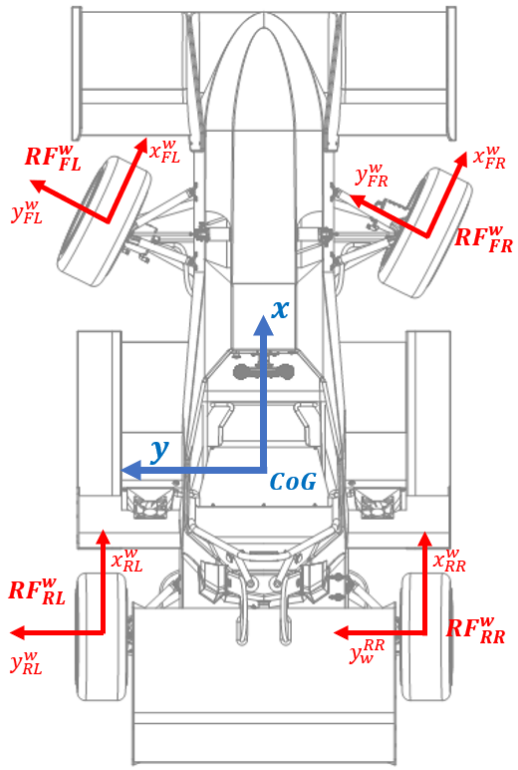

Figure 2.2: Reference Frames: RF^1 and RF^w.

It is convenient to evaluate the transformation matrices that are used when a

change of reference system is needed. In particular, it will be useful later on to change coordinates between RF^0 and RF^1.
Based on the Formula Student prototype application on which this thesis work is focusing, it can be assumed that the movement only occurs along the XY plane, thus neglecting any movement along the Z-axis. Moreover, later on it will be described a method to address roll and pith dynamics of the vehicle chassis, without taking into consideration actual rotations around the vehicle's x-axis and y-axis. Thanks to this assumptions, only three degrees of freedom are necessary to fully describe the chassis motion and its reference frame RF^1, and the Yaw Angle ψ is the only one needed to describe the different orientation between RF^0 e RF^1. Moreover, the Yaw Rate r is defined, as the rate of change of the angle ψ at the time instant k:

$$\begin{aligned} r(t) &= \dot{\psi}(t), \\ \psi_k &= \psi_0 + \int_0^k r(t)dt \end{aligned} \tag{2.1}$$

At a given time instant k, the reference frame RF^1 can now be described as the result of a rotation around its z-axis of the angle ψ_k, described by the rotation matrix

$$\mathbf{R^0_{1,k}} = \begin{bmatrix} \cos\psi_k & -\sin\psi_k & 0 \\ \sin\psi_k & \cos\psi_k & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

and a rigid translation of the vector:

$$\mathbf{t^0_{01}} = \begin{bmatrix} X_k & Y_k & 0 \end{bmatrix}^T \tag{2.3}$$

that expresses the distance vector between the two frame's origins described in RF^0, so the position of the CoG of the vehicle at the time instant k expressed in the fixed frame.
At a given time instant k, a point in the 3D space P can be expressed as a set of coordinates in both reference frames, where for the inertial frame RF^0, upper case letters X, Y, Z will be used, while for the moving frame RF^1 lower-case letters x, y, z will be used instead:

$$\begin{aligned} P^0_k &= \begin{bmatrix} X_{P,k} & Y_{P,k} & Z_{P,k} \end{bmatrix} \\ P^1_k &= \begin{bmatrix} x_{P,k} & y_{P,k} & z_{P,k} \end{bmatrix} \end{aligned} \tag{2.4}$$

It can be found that the following holds:

$$\begin{bmatrix} X_{P,k} \\ Y_{P,k} \\ Z_{P,k} \\ 1 \end{bmatrix} = \mathbf{t^0_{01}} + \mathbf{R^0_{1,k}} \cdot \begin{bmatrix} x_{P,k} \\ y_{P,k} \\ z_{P,k} \\ 1 \end{bmatrix} \tag{2.5}$$

and can be rewritten as

$$\begin{bmatrix} X_{P,k} \\ Y_{P,k} \\ Z_{P,k} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R^0_{1,k}} & \mathbf{t^0_{01}} \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} x_{P,k} \\ y_{P,k} \\ z_{P,k} \\ 1 \end{bmatrix} = \mathbf{T^0_{1,k}} \cdot \begin{bmatrix} x_{P,k} \\ y_{P,k} \\ z_{P,k} \\ 1 \end{bmatrix} \tag{2.6}$$

where, by making explicit all the components, the transformation matrix can be obtained:

$$\mathbf{T^0_{1,k}} = \begin{bmatrix} \cos\psi_k & -\sin\psi_k & 0 & X_k \\ \sin\psi_k & \cos\psi_k & 0 & Y_k \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

The transformation matrix $\mathbf{T^0_{1,k}}$ obtained describes how to compute the coordinates of a point in RF^0 by using its coordinates in RF^1.
Similarly, the inverse transformation matrix can be derived, used to move from the inertial frame RF^0 to the local frame RF^1:

$$\begin{bmatrix} x_{P,k} \\ y_{P,k} \\ z_{P,k} \\ 1 \end{bmatrix} = \mathbf{T^1_{0,k}} * \begin{bmatrix} X_{P,k} \\ Y_{P,k} \\ Z_{P,k} \\ 1 \end{bmatrix} \tag{2.8}$$

$$\mathbf{T^1_{0,k}} = \begin{bmatrix} \cos\psi_k & \sin\psi_k & 0 & -X_k\cos\psi_k - Y_k\sin\psi_k \\ -\sin\psi_k & \cos\psi_k & 0 & +X_k\sin\psi_k - Y_k\cos\psi_k \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

Note that this transformation matrices, used for points in the space, can also be applied to vectors, that can always be described by their components in a given reference frame. When dealing with vectors transformation between reference frames, usually only their direction and magnitude are of interest and not their point of application, so when passing from a reference frame representation to another the vectors are rigidly moved to the actual frame's origin, that translates into applying the following:

$$\begin{bmatrix} v^0_x \\ v^0_y \\ v^0_z \\ 1 \end{bmatrix} = \mathbf{T^0_1} * \begin{bmatrix} v^1_x \\ v^1_y \\ v^1_z \\ 1 \end{bmatrix} - \mathbf{t^0_{01}} \tag{2.10}$$

which, with some simple manipulation, can also be written in the following simplified relation:

$$\begin{bmatrix} v^0_x \\ v^0_y \\ v^0_z \end{bmatrix} = \mathbf{R^0_1} * \begin{bmatrix} v^1_x \\ v^1_y \\ v^1_z \end{bmatrix} \tag{2.11}$$

2.2 Vehicle Dynamics Equations

2.2.1 Wheel dynamics

In order to evaluate the dynamic of the whole vehicle, it is convenient to divide it into subsystems and analyse them individually, making explicit forces and torques exchanged at the interface with other systems.
Starting from the wheel subsystem, which reference frame is shown in Figure 2.1, it is subject to:

- Force exchanged with the ground, with components along the three directions of the frame
- Motor torque T_{motor}
- Braking torque T_{brake}
- Rolling resistance moment $M_{y,w}$
- Force exchanged at the interface with the vehicle
- Inertia

Some assumptions and simplifications are made in order to simplify the analysis:

- Wheel-rim, tire, motor and transmission are considered as a unique rotating object with mass m_w and equivalent moment of inertia $J_{y,w}$, which has been evaluated in the design phase of the fixed ratio transmission gear
- The force exchanged with the vehicle are considered as acting at the center of mass of the wheel subsystem: this implies null overturning moment along the wheel's longitudinal axis due to the attachment position of the suspension system, resulting in a static camber during motion
- The self aligning moment is not influencing the steering actuation: this can be considered valid since the low level steering controller compensates for it
- Motor torque and braking torque are considered as a unique torque acting on the system: this hypothesis is not limiting the analysis since the low level torque vectoring translates braking torque into suitable combination of braking action and regenerative motor torque

Four equilibrium equations can be derived for the wheel subsystem:

$$\begin{cases} x : F_{x,w} - F_{x,car} = m_w \ddot{x} \\ y : F_{y,w} - F_{y,car} = m_w \ddot{y} \\ z : F_{z,w} - F_{z,car} - m_w g = m_w \ddot{z} \\ \omega : T_{in} \cdot \tau - M_{y,w} - F_{x,w} \cdot R_{roll} = J_{y,w} \dot{\omega}_w \end{cases} \tag{2.12}$$

Where T_{in} is the input torque as output of the motor, so it is multiplied by the fixed transmission ratio τ, and R_{roll} is the rolling radius. The linear influence of the inertia is considered null, due to the low mass of the wheel subsystem and its suspension constraints along the x and y local directions, together with an ideal flat surface assumed for the track. As a result, the following set of equations are obtained:

$$\begin{cases} F_{x,car} = F_{x,w} \\ F_{y,car} = F_{y,w} \\ F_{z,w} = F_{z,car} + m_w g \\ \dot{\omega}_w = \frac{T_{in} \cdot \tau - M_{y,w} - F_{x,w} \cdot R_{roll}}{J_{y,w}} \end{cases} \tag{2.13}$$

Where the unknowns are $F_{x,w}$, $F_{y,w}$, $F_{z,car}$ and $M_{y,w}$. While $F_{z,car}$ depends on the chassis dynamics and the suspension effects during the motion, the other unknowns derive from the interaction between pneumatic and ground. The pneumatic characterisation and its dynamics have been at the center of the scientific literature for years. While simplified models are easier to understand, although they do not consider non-linearities and transient behaviours, more complex models have been developed: the "brush model" and the "Pacejka model". The first describes in a rigorous analytic way the forces generated by the pneumatic-ground interaction, meeting solid experimental data validation especially in non-transient behaviours [4]. The Pacejka model instead is an empirical-derived model, which uses a parametric expression, also known as "magic formula" [5], to evaluate the pneumatic-ground forces:

$$F_i = D_i \sin(C_i \arctan((1 - E_i) B_i s_i + E_i \arctan(B_i s_i))) + V_i \tag{2.14}$$

where B_i is called stiffness factor, C_i shape factor, D_i peak value, E_i curvature factor, V_i is the vertical offset, and finally s_i is the slip, and is function of the velocities of the wheel: in particular, for the longitudinal force, the slip is known as longitudinal slip, is a-dimensional and is evaluated as:

$$s_x = \sigma = \frac{\omega \cdot R_{roll} - v_{x,w}}{v_{x,w}} \tag{2.15}$$

while for the lateral force the slip is known as side-slip angle, has the dimensions of an angle (radians) and is evaluated as:

$$s_y = \alpha = \arctan\left(\frac{v_{y,w}}{v_{x,w}}\right) \tag{2.16}$$

The expression 2.14 can take into account for different operative conditions, like camber, vertical load, but is used only in steady state condition. The complete formulation takes into account also for transient and non-linear behaviours, and can be find in Pacejka's book [5] and different published articles, like [6].

2.2.2 Chassis dynamics

The forces developed at the tyre-ground interface are transmitted to the vehicle chassis, following 2.13. This forces act at the same time on the chassis, but in different points of the vehicle, in particular on the suspended mass: in order to evaluate the dynamic of the vehicle, the vehicle's chassis is considered a rigid body, that moves in the reference frame RF^0, considering only its planar motion in the plane XY. In this way, the chassis has a total of only 3 degrees of freedom, that are the linear motion along the X-axis and the Y-axis, and the rotation around the vehicle's z-axis, as discussed in 2.1. This 2D motion is caused by the action of two different kind of forces:

- Wheel forces transmitted to the chassis $F^i_{x,car}$ and $F^i_{y,car}$, where i is represents one of the four wheels, $i = FL, FR, RL, RR$
- Aerodynamic force, which is acting on the center of pressure (CoP), and in general has a longitudinal and a vertical components:

$$\begin{aligned} F_{x,aero} &= \tfrac{1}{2}\rho S c_x v_x^2 \\ F_{z,aero} &= \tfrac{1}{2}\rho S c_z v_x^2 \end{aligned} \tag{2.17}$$

  Where ρ is the air density, S is the effective surface of the aerodynamic devices, c_x is called drag coefficient, and v_x is the longitudinal velocity.

The air density is considered constant and equal to $1.18\frac{kg}{m^3}$, that is the average air density in Turin, while the surface and the drag coefficient have been previously evaluated by the team via CFD simulation, being $S = 1m^2$ and $c_x = 3.2$ for the VaLentina prototype. The forces contribution can be applied to the vehicle model, represented in Figure 2.3, where the geometric parameters and the steering geometry are taken into account.

Separating the longitudinal and lateral components of the force vectors, the following system of three equilibrium equations can be derived:

$$\begin{cases} ma_x &= F_x^{FR}\cos\delta^{FR} + F_x^{FL}\cos\delta^{FL} - F_y^{FR}\sin\delta^{FR} - F_y^{FL}\sin\delta^{FL} + \\ &\quad + F_x^{RR} + F_x^{RL} + F_{x,aero} \\ ma_y &= F_x^{FR}\sin\delta^{FR} + F_x^{FL}\sin\delta^{FL} + F_y^{FR}\cos\delta^{FR} + F_y^{FL}\cos\delta^{FL} + \\ &\quad + F_y^{RR} + F_y^{RL} \\ I_z r &= F_x^{FR}\left(a\sin\delta^{FR} + \tfrac{t}{2}\cos\delta^{FR}\right) + F_x^{FL}\left(a\sin\delta^{FL} - \tfrac{t}{2}\cos\delta^{FL}\right) + \\ &\quad + F_y^{FR}\left(a\cos\delta^{FR} + \tfrac{t}{2}\sin\delta^{FR}\right) + F_y^{FL}\left(a\cos\delta^{FL} - \tfrac{t}{2}\sin\delta^{FL}\right) + \\ &\quad + \tfrac{t}{2}F_x^{RR} - \tfrac{t}{2}F_x^{RL} - bF_y^{RR} - bF_y^{RL} \end{cases} \tag{2.18}$$

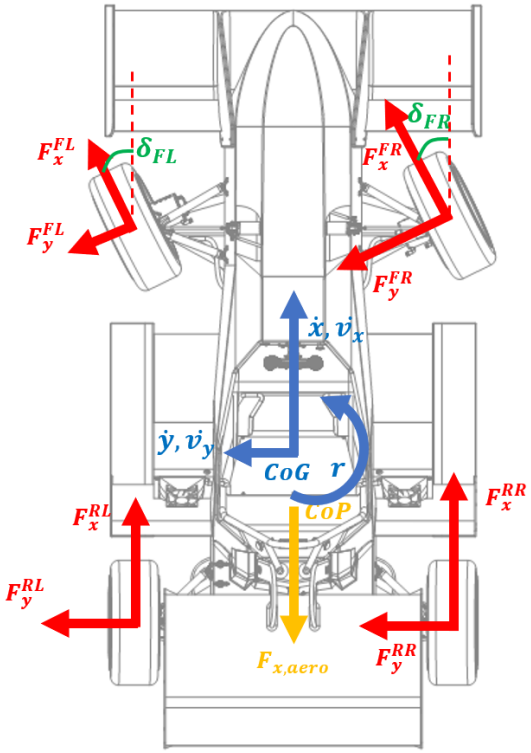

Figure 2.3: Forces acting on the vehicle, overhead view.

Regarding the acceleration terms, they are the components of the acceleration vector $\mathbf{a_{CoG}}$ evaluated as derivative of the velocity vector $\mathbf{v_{CoG}}$. Since the reference frame of these two vectors in a moving reference frame, the following is applied:

$$\mathbf{a_{CoG}} = \frac{d\mathbf{v_{CoG}}}{dt} = \frac{d(v_x\mathbf{i})}{dt} + \frac{d(v_y\mathbf{j})}{dt} = \frac{dv_x}{dt}\mathbf{i} + v_x\frac{d\mathbf{i}}{dt} + \frac{dv_y}{dt}\mathbf{j} + v_y\frac{d\mathbf{j}}{dt} \tag{2.19}$$

$$\mathbf{a_{CoG}} = \frac{d\mathbf{v_{CoG}}}{dt} = \dot{v}_x\mathbf{i} + v_x r\mathbf{j} + \dot{v}_y\mathbf{j} - v_y r\mathbf{i} = (\dot{v}_x - v_y r)\mathbf{i} + (\dot{v}_y + v_x r)\mathbf{j} \tag{2.20}$$

$$\mathbf{a_{CoG}} = \begin{bmatrix} a_x \\ a_y \end{bmatrix} = \begin{bmatrix} \dot{v}_x - v_y r \\ \dot{v}_y + v_x r \end{bmatrix} \tag{2.21}$$

Finally, the system of equation 2.18 can be rewritten as:

$$\begin{cases} \dot{v_x} &= \frac{1}{m}\Big[F_x^{FR}\cos\delta^{FR} + F_x^{FL}\cos\delta^{FL} - F_y^{FR}\sin\delta^{FR} - F_y^{FL}\sin\delta^{FL} + \\ &\quad + F_x^{RR} + F_x^{RL} + F_{x,aero}\Big] + v_y r \\ \\ \dot{v_y} &= \frac{1}{m}\Big[F_x^{FR}\sin\delta^{FR} + F_x^{FL}\sin\delta^{FL} + F_y^{FR}\cos\delta^{FR} + F_y^{FL}\cos\delta^{FL} + \\ &\quad + F_y^{RR} + F_y^{RL}\Big] - v_x r \\ \\ \dot{r} &= \frac{1}{m}\Big[F_x^{FR}\left(a\sin\delta^{FR} + \frac{t}{2}\cos\delta^{FR}\right) + F_x^{FL}\left(a\sin\delta^{FL} - \frac{t}{2}\cos\delta^{FL}\right) + \\ &\quad + F_y^{FR}\left(a\cos\delta^{FR} + \frac{t}{2}\sin\delta^{FR}\right) + F_y^{FL}\left(a\cos\delta^{FL} - \frac{t}{2}\sin\delta^{FL}\right) + \\ &\quad + \frac{t}{2}F_x^{RR} - \frac{t}{2}F_x^{RL} - bF_y^{RR} - bF_y^{RL}\Big] \end{cases} \tag{2.22}$$

These quantities can be numerically integrated in the simulation environment, obtaining in this way quantities v_x, v_y and r. From these quantities, that fully describe the motion of the mobile reference frame RF^1 and of the CoG of the vehicle, also the linear velocity of each wheel's reference frame can be obtained by simple geometric considerations:

$$\begin{cases} v_x^{FR} &= v_x\cos\delta^{FR} + v_y\sin\delta^{FR} + r\left(a\sin\delta^{FR} + \frac{t}{2}\cos\delta^{FR}\right) \\ v_y^{FR} &= v_y\cos\delta^{FR} - v_x\sin\delta^{FR} + r\left(a\cos\delta^{FR} - \frac{t}{2}\sin\delta^{FR}\right) \\ \\ v_x^{FL} &= v_x\cos\delta^{FL} + v_y\sin\delta^{FL} + r\left(a\sin\delta^{FL} - \frac{t}{2}\cos\delta^{FL}\right) \\ v_y^{FL} &= v_y\cos\delta^{FL} - v_x\sin\delta^{FL} + r\left(a\cos\delta^{FL} + \frac{t}{2}\sin\delta^{FL}\right) \\ \\ v_x^{RR} &= v_x + r\cdot\frac{t}{2} \\ v_y^{RR} &= v_y - r\cdot b \\ \\ v_x^{RL} &= v_x - r\cdot\frac{t}{2} \\ v_y^{RL} &= v_y - r\cdot b \end{cases} \tag{2.23}$$

This velocities are then passed to each wheel subsystem, so that the slips can be evaluated. The last unknown is the vertical load acting on each wheel, that in the Pacejka model has a great influence on the evaluation of the force developed at the ground contact point. The vertical load is deeply influenced by the presence of the suspension system. In order to fully describe them, two more degrees of freedom should be added to the model chassis about the roll and pitch motion. Since this would deeply complicate the model analysis, a simplified approach has been followed.

The load acting on each wheel is influenced by four components:

- The static load due to the mass of the vehicle when standing still;
- The longitudinal load transfer due to the presence of longitudinal acceleration;
- The lateral load transfer due to the presence of longitudinal acceleration;
- The aerodynamic devices presence, which contribution can be divided into a longitudinal component with coefficient c_x and a vertical one with coefficient c_z (2.17.

For the lateral load transfer, it is possible to easily take into account for the presence of the suspension system without including the suspension kinematic: following the methods presented in [7] , the roll stiffness distribution and roll axis position have been evaluated by the team Squadra Corse, and parameters like the front and rear roll stiffness $k_{\phi,F}$ and $k_{\phi,R}$, and the height of the roll axis $z_{\phi,F}$ and $z_{\phi,F}$ have been obtained. Now the lateral load transfer due to the lateral acceleration can be evaluated, as presented in [8]:

$$\begin{aligned} \Delta F^y_{z,F} &= \frac{m}{t}\left(\frac{k_{\phi,F}}{k_{\phi,F}+k_{\phi,R}} d_{CoG} + \frac{b}{L} z_{\phi,F}\right) a_y \\ \Delta F^y_{z,R} &= \frac{m}{t}\left(\frac{k_{\phi,F}}{k_{\phi,R}+k_{\phi,R}} d_{CoG} + \frac{a}{L} z_{\phi,R}\right) a_y \end{aligned} \tag{2.24}$$

where $d_{CoG} = h_{CoG} - z_{CoG}$ is the vertical distance between the center of gravity and the roll axis, as shown in Figure 2.4

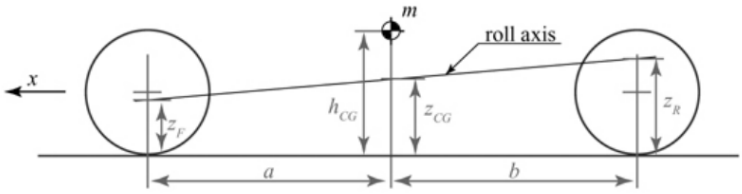

Figure 2.4: Ride Roll distances.

For what regards the longitudinal load transfer, the suspension effect is neglected, obtaining:

$$\begin{aligned} \Delta F^x_{z,F} &= -\frac{m h_{CoG}}{L} |a_x| \\ \Delta F^x_{z,R} &= +\frac{m h_{CoG}}{L} |a_x| \end{aligned} \tag{2.25}$$

For what concerns the aerodynamic effect on the vertical load, the aerodynamic force is considered as acting in the CoP following 2.17. Summing up, the load distribution between the four wheels can be evaluated as:

$$\begin{cases} F_z^{FR} &= mg\frac{b}{L} - \Delta F_{z,F}^x + \Delta F_{z,F}^y + F_{z,aero}\frac{b_{CoP}}{L} - F_{x,aero}\frac{h_{CoP}}{L} \\ F_z^{FL} &= mg\frac{b}{L} - \Delta F_{z,F}^x - \Delta F_{z,F}^y + F_{z,aero}\frac{b_{CoP}}{L} - F_{x,aero}\frac{h_{CoP}}{L} \\ F_z^{RR} &= mg\frac{a}{L} + \Delta F_{z,R}^x + \Delta F_{z,R}^y + F_{z,aero}\frac{a_{CoP}}{L} + F_{x,aero}\frac{h_{CoP}}{L} \\ F_z^{RL} &= mg\frac{a}{L} + \Delta F_{z,R}^x - \Delta F_{z,R}^y + F_{z,aero}\frac{a_{CoP}}{L} + F_{x,aero}\frac{h_{CoP}}{L} \end{cases} \tag{2.26}$$

2.3 Simulation Environment

Considering the complexity of the complete tyre model analysied in 2.2.1, being the simulation model developed with MATLAB® Simulink, the MFeval library can be used [9]: using a .tir file that describes the tyre behaviour and constitutive parameters, it is possible to evaluate forces and moments acting on the wheel.

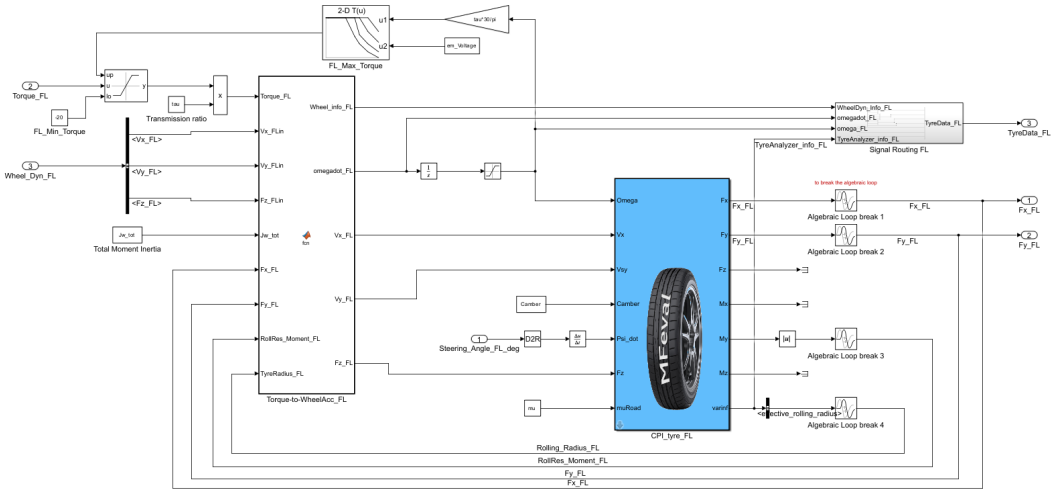

Figure 2.5: Wheel subsystem simulation model developed on MATLAB® Simulink.

The developed tyre subsystem is able to evaluate the forces $F_{x,w}$, $F_{y,w}$ and the rolling resistance moment $M_{y,w}$ of 2.13, starting from the input motor torque and the wheel's hub velocity, which is derived from the chassis overall dynamics. A look-up table for torque limits evaluation has been added, following the motor's map provided by the manufacturer. Moreover, all the signals are collected into a signal routing block, so that all the tyre info are also available outside the model for scope purposes.
The tyre forces are considered as acting on the car following 2.13, which effect on the chassis motion can be evaluated with the dynamic equations 2.22. The chassis

model is developed, as in Figure 2.6 starting from the input force contributions. The wheel's hub velocities 2.23 and the vertical load on each wheel 2.26 are then fedback to each wheel subsystem. Other useful signals are routed outside the model in order to obtain easy access to the chassis dynamics, similar to what has been done with each wheel.

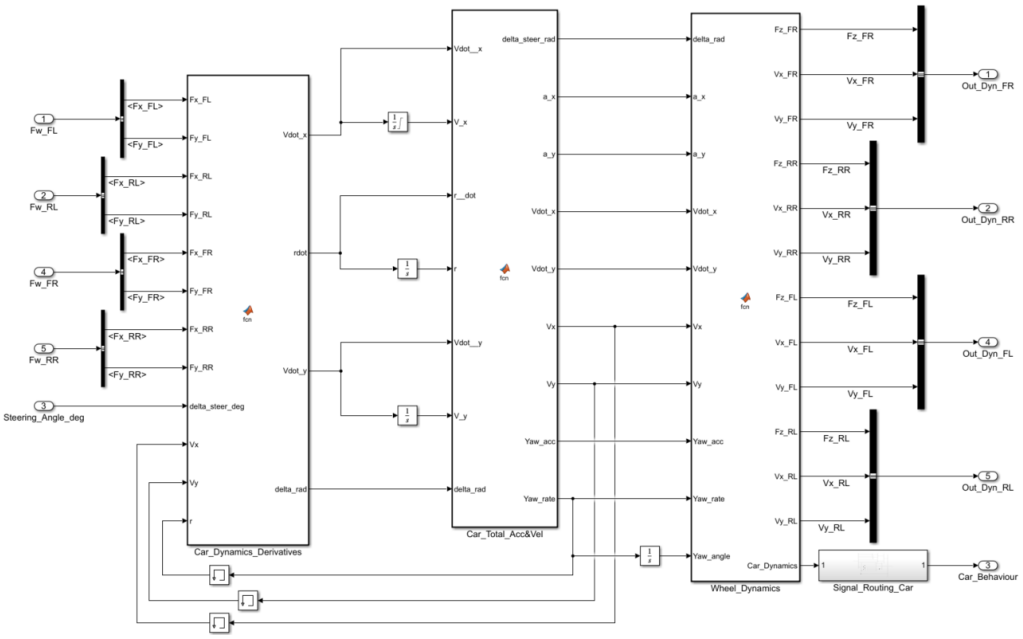

Figure 2.6: Chassis subsystem simulation model developed on MATLAB® Simulink.

The overall simulation model can be obtained, as shown in Figure 2.7. In order to make the model easily portable and tunable with experimental data, a mask has been created, that initialises all the constants and parameters used in the model.

Of course, even with this level of complexity of the model, parameter uncertainties and unmodelled behaviours affect the accuracy of the model. Thanks to track-test data collected by the team Squadra Corse Driverless, it is possible to perform some parameters correction, in order to fit the experimental data and make the simulation model more accurate. In the real time application, two of the vehicle states can be directly measured or estimated: in particular, while the Yaw Rate r can be directly measured by means of the IMU sensor installed, the longitudinal velocity v_x is estimated in real time thanks to a Fuzzy Logic estimator [2]. Since the odometry information of the vehicle is derived by these two quantities, in order to validate and tune the developed model a comparison between the actual track-test data and the two states is addressed. The inputs of the model during the simulation

are the data registered during the track tests of the the TV controller and the steering controller outputs: in this way, it is possible to take into account for control input delays, unmodelled behaviours or disturbances of the real system directly inside the model, fitting the experimental data by tuning global parameters. This procedure allows to tune the model parameters up until obtaining a sufficiently low error, and in future simulations it will be possible to directly attach the low level controllers directly to the developed model.
The results of the fitting procedure are shown in Figure 2.8: in particular, the longitudinal velocity reached an absolute maximum error of $0.1 m/s$ over $5 m/s$ velocity, meaning a relative error of about 2% between the developed model and the actual real time measurement; the Yaw Rate shows a maximum error of $0.05 rad/s$, but a relative error can not be evaluated, since when the Yaw Rare goes to zero the relative error would go to infinity. In order to evaluate a performance parameter for the Yaw Rate accuracy, the integral value of the error can be observed: over a complete circuit lap, the total error is of about 7.45 deg, which corresponds to a 2% relative error of the Yaw Angle evaluation.

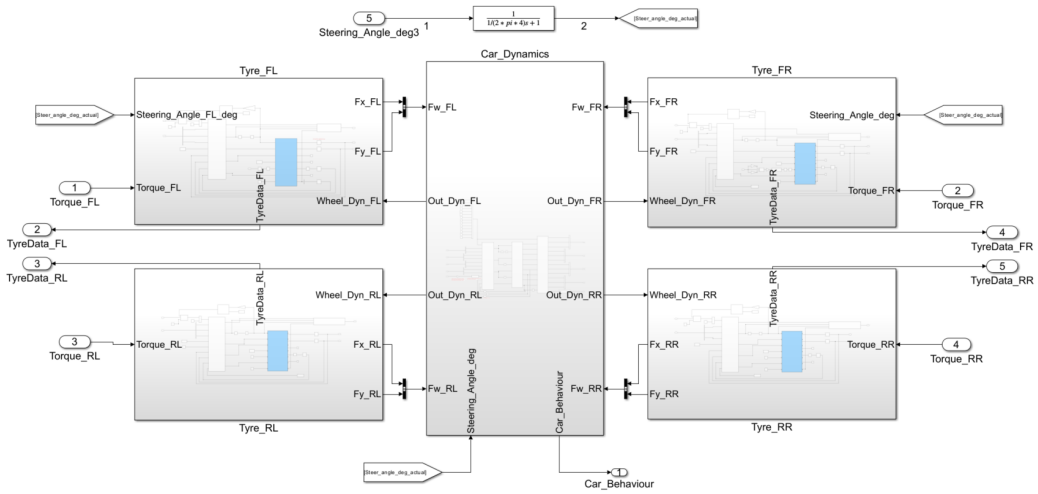

Figure 2.7: Complete simulation model developed on MATLAB® Simulink.

2.4 Prediction model

As discussed in the outline of this thesis work, in order to develop a proper MPC, a prediction model must be selected. Following similar works in the Formula SAE environment, a single-track bicycle model has been evaluated as a possible candidate. In the Formula Student Driverless competitions, the single-track bicycle model has proved to be a good trade-off between simplicity, that helps keeping

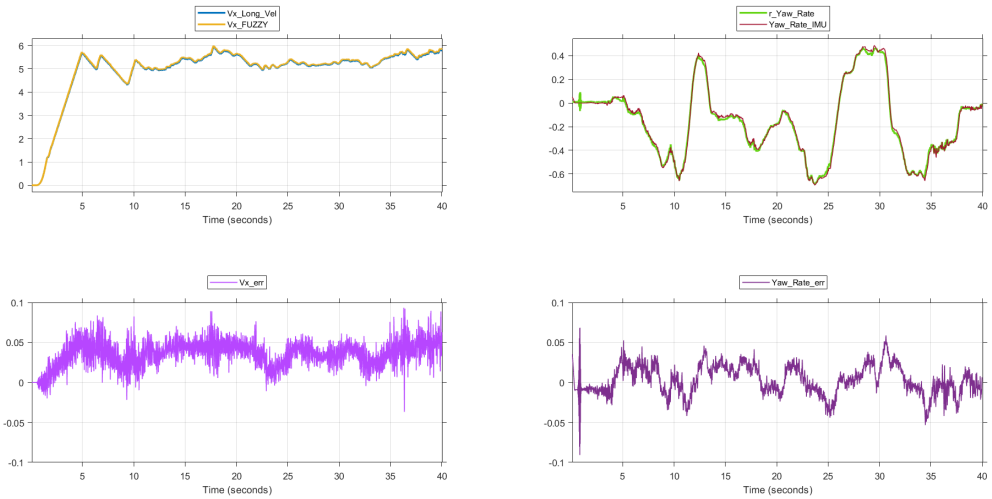

Figure 2.8: Data comparison between the simulation model and track-tests data.

the computational effort of the MPC less heavy, and accuracy of the predicted dynamic. In the particular case of the Squadra Corse Driverless team, this model finds greater accuracy with respect to similar automotive applications, since the low level TV already deployed in the prototype car adopts as reference dynamic the single-track bicycle model one.
A detailed description of the single-track model is well-documented in literature: in particular, in this work the book [10] as been taken as reference.

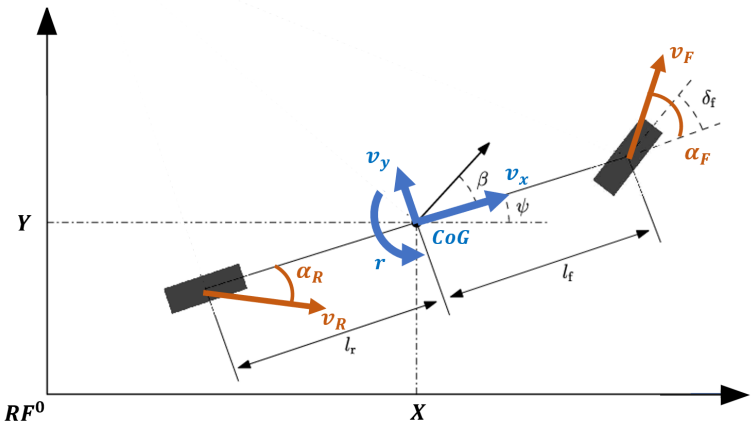

Figure 2.9: Bicycle Model representation in RF^0.

2.4.1 Kinematic equations

The single-track model adopted for this work considers only front-wheel steering and combined longitudinal and lateral dynamic. The vehicle's kinematic is described by the motion of its center of gravity (CoG), in which the reference frame RF^1, described in 2.1, is placed. This frame moves in the 3D space with velocity $\mathbf{v}_{CoG}$ that has coordinates (v_x, v_y, v_z) expressed in the local reference frame RF^1, and the controller must ensure its tracking of a reference trajectory, that is expressed as the set of coordinates $[(X^{ref}, Y^{ref})]$ in the global reference system RF^0.
Based on the Formula Student prototype application on which this thesis work is focusing as described in section 1.1, the first assumption is that the vehicle moves in a planar 2D environment, thus neglecting its movement along the z-axis.
Called (X_k, Y_k) the global coordinates of the origin of the reference frame RF^1 at time instant k, which moves with velocity $v_{CoG} = (v_x, v_y, 0)$, and being ψ_k the angle between the two reference systems x-axis above described, the rate of change of (X_k, Y_k) expresses the velocity vector of the vehicle expressed in the inertial frame. This can be obtained by applying 2.11 to $\mathbf{v}_{CoG}$, so the following set of equations that generally describe the kinematic of the vehicle expressed in RF^0 can be obtained:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = \begin{bmatrix} v_x \cos\psi - v_y \sin\psi \\ v_x \sin\psi + v_y \cos\psi \end{bmatrix} \tag{2.27}$$

2.4.2 Dynamic equations

Next step is to derive the dynamic equations that describe the laws of motion of the single-track bicycle model. Since the vehicle is considered to move in the 2D space, only three equations are needed to fully describe the dynamical system: two forces equilibrium equations along the longitudinal x-axis and lateral y-axis, and one moment equilibrium equation to describe the yaw dynamic around the z-axis. The forces and moments that are considered to be acting on the bicycle model are:

- Longitudinal and lateral tire forces exchanged with the ground, as explained in chapter 2.2
- Drag forces due to the presence of air
- An external moment, called τ_{TV}, that takes into account any additional moment caused by the low-level torque vectoring algorithm

Other forces, like vertical load, rolling resistance and downforce due to the aerodynamic devices are not considered in this analysis.

Starting from Figure 2.10 where all the forces above mentioned have been made explicit, the equilibrium equations can be easily derived:

$$ma_x = F_{x,f}\cos\delta_f - F_{y,f}\sin\delta_f + F_{x,r} - F_{x,aero} \tag{2.28}$$

$$ma_y = F_{x,f}\sin\delta_f + F_{y,f}\cos\delta_f + F_{y,r} \tag{2.29}$$

$$I_z\dot{r} = F_{x,f}l_f\sin\delta_f + F_{y,f}l_f\cos\delta_f - F_{y,r}l_r + \tau_{TV} \tag{2.30}$$

By including 2.21, and solving for $\dot{v_x}$, $\dot{v_y}$ and $\dot{r}$, the following system of equations can be obtained:

$$\begin{bmatrix}\dot{v_x}\\ \dot{v_y}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}\frac{1}{m}(F_{x,f}\cos\delta - F_{y,f}sin\delta + F_{x,r} - F_{x,aero} + mv_yr)\\ \frac{1}{m}(F_{x,f}\sin\delta + F_{y,f}cos\delta + F_{y,r} - mv_xr)\\ \frac{1}{I_z}(F_{x,f}l_f\sin\delta + F_{y,f}l_f\cos\delta - F_{y,r}l_r + \tau_{TV})\end{bmatrix} \tag{2.31}$$

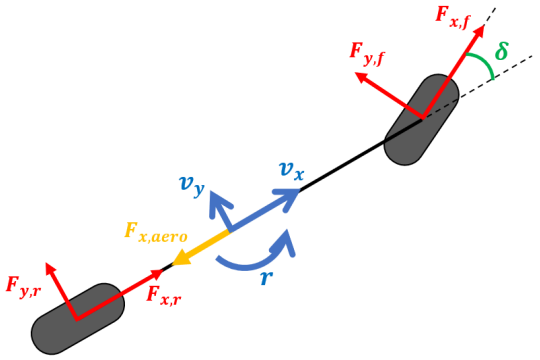

Figure 2.10: Bicycle model with acting forces.

2.4.3 Linearized model

By putting together 2.27, 2.1 and 2.31, a single system of equation describing the single-track bicycle model's kinematic and dynamic can be obtained:

$$\begin{bmatrix}\dot{X}\\ \dot{Y}\\ \dot{\psi}\\ \dot{v_x}\\ \dot{v_y}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}v_x\cos\psi - v_y\sin\psi\\ v_x\sin\psi + v_y\cos\psi\\ r\\ \frac{1}{m}(F_{x,f}\cos\delta - F_{y,f}\sin\delta + F_{x,r} - F_{x,aero} + mv_yr)\\ \frac{1}{m}(F_{x,f}\sin\delta + F_{y,f}\cos\delta + F_{y,r} - mv_xr)\\ \frac{1}{I_z}(F_{x,f}l_f\sin\delta + F_{y,f}l_f\cos\delta - F_{y,r}l_r + \tau_{TV})\end{bmatrix} \tag{2.32}$$

where m and I_z are mass and yaw inertia of the vehicle, l_f and l_r the distances from the center of gravity to the front and rear wheels, $F_{x,i}$ and $F_{y,i}$ are the forces

due to the tyres interaction with the ground, $F_{x,aero}$ is the aerodynamic drag.

Having defined the system equations, since the end goal is to derive a linear MPC, the objective is to transform the system 2.32 into a linear system in the form

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{2.33}$$

where the state vector $\mathbf{x}$ is

$$\mathbf{x} = \Big[X, Y, \psi, v_x, v_y, r\Big]^T \tag{2.34}$$

and the input vector must be defined.

As it can be easily noticed, the system 2.32 is non-linear in its implicit form, due to the presence of cosine and sine functions, and products between states. The objective of this thesis work is to develop a linear MPC that can guarantee a satisfying trade-off between computational effort and performance of the prototype. Thus the next step is to make explicit all the variables 2.34 and obtain at the end a linear system.
First, the low level controller of the prototype has a great influence on the system to control, thus some preliminary considerations about it have to be done: due to the specific application prototype described in previous chapters, a suitable TV algorithm has been developed by the team Squadra Corse Driverless PoliTO, that aims to translate the instantaneous power request and the actual steering angle into suitable torques to each wheel's motor. The TV algorithm main objective is to provide traction force by translating a reference acceleration command into an instantaneous torque request that has to be delivered by the motors, avoiding traction loss and instability. The reference command can originate by the accelerator pedal, if a physical driver is present inside the vehicle, or by a reference signal provided by the high level control if it is driving in autonomous mode. Thus the longitudinal forces $F_{x,i}$ in (2.32) can be considered as acting directly on the center of gravity of the model instead of being divided between front and rear, and rewrite the total longitudinal force as

$$F_{x,tot} = F_{x,f} + F_{x,r} \quad = \frac{P_{tot}}{v_x} \tag{2.35}$$

Moreover, the TV second objective is to compensates for understeering or oversteering behaviour of the car, by controlling the vehicle's yaw rate, trying to reach a suitable online computed reference. This has been set to be based on the ideal behaviour of the single-track bicycle model. This allows to greatly simplify the high level control, in particular the MPC problem, since it can be

assumed that the dynamics of the plant to control is predictable and actually similar to the single-track model used, so also the contribution of τ_{TV} can be set to null.

Taking into account also the aerodynamic drag force, which expression has been derived in chapter 2.2.2, the system of equations 2.32 can thus be rewritten as:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\psi} \\ \dot{v_x} \\ \dot{v_y} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} v_x \cos\psi - v_y \sin\psi \\ v_x \sin\psi + v_y \cos\psi \\ r \\ \frac{1}{m}(\frac{P_{tot}}{v_x} - F_{y,f} \sin\delta - \frac{1}{2}\rho S c_x v_x^2 + m v_y r) \\ \frac{1}{m}(F_{y,f} \cos\delta + F_{y,r} - m v_x r) \\ \frac{1}{I_z}(F_{y,f} l_f \cos\delta - F_{y,r} l_r) \end{bmatrix} \tag{2.36}$$

Notice that, following 2.35, the contribution of $F_{x,i}$ disappears from both $\dot{v_y}$ and $\dot{r}$ equations.
The lateral forces $F_{y,i}$ can be derived by he Pacejka tyre model already analyzed in chapter 2.2.1. As it has been described, the tyre behaviour is highly non-linear, considering its dependence from the side slip angle, the vertical load, the road adhesion coefficient etc. Since the objective is to derive a linear model of the system, a possible solution is to use the linearized Pacejka model which is based on the following assumptions:

- small side slip angles, that guarantee the model to remain in the linear region of the characteristic curve;
- F_y is null for straight line driving;
- the cornering stiffness is considered constant, so it's neglected its dependency from the vertical load and the road adhesion coefficient.

The first assumption can be considered valid for small lateral velocity and yaw rate: since this is a first approach to a controller that will be tested on a full scale prototype, the tests will not be conducted trying to reach high dynamics, so the assumption can be considered valid. The second assumption is always valid, since in the real vehicle, being double-track, the pneumatics are usually mounted so that offsets in lateral force is compensated between right and left wheel. The last assumption is probably the most critical, since the vertical load changes based on the weight transfer during motion and on the presence of aerodynamic devices that generate downforce. It also must be considered that, during cornering in a dual track vehicle, vertical load significantly changes between left and right side of the vehicle, as largely discussed in 2.2. In order to keep the model simple, an average value of the cornering stiffness has been evaluated, based on the average vertical

load on each wheel. The lateral tire forces are then modelled as:

$$\begin{aligned} F_{y,f} &= 2C_f\alpha_f, \\ F_{y,r} &= 2C_r\alpha_r \end{aligned} \tag{2.37}$$

As analysed in chapter 2.2.1, the side slip angle can be evaluated as the angle between the total velocity of the wheel and the longitudinal direction: this can be translated in the following function:

$$\alpha_w = \arctan\left(\frac{v_{y,w}}{|v_{x,w}|}\right) \tag{2.38}$$

Considering that the vehicle is moving with velocity (v_x, v_y) and rotating with angular velocity r, and that a steering angle δ is present at the front wheel, the following expressions for the side slip angles can be found by simple algebraic manipulation (the longitudinal velocity is considered always positive since the vehicle is forbidden from Formula SAE regulations to move in reverse):

$$\begin{aligned} \alpha_f &= \delta - \left(\arctan\frac{v_y + l_f r}{v_x}\right), \\ \alpha_r &= -\left(\arctan\frac{v_y - l_r r}{v_x}\right) \end{aligned} \tag{2.39}$$

In the hypothesis of small side slip angles, the following approximation can be made:

$$\begin{aligned} \alpha_f &\simeq \delta - \frac{v_y + l_f r}{v_x}, \\ \alpha_r &\simeq -\frac{v_y - l_r r}{v_x} \end{aligned} \tag{2.40}$$

The lateral forces can then be written as:

$$\begin{aligned} F_{y,f} &= -2C_f\frac{v_y}{v_x} - 2C_f l_f\frac{r}{v_x} + 2C_f\delta, \\ F_{y,r} &= -2C_r\frac{v_y}{v_x} + 2C_r l_r\frac{r}{v_x} \end{aligned} \tag{2.41}$$

Finally, the original system of equations 2.32 can be written in its explicit form as

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\psi} \\ \dot{v_x} \\ \dot{v_y} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} v_x\cos\psi - v_y\sin\psi \\ v_x\sin\psi + v_y\cos\psi \\ r \\ \frac{P_{tot}}{mv_x} - \frac{2C_f}{m}\left(-\frac{v_y}{v_x}\sin\delta - l_f\frac{r}{v_x}\sin\delta + \delta\sin\delta\right) - \frac{\rho S c_x}{2m}v_x^2 + v_y r \\ \frac{2C_f}{m}\left(-\frac{v_y}{v_x}\cos\delta - l_f\frac{r}{v_x}\cos\delta + \delta\cos\delta\right) + \frac{2C_r}{m}\left(-\frac{v_y}{v_x} + l_r\frac{r}{v_x}\right) - v_x r \\ \frac{2C_f l_f}{I_z}\left(-\frac{v_y}{v_x}\cos\delta - l_f\frac{r}{v_x}\cos\delta + \delta\cos\delta\right) - \frac{2C_r l_r}{I_z}\left(-\frac{v_y}{v_x} + l_r\frac{r}{v_x}\right) \end{bmatrix} \tag{2.42}$$

where it have been made explicit the dependencies from the states of the system defined in 2.34

$$\mathbf{x} = \Big[X, Y, \psi, v_x, v_y, r\Big]^T \tag{2.43}$$

and the external inputs:

$$\mathbf{u} = \Big[P_{tot}, \delta\Big]^T \tag{2.44}$$

Having derived the explicit dynamic equations and both the state and input vectors, the next step is to obtain a linear system in the form defined in 2.33. The followed approach is the Jacobian linearization method ([11]), that approximates the non-linear equation with its Taylor expansion to the first order evaluated in the operating point $(\overline{\mathbf{x}}, \overline{\mathbf{u}})$. In this way, each non-linear equation can be written as:

$$f_i(\mathbf{x}, \mathbf{u}) = f_i(\overline{\mathbf{x}}, \overline{\mathbf{u}}) + \left.\frac{\partial f_i}{\partial x_j}\right|_{\overline{\mathbf{x}},\overline{\mathbf{u}}} (x_j - \overline{x_j}) + \left.\frac{\partial f_i}{\partial u_k}\right|_{\overline{\mathbf{x}},\overline{\mathbf{u}}} (u_k - \overline{u_k}) \tag{2.45}$$

and by defining the deviation variables $\Delta\mathbf{x}$ and $\Delta\mathbf{u}$ as

$$\begin{aligned} \Delta x_j &= x_j - \overline{x_j} \\ \Delta u_k &= u_k - \overline{u_k} \end{aligned} \tag{2.46}$$

the equation 2.45 becomes linear with respect to the new state vector $\Delta\mathbf{x}$ and the new input vector $\Delta\mathbf{u}$. The linearization method allows to obtain a linear model, but it only approximates the non-linear one around the considered working point, in other words it is only valid for small deviations of the states and the inputs. For this reason the operating point selection has a great influence on the quality of approximation. By comparison with the objective linear system 2.33, the only difference is the presence of the term $f_i(\overline{\mathbf{x}}, \overline{\mathbf{u}})$. In this cases, the usual approach is to try to find a suitable point $(\overline{\mathbf{x}}, \overline{\mathbf{u}})$ such that this term goes to zero. This particular point (or set of points) of the system is called equilibrium point, and can be found by imposing

$$f_i(\overline{\mathbf{x}}, \overline{\mathbf{u}}) = 0, i = 1:6 \tag{2.47}$$

that, for the system in analysis, translates into solving the system of equations:

$$\begin{cases} v_x \cos\psi - v_y \sin\psi = 0 \\ v_x \sin\psi + v_y \cos\psi = 0 \\ r = 0 \\ \frac{P_{tot}}{m v_x} - \frac{2C_f}{m}\left(-\frac{v_y}{v_x}\sin\delta - l_f\frac{r}{v_x}\sin\delta + \delta\sin\delta\right) - \frac{\rho S c_x}{2m}v_x^2 + v_y r = 0 \\ \frac{2C_f}{m}\left(-\frac{v_y}{v_x}\cos\delta - l_f\frac{r}{v_x}\cos\delta + \delta\cos\delta\right) + \frac{2C_r}{m}\left(-\frac{v_y}{v_x} + l_r\frac{r}{v_x}\right) - v_x r = 0 \\ \frac{2C_f l_f}{I_z}\left(-\frac{v_y}{v_x}\cos\delta - l_f\frac{r}{v_x}\cos\delta + \delta\cos\delta\right) - \frac{2C_r l_r}{I_z}\left(-\frac{v_y}{v_x} + l_r\frac{r}{v_x}\right) = 0 \end{cases} \tag{2.48}$$

In a non-linear system of equations, where the number of equations to be solved is less than the number of variables, the goal of finding equilibrium points could result in the impossibility of finding an algebraic solution, or even in having an infinite number of solutions. In the particular case of the system 2.48, there are six equations and eight variables in total, that are:

$$(\overline{\mathbf{x}}, \overline{\mathbf{x}}) = (\overline{X}, \overline{Y}, \overline{\psi}, \overline{v_x}, \overline{v_y}, \overline{r}, \overline{P_{tot}}, \overline{\delta}) \tag{2.49}$$

First, an existing condition on the system must be imposed: it can be noticed in fact that in some equations the velocity v_x is present at the denominator, thus the existing condition that must always be verified is:

$$v_x \neq 0 \tag{2.50}$$

This represents a common problem of the single track model, that is said to be ill-formed, since the definition of lateral slip has v_x at the denominator. In this cases, usually a simpler kinematic model is adopted at low velocities, where the interaction between pneumatic and ground is not considered, then a blending procedure can be adopted to smooth the transition between the kinematic model and the dynamic one ([12]). For the particular implementation of the MPC developed in this thesis work, the controller is designed to act when the car is already moving, and the longitudinal velocity is always different from zero. Anyway, as it will be discussed later on, also the problem of low velocity will be addressed and solved.
Having set the existing condition for all the equations, the system 2.48 can be solved. The first equation to take into account is the third one, $r = 0$, that admits of course a unique solution for the yaw rate variable:

$$\overline{r} = 0 \tag{2.51}$$

This solution implies that the vehicle should move rigidly in the 2D space without any rotation around the vehicle's z-axis. Intuitively, this condition alone corresponds to a straight line driving with null steering, but this can also be demonstrated analytically. Substituting $r = 0$ in the fifth and sixth equation, a subsystem of two equations can be obtained:

$$\begin{cases} \frac{2C_f}{m}\left(-\frac{v_y}{v_x}\cos\delta + \delta\cos\delta\right) + \frac{2C_r}{m}\left(-\frac{v_y}{v_x}\right) = 0 \\ \frac{2C_f l_f}{I_z}\left(-\frac{v_y}{v_x}\cos\delta + \delta\cos\delta\right) - \frac{2C_r l_r}{I_z}\left(-\frac{v_y}{v_x}\right) = 0 \end{cases} \tag{2.52}$$

By means of simple algebraic manipulation, the following system is derived:

$$\begin{cases} v_y\left(\frac{C_r}{C_f}\frac{1}{v_x}\right) = \delta\cos\delta - \frac{v_y}{v_x}\cos\delta \\ v_y\left(-\frac{C_r l_r}{C_f l_f}\frac{1}{v_x}\right) = \delta\cos\delta - \frac{v_y}{v_x}\cos\delta \end{cases} \tag{2.53}$$

By means of substitution, the following result is obtained:

$$v_y \left(\frac{C_r}{C_f} \frac{1}{v_x} \right) = v_y \left(-\frac{C_r l_r}{C_f l_f} \frac{1}{v_x} \right) \tag{2.54}$$

In the hypothesis that v_x is non null and non infinite (the first condition would lead to a standing still vehicle, that is not an interesting condition to study, while the second one is infeasible in reality), the only solution that satisfies 2.54 is

$$\overline{v_y} = 0 \tag{2.55}$$

The two solutions 2.51 and 2.55 can be used to evaluate directly the equilibrium condition for δ: by simply substituting them in either the fifth or sixth equations of 2.48, and taking into considerations that the physical angle δ on a road vehicle is limited, in the specific case of the Formula Student prototype in analysis is limited to $\pm 20°$, the only feasible solution is:

$$\overline{\delta} = 0 \tag{2.56}$$

The equilibrium states derived until now 2.51, 2.55 and 2.56 describe a straight line driving condition as the equilibrium for the considered system. The fourth equation of 2.48 can now be solved, that easily drives to:

$$\overline{P_{tot}} = \frac{\rho S c_x}{2} v_x^3 = P_{drag} \tag{2.57}$$

Which is the dissipated power due to the aerodynamic drag force. Note that, from the same equation, a dissipated power due to the side slip presence in case of steering condition can be identified:

$$P_{alpha} = 2C_f \left(-\frac{v_y}{v_x} \sin\delta - l_f \frac{r}{v_x} \sin\delta + \delta \sin\delta \right) v_x = 2C_f \alpha_f v_x \sin\delta \tag{2.58}$$

Which is of course not influencing the equilibrium state $\overline{P_{tot}}$ since in straight driving condition P_{alpha} is null.
The remaining conditions on $\overline{\psi}$ and $\overline{v_x}$ should be derived from the first and second equations of 2.48, but unfortunately no solution can be obtained that satisfies both: considering 2.55, the two simultaneous conditions to be satisfied are:

$$\begin{cases} v_x \cos\psi = 0 \\ v_x \sin\psi = 0 \end{cases} \tag{2.59}$$

which are both true only if $\overline{v_x} = 0$, condition that is in contrast with 2.50. It derives that no equilibrium condition can be defined for both $\overline{v_x}$ and $\overline{\psi}$, together

with $\overline{X}$ and $\overline{Y}$ that do not appear in the system of equations.

Summing up, the expression of the dynamic equations at equilibrium $f_i(\overline{x}, \overline{u})$ are:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\psi} \\ \dot{v_x} \\ \dot{v_y} \\ \dot{r} \end{bmatrix}_{\overline{\mathbf{x}},\overline{\mathbf{u}}} = \begin{bmatrix} \overline{v_x}\cos\overline{\psi} \\ \overline{v_x}\sin\overline{\psi} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.60}$$

The linearized equations can be now derived following 2.45, then the equilibrium states of the vectors $\Delta\mathbf{x}$ and $\Delta\mathbf{u}$ can be multiplied by the Jacobian matrix, obtaining the following formulation derived form 2.45:

$$\begin{aligned} f_i(\mathbf{x},\mathbf{u}) &= f_i(\overline{\mathbf{x}},\overline{\mathbf{u}}) + \left.\frac{\partial f_i}{\partial x_j}\right|_{\overline{\mathbf{x}},\overline{\mathbf{u}}} (x_j - \overline{x_j}) + \left.\frac{\partial f_i}{\partial u_k}\right|_{\overline{\mathbf{x}},\overline{\mathbf{u}}} (u_k - \overline{u_k}) = \\ &= \left.\frac{\partial f_i}{\partial x_j}\right|_{\overline{\mathbf{x}},\overline{\mathbf{u}}} x_j + \left.\frac{\partial f_i}{\partial u_k}\right|_{\overline{\mathbf{x}},\overline{\mathbf{u}}} u_k + \left(f_i(\overline{\mathbf{x}},\overline{\mathbf{u}}) - \left.\frac{\partial f_i}{\partial x_j}\right|_{\overline{\mathbf{x}},\overline{\mathbf{u}}} \overline{x_j} - \left.\frac{\partial f_i}{\partial u_k}\right|_{\overline{\mathbf{x}},\overline{\mathbf{u}}} \overline{u_k} \right) \end{aligned} \tag{2.61}$$

where the first two terms are the elements of the matrix representation 2.33, while the last term is constant but acts as a sort of disturbance to the system.

$$\begin{aligned} \dot{X} &= \overline{v_x}\cos\overline{\psi} + \begin{bmatrix} 0 \\ 0 \\ -v_x\sin\psi - v_y\cos\psi \\ \cos\psi \\ -\sin\psi \\ 0 \end{bmatrix}^T_{\overline{\mathbf{x}},\overline{\mathbf{u}}} \begin{bmatrix} X - \overline{X} \\ Y - \overline{Y} \\ \psi - \overline{\psi} \\ v_x - \overline{v_x} \\ v_y - \overline{v_y} \\ r - \overline{r} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}^T_{\overline{\mathbf{x}},\overline{\mathbf{u}}} \begin{bmatrix} P_{tot} - \overline{P_{tot}} \\ \delta - \overline{\delta} \end{bmatrix} = \\ &= \begin{bmatrix} 0, 0, -\overline{v_x}\sin\overline{\psi}, \cos\overline{\psi}, -\sin\overline{\psi}, 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0, 0 \end{bmatrix} \mathbf{u} - \overline{\psi}(-\overline{v_x}\sin\overline{\psi}) \end{aligned} \tag{2.62}$$

$$\dot{Y} = \overline{v_x}\sin\overline{\psi} + \begin{bmatrix} 0 \\ 0 \\ v_x\cos\psi - v_y\sin\psi \\ \sin\psi \\ \cos\psi \\ 0 \end{bmatrix}^T_{\overline{\mathbf{x}},\overline{\mathbf{u}}} \begin{bmatrix} X-\overline{X} \\ Y-\overline{Y} \\ \psi-\overline{\psi} \\ v_x-\overline{v_x} \\ v_y-\overline{v_y} \\ r-\overline{r} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}^T_{\overline{\mathbf{x}},\overline{\mathbf{u}}} \begin{bmatrix} P_{tot}-\overline{P_{tot}} \\ \delta-\overline{\delta} \end{bmatrix} = \tag{2.63}$$

$$= \begin{bmatrix} 0, 0, \overline{v_x}\cos\overline{\psi}, \sin\overline{\psi}, \cos\overline{\psi}, 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0, 0 \end{bmatrix} \mathbf{u} - \overline{\psi}(\overline{v_x}\cos\overline{\psi})$$

$$\dot{\psi} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}^T_{\overline{\mathbf{x}},\overline{\mathbf{u}}} \begin{bmatrix} X-\overline{X} \\ Y-\overline{Y} \\ \psi-\overline{\psi} \\ v_x-\overline{v_x} \\ v_y-\overline{v_y} \\ r-\overline{r} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}^T_{\overline{\mathbf{x}},\overline{\mathbf{u}}} \begin{bmatrix} P_{tot}-\overline{P_{tot}} \\ \delta-\overline{\delta} \end{bmatrix} = \tag{2.64}$$

$$= \begin{bmatrix} 0, 0, 0, 0, 0, 0, 1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0, 0 \end{bmatrix} \mathbf{u}$$

$$\begin{aligned} \dot{v_x} = &\begin{bmatrix} 0 \\ 0 \\ 0 \\ -\frac{P_{tot}}{mv_x^2} - \frac{2C_f}{m}\left(\frac{v_y}{v_x^2}\sin\delta + l_f\frac{r}{v_x^2}\sin\delta\right) - \frac{\rho S c_x}{m}v_x \\ \frac{2C_f}{m}\left(\frac{1}{v_x}\sin\delta\right) + r \\ \frac{2C_f l_f}{m}\left(\frac{1}{v_x}\sin\delta\right) + v_y \end{bmatrix}^T_{\overline{\mathbf{x}},\overline{\mathbf{u}}} \begin{bmatrix} X-\overline{X} \\ Y-\overline{Y} \\ \psi-\overline{\psi} \\ v_x-\overline{v_x} \\ v_y-\overline{v_y} \\ r-\overline{r} \end{bmatrix} + \\ &+ \begin{bmatrix} \frac{1}{mv_x} \\ -\frac{2C_f}{m}\left(-\frac{v_y}{v_x}\cos\delta - l_f\frac{r}{v_x}\cos\delta + \delta\cos\delta + \sin\delta\right) \end{bmatrix}^T_{\overline{\mathbf{x}},\overline{\mathbf{u}}} \begin{bmatrix} P_{tot}-\overline{P_{tot}} \\ \delta-\overline{\delta} \end{bmatrix} = \\ &= \begin{bmatrix} 0, 0, -\frac{3}{2}\frac{\rho S c_x}{m}\overline{v_x}, 0, 0, 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} \frac{1}{m\overline{v_x}}, 0 \end{bmatrix} \mathbf{u} + \frac{\rho S c_x}{m}\overline{v_x}^2 \end{aligned} \tag{2.65}$$

$$
\begin{aligned}
\dot{v_y} = &\begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{2C_f}{m}\left(\frac{v_y}{v_x^2}\cos\delta + l_f\frac{r}{v_x^2}\cos\delta\right) + \frac{2C_r}{m}\left(\frac{v_y}{v_x^2} - l_r\frac{r}{v_x^2}\right) - r \\ -\frac{2C_f}{m}\left(\frac{1}{v_x}\cos\delta\right) - \frac{2C_r}{m}\left(\frac{1}{v_x}\right) \\ -\frac{2C_f l_f}{m}\left(\frac{1}{v_x}\cos\delta\right) + \frac{2C_r l_r}{m}\left(\frac{1}{v_x}\right) - v_x \end{bmatrix}^T_{\overline{\mathbf{x}},\overline{\mathbf{u}}} \begin{bmatrix} X-\overline{X} \\ Y-\overline{Y} \\ \psi-\overline{\psi} \\ v_x-\overline{v_x} \\ v_y-\overline{v_y} \\ r-\overline{r} \end{bmatrix} + \\
&+ \begin{bmatrix} 0 \\ \frac{2C_f}{m}\left(\frac{v_y}{v_x}\sin\delta + l_f\frac{r}{v_x}\sin\delta + \cos\delta - \delta\sin\delta\right) \end{bmatrix}^T_{\overline{\mathbf{x}},\overline{\mathbf{u}}} \begin{bmatrix} P_{tot}-\overline{P_{tot}} \\ \delta-\overline{\delta} \end{bmatrix} = \\
&= \left[0,0,0,0,-\tfrac{2(C_f+C_r)}{m\overline{v_x}},-\tfrac{2(C_f l_f - C_r l_r)}{m\overline{v_x}} - \overline{v_x}\right]\mathbf{x} + \left[0,\tfrac{2C_f}{m}\right]\mathbf{u}
\end{aligned}
\tag{2.66}
$$

$$
\begin{aligned}
\dot{r} = &\begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{2C_f l_f}{I_z}\left(\frac{v_y}{v_x^2}\cos\delta + l_f\frac{r}{v_x^2}\cos\delta\right) - \frac{2C_r l_r}{I_z}\left(\frac{v_y}{v_x^2} - l_r\frac{r}{v_x^2}\right) \\ -\frac{2C_f l_f}{I_z}\left(\frac{1}{v_x}\cos\delta\right) + \frac{2C_r l_r}{I_z}\left(\frac{1}{v_x}\right) \\ -\frac{2C_f l_f^2}{I_z}\left(\frac{1}{v_x}\cos\delta\right) - \frac{2C_r l_r^2}{I_z}\left(\frac{1}{v_x}\right) \end{bmatrix}^T_{\overline{\mathbf{x}},\overline{\mathbf{u}}} \begin{bmatrix} X-\overline{X} \\ Y-\overline{Y} \\ \psi-\overline{\psi} \\ v_x-\overline{v_x} \\ v_y-\overline{v_y} \\ r-\overline{r} \end{bmatrix} + \\
&+ \begin{bmatrix} 0 \\ \frac{2C_f l_f}{I_z}\left(\frac{v_y}{v_x}\sin\delta + l_f\frac{r}{v_x}\sin\delta + \cos\delta - \delta\sin\delta\right) \end{bmatrix}^T_{\overline{\mathbf{x}},\overline{\mathbf{u}}} \begin{bmatrix} P_{tot}-\overline{P_{tot}} \\ \delta-\overline{\delta} \end{bmatrix} = \\
&= \left[0,0,0,0,-\tfrac{2(C_f l_f - C_r l_r)}{I_z\overline{v_x}},-\tfrac{2(C_f l_f^2 + C_r l_r^2)}{I_z\overline{v_x}}\right]\mathbf{x} + \left[0,\tfrac{2C_f l_f}{I_z}\right]\mathbf{u}
\end{aligned}
\tag{2.67}
$$

In conclusion, the linearized model obtained for the single-track vehicle model

equations 2.42 can be expressed in matrix form as:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\psi} \\ \dot{v_x} \\ \dot{v_y} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -\overline{v_x}\sin\overline{\psi} & \cos\overline{\psi} & -\sin\overline{\psi} & 0 \\ 0 & 0 & \overline{v_x}\cos\overline{\psi} & \sin\overline{\psi} & \cos\overline{\psi} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{3}{2}\frac{\rho S c_x}{m}\overline{v_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{2(C_f+C_r)}{m\overline{v_x}} & -\frac{2(C_f l_f - C_r l_r)}{m\overline{v_x}} - \overline{v_x} \\ 0 & 0 & 0 & 0 & -\frac{2(C_f l_f - C_r l_r)}{I_z\overline{v_x}} & -\frac{2(C_f l_f^2 + C_r l_r^2)}{I_z\overline{v_x}} \end{bmatrix} \begin{bmatrix} X \\ Y \\ \psi \\ v_x \\ v_y \\ r \end{bmatrix} + \\ + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{m\overline{v_x}} & 0 \\ 0 & \frac{2C_f}{m} \\ 0 & \frac{2C_f l_f}{I_z} \end{bmatrix} \begin{bmatrix} P_{tot} \\ \delta \end{bmatrix} + \begin{bmatrix} -\overline{\psi}(-\overline{v_x}\sin\overline{\psi}) \\ -\overline{\psi}(\overline{v_x}\cos\overline{\psi}) \\ 0 \\ \frac{\rho S c_x}{m}\overline{v_x}^2 \\ 0 \\ 0 \end{bmatrix} \tag{2.68}$$

Where the state matrix **A** and the input matrix **B** can be identified, and are parametric since they still depend on the choice of $\overline{v_x}$ and $\overline{\psi}$. The additional disturbance vector is not dependant by any state of the system, only by their initial value of the state of the system. Since this disturbances have a mathematical expression that can be evaluated, this vector is usually called measured disturbance vector.

Being the linearized system an approximation of the original non-linear model, its accuracy must be analysed: since for this thesis work the controller to be designed will provide lateral dynamic control, in order to track a desired reference trajectory, the accuracy about the three states X, Y, and ψ are analysed: simulating multiple times the two systems, the non-linear and the linearized one, over different time intervals, giving constant $P_{tot} = \overline{P_{tot}}$ and δ for each simulation, but changing the steering nput between simulations, it is possible to evaluate the differences between the final state (X, Y, ψ) reached by the two models. In particular, this analysis has been developed for 1 second and 1.5 seconds time intervals, which are interesting intervals for later anaysis.

As shown in Figure 2.11, the error about the X and Y position is below 1 meter if simulating for 1 second, values that grow exponentially for higher time intervals, for example they are more than doubled after just 0.5s. This results will be useful when discussing the prediction horizon of the MPC controller.

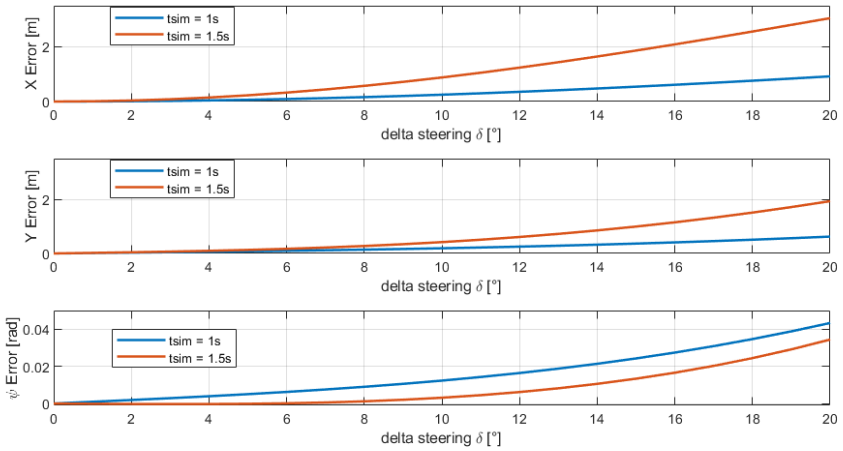

Figure 2.11: Prediction error over 1 second and 1.5 seconds at constant inputs.

Chapter 3

MPC Problem Formulation

Being the MPC a discrete time controller, acting on a physical system that therefore is continuous, some preliminary considerations about the real time implementation have to be made. On the prototype for which the MPC is designed, it is already deployed a electronic control unit (ECU) which objective is to collect data from all sensors on the car (except the environment perceptive ones) via CANbus communication, elaborate the control inputs for the actuators and send suitable signals to them. In particular, for the steering actuator, the reference position is set by the ACU, on which the MPC will be deployed, and the ECU translates this reference into actual control input signal. This control unit runs at higher speeds with respect to the ACU, in the particular Squadra Corse Driverless car it runs at 200Hz. When a target steering input is received from the ACU, the ECU applies a zero-order-hold technique to the reference signal and keeps it constant until a new reference is received from the ACU. An high level representation of the actual steering control pipeline is shown in Figure 3.1.

The series of low-level steering control, steering actuator and vehicle, also thanks to the presence of torque vectoring algorithm acting in parallel, can be considered as a single-track vehicle model, as discussed in chapter 2.4.3, which overall dynamic is represented by the system 2.36.

3.1 Discrete time prediction model

3.1.1 Local frame model formulation

In order to realise an efficient Model Predictive Controller, it is fundamental to study the prediction model, adjust it to he desired problem formulation and evaluate a proper cost function that is the key element for the optimization problem.
Starting from the prediction model, as discussed in chapter 2.4, in order to keep

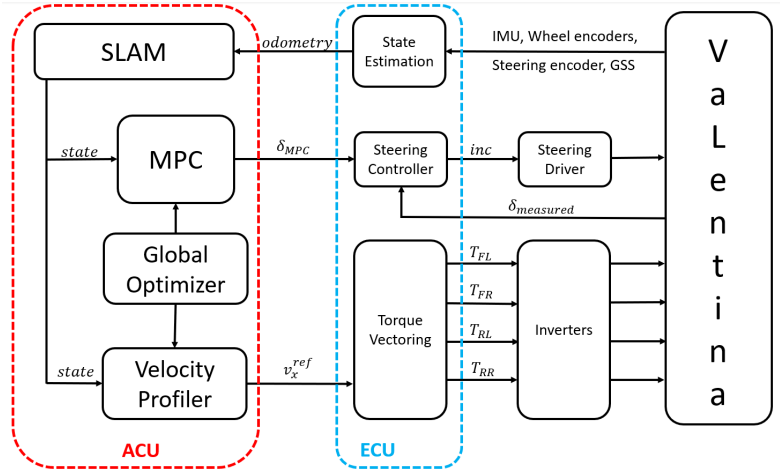

Figure 3.1: High level overview of the control scheme in which the MPC will be deployed.

the controller simple and computationally efficient, a linear representation has been chosen in the form

$$\dot{\mathbf{x}}(t) = A_c\mathbf{x}(t) + B_c\mathbf{u}(t) \tag{3.1}$$

Where A_c and B_c are the continuous time system matrices.

The derived dynamic model, obtained through the Jacobian linearization method, is expressed as

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\psi} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -\overline{v_x}\sin\overline{\psi} & \cos\overline{\psi} & -\sin\overline{\psi} & 0 \\ 0 & 0 & \overline{v_x}\cos\overline{\psi} & \sin\overline{\psi} & \cos\overline{\psi} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{3}{2}\frac{\rho S c_x}{m}\overline{v_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{2(C_f+C_r)}{m\overline{v_x}} & -\frac{2(C_f l_f - C_r l_r)}{m\overline{v_x}} - \overline{v_x} \\ 0 & 0 & 0 & 0 & -\frac{2(C_f l_f - C_r l_r)}{I_z\overline{v_x}} & -\frac{2(C_f l_f^2 + C_r l_r^2)}{I_z\overline{v_x}} \end{bmatrix} \begin{bmatrix} X \\ Y \\ \psi \\ v_x \\ v_y \\ r \end{bmatrix} + \\ + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{m\overline{v_x}} & 0 \\ 0 & \frac{2C_f}{m} \\ 0 & \frac{2C_f l_f}{I_z} \end{bmatrix} \begin{bmatrix} P_{tot} \\ \delta \end{bmatrix} + \begin{bmatrix} -\overline{\psi}(-\overline{v_x}\sin\overline{\psi}) \\ -\overline{\psi}(\overline{v_x}\cos\overline{\psi}) \\ 0 \\ \frac{\rho S c_x}{m}\overline{v_x}^2 \\ 0 \\ 0 \end{bmatrix} \tag{3.2}$$

It can be noticed that the two systems are similar, except for the contribution of the previously defined measured disturbance vector. This disturbance can lead to unmodelled dynamics in the prediction model, because the MPC approach

that will be followed can not take into account for a measured disturbance. It is possible to get rid of this undesired contribution by slightly changing the approach to the trajectory reference tracking problem. The initial working conditions of the controller to be designed has been set to the global reference frame RF^0, because the reference trajectory is expressed as a vector in the inertial fixed frame. By simply changing the reference frame in which the controller operates, going from the fixed frame RF^0 to the local one RF^1, it is possible to locally set the initial state to be null:

$$\begin{aligned} \overline{X} &= \overline{x} = 0, \\ \overline{Y} &= \overline{y} = 0, \\ \overline{\psi} &= 0 \end{aligned} \tag{3.3}$$

making possible also to bring to zero two of the measured disturbance contributions, and simplifying the system matrices too. This result can be obtained by applying, at the beginning of each prediction interval of the MPC routine, a roto-translation of the reference vector to the local reference frame, using the transformation matrix $\mathbf{T^1_{0,k}}$ derived in chapter 2.1, taking the initial state of the vehicle X_0, Y_0, ψ_0 as parameters of the matrix.
Moreover, the objective of this thesis is to design a lateral dynamics controller for the problem in analysis, and this implies that the longitudinal behaviour of the vehicle is not being actually controlled. By making the hypothesis of constant longitudinal velocity along the prediction horizon, thus making null its rate of change $\dot{v}_x$, the prediction model can be formulated in a simpler and more convenient expression:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{v_x} \\ \dot{v_y} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & v_0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{2(C_f+C_r)}{mv_0} & -\frac{2(C_f l_f - C_r l_r)}{mv_0} - v_0 \\ 0 & 0 & 0 & 0 & -\frac{2(C_f l_f - C_r l_r)}{I_z v_0} & -\frac{2(C_f l_f^2 + C_r l_r^2)}{I_z v_0} \end{bmatrix} \begin{bmatrix} x \\ y \\ \psi \\ v_x \\ v_y \\ r \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \frac{2C_f}{m} \\ \frac{2C_f l_f}{I_z} \end{bmatrix} \begin{bmatrix} \delta \end{bmatrix} \tag{3.4}$$

Note that this prediction model is the same of 2.68, so also the analysis on its range of validity shown in 2.11 is still valid, since it has been conducted at constant velocity.
The simplified prediction model obtained can now be used in the linear MPC formulation adopted, since it can be fully expressed in the form described in 3.1. It can be noticed that the matrix A_c is still not constant, since it depends on the initial longitudinal velocity v_0, making the system a linear parameter varying (LPV) system.

3.1.2 Physical constraints

In order to address the formulation of the MPC problem, the constraints of the actual system must be taken into account first. Starting from the state constraints, the first three states x, y, ψ have no actual constraint, since in principle the vehicle is free to move in the 2D space, and the tracks limits are considered above, when evaluating the desired trajectory to be followed. For what regards the other system states, as v_x, v_y, r, their limitations are not of interest for the designed controller, since its objective is to track a reference trajectory, and the vehicle limits of velocity and handling are taken into account before the MPC routine, in the evaluation of the desired trajectory. Summing up, the states constraints to be taken into account in the MPC problem can be all put to infinite, this is also useful for the actual real time implementation, since the solver, recognising the infinite value, avoids checking limits on the states and therefore the computational time can be reduced. For what regards the input constraints, in particular for the steering input δ, there are two types of constraints that must be imposed: the first one is on its range of motion, that is limited to $\pm 20°$, the second on its speed of actuation: the deployed actuator on the prototype shows a low pass behaviour, with cut-off frequency at 4Hz. The rate of change of the input will be made explicit when deriving the discrete time prediction model.

3.1.3 Discrete time model

In order to set up the optimization problem for the MPC algorithm, the prediction model must be expressed in its discrete time equivalent. In signal processing theory, the discretization of a continuous time signal is made by sampling the signal in a uniformly-spaced time intervals, called sampling period T_s. This creates a sequence of values that constitute the discrete time signal. When going in the opposite direction, that is when converting a discrete signal into a continuous one, not knowing at priory the entire sequence of the signal, usually the zero-order hold conversion method is used, which simply holds the value of the discrete sequence for the period T_s, creating a stair-like continuous time signal. Of course, this "deconstruction" and "reconstruction" procedure modifies the original signal, in particular information is lost between sampling intervals.
When dealing with dynamical systems expressed in matrix form 3.1, the goal is to obtain an equivalent discrete time system:

$$\mathbf{x}[k+1] = A_d\mathbf{x}[k] + B_d\mathbf{u}[k] \tag{3.5}$$

that better approximates the continuous one, where k is the sampled time instant [13]. It follows that the derivative of the state at time t is approximated as its

value after a period T_s: being the general solution of the system 3.1:

$$\mathbf{x}(t) = e^{A_c t}\mathbf{x}(0) + \int_0^t e^{A_c(t-\tau)} B_c \mathbf{u}(\tau) d\tau \tag{3.6}$$

the discrete matrices A_d and B_d can be obtained as:

$$\begin{aligned} A_d &= e^{A_c T_s} \\ B_d &= \left(\int_0^{T_s} e^{A_c \tau} d\tau \right) B_c \end{aligned} \tag{3.7}$$

The expressions 3.7 represent the exact solution to the problem of finding the discrete matrices of an LTI system.
Generally, the matrix exponential is a complex operation that is difficult to evaluate numerically, resulting in a computationally expensive task to be performed. In the specific case in analysis, where the system matrices are not constant and change in time, the operation of matrix discretization must be computed online based on the actual state of the model, in this case based on the initial longitudinal velocity $\overline{v_x}$. Different numerical approaches are present to obtain a close approximation of the matrix exponential, but they usually are numerically expensive. In this thesis work, the Euler Backward method is applied, which approximates a function derivative over the interval T_s as

$$\dot{x_i} \simeq \frac{x_i[k] - x_i[k-1]}{T_s} \tag{3.8}$$

avoiding numerical instabilities of the state vector which can arise with similar methods, like Forward Euler. By means of substitution, the following is obtained:

$$(I - T_s A)\mathbf{x}[k] = \mathbf{x}[k-1] + TsB_c\mathbf{u}[k] \tag{3.9}$$

and finally the discrete time system, forward-shifting of one sample instant the expression 3.9, can be expressed as:

$$\mathbf{x}[k+1] = A_d\mathbf{x}[k] + B_d\mathbf{u}[k+1] \tag{3.10}$$

where

$$\begin{aligned} A_d &= (I - T_s A_c)^{-1} \\ B_d &= T_s A_d B_c \end{aligned} \tag{3.11}$$

Note that, using the backward Euler formulation 3.10, the input vector is no more $\mathbf{u}[k]$ as in 3.5, but is forward-shifted of one time instant in $\mathbf{u}[k+1]$. By considering that

$$\mathbf{u}[k+1] = \mathbf{u}[k] + \Delta\mathbf{u}[k] \tag{3.12}$$

and changing the system discrete formulation as

$$\begin{bmatrix}\mathbf{x}[k+1]\\ \mathbf{u}[k+1]\end{bmatrix} = \begin{bmatrix}A_d & B_d\\ 0 & I\end{bmatrix}\begin{bmatrix}\mathbf{x}[k]\\ \mathbf{u}[k]\end{bmatrix} + \begin{bmatrix}B_d\\ I\end{bmatrix}\Delta\mathbf{u}[k] \tag{3.13}$$

the final discrete time representation of the controlled system can be obtained, called augmented system since $\mathbf{u}[k]$ is now a state of the system, and the actual input has become $\Delta\mathbf{u}$. The augmented formulation allows to take into account also for constraints about the rate of change of the input, which in the specific case of this thesis work is of great importance as discussed in 3.1.2.

3.1.4 Reference trajectory discretization

The objective of the MPC, as discussed in chapter 1.2.3, is to track a reference trajectory expressed as a set of coordinates $[(X_i^{ref}, Y_i^{ref})]$ in the global reference frame RF^0. In chapter 3.1.1 the change of reference frame from global into local has been discussed, then the actual reference trajectory is passed to the MPC in the local frame RF^1 by applying to every set of coordinates the transformation 2.8. The overall trajectory to be tracked, in the environment where the MPC will be deployed, is generated by a ROS2 node developed outside the controller routine, and it is evaluated when the full map has been acquired, that is after the first lap is finished. This can be obtained with different methods, with the aim to evaluate the optimal trajectory in order to optimize the lap time. At the end, a matrix containing the global set of coordinates $[(X_i^{ref}, Y_i^{ref})]$ for the reference trajectory is passed to the MPC node, but its points do not take into account for vehicle dynamics along the track, in particular its instantaneous velocity. The reference trajectory is in fact discretized with a constant distance of $10cm$ between consecutive points. In order to have the MPC tracking a feasible trajectory and achieve an optimal control input sequence, the reference trajectory must be suitably modified taking into account how the controller evaluates the input sequence and the predicted states.
The MPC uses a discrete time model, evaluated in the previous chapter, to predict the future states in which the vehicle is going to be if the evaluated input sequence is applied. At the beginning of each prediction interval, the vehicle is not said to be already in track: it is almost always present a Cross Track error $CrossTrack$, that is the distance between the vehicle's CoG and the reference trajectory, as shown in Figure 3.2. This error will also be a fundamental parameter to evaluate the performances of the controller, since a low Cross Track error translates into a vehicle being near to the desired trajectory. In order to evaluate this error, instead of taking the closest point of the reference trajectory matrix, that as shown in Figure 3.2 could lead to an error due to the original trajectory discretization, an approximation of it $CrossTrack^*$ is evaluated, by finding the linear interpolation

between the two closest points and its perpendicular passing through the vehicle's CoG. The intersection point is also used as first point of the reference vector passed to the MPC (X_0^{ref}, Y_0^{ref}).

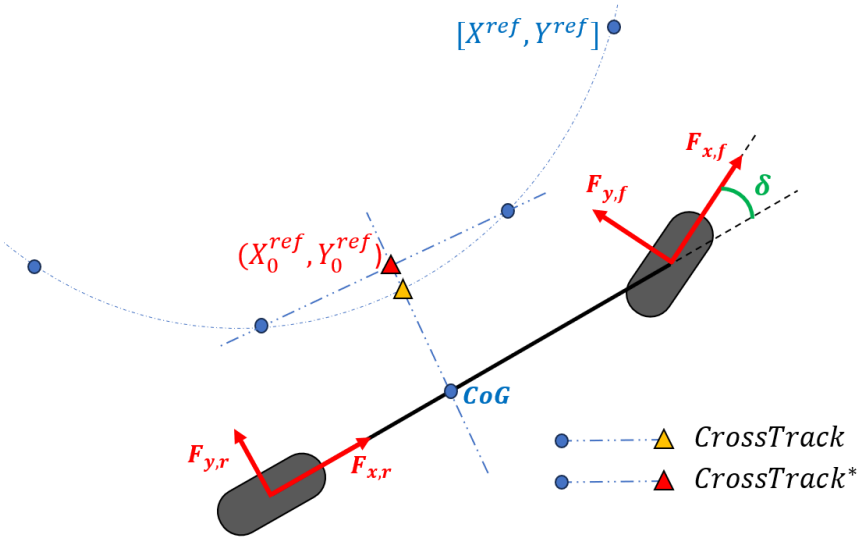

Figure 3.2: Cross Track Error between the vehicle and the reference trajectory

Being the prediction model evaluated at constant longitudinal velocity, that in each prediction interval is equal to the initial velocity v_0, for every predicted state the covered distance will be therefore constant, equal to $v_0 \cdot T_s$. Based on this consideration, each reference point (X_i^{ref}, Y_i^{ref}) is evaluated by tracing a circumference of radius $v_0 \cdot T_s$ and center in the previous point, and computing its intersection with the linear piece-wise interpolation of the points following (X_0^{ref}, Y_0^{ref}) along the original reference trajectory. By recursively applying this procedure, a set of N points can be obtained that constitute the discretized reference state passed to MPC. This set of points are then transformed in the local reference frame by applying the transformation 2.8 obtaining the set of local coordinates $[(x_j^{ref}, y_j^{ref})], j = 1 : N$.

3.2 Optimization problem

The optimization problem formulation is the core of the MPC development, since it requires a well established knowledge of the system to be controlled and the parameters that influence it, so it is directly linked with the performances of the final controller.
Summing up what has been obtained until now:

- The continuous time prediction model 3.4:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & v_0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{2(C_f+C_r)}{mv_0} & -\frac{2(C_f l_f - C_r l_r)}{mv_0} - v_0 \\ 0 & 0 & 0 & 0 & -\frac{2(C_f l_f - C_r l_r)}{I_z v_0} & -\frac{2(C_f l_f^2 + C_r l_r^2)}{I_z v_0} \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \frac{2C_f}{m} \\ \frac{2C_f l_f}{I_z} \end{bmatrix} \delta \quad (3.14)$$

- The discrete time matrices A_d and B_d, obtained following the Backward Euler method 3.8:

$$\begin{aligned} A_d &= (I - T_s A_c)^{-1} \\ B_d &= T_s A_d B_c \end{aligned} \quad (3.15)$$

- The augmented system formulation, where the input can be directly written as δ:

$$\begin{bmatrix} \mathbf{x}[k+1] \\ \delta[k+1] \end{bmatrix} = \begin{bmatrix} A_d & B_d \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}[k] \\ \delta[k] \end{bmatrix} + \begin{bmatrix} B_d \\ 1 \end{bmatrix} \Delta\delta[k] \quad (3.16)$$

- The states to be controlled are only $\mathbf{y} = [(x_i, y_i)]$, that can be obtained as:

$$\mathbf{y}_k = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}_k = \mathbf{C}\mathbf{x}_k \quad (3.17)$$

- The local reference trajectory $\mathbf{y^{ref}} = [(x_i^{ref}, y_i^{ref})]$, obtained by re-discretizing the global reference trajectory based on the actual velocity of the vehicle and roto-traslating the objective points to the local reference frame by applying 2.9 in the point (X_0, Y_0, ψ_0):

$$\mathbf{T_0^1} = \begin{bmatrix} \cos\psi_0 & \sin\psi_0 & 0 & -X_0\cos\psi_0 - Y_0\sin\psi_0 \\ -\sin\psi_0 & \cos\psi_0 & 0 & +X_0\sin\psi_0 - Y_0\cos\psi_0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.18)$$

- The system constraints:

  $\underline{\delta} = -20° \leq \delta \leq 20° = \overline{\delta}$

  $\underline{\Delta\delta} = -2\pi f_{lim} \cdot T_s \leq \Delta\delta \leq 2\pi f_{lim} \cdot T_s = \overline{\Delta\delta}, \quad f_{lim} = 4Hz$

- The state is either measured or estimated outside the MPC:

  $\mathbf{x_0} = \hat{\mathbf{x}}$

  $\delta_0 = \hat{\delta}$

Finally, the desired optimization problem (OP) can be derived, which in its general formulation is expressed as:

$$\begin{aligned}
\min_{\Delta u_{1:N}} \quad & \textstyle\sum_{k=0}^{N} \left(\left\| \mathbf{x}_k - \mathbf{x}_k^{ref} \right\|_{\mathbf{Q}}^2 + \|\mathbf{u}_k\|_{\mathbf{R_u}}^2 + \|\mathbf{\Delta u}_k\|_{\mathbf{R}_{\Delta u}}^2 \right) \\
\text{s.t.} \quad & \mathbf{x}_{k+1} = \mathbf{A_d}\mathbf{x}_k + \mathbf{B_d}\mathbf{u}_{k+1} \quad k = 0, \dots, N \\
& \mathbf{u}_{k+1} = \mathbf{u}_k + \Delta \mathbf{u}_k \quad k = 0, \dots, N \\
& \Delta \mathbf{u}_0 = 0 \\
& \mathbf{x}_0 = \hat{\mathbf{x}} \\
& \mathbf{u}_0 = \hat{\mathbf{u}} \\
& \underline{\mathbf{x}} \leq \mathbf{x}_k \leq \overline{\mathbf{x}} \quad k = 1, \dots, N \\
& \underline{\mathbf{u}} \leq u_k \leq \overline{\mathbf{u}} \quad k = 1, \dots, N \\
& \underline{\Delta \mathbf{u}} \leq \Delta \mathbf{u}_k \leq \overline{\Delta \mathbf{u}} \quad k = 1, \dots, N.
\end{aligned} \tag{3.19}$$

Where matrices $\mathbf{Q}$, $\mathbf{R}_u$ and $\mathbf{R}_{\Delta u}$ are the weight matrices, which contain the tuning parameters that influence the final performances of the controller in therms of reference tracking and input amplitude.
Since only the tracking of the coordinates (x, y) is of interest and the input vector has only one element δ, 3.19 can be specifically formulated for the problem in analysis as:

$$\begin{aligned}
\min_{\Delta \delta_{1:N}} \quad & \textstyle\sum_{k=0}^{N} \left(\left\| \mathbf{y}_k - \mathbf{y}_k^{ref} \right\|_{\mathbf{Q}_y}^2 + R\delta_k^2 + R_\Delta \Delta \delta_k^2 \right) \\
\text{s.t.} \quad & \mathbf{y}_k = \mathbf{C}\mathbf{x}_k \\
& \mathbf{x}_{k+1} = \mathbf{A_d}\mathbf{x}_k + B_d \delta_{k+1} \quad k = 0, \dots, N \\
& \delta_{k+1} = \delta_k + \Delta \delta_k \quad k = 0, \dots, N \\
& \Delta \delta_0 = 0 \\
& \mathbf{x}_0 = \hat{\mathbf{x}} \\
& \delta_0 = \hat{\delta} \\
& \underline{\delta} \leq \delta_k \leq \overline{\delta} \quad k = 1, \dots, N \\
& \underline{\Delta \delta} \leq \Delta \delta_k \leq \overline{\Delta \delta} \quad k = 1, \dots, N.
\end{aligned} \tag{3.20}$$

The optimization problem as been expressed by means of a quadratic cost function and some linear constraints. In control theory, the formulation 3.20 is also addressed as Finite-Horizon Optimal Control problem.

3.3 Real Time Implementation

3.3.1 Solver choice

The MPC optimization problem that has been derived has the relevant characteristic of being well-suited for the Formula Student problem in analysis and of easy real time implementation, since as of today a large number of solvers have been developed that are able to efficiently solve a quadratic control problem.

In particular, these solvers can find the optimal solution over a Finite Time Horizon of a quadratic program (QP) expressed in the form:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{z}^T\mathbf{P}\mathbf{z} + \mathbf{q}^T\mathbf{z} \\ \text{subject to} & \mathbf{b_l} \leq \mathbf{A}\mathbf{z} \leq \mathbf{b_u} \end{array} \tag{3.21}$$

Where:

- $\mathbf{z}$ is the optimization variable
- $\mathbf{P}$ is called quadratic objective and is a positive semi-definite matrix
- $\mathbf{q}$ is the linear objective and is a positive semi-definite vector
- $\mathbf{A}$ is the linear constraint matrix
- $\mathbf{b_l}$ and $\mathbf{b_u}$ are the lower and upper bound vectors.

Between the most used linear solvers, especially in Formula Student applications, some relevant ones are for example OSQP [14], qpDUNES [15], HPIPM [16], qpOASES [17]. Between these solutions, OSQP has been adopted, since it is computationally fast, can be easily implemented in different programming languages such as C++ or MATLAB®, and has plenty of documentation and examples in GitHub.

3.3.2 MPC problem casting to a QP problem

In order to develop the actual code for the MPC controller, the Optimal Control Problem 3.20 has to be translated in the Quadratic Problem 3.21.
The optimization variable $\mathbf{z}$ of the QP problem is the vector that includes all the states and inputs of the Finite Time Horizon, starting from $k = 0$ to $k = N$. For the designed MPC controller, the optimized variable is expressed as:

$$\mathbf{z} = \begin{bmatrix} \mathbf{x}_0, & \mathbf{x}_1, \dots & \mathbf{x}_N, & \delta_0, & \delta_1, \dots & \delta_N, & \Delta\delta_1, \dots & \Delta\delta_N \end{bmatrix} \tag{3.22}$$

While $\mathbf{P}$ is a block-diagonal matrix containing the weight matrices associated to the states and inputs of $\mathbf{z}$, and $\mathbf{q}$ contains the weighted reference state:

$$\mathbf{P} = \begin{bmatrix} Q_0 & & & & & & & & \\ & \ddots & & & & & & & \\ & & Q_N & & & & & & \\ & & & R_{u,0} & & & & & \\ & & & & \ddots & & & & \\ & & & & & R_{u,N} & & & \\ & & & & & & R_{\Delta u,1} & & \\ & & & & & & & \ddots & \\ & & & & & & & & R_{\Delta u,N} \end{bmatrix} \tag{3.23}$$

$$\mathbf{q} = \begin{bmatrix} -\mathbf{Q}_0\mathbf{x}_0^{ref} \\ -\mathbf{Q}_1\mathbf{x}_1^{ref} \\ \vdots \\ -\mathbf{Q}_N\mathbf{x}_N^{ref} \\ \mathbf{0}_{nu} \\ \mathbf{0}_{ndu} \end{bmatrix} \tag{3.24}$$

This formulation for the objective function takes into account only for the non-constant terms of the cost function in 3.19: it can be shown that, for the generic simplified case of single state and single input (similar reasoning can be applied to multiple state and input), the two cost function formulations for $N = 0$ become:

$$\begin{aligned} \text{OP Cost Function} &= (x_0 - x_0^{ref})Q_0(x_0 - x_0^{ref}) + u_0 R_{u,0} u_0 = \\ &= Q_0 x_0^2 + Q_0 (x_0^{ref})^2 - 2Q_0 x_0 x_0^{ref} + R_{u,0} u_0^2 \\ \text{QP Cost Function} &= \tfrac{1}{2} z P z + q^T z = \\ &= \tfrac{1}{2}(Q_0 x_0^2 - 2Q_0 x_0 x_0^{ref} + R_{u,0} u_0^2) \end{aligned} \tag{3.25}$$

The two formulations are equivalent, except for the constant term $Q_0(x_0^{ref})^2$ that is discarded and the multiplying factor 1/2. This has no influence in the problem solution since the QP aims to minimize the cost function, and the minimum of the function is found for the same control sequence in both cases, the only difference is in the value of the cost function in the minimum.
The linear constraint matrix **A** is composed of two sub-matrices: the first is defined as Equality Matrix, and evaluates the dynamic evolution of the system by solving

for each time step the system of equations 3.13:

$$\begin{cases} \mathbf{I}_{nx}\mathbf{x}[0] = \mathbf{x}_0 \\ \mathbf{I}_{nu}\mathbf{u}[0] = \mathbf{u}_0 \\ -\mathbf{A}_d\mathbf{x}[k] + \mathbf{x}[k+1] - \mathbf{B}_d\mathbf{u}[k] - \mathbf{B}_d\mathbf{x}\boldsymbol{\Delta}\mathbf{u}[k] = 0 \\ -\mathbf{I}_{nu}\mathbf{u}[k] + \mathbf{I}_{nu}\mathbf{u}[k+1] - \mathbf{I}_{ndu}\boldsymbol{\Delta}\mathbf{u}[k] = 0 \end{cases} \tag{3.26}$$

The lower part is defined as Inequality Matrix, which is an identity diagonal matrix and applies the inequality constraints of the QP formulation to check for state and input constraints.

$$\mathbf{A} = \left[\begin{array}{cccccccccccccc} \mathbf{I}_{nx} & & & & & & & & & & & & & \\ & & & & \mathbf{I}_{nu} & & & & & & & & & \\ -\mathbf{A}_d & \mathbf{I}_{nx} & & & -\mathbf{B}_d & & & & & -\mathbf{B}_d & & & & \\ & \ddots & \ddots & & & \ddots & & & & & & \ddots & & \\ & & \ddots & \ddots & & & \ddots & \ddots & & & & & & \\ & & & -\mathbf{A}_d & \mathbf{I}_{nx} & & & & -\mathbf{B}_d & & & & -\mathbf{B}_d & \\ & & & & & & & & -\mathbf{I}_{nu} & \mathbf{I}_{nu} & & & -\mathbf{I}_{ndu} & \\ \hdashline \mathbf{I}_{nx} & & & & & & & & & & & & & \\ & \ddots & & & & & & & & & & & & \\ & & & \mathbf{I}_{nx} & & & & & & & & & & \\ & & & & & \mathbf{I}_{nu} & & & & & & & & \\ & & & & & & & \ddots & & & & & & \\ & & & & & & & & & \mathbf{I}_{nu} & & & & \\ & & & & & & & & & & \mathbf{I}_{nu} & & & \\ & & & & & & & & & & & \ddots & & \\ & & & & & & & & & & & & & \mathbf{I}_{nu} \end{array}\right] \tag{3.27}$$

In the particular case of the OSQP solver, for the Equality part of the matrix, instead of evaluating the solution of the equations 3.26 the solver applies to each equation the simultaneous set of inequalities:

$$\begin{cases} \mathbf{x}_0 \leq \mathbf{I}_{nx}\mathbf{x}[0] \leq \mathbf{x}_0 \\ \mathbf{u}_0 \leq \mathbf{I}_{nu}\mathbf{u}[0] \leq \mathbf{u}_0 \\ \mathbf{0}_{nx} \leq -\mathbf{A}_d\mathbf{x}[k] + \mathbf{x}[k+1] - \mathbf{B}_d\mathbf{u}[k] - \mathbf{B}_d\mathbf{x}\boldsymbol{\Delta}\mathbf{u}[k] \leq \mathbf{0}_{nx} \\ \mathbf{0}_{nu} \leq -\mathbf{I}_n u\mathbf{u}[k] + \mathbf{I}\mathbf{u}[k+1] - \mathbf{I}\boldsymbol{\Delta}\mathbf{u}[k] \leq \mathbf{0}_{nu} \end{cases} \tag{3.28}$$

As a result, the vectors $\mathbf{b_l}$ and $\mathbf{b_u}$ are derived as:

$$\begin{aligned} \mathbf{b_l} &= \begin{bmatrix} \mathbf{x}_0 & \mathbf{u}_0 & \mathbf{0}_{nx} & \mathbf{0}_{nu} & \mathbf{0}_{ndu} & \underline{\mathbf{x}} & \underline{\mathbf{u}} & \underline{\Delta \mathbf{u}} \end{bmatrix}^T \\ \mathbf{b_u} &= \begin{bmatrix} \mathbf{x}_0 & \mathbf{u}_0 & \mathbf{0}_{nx} & \mathbf{0}_{nu} & \mathbf{0}_{ndu} & \overline{\mathbf{x}} & \overline{\mathbf{u}} & \overline{\Delta \mathbf{u}} \end{bmatrix}^T \end{aligned} \tag{3.29}$$

Chapter 4

HiL Simulation and Controller Tuning

4.1 ROS2 integration and simulation

As anticipated in chapter 1.2.3, the MPC algorithm is developed in a ROS2 environment using the C++ language. In order to obtain a first feedback about the developed controller, a simulation model on the same environment must be developed. Since the objective of this simulation model is not to validate and properly tune the MPC parameters, but rather to check if the followed procedure is correct, it is not necessary to develop an highly non-linear and accurate model as in chapter 2.3. For this purpose, a single-track bicycle model has been adopted, similarly to what has been done for the prediction model, following 2.31. In order to have a first approach to the effects of the model mismatch between prediction and simulation model, the non-linear formulation is adopted, where the lateral forces $F_{y,f}$ and $F_{y,r}$ are evaluated following the Pacejka formulation 2.14, for which the parameters can be obtained from the .tir file of the actual pneumatics mounted on the prototype. Since the magic formula parameters are function of the vertical load and the friction coefficient, instead of taking into account for their variation both on the ROS2 simulation model and on the prediction model, constant values are assumed. In particular, unitary friction coefficients are assumed in both longitudinal and lateral direction, while for the vertical load, straight driving conditions at constant $v_x = 10m/s$ are assumed (taking the higher velocity of the expected operative range for the actual prototype, and therefore higher vertical load, the lateral force will be higher too, in the attempt to partially compensate for the prediction model error described in figure 2.11).

Notice that, when using a single-track model, the lateral force resulting from

Parameter	Front wheel ($i = f$)	Rear wheel ($i = r$)
$F_{z,i}$ [N]	657	769
B_y [-]	8.9290	8.9475
C_y [-]	1.2441	1.2441
D_y [N/rad]	886.48	1029.90
E_y [-]	0.0128	0.0200
V_y [N]	28.3	31.8
$C_{\alpha,i}$ [N/rad]	9847	11464

Table 4.1: Tyre parameters of the Pacejka "Magic Formula" equation, evaluated with static load experienced by each vehicle wheel during straight driving at $v_x = 10m/s$.

2.14 must be doubled, but the vertical offset V_y is null: in real vehicles, the tyres are mounted symmetrical between right and left side, in order to balance the vertical offset during straight driving. From 2.14 it can also be found that the cornering stiffness C_α is the derivative of the function for null slip, and is equal to $C_\alpha = B_y C_y D_y$.
The MPC C++ code has been developed, following the OP derived in 3.19 and translated into a QP following 3.21. The MPC node has been integrated into the ROS2 workspace developed by the Squadra Corse Driverless team, which contains the real time workspace deployed on the ACU and a simulation model which simulates the real prototype behaviour. The simulation model has been developed as a single track bicycle model, as the one used for the prediction model, with the difference that it is implemented in its non-linear formulation 2.32, using the Pacejka "Magic Formula" 2.14 to evaluate the $F_{y,i}$ contributions. This different model formulation allowed to have a first approach to the differences between the prediction model of the MPC and a more realistic behaviour of the real system. Moreover, the simulation workspace includes the entire software stack of the RTI implementation, including environmental perception, state estimation, SLAM algorithms, and communication. In particular, the feedback state used by the MPC comes from the SLAM algorithm and not directly from the simulation model: this will allow to take into account for noise and disturbances acting on the real time signals, and to make the overall project easily portable to the target ACU. This made possible to follow a software in the loop (SiL) validation approach, having developed the controller directly in C++ language as it will be deployed in the target hardware.
For an MPC algorithm, as a preliminary step, the controller frequency T_s and

an initial value for the prediction horizon N must be chosen. For the sampling time T_s, its value depends on the signals that are processed by the algorithm. The SLAM algorithm uses the odometry information running at $100Hz$, which is corrected using environmental perception data available at $10Hz$ frequency and data association algorithms. Between two optimization instants, the information about the state can be integrated with the odometry data. As a result the state information can be obtained up to $100Hz$ if the odometry error is sufficiently low, otherwise at $10Hz$ if only the optimized state is considered reliable. After some simulation and real time tests conducted by the Squadra Corse Driverless PoliTO team, the update frequency of the SLAM algorithm has been set to $20Hz$, that has proved to be a good compromise between data accuracy and computation cost of the overall pipeline on the ACU target hardware. This sets the limit for the controller frequency to $20Hz$, that translates into a sampling time of $T_s = 0.05s$, since higher frequencies means that the state is not updated between two sampling instants. For what regards the prediction horizon N, based on the analysis about the prediction model error of chapter 2.4.3, an upper bound of $1s$ has been evaluated for the prediction model to be sufficiently accurate. Based on the initial value set for the sampling time, the upper bound for the prediction horizon is set to $N = 20$. The initial SiL tuning procedure of the controller's parameters $\mathbf{Q}$, $\mathbf{R_u}$ and $\mathbf{R_{du}}$ has been performed on the ROS2 simulation environment in two different conditions: first, using nominal conditions both for the plant and the localization algorithm, then including Gaussian noise on different levels of the simulation, in particular for the simulated sensors such as LiDAR, Cameras (used by the SLAM localization algorithm), IMU and steering sensor, with standard deviations typical of the real prototype.

Here are presented the results and performances achieved by the controller. The performance is evaluated using the $CrossTrack*$ error absolute value as parameter, and its RMS value during 1 lap of the circuit. The SiL tuning procedure has been conducted at constant longitudinal velocity, since in the linearized model the influence of longitudinal acceleration has been neglected. In the next tuning process, involving a more complicated simulation model with disturbances, the velocity will be not constant as it is in the real time prototype scenario, with the objective to obtain a final value for the design parameters that try to achieve robust stability and performance of the controller.
About the weight matrices, some preliminary considerations have to be made. Based on the applied cost function 3.19 to be minimized, since the tracking of the x^{ref} coordinate has the same importance of the tracking of y^{ref}, equal weight q_{xy} is assigned to the first two elements of the $\mathbf{Q}$ matrices. Since no reference is provided for the remaining states, null weight is assigned for the remaining elements. Moreover, the trade-off between performances and actuation cost is not expressed as the absolute value of the weights but rather as the ratio between these values

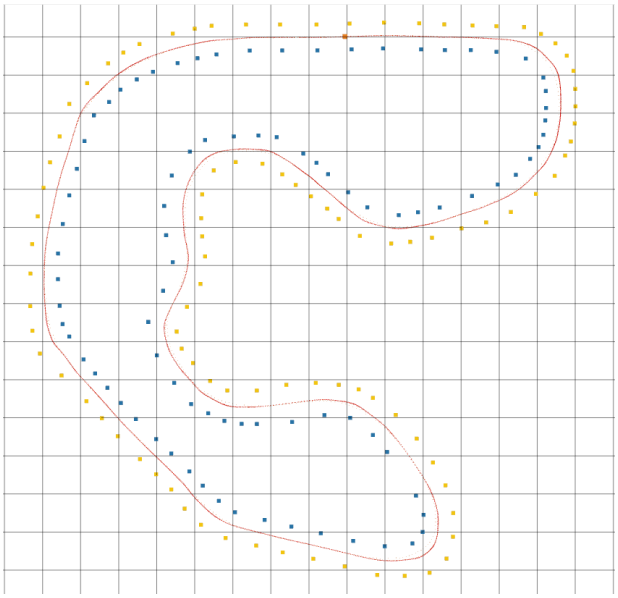

Figure 4.1: Simulation race track, with cones (in blue and yellow) and Reference Trajectory (in red), 5 meters for each grid's square.

$\frac{r_\delta}{q_{xy}}$ and $\frac{r_{\Delta\delta}}{q_{xy}}$. The weight matrices are then expressed and tuned as:

$$\mathbf{Q} = \begin{bmatrix} q_{xy} & & & & & \\ & q_{xy} & & & & \\ & & 0 & & & \\ & & & 0 & & \\ & & & & 0 & \\ & & & & & 0 \end{bmatrix}, \mathbf{R}_u = \begin{bmatrix} r_\delta \end{bmatrix}, \mathbf{R}_{\Delta u} = \begin{bmatrix} r_{\Delta\delta} \end{bmatrix} \tag{4.1}$$

Notice that q_{xy} refers to a state, the distance, that is of the order of magnitude of $1 - 10$, while the steering and its rate are of the order of magnitude of $0.01 - 0.1$. For this reason it is preferred to adopt $q_{xy} = 1$, so that the weights r_δ and $r_{\Delta\delta}$ can be integers greater than 1 in order to have comparable contributions on the cost function 3.19.
Based on the prototype application, the steering actuator full range does not represent a problem to be controlled, but referring to the analysis made in 2.4.3 a large value for the steering increases the presence of errors between the prediction model and the actual system. In principle it derives that keeping the steering input low will decrease the prediction error, but the effect on the $CrossTrack*$ error is not defined yet. Simulations in nominal conditions have been conducted in order to establish the effects of a positive weight r_δ on the $CrossTrack*$ error, shown in Figure 4.2: increasing the command input weight, allows to obtain an

overall smaller steering input δ as expected, but the performance parameter rapidly increases.

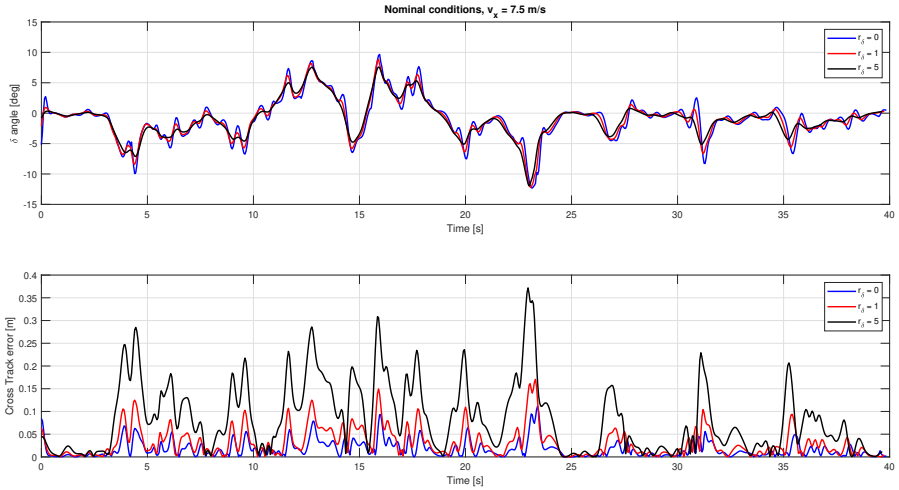

Figure 4.2: Effects of r_δ on the control input and on the performance parameter Cross Track error in nominal conditions.

In Figure 4.3 the simulation results are displayed, over a single lap of the circuit of Figure 4.1, in nominal conditions with constant longitudinal velocity. In Figure 4.4 the comparison between the nominal and disturbed conditions are shown.

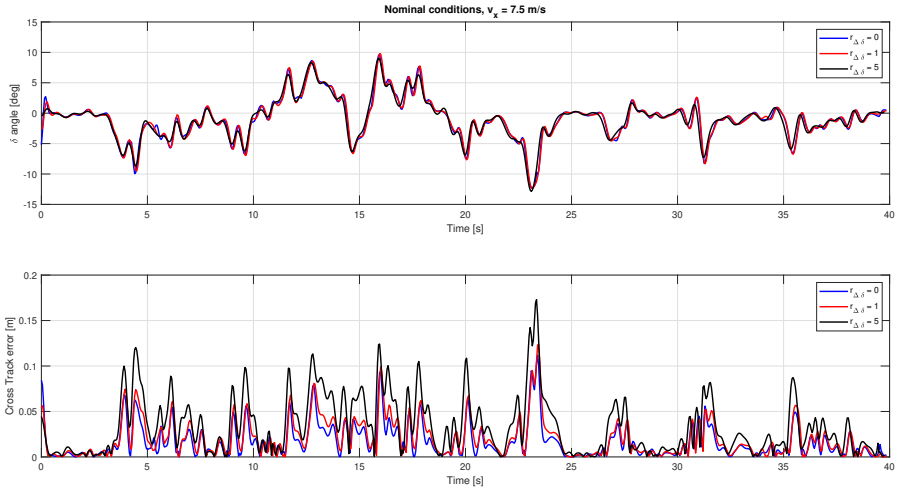

Figure 4.3: Effects of $r_{\Delta\delta}$ on the steering angle δ and on the performance parameter Cross Track error in nominal conditions.

It can be noticed that, while in nominal conditions $r_\delta = 0$ provides the best performances, when Gaussian disturbance is added on the overall simulation

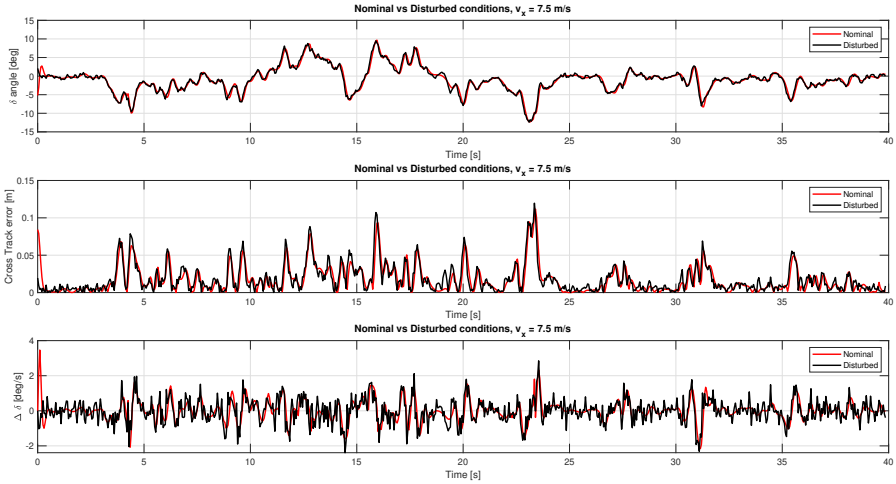

Figure 4.4: Comparison between Nominal and Disturbed conditions.

environment small corrections have to be made on δ. In order to filter out this contributions, that are feasible for the real actuator but are undesired and can lead to an excessive number of corrections, $r_{\Delta\delta}$ is increased, making the actuation smoother in exchange for a slightly greater $CrossTrack*$ error. The RMS values for the $CrossTrack*$ errors are displayed in table 4.2

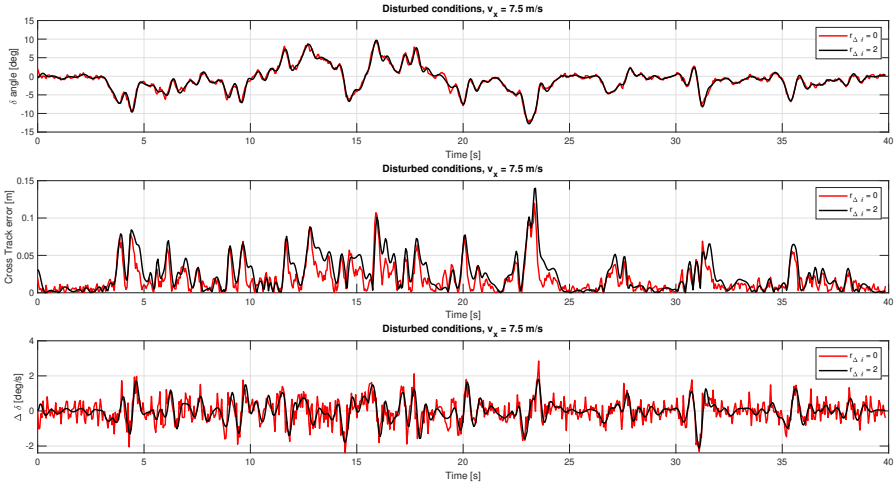

Figure 4.5: Effects of $r_{\Delta\delta}$ on the performance parameter and noise rejecting in disturbed conditions.

The results obtained from the SiL controller tuning process show that, while in nominal plant conditions it is not necessary to take into account for the input contribution in the tuning process, thus keeping null r_δ and $r_{\Delta\delta}$, when considering

Cross Track Error [m]		
	Nominal Conditions	Disturbed Conditions
$r_{\Delta\delta} = 0$	0.025	0.026
$r_{\Delta\delta} = 1$	0.027	0.029
$r_{\Delta\delta} = 2$	0.033	0.034
$r_{\Delta\delta} = 5$	0.045	0.053

Table 4.2: Cross Track error comparison between nominal and disturbed conditions during SiL simulation

a disturbed plant, much more similar to the real system and with higher differences with respect to the prediction model, the simulated noise effect on the steering angle can be filtered by weighting the steering rate.
After a trial and error procedure, the following weight matrices are considered as the best trade-off between performances and disturbance filtering:

$$\mathbf{Q} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 0 & & & & \\ & & & 0 & & & \\ & & & & 0 & & \\ & & & & & 0 & \\ & & & & & & 0 \end{bmatrix}, \mathbf{R}_u = \begin{bmatrix} 0 \end{bmatrix}, \mathbf{R}_{\Delta u} = \begin{bmatrix} 2 \end{bmatrix} \tag{4.2}$$

4.2 HiL Simulation and Tuning

The last part of the tuning process presented in this thesis work consists in performing an hardware in the loop simulation, where the controller is deployed directly on the target ACU inside the ROS2 workspace that runs on the prototype. In this way, Real Time performances can be evaluated in therms of computational cost of the controller code. The data stream of the LiDAR and Cameras are simulated by a dedicated ROS2 node, that is the same of the Simulator used in the previous chapter, which based on the actual state of the vehicle simulates the cones perception including disturbance about their position, similar to the one actually measured from the real prototype on-track tests. The evaluated outputs of the ACU are sent via the Serial-to-CAN interface to the dSPACE ECU, in the same configuration as in real prototype. As shown in Figures 1.5 and 3.1, the real time ECU runs the low level controller (LLC) used on the prototype, which includes TV, Steering Controller and State Estimation, together to other fundamental applications like the State machine, I/O communication and CANbus communication. All of this components of the LLC have been developed and fully tested on the prototype

by the SCD team, using the MATLAB Simulink environment. The LLC runs on a single Task, which is the set of operations that have to be performed by the micro-controller (μC) inside a predifined time window, corresponsing to the fundamental time at which the LLC is desired to run. If the Task is executed inside that window, the hardware is said to be running on real time, while if the Task can not be completed inside the time window, an Overrun occurs and the platform is not running in real time. For the VaLentina prototype, the fundamental time of the ECU has been set to $200Hz$.
In order to perform HiL validation, the vehicle must be simulated on a real time hardware, using a simulation model that is as close as possible to the real system. Usually, the vehicle model runs on a separate real time hardware, which fully simulates the prototype's on-board communication from the ECU to all the subsystems (like boards, sensors, actuator drivers) and includes the system model which returns the system dynamics. This validation process can be expensive since it requires a separate Real Time Hardware for the simulation. In the case of the VaLentina prototype, the on-board ECU is a real time system that is also used for prototyping and testing both in-vehicle and on-bench, thus it has high computational capabilities: in particular the dSPACE MicroAutoBox III runs on a quad-core processor, where each core can run a different Task to which a different MATLAB Simulink model can be assigned, and each one can communicate between each other and access the I/O interface. Since the LLC currently running on the ECU has been developed on a single Task running at $200Hz$, the simulation environment can be developed on a separated Task on the same hardware, avoiding the use of a separate Real Time system thus reducing the HiL validation process' cost. The two MATLAB Simulink models can communicate internally on the dSPACE, so only additional blocks have to be added to the original LLC to enable internal communication, thus keeping it almost unchanged with respect to the real configuration. Of course, a suitable state machine has been developed on the Simulation Task model in order to emulate the start-up procedure of the vehicle. The full hardware setup has been reproduced on the test bench shown in Figure 4.6: the NVIDIA Jetson AGX Orin (the ACU) communicates via Kvaser Leaf Light V2 (the serial-to-CAN interface) to the dSPACE MicroAutoBox III (the ECU), and a monitor is used for debug purpose and system functionality checks.

The vehicle model used in the Simulation Task is the one developed in chapter 2.3, where the input torques come directly form the torque vectoring deployed on the LLC, while the steering angle is converted from the Steering controller output, that is expressed in increments of the stepper motor used from the steering actuator, to steering wheel degrees that is the actual sensor measure on the prototype and is the input of the simulation model. The Simulation Task runs at $200Hz$ because the model complexity, in particular the MFeval functions, do not allow lower execution

Figure 4.6: HiL setup: the NVIDIA Jetson AGX Orin (in the middle) communicating with the dSPACE MicroAutoBox III (on the left) via Kvaser Leaf Light V2 (on the bottom).

times on the dSPACE without experiencing Overrun.
The first simulation is conducted with the same parameters as in the SiL best simulation with weight matrices 4.2, constant longitudinal velocity, simulated Gaussian error on perception sensors and odometry, and prediction horizon $N = 20$.

The first and most important difference between the two simulations occurs on the steering angle signal which is higher in tight curves for the HiL simulation with respect to the SiL case. This is explained by the understeering behaviour of the real vehicle, which is well represented by the Simulation model on MATLAB Simulink since it has been tuned using track test data. The understeering of the real prototype is partially compensated by the torque vectoring algorithm, which uses as reference yaw rate the single track model one, but of course the TV controller correction is not instantaneous. Anyway, the MPC compensates for this mismatch between the prediction model and the actual controlled system increasing the control input if the expected dynamic is not satisfied.
The Cross Track error behaviour is very similar between the two simulation conditions, with a RMS value of $0.035m$ with respect to the $0.034m$ of the SiL simulation,

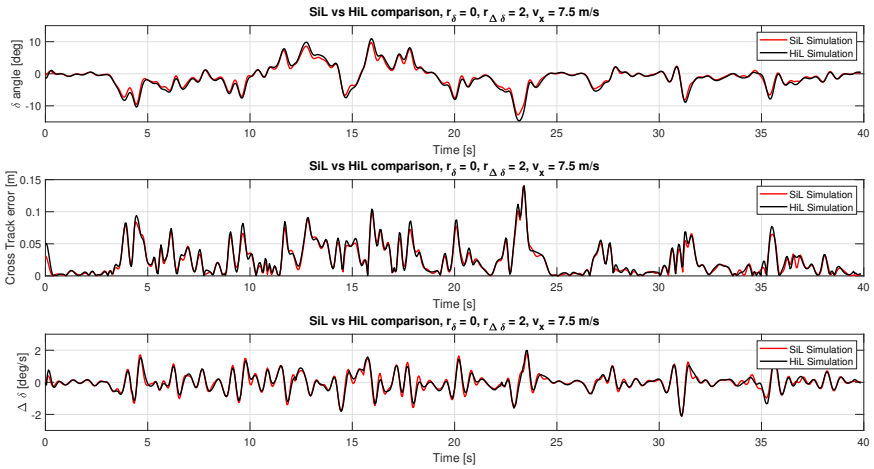

Figure 4.7: Comparison between SiL simulation and HiL simulation.

so the obtained performances are similar in both cases. For the sake of completeness, simulations have been conducted with the rate input weight $r_{\Delta\delta}$ in the range $0:5$ and the results compared to Table 4.2: also in the HiL case the value $r_{\Delta\delta} = 2$ has proved to be the best trade off between performances and disturbance filtering.
In order to compensate for the understeering behaviour previously discussed, a possible approach could be to tune also the r_δ parameter in order to obtain smaller steering angles: similar to the results obtained in the SiL case and shown in Figure 4.2, the obtained results is to have a more similar behaviour to the SiL case, but the Cross Track Error increases. Since the objective is to optimize performance and disturbance rejection, the actual behaviour of the steering angle is not relevant, as soon as it remains in the actuation limit of $\pm 20\,\text{deg}$.

Having tuned the controller's parameters and obtained robust performances in the case of disturbed plant, a further step is to include the the velocity profiler as a reference generator for the longitudinal velocity. In this way the MPC robustness can be tested also in non-null longitudinal acceleration conditions, which is in contrast with the prediction model hypothesis (as discussed in 3.1.1), thus representing an unmodelled disturbance.

The obtained results are shown in Figure 4.8, with the longitudinal velocity shown in Figure 4.9. First important result is that, while the Cross Track Error RMS value has slightly increased, from $0.035m$ for the constant velocity case to $0.037m$, due to the unmodelled longitudinal acceleration that the MPC can not properly evaluate, its maximum value has decreased instead. The maximum Cross Track Error could be found in the tightest turn of the circuit, where the vehicle

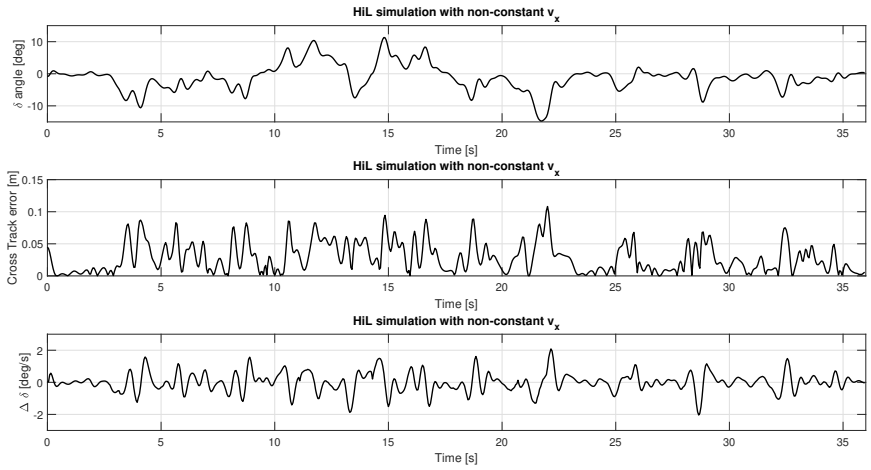

Figure 4.8: Complete HiL Simulation including longitudinal and lateral control, MPC performances.

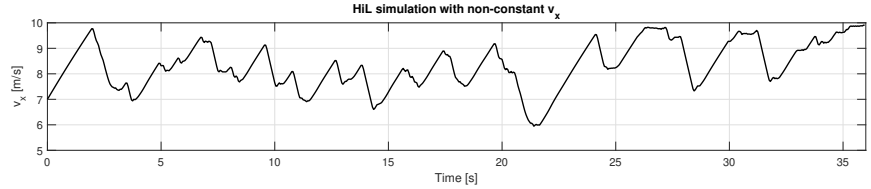

Figure 4.9: Complete HiL Simulation including longitudinal and lateral control, longitudinal dynamics.

was forced to turn at $7.5m/s$, while with the velocity profiler the longitudinal velocity is kept lower, providing more controllability of the system. Even with higher velocities, the designed MPC has proved to achieve robust performances in the presence of non-null longitudinal acceleration, which is the most realistic case for the real prototype.

Having obtained satisfying performances and noise rejection, the prediction horizon N can be also tuned, with the objective to reduce the computational time of the optimization problem. This procedure is conducted by gradually decreasing the prediction horizon, that will have as a consequence a decrease in the performances, that will be acceptable up until a certain value of N. The obtained values for RMS cross track error and Computational time of the whole MPC node exhecution are shown in Table 4.3.

It is easily shown that the rate of performance degradation is very low up to $N = 10$, while the computational time is proportional to N. When further reducing

	CrossTrack RMS [m]	Computational time [10^{-3} s]
$N = 20$	0.037	1.029
$N = 15$	0.037	0.831
$N = 10$	0.039	0.585
$N = 5$	0.054	0.294

Table 4.3: Cross Track error and computational time comparison when decreasing the prediction horizon.

the prediction horizon up to $N = 5$, the Cross Track Error experiences a rapid increase. In order to keep the performances in a "linear region" of the performance degradation rate, with a sufficient margin for further tuning in the real prototype, it is preferred to keep $N = 15$, since the computational time of the controller is very low, especially if compared with other nodes running on the ACU.

Chapter 5

Conclusions

5.1 Results

In this thesis work, a simple but effective solution for the problem of path tracking of an autonomous vehicle has been presented, with the particular objective of implementing the controller on a real Formula SAE prototype. The problem has been solved by implementing a linear model predictive controller which satisfies both requirements of computational efficiency ($1.029 * 10^{-3}s$ on the target hardware) and performances (expressed in therms of mean Cross Track Error on a simulated circuit). The controller has been tested extensively in two steps. The first validation has been conducted using a software in the loop approach, by deploying the controller as a ROS2 node developed in C++ language, in the final configuration as in the target hardware, and performing the controller parameters tuning using a virtual simulator inside the ROS2 simulation workspace. The second validation step has been conducted using an hardware in the loop approach, where the MPC has been deployed on the workspace of the target hardware (called ACU), recreating the actual communication with the downstream architecture of the prototype, using the same Real Time Hardware (called ECU) that is deployed in the prototype. The simulation environment has been developed using the same electronic control unit, since it has high computational capabilities suitable for real time testing, which allowed to keep the costs for an HiL simulation to the minimum, where the self-developed vehicle model (which has been validated using track test data) has been integrated downstream the control pipeline.This process allowed a full validation of the software stack starting from the cones identification to the low level control action.
The result of this thesis work is a fully tuned and integrated MPC on the target hardware, which is ready for testing on the real prototype, and has been already

tested in stressed conditions to evaluate the controller's robust stability and performances in the presence of the disturbances that have been observed are affecting the real prototype.

5.2 Future works

The results presented in this thesis work are well promising in terms of Real Time application on the Formula Student prototype of the Squadra Corse Driverless team. Next step for the controller is to organize track tests and have a final validation for the proposed solution.

One of the major drawbacks found during HiL simulation was the understeering behaviour of the Simulation model and the actuation delay of the TV. In order to overcome this, a possible future development could be to use the predicted yaw rate of the prediction model as yaw reference, instead of the ideal yaw rate of a single track model to which is applied the actual steering angle. If properly tuned, this change of the yaw reference can guarantee a "predictive" behaviour of the TV when the MPC is running, allowing more predictable performances and a more realistic prediction model if compared with the real system.

Another possible future approach can also be to develop a MPC that accounts for both lateral and longitudinal dynamic control, with the objective to optimize lap time and have a single controller instead of two different pipelines.

Finally, the designed controller has proved to be computationally efficient and adapt for Real Time applications, but to achieve this goal the prediction model has been linearized and simplified, and this process brings errors in the predicted dynamic. Since there is wiggle room for improvements in terms of available computing power, a possible future step is to implement non-linear solutions, supported by last generation non-linear solvers (ACADOS) that present slightly higher computational times in exchange for higher accuracy.

Bibliography

[1] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. «Simultaneous Localization and Mapping With Sparse Extended Information Filters». In: *International Journal of Robotics Research* (2004) (cit. on p. 4).

[2] Angelo Bonfitto, Stefano Feraco, Marco Rossini, and Francesco Carlomagno. «Fuzzy Logic Method for the Speed Estimation in All-Wheel Drive Electric Racing Vehicles». In: 23 (Jan. 2021), B117–B129 (cit. on pp. 6, 19).

[3] Steven M. LaValle. *Planning Algorithms*. Cambridge, England: Cambridge University press, 2006 (cit. on p. 7).

[4] Guiggiani M. *The science of vehicle dynamics*. Springer Nature, 2019 (cit. on p. 13).

[5] Pacejka H.B. *Tyre and Vehicle Dynamics*. Oxford, UK: Butterworth-Heinemann, 2006 (cit. on p. 13).

[6] Pacejka H.B. Besselink I.J.M. Schmeitz A.J.C. «An improved Magic Formula/Swift tyre model that can handle inflation pressure changes». In: *Taylor Francis* 48 (2010), pp. 337–352 (cit. on p. 13).

[7] Metz L.D. Milliken D.L. Kasprzak E.M. *Race Car Vehicle Dynamics*. SAE International, 2013 (cit. on p. 17).

[8] Crocombe A. Sanpo' E. Sorniotti A. «Chassis Torsional Stiffness: Analysis of the influence on Vehicle Dynamics». In: *SAE International* 0094 (2010) (cit. on p. 17).

[9] Furlan M. *MFeval*. 2021. URL: `https://www.mathworks.com/matlabcentral/fileexchange/63618-mfeval` (cit. on p. 18).

[10] Genta G. *Motor Vehicle Dynamics: Modelling and Simulation*. Singapore: World Scientific Publishing Company, 1997 (cit. on p. 21).

[11] Horowitz Packard Poola. «Dynamic System and Feedback». In: Barkeley, California: Department of Mechanical Engineering University of California, 2002. Chap. 19 (cit. on p. 27).

[12] Juraj Kabzan et al. «AMZ Driverless: The Full Autonomous Racing System». In: *CoRR* abs/1905.05150 (2019). arXiv: 1905.05150. URL: http://arxiv.org/abs/1905.05150 (cit. on p. 28).

[13] D'Andrea. *Signals and Systems*. Zurich, Switzerlanda: ETH Zurich, 2018 (cit. on p. 38).

[14] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. «OSQP: an operator splitting solver for quadratic programs». In: *Mathematical Programming Computation* 12.4 (2018), pp. 637–672 (cit. on p. 44).

[15] M. Diehl J. V. Frasch S. Sager. «A parallel quadratic programming method for dynamic optimization problems». In: *Mathematical Programming Computation,* 2003.02547 (2015) (cit. on p. 44).

[16] M. Diehl G. Frison. «HPIPM: a high-performance quadratic programming framework for model predictive control». In: *arXiv preprint* 7.3 (2020), pp. 289–329 (cit. on p. 44).

[17] M. Diehl J. V. Frasch S. Sager. «A parametric active-set algorithm for quadratic programming». In: *Mathematical Programming Computation,* 6.4 (2014), pp. 327–636 (cit. on p. 44).