TÉCNICO LISBOA

Model Predictive Control with a Neural Network Model of a Formula Student Prototype

Henrique Manuel Caldeira Pimentel Furtado

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisor: Prof. Miguel Afonso Dias de Ayala Botto

Examination Committee

Chairperson: Prof. Paulo Jorge Coelho Ramalho Oliveira
Supervisor: Prof. Miguel Afonso Dias de Ayala Botto
Member of the Committee: Prof. Mário José Gonçalves Cavaco Mendes

November 2020

Acknowledgments

First of all, I would like to thank my supervisor, Professor Miguel Ayala Botto, for his availability and support throughout this work, which would not have been the same without all the shared knowledge and advice given.

I would also like to thank the current and former teams of FST Lisboa for all the help and information provided, as well as for all the friendships and moments experienced.

To my family and girlfriend for all the love and support and also to my parents for giving me the opportunity to study and to reach my goals.

Lastly, but not least, to my friends for their motivation and everlasting friendship and without whom this 5 years would not be the same.

Resumo

Os carros autónomos estão a tornar-se cada vez mais populares entre o público em geral e as competições de Formula Student encorajam cada vez mais o desenvolvimento desta tecnologia. O interesse por esta área tem aumentado nos últimos anos, sendo motivado pela segurança e conveniência acrescidas, bem como a redução de custos.

O objetivo deste trabalho é projetar um controlador através da metodologia de Controlo por Modelo Preditivo (MPC), para um protótipo de Formula Student, que funcione nas respetivas pistas de competição. A solução proposta consiste na implementação de um MPC não linear com um modelo de previsão baseado numa Rede Neuronal Artificial (RNA).

Um modelo dinâmico não linear do protótipo foi apresentado e desenvolvido, sendo depois este modelo usado para implementar e simular os algoritmos de controlo. Este modelo dinâmico foi validado introduzindo como entradas dados do ângulo de direção e de velocidade e comparando as saídas simuladas com dados reais do protótipo. Foi feita a identificação deste modelo dinâmico, utilizando uma RNA com base em sinais adequados que cobrem o envelope de funcionamento do sistema. As trajetórias de referência foram parametrizadas em relação ao comprimento de arco de um conjunto de polinómios de terceira ordem e o erro de contorno em relação à trajetória foi definido. Foi também definida uma função de custo que otimizando os estados previstos pelo modelo de RNA, minimiza o erro de contorno enquanto maximiza a progressão ao longo da pista. O controlador MPC desenvolvido seguiu com sucesso as trajetórias de referência. Explorou-se ainda o treino de uma RNA que recebe como entrada a trajetória de referência e os estados do sistema e devolve as ações de controlo respetivas, substituindo assim o MPC e a necessidade de calcular otimizações em tempo real.

Palavras-chave: Controlo por Modelo Preditivo, Redes Neuronais Artificiais, Formula Student, Condução Autónoma

Abstract

Autonomous vehicles are becoming increasingly popular amongst the general public and Formula Student competitions are encouraging the development of this technology. The interest in this area has increased over the last years, motivated by the added safety and convenience, as well as cost reduction.

The objective of this work is to design a controller with the Model Predictive Control (MPC) methodology for a Formula Student prototype that works on the respective competition tracks. The proposed solution consists in implementing a nonlinear MPC with an Artificial Neural Network (ANN) prediction model.

A nonlinear prototype dynamical model was presented and developed, on which control algorithms were implemented and simulated. This dynamical model was validated by introducing in its inputs recorded data from the steering angle and velocity and comparing the simulated outputs with real data from the prototype. The dynamical model was then identified, using an ANN based on adequate signals that capture the entire operation envelope of the system. Reference trajectories were parameterized with respect to the arc length of a series of third order polynomials and the contour error relative to the trajectory was defined. A cost function optimizes the evolution of the states predicted by the ANN model, such that the contour error is minimized while the progression along the track is maximized, during the prediction horizon. The developed MPC controller tracked well the reference trajectories. A different ANN is explored, receiving as inputs the reference trajectory and the states of the system and returning as outputs the respective control actions, thus replacing the MPC and the need of optimizing in real-time.

Keywords: Model Predictive Control, Artificial Neural Networks, Formula Student, Autonomous Driving

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

α	Slip angle.
β	Sideslip angle.
Δ	Variation.
δ	Steering angle.
γ	Camber angle.
ω	Angular velocity.
Φ	Angle of line tangent to the trajectory, with respect to the X-axis.
ϕ, θ, ψ	Roll, pitch and yaw angle.
ρ_{air}	Air density.

Roman symbols

$\dot{h}_i$	Vertical velocity of wheel i $(i = 1, 2, 3, 4)$.
$\overline{v_i}$	Velocity vector of wheel i $(i = 1, 2, 3, 4)$.
$\overline{v}$	Velocity vector of the CoG.
a	Distance between the front wheels and the CoG.
A_x^{proj}	Projected section of the car, perpendicular to the X direction of velocity.
A_z^{proj}	Projected section of the car, perpendicular to the Z direction of velocity.
b	Distance between the rear wheels and the CoG.
c	Distance between the left wheels and the CoG.
C_d	Coefficient of drag.
C_l	Coefficient of lift.
C_i	Adjusted damping coefficient of wheel i $(i = 1, 2, 3, 4)$.

C_{pi} Damping coefficient of tyre i $(i = 1, 2, 3, 4)$.

C_{si} Damping coefficient of wheel i $(i = 1, 2, 3, 4)$.

d Distance between the right wheels and the CoG.

d_i Position vector from CoG to wheel i $(i = 1, 2, 3, 4)$.

e_n Contouring error normal component.

F Force.

G_i Ground height of wheel i $(i = 1, 2, 3, 4)$.

G_e Equivalent impedance of inverter and motor assembly.

H_c Control horizon.

h_i Vertical position of wheel i $(i = 1, 2, 3, 4)$.

H_p Prediction horizon.

i Current.

I_Φ Moment of inertia around x-axis.

I_ψ Moment of inertia around z-axis.

I_θ Moment of inertia around y-axis.

J Cost function.

J_w Rotational inertia of the wheel.

K_i Equivalent stiffness of wheel i $(i = 1, 2, 3, 4)$.

K_t Motor torque constant.

K_{pi} Equivalent stiffness of tyre i $(i = 1, 2, 3, 4)$.

K_{si} Equivalent stiffness of spring i $(i = 1, 2, 3, 4)$.

M Moment.

m Vehicle mass.

m_{ch} Vehicle sprung mass.

m_u Vehicle unsprung mass of wheel i $(i = 1, 2, 3, 4)$.

R_w Wheel radius.

s Longitudinal slip ratio.

s_s Lateral slip ratio.

T Torque.

t Arc length parameter.

t_r Track width of the vehicle.

T_s Sampling time.

v Velocity magnitude.

v_i Velocity magnitude of wheel i.

v_x Longitudinal velocity of the CoG .

v_y Lateral velocity of the CoG .

V_{in} Voltage input.

v_{ref} Reference longitudinal velocity of the CoG .

w_b Wheelbase of the vehicle.

w_i Tuning weight of parameter i of cost function.

X, Y Position of the vehicle in the global frame.

Z Centre of gravity vertical position.

$\mathbf{u}(k)$ Input vector of model at instant k.

$\mathbf{y}(k)$ Output vector of model at instant k.

Subscripts

F, R Front and rear.

FL, FR, RL, RR Front left, front right, rear left and rear right.

i Quarter suspension index.

mot Motor.

ref Reference.

$stdd$ Standardized.

x, y, z Cartesian components.

Superscripts

T Transpose.

Glossary

ANN	Artificial Neural Network.
CoG	Centre of Gravity.
DOF	Degrees of freedom.
FS East	Formula Student East competition in Hungary.
FSG	Formula Student Germany competition.
FSS	Formula Student Spain competition.
FST Lisboa	Formula Student team from University of Lisbon.
FST09e	Ninth prototype of FST Lisboa.
GPS	Global Positioning System.
MIMO	Multiple Input Multiple Output.
MPC	Model Predictive Control.
MR	Motion Ration.
MSE	Mean Squared Error.
NARX	Nonlinear Auto Refressive eXogenous.
NFIR	Nonlinear Finite Impulse Response.
OEM	Original Equipment Manufacturer.
PID	Proportional-Integral-Derivative.
RCA	Radio Corporation of America.
RPM	Rotations per minute.

SISO Single Input Single Output.

Chapter 1

Introduction

In recent years, the buzz about autonomous vehicles has enabled every major OEM to pool in resources to develop their own unique solutions in tackling the challenges of self driving vehicles. With major advancements in the fields of vehicle-to-vehicle communication and intelligent transport, many vehicles have already displayed semi-autonomous behaviour and are moving steadily towards full-autonomy.

The emergent interest in autonomous vehicles, as evidenced by the way Formula Student competitions are promoting the development of this technology, provides a good opportunity to explore this area.

1.1 Motivation

In Formula Student (FS) competitions, students are challenged to design, build and test a race car according to a specific set of rules stated by Formula Student Germany (FSG) and Formula Society of Automotive Engineers (FSAE). The Formula Student Team of Técnico Lisboa (FST Lisboa) has been developing cars for these competitions since 2001. In 2019, the team, with more than 40 members, began pursuing two projects simultaneously for its 10^{th} generation of cars. It is building a new electric prototype - FST10e - and at the same time empowering FST09e, the car that competed in the summer of 2019, with autonomous features, which will become FST10d. The base vehicle, FST09e [1] , is the 5^{th} electric prototype of FST Lisboa, and it achieved 9^{th} place overall in FSG 2019 (Figure 1.1), one of the most competitive events on the Formula Student season. I was happy to be part of the team from September 2018 to September 2020, which also motivated me to develop this work.

Each FS competition has three static events and four dynamic events. In the static events the team defends its design, manufacturing, cost and business plan, while in the dynamic events, where the performance of the car is evaluated, are as follows:

- Acceleration: a straight line with a length of 75 m;

[1] For more information about FST09e and the team visit fstlisboa.com and for a video of FST09e in action visit https://youtu.be/r-Ra5g7uQGw

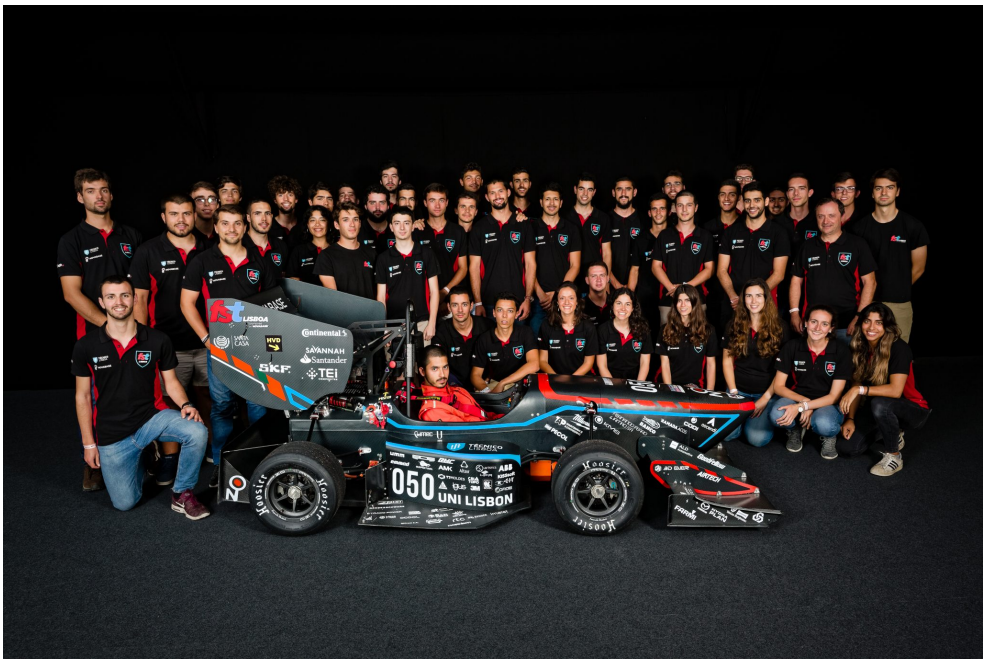

Figure 1.1: FST09e team at Formula Student Germany 2019

- Skidpad: an 8-shaped track to test lateral acceleration;
- Autocross: a single lap around a handling track with an approximate length of 1 km;
- Endurance/Trackdrive & Efficiency: several laps around a closed circuit (Trackdrive is for Driverless cars and Endurance for the remaining).

Up until 2019, most FS competitions allowed 3 separate car classes: combustion, electric and driverless. Vehicles competing in the driverless class had to complete all four dynamic events mentioned autonomously. Following the growing investment from the automotive industry in autonomous driving [1], the organization of FSG has decided that from 2021 onwards its competition won't have a separate driverless class. Instead, all participating vehicles will have to complete the acceleration event in driverless mode and from 2022 onwards, driverless is also mandatory for the skidpad. If teams are not capable of completing these driverless events, they will reduce their chances of scoring many points, hence reinforcing the incentive to develop this technology.

Many top tier companies in the automotive industry sponsor FS competitions and also provide officials and design judges for the competitions. These companies also help to establish the path of the competition to fulfill their needs in terms of the skills of their future engineers.

The January 1958 issue of Electronic Age, the quarterly magazine from Radio Corporation of America (RCA), included its vision of the "highway of the future", where the driver simply presses the 'Electronic Drive' button and allows the vehicle to take control of driving, while the human driver may read or even catch up on office work. Of course, this was simply their vision of the future at the time, propelled by the first step they achieved towards this goal, which reporters got to experience for themselves two

and a half years later. On a test track in Princeton, cars drove autonomously, using an electrical cable incorporated in the road, which was detected by sensors on the front bumpers. The cable carried signals warning of obstructions ahead (such as road work or a stopped vehicle), and the car had autonomy to apply its brakes or switch lanes accordingly. The dashboard contained a special receiver which would interrupt the radio of the car to announce information about upcoming exits [2].

The main motivation behind this ”highway of the future” was the increasing number of road traffic fatalities. According to the RCA vision, all highway driving would be autonomous in just a decade or two, with human drivers only intervening when their exit was near. Well over 50 years later, autonomous driving is still a challenge to implement in the real world, despite the advantages it may bring of improved safety, convenience, efficiency and reduced costs compared to conventional vehicles.

The 5 development stages of a fully autonomous vehicle have been defined by SAE International in 2014 with the J3016 Standard. In recent years, many companies like Tesla, Daimler and Uber have been developing autonomous technology, but in the present day there is no vehicle available to the public with level 5 full automation [3], the most autonomous level, where the vehicle takes care of all aspects of driving in every road and environmental condition, according to the J3016 Standard. Currently, the public can purchase level 3 technology, which despite being able to operate autonomously under certain conditions, still requires a driver in the vehicle ready to take control at all times with notice.

1.2 State of the Art

FST Lisboa is currently developing its first autonomous vehicle, so more emphasis is being placed on reliability rather than on performance. Hence, the control algorithms being implemented are relatively simple and based on PID controllers having a large improvement margin. A general structure for an autonomous vehicle is shown in Figure 1.2, where it can be seen that in order to implement any control strategy, the previous stages need to be working reliably.

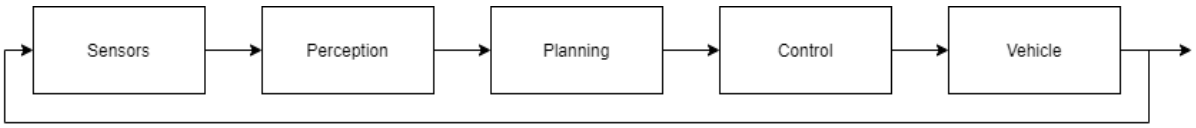

Figure 1.2: Autonomous vehicle pipeline

Some FS teams have successfully implemented Model Predictive Control (MPC) in their cars, like AMZ from Zurich [4] and Oxford Brookes Racing [5]. MPC dates back to the 1970s and it started to emerge industrially in the 1980s [6]. In recent years, with the increasing computing power of microprocessors, its use has spread to many other fields including automotive and aerospace, being used for

example in power system balancing models [7] and in power electronics [8].

MPC has the advantage of computing the optimal solution by using a prediction model which allows the controller to deal with a replica of the system dynamics, improving the control quality [9]. It also has the advantage of allowing constraints on the inputs, outputs, and states of the system. The prediction model typically involves modelling the physics of the system through mathematical expressions. Most MPC algorithms are based on a linear model of the system. When the goal is to maintain the system at a desired steady state (which happens often in industrial processes), rather than moving rapidly between different operating points, a precisely identified linear model is sufficiently accurate around a certain operating point [10]. If the system is highly nonlinear and large frequent disturbances are present, a nonlinear model is necessary to describe the dynamics [11].

In situations where a nonlinear model is required, the task of obtaining an accurate dynamical model is more difficult. Artificial neural networks (ANN) provide an easier way to model complex systems due to their ability to learn and approximate nonlinear functions. These models can then be used as a prediction model for the MPC.

1.3 Objectives and Contributions

The primary objective of this thesis is to explore the viability of a nonlinear MPC to control a FS prototype along a defined trajectory. To do this, it is necessary to develop and improve the existing vehicle dynamical model used by the team, which can serve as a platform for testing alternative control techniques, as well as a tool to evaluate how different parameters affect the performance of the car. This model can be easily adapted to other FS prototypes by changing its parameters.

A vehicle model is identified with ANN using adequate signals covering the entire performance envelope of the vehicle. ANN identification is a black box modelling approach which does not require any prior knowledge of the internal system dynamics.

A nonlinear MPC algorithm is proposed, which can be further improved and implemented or act as a starting point to test other control techniques.

1.4 Thesis Outline

The present thesis is divided in 6 Chapters. Following the Introduction, Chapter 2 introduces the vehicle model used throughout the thesis. The coordinate system of the vehicle is defined, together with the physical relations and equations that describe the motion of the vehicle. A simulation of the model is compared to real world data of the same car.

In Chapter 3 an ANN model of the vehicle dynamics is identified based on appropriate data. The learning results of the model are then presented.

Chapter 4 formulates the MPC problem for trajectory tracking and describes its implementation in MATLAB®. A reference trajectory for a Formula Student track is given and the simulation results are analyzed. In order to decrease the computation time of the control algorithm, a tentative solution of having a neural network structure learning the MPC is explored in Chapter 5.

Finally, Chapter 6 presents conclusions and some suggestions for future work.

Chapter 2

Vehicle Dynamics Model

Vehicle models are useful to allow engineers to analyze several parameters and determine if requirements are met for each design. Running a computer simulation is faster and more cost effective than building an actual prototype of a car, hence the significant increase of the usage of simulations over recent years [12].

This chapter presents a formulation for a dynamical model for the FST09e prototype, explaining the basic principles behind its dynamics. This model has been developed on Simulink®, a toolbox software from Mathworks®. The dynamical model developed in this work is built upon existing models of previous FST Lisboa prototypes [13, 14]. Major improvements are made on the tyre model, steering kinematics, motor controllers and wheel dynamics.

A car is a complex system which can be divided in several subsystems. Taking into consideration the coordinate system defined in Figure 2.1, the car subsystems and how they interact with each other can be seen in Figure 2.2.

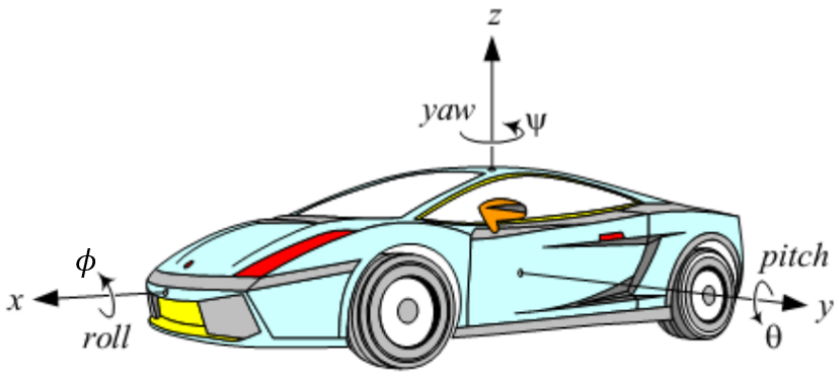

Figure 2.1: Vehicle coordinate system defined by SAE with roll, pitch and yaw angles [15]

There are three subsystems that have a major impact in the forces applied on a car: the powertrain, the aerodynamics and the tyres. The planar model receives these forces and determines the horizontal movement of the vehicle, as well as the accelerations ($\dot{v}_x$ and $\dot{v}_y$). These accelerations generate load

transfers, on top of which the aerodynamic loads are added and the vertical load on each tyre (F_z) is then calculated in the vertical model, by simulating the response of the suspension. The tyre model takes the vertical loads on the tyres together with the velocity components and the steering angle (δ) to determine the tyre lateral forces (F_{yi}) given to the planar model. The driver controls the steering angle and the reference velocity, which is then converted to a certain percentage of throttle or brake pedal.

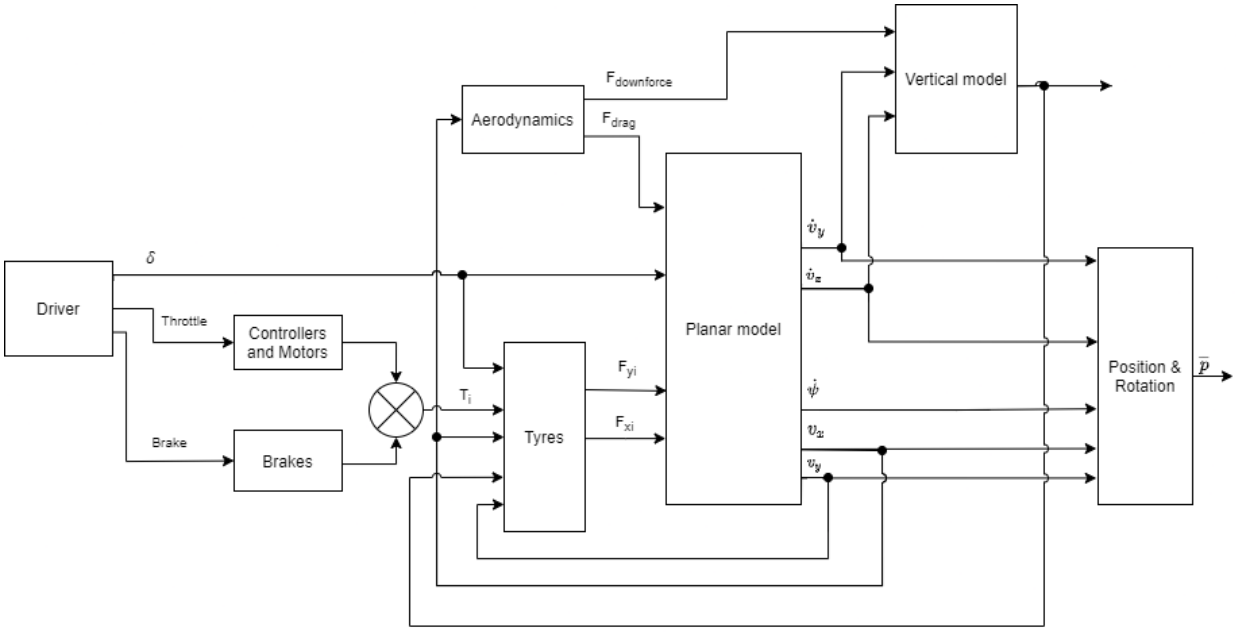

Figure 2.2: General Structure of the Vehicle Model

2.1 Motors Model

The car has four motors, each one connected to one wheel. The inputs of the motors are the desired angular velocities computed by the controllers, which will be presented in section 2.8.

The motor torque vector as well as the required force for each of the four electric motors is calculated according to the model in Figure 2.3, based on information provided by the motor manufacturer [16].

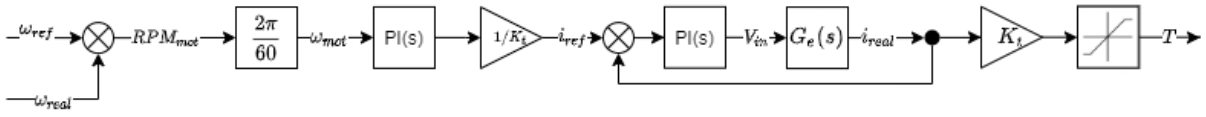

Figure 2.3: Motor subsystem controller model [16]

The torque T is calculated given the difference between the real and the desired values of motor rotation (ω_{real} and ω_{ref}, respectively). This difference in angular velocity is then converted from RPM to rad/s:

$$w_{mot} = RPM_{mot} \times \frac{2\pi}{60} \tag{2.1}$$

which is followed by a proportional-integral (PI) control action, so that an appropriate setpoint is defined, which is subsequently multiplied by the current constant, $1/K_t$, to obtain the current setpoint, i_{ref}. A PI controller then takes the current error and defines the input voltage V_{in}.

The feedback loop simulates the inverter and permanent-magnet synchronous motor assembly by means of an equivalent impedance $G_e(s)$. The real current value, i_{real} is then multiplied by the torque constant, K_t, which yields the actual torque output. Finally, a saturation block is used to guarantee that the torque, T, is between the interval of minimum and maximum torque limits. The values for the equivalent impedance, torque limits and other parameters can be found in the motor datasheet [16].

Figure 2.4 shows the maximum torque and power of the motors as a function of the RPM, where the orange curves represent nominal values while the blue curves represent maximum values. The dashed lines represent the transition to field weakening, a phenomenon that reduces torque but allows the motor to reach higher angular speeds, for different voltages.

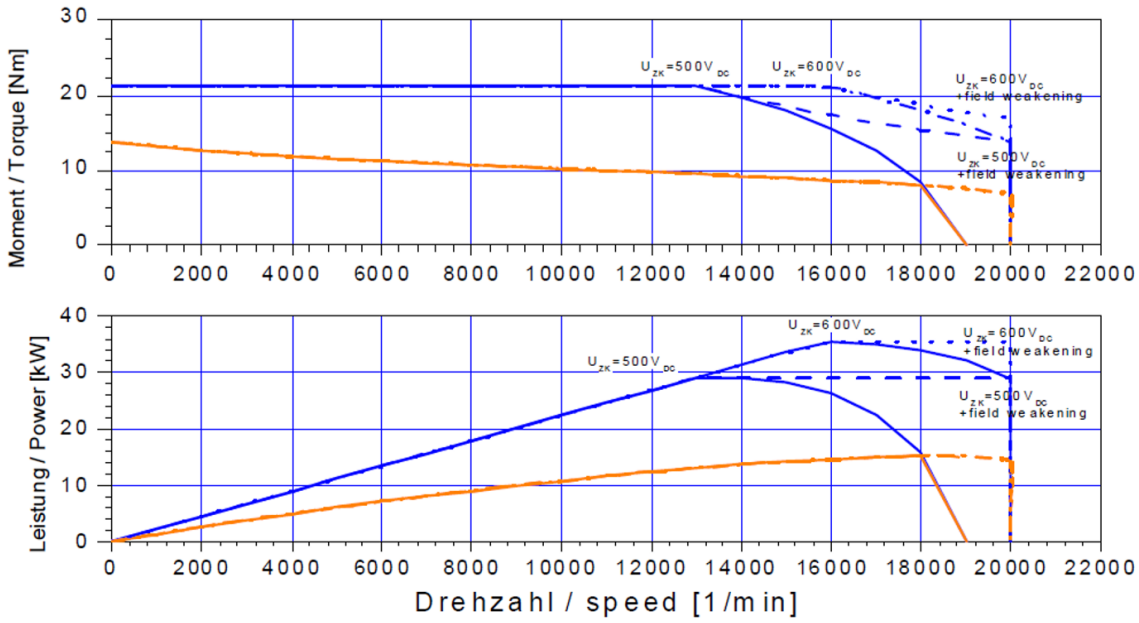

Figure 2.4: Motors torque (top) and power (bottom) curves as a function of RPM [16]

2.2 Aerodynamics Model

This subsystem calculates the aerodynamic forces acting on the car. It accounts for the influence of drag and downforce. Drag is the force component acting parallel and opposite to the velocity of the vehicle and lift is the force component acting perpendicular to the velocity of the vehicle. In the case of a racecar this force points downwards, thus being frequently called downforce. These loads are described by the following equations:

$$F_{drag} = \frac{1}{2}\rho_{air} A_x^{proj} C_d v_x^2 \tag{2.2}$$

$$F_{downforce} = \frac{1}{2}\rho_{air} A_z^{proj} C_l v_x^2 \tag{2.3}$$

where ρ_{air} is the air density, A_x^{proj} and A_z^{proj} are the projected section areas of the vehicle, perpendicular to the x-axis and z-axis of the velocity vector, while C_d and C_l are the drag and lift coefficients.

It is very complex to accurately simulate the aerodynamics response of the vehicle over time, due to the constant variations of velocity magnitude and direction, as well as chassis movement. In order to simplify, the values of A_x^{proj}, A_z^{proj}, C_d, C_l and ρ_{air} are assumed constant and are estimated using CFD (Computational Fluid Dynamics) simulations and wind tunnel tests performed by FST Lisboa.

2.3 Vertical Model

The vertical dynamics of the car is based on a 7 degrees of freedom (DOF) model. The model combines four quarter suspension models. The suspension is represented with two mass-spring-damper systems in series. In the literature it is quite common to find the whole suspension only by one quarter or half of a car [15, 17]. By having the four quarters combined, interactions between wheels may be taken into account. This model is illustrated in Figure 2.5.

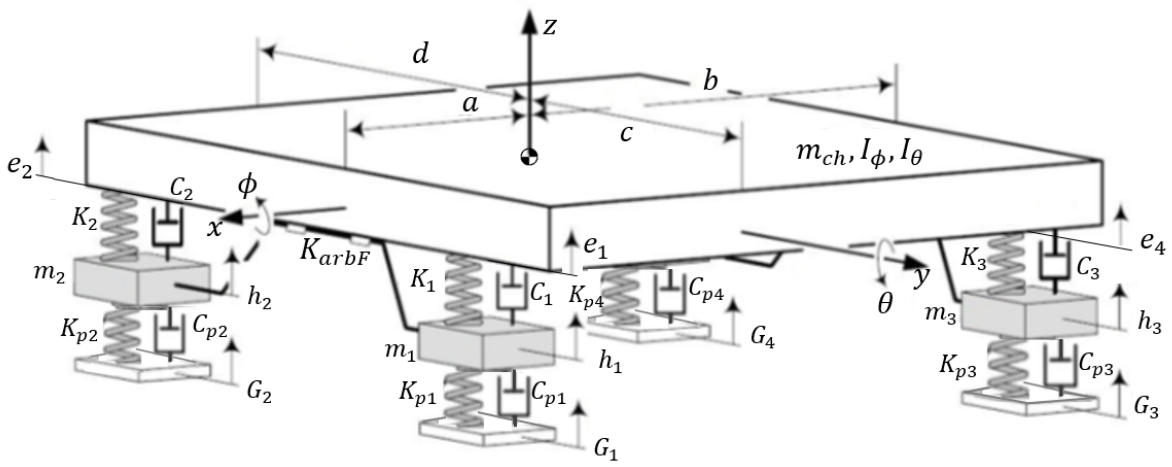

Figure 2.5: 7 DOF vibrational model [15]

The 7 DOF of the model are: roll (ϕ), pitch (θ), vertical position (Z) of the sprung mass of the car and the height of each unsprung component (h_i). Each unsprung component is made up of wheel, tyre, brake, upright and all remaining components which are not suspended by the dampers. Each of the quarters has a number $i = 1, 2, 3, 4$. The sprung component is represented by its mass (m_{ch}) and inertias (I_ϕ and I_θ) while the unsprung components are represented only by their mass (m_i).

Each suspension quarter (i) is defined as an equivalent linear spring-damper parallel system, where K_i is the wheel stiffness and C_i is the adjusted damping coefficient, which are calculated using the spring stiffness K_{si}, the damping coefficient C_{si} and the suspension motion ratio (MR), as shown in Equations (2.4):

$$K_i = \frac{K_{si}}{MR^2} \qquad C_i = \frac{C_{si}}{MR^2} \tag{2.4}$$

Similarly, the tyres are also represented by a spring-damper system where the tyre stiffness is represented by K_{pi} and the tyre damping coefficient is represented by C_{pi}. The left and right quarters are connected by an anti-roll bar, represented by K_{arb_F} for the front and K_{arb_R} for the rear.

The model inputs are: vertical load, F_z, given by the sum of the weight and total downforce of the vehicle; moments applied to roll, M_ϕ, and pitch, M_θ, calculated according to Equations (2.5); and ground height below each wheel, G_i, which will be simplified as constant and equal to zero, representing a smooth and levelled track.

$$M_\phi = m \cdot \dot{v}_y \cdot h_{CoG} \qquad M_\theta = -m \cdot \dot{v}_x \cdot h_{CoG} \tag{2.5}$$

To keep this model linear, a small angle approximation is made, where the displacements of the top suspension quarters, e_i, are calculated as follows:

$$\begin{aligned} e_1 &= Z + c\phi - a \cdot \theta \qquad & e_2 &= Z - d\phi - a \cdot \theta \\ e_3 &= Z + c\phi + b \cdot \theta \qquad & e_4 &= Z - d\phi + b \cdot \theta \end{aligned} \tag{2.6}$$

where c is the distance between left wheels and CoG, d is the distance between right wheels and CoG, a is the distance between the front wheels and the CoG and b is the distance between the rear wheels and the CoG, as depicted in Figure 2.5.

The equilibrium forces for each quarter ($i = 1, 2, 3, 4$) can be defined with these displacements, where $\Delta_i = e_i - h_i$ and $\dot{\Delta}_i = \dot{e}_i - \dot{h}_i$:

$$m_{ch}\ddot{Z} + \sum_{i=1}^{4} K_i\Delta_i + \sum_{i=1}^{4} C_i\dot{\Delta}_i = F_z \tag{2.7}$$

$$I_\theta\ddot{\theta} + a\Big(\sum_{i=1}^{2} K_i\Delta_{si} + \sum_{i=1}^{2} C_i\dot{\Delta}_{si}\Big) - b\Big(\sum_{i=3}^{4} K_i\Delta_{si} + \sum_{i=3}^{4} C_i\dot{\Delta}_i\Big) = M_\theta \tag{2.8}$$

$$I_\phi\ddot{\phi} + a\Big(\sum_{i=1,3} K_i\Delta_i + \sum_{i=1,3} C_i\dot{\Delta}_i\Big) - b\Big(\sum_{i=2,4} K_i\Delta_i + \sum_{i=2,4}^{C} {}_i\dot{\Delta}_i\Big) = M_\phi \tag{2.9}$$

$$m_1\ddot{h}_1 - K_1\Delta_1 - C_1\dot{\Delta}_1 + K_{p1}h_1 + C_{p1}\dot{h}_1 + K_{arbF}(h_1 - h_2) = 0 \tag{2.10}$$

$$m_2\ddot{h}_2 - K_2\Delta_2 - C_2\dot{\Delta}_2 + K_{p2}h_2 + C_{p2}\dot{h}_2 + K_{arbF}(h_2 - h_1) = 0 \tag{2.11}$$

$$m_3\ddot{h}_3 - K_3\Delta_3 - C_3\dot{\Delta}_3 + K_{p3}h_3 + C_{p3}\dot{h}_3 + K_{arbR}(h_3 - h_4) = 0 \tag{2.12}$$

$$m_4\ddot{h}_4 - K_4\Delta_4 - C_4\dot{\Delta}_4 + K_{p4}h_4 + C_{p4}\dot{h}_4 + K_{arbR}(h_4 - h_3) = 0 \tag{2.13}$$

The expansion of these equations is performed in Appendix A, and the system of equations is implemented as a state space system, where the state variables are $[Z, \dot{Z}, \theta.\dot{\theta}, \phi, \dot{\phi}, h_1, \dot{h}_1, h_2, \dot{h}_2, h_3, \dot{h}_3, h_4, \dot{h}_4]^T$ and the inputs of the system $[F_z, M_\theta, M_\phi, G_1, G_2, G_3, G_4]$. The roll ($\phi$) and pitch ($\theta$) are the outputs of the system, while the tyre vertical load is computed with Equation (2.14) assuming the tyre as a linear spring damper [18]:

$$F_{zi} = K_{pi}(h_i - G_i) + C_{pi}(\dot{h}_i - \dot{G}_i), \qquad i = 1, 2, 3, 4 \tag{2.14}$$

2.4 Steering Kinematics

In Figure 2.6 the rack and pinion steering system of FST09e is depicted. The driver turns the steering wheel, which transmits the rotation to the upper steering column. Due to packaging constraints, the lower column needs to be at an angle relatively to the upper column. This is done with a cardan joint (also known as universal joint), which connects the rotation motion of both columns. The lower column is then connected to the pinion, shown in Figure 2.7. The pinion meshes with the rack and the rotation of the pinion is converted to linear motion of the rack. The rack is then connected to each of the steering yokes, which are linked to the front wheels through the steering arms, and this is how the linear motion of the rack makes the wheels turn.

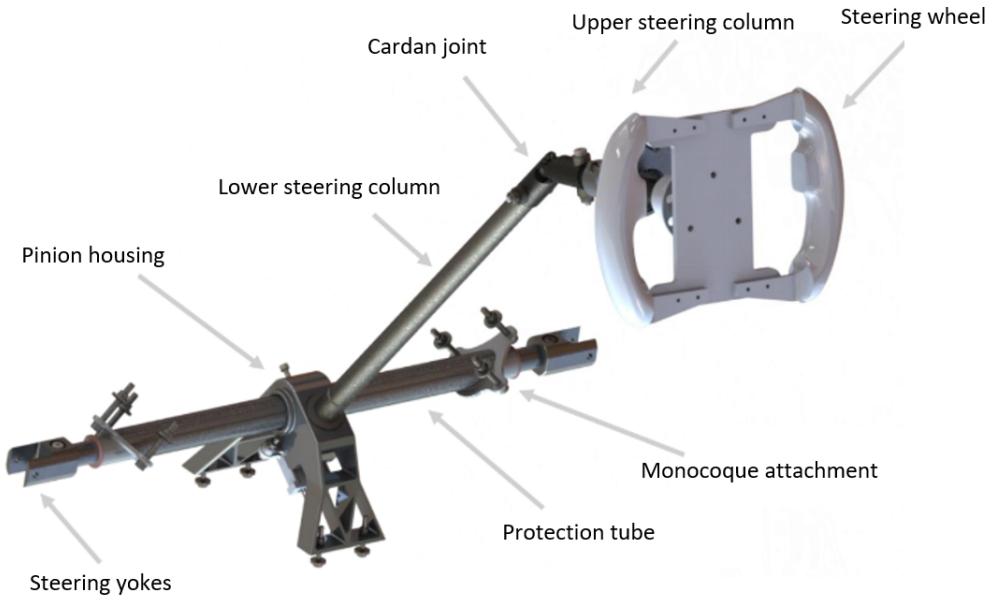

Figure 2.6: FST09e steering assembly

The cardan joint can be defined as a mechanical connection used to transfer mechanical power between two shafts with its axes at an angle to each other. This joint may seem quite simple, however the physics behind their mechanism is rather intricate. As seen in Figure 2.8, a cardan joint is formed by: two yokes, one connected to the input shaft and another connected to the output shaft, and a cross, connecting both yokes through a bearing interface, which allows the cross to spin around its axes. The input shaft corresponds to the upper steering column while the output shaft corresponds to the lower steering column. The yokes are usually made of steel and they are designed to overcome any plastic deformations and to keep aligned the bearing seats of the cross, which are located at the cross tips. The cross is also made of steel and its size is determined in order to obtain the best compromise between the dynamic features of the rotating elements of the joint and its flexural properties.

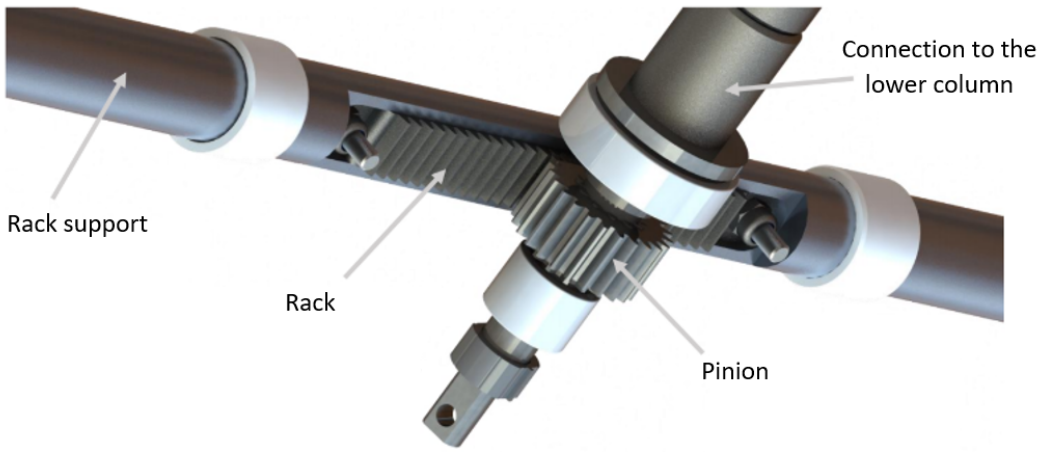

Figure 2.7: Close-up of rack and pinion of FST09e

In Figure 2.8 it can be seen that the cross is divided in two axes, the red axis (connecting the two red circles) and the green axis (connecting the green circle to the other circle which is not visible). The green axis rotates in a different plane than the red axis. This is possible because the green axis not only rotates around the vertical plane, it also spins around the green axis itself. Without this spin, the motion of the output shaft at a different angle of the input shaft would be impossible. However, the spin of this axis causes an added effect on the speed of the output shaft, meaning that it ends up having a different speed from the input shaft, i.e. the transmission ratio is not constant overtime, presenting a smooth periodic oscillation with a sine wave pattern [19]. As a consequence of this speed fluctuation, the angle of rotation of the steering wheel is not the same as the angle of rotation of the pinion. In the case of FST09e, the discrepancy between the angle of rotation of the steering wheel and the angle of rotation of the pinion can reach the maximum value of 6° , which is rather significant.

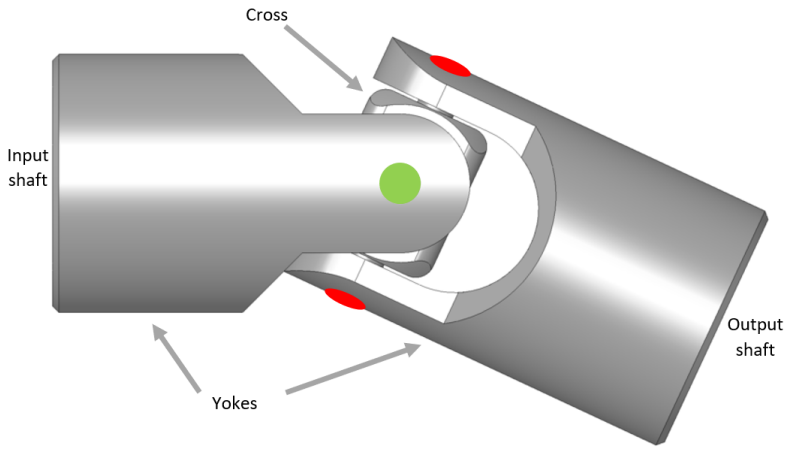

Figure 2.8: Cardan joint [20]

Throughout this work, the steering angle δ refers to the angle of rotation of the pinion because that is the angle that the steering rotary potentiometer measures, and the data from this sensor will be used to validate the model. This is also the angle that the steering actuator of FST10d, the autonomous vehicle based on FST09e, will control. As explained, the angle of rotation of the pinion is not the same as the angle of rotation of the steering wheel because of the fluctuations of the cardan joint.

2.4.1 Ackerman Steering

When a four-wheeled vehicle is negotiating a corner, in order to not have any wheel slip or skidding, the geometric centre of the corner must be located at a point in the same line extending from the rear axle [21], as depicted in Figure 2.9, where a vehicle is seen from above during a corner.

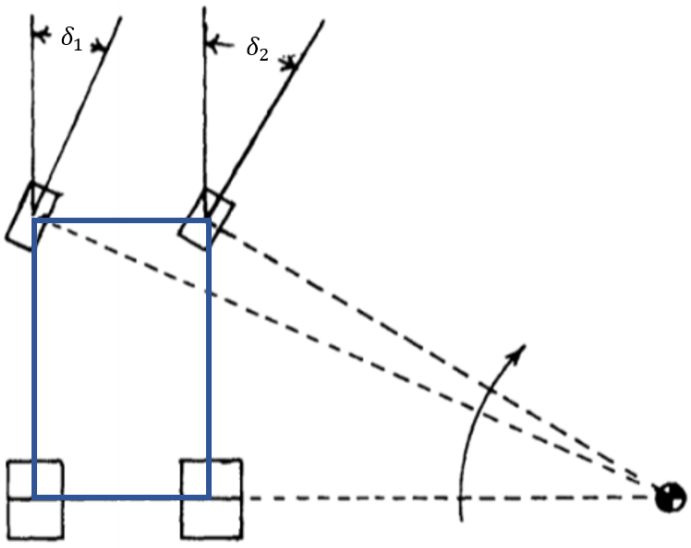

Figure 2.9: Ackermann steering principle [21]

A steering mechanism which behaves like this is said to have a perfect Ackermann geometry. However, the steering geometry with optimal performance is not necessarily the geometry with zero wheel slip when cornering, because as can be seen in Figure 2.13, the lateral force of a tyre is maximal for a certain slip angle α (Figure 2.12 illustrates the slip angle). In the case of FST09e, Ackermann geometry is present because the inside wheel turns more than the outside wheel when cornering, but it is not a perfect Ackermann geometry because there is some tyre slip when cornering.

The driver inputs one steering angle which is then converted to the different steering angles of the front left and front right wheels, δ_1 and δ_2, respectively. This is done by interpolating a table which has the values of δ_1 and δ_2 for each value of δ. This table is provided by FST Lisboa when the steering geometry is defined.

2.5 Planar Car Model

2.5.1 Lateral Dynamics

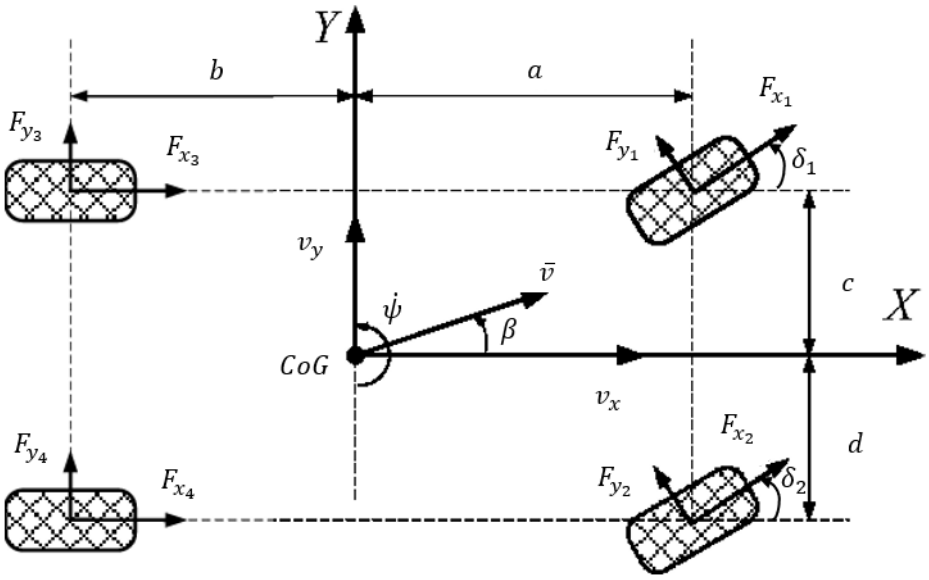

Figure 2.10: Forces Applied on the Vehicle - Planar Model

In this model, it is assumed that the vehicle only moves in the x-y plane. It is also assumed that the vehicle is a rigid body with dynamics that can be expressed by the Newton-Euler equations of motion. Considering a reference frame attached to the CoG like the one in Figure 2.10 results in the following balances:

$$F_x = m \cdot \dot{v}_x - m \cdot \dot{\psi} \cdot v_y \tag{2.15a}$$

$$F_y = m \cdot \dot{v}_y + m \cdot \dot{\psi} \cdot v_x \tag{2.15b}$$

$$M_\psi = \ddot{\psi} \cdot I_\psi \tag{2.15c}$$

where $\dot{\psi}$ is the angular velocity around the z-axis, also known as the yaw rate of the car, m is the mass of the vehicle, F_x and F_y are the total longitudinal and lateral forces acting on the car, v_x and v_y are the longitudinal and lateral velocities of the car and I_ψ is the inertia around the z-axis.

Figure 2.10 illustrates the forces applied on the car in the planar model. Only the front wheels are steerable and each wheel has a different steering angle, due to the Ackermann geometry, as explained in Section 2.4. The track width is the same at the front and at the rear. Equations (2.16) are the combination of the force balance (2.15) with the forces from Figure 2.10:

$$\dot{v}_x = v_y \cdot \dot{\psi} - \frac{1}{m}[\sum_{i=1}^{2} {F_{yi}}^F sin(\delta_i) - \sum_{i=1}^{2} {F_{xi}}^F cos(\delta_i) - F_x^R] \tag{2.16a}$$

$$\dot{v}_y = -v_x \cdot \dot{\psi} - \frac{1}{m}[\sum_{i=1}^{2} {F_{yi}}^F cos(\delta_i) + F_y^R + \sum_{i=1}^{2} {F_{xi}}^F sin(\delta_i)] \tag{2.16b}$$

$$\ddot{\psi} = \frac{1}{I_\psi} a[\sum_{i=1}^{2} {F_{yi}}^F cos(\delta_i) + {F_{xi}}^F sin(\delta_i)] - \frac{1}{I_\psi} b F_y^R \tag{2.16c}$$

where the superscripts F and R denote the total forces in the respective component at the front and at the rear.

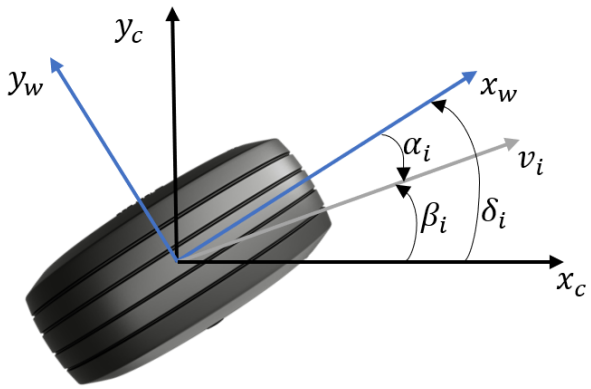

Figure 2.11: Wheel velocity and angle vectors. The velocity vector is given by v while the subscripts w and c denotes the wheel and car reference frames, respectively

The sideslip of the car, β, defined as the angle between the velocity vector and the heading of the car, is shown in Figure 2.10. This angle is given by Equation (2.17) with the velocities in the vehicle reference frame:

$$\beta = \arctan(\frac{v_y}{v_x}) \tag{2.17}$$

Afterwards, the slip angles of each wheel, α_i are calculated using Equation (2.18) where β_i is the projected sideslip angle of the car on wheel i:

$$\alpha_i = \beta_i - \delta_i \tag{2.18}$$

It should be noted that for the rear wheels, $\delta_i = 0$. The velocity components at each wheel, $\overline{v_i}$, which are associated with the angle β_i seen in Figure 2.11, can be calculated with Equation (2.19):

$$\overline{v_i} = \overline{v} + \overline{\omega} \times \overline{d_i} = \begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ r \end{bmatrix} \times \begin{bmatrix} x_i \\ y_i \\ 0 \end{bmatrix} = \begin{bmatrix} v_x - y_i \cdot \dot{\psi} \\ v_y + x_i \cdot \dot{\psi} \\ 0 \end{bmatrix} \tag{2.19}$$

It is assumed that the wheel i is at a distance $\overline{d_i} = (x_i, y_i)$ of the CoG, that $\overline{v}$ is the vector with the velocity components at the CoG and that $\overline{\omega}$ is the vector with the angular velocity components at the CoG, which only has the angular velocity around the z-axis, since a planar model is assumed.

Finally, Equations (2.19) and (2.17) can be substituted into (2.18) to obtain (2.20), a general expression for the wheel slip angle:

$$\alpha_i = \arctan\left(\frac{v_y + x_i \cdot \dot{\psi}}{v_x - y_i \cdot \dot{\psi}}\right) - \delta_i \tag{2.20}$$

2.5.2 Longitudinal Dynamics

To the torque applied at the hub one must subtract the reaction torque due to the longitudinal force exerted by the tyre on the road. To obtain such torque one must multiply this force by the wheel radius. The net torque acting on the hub must overcome the rotational inertia of the wheel assembly, resulting in a given angular velocity. This rotational inertia, J_w, is influenced by the individual inertias of the tyre, hub, transmission and rim. These effects are represented by the following transfer function:

$$T = J_{wheel} \cdot \alpha <=> \alpha = \frac{T}{J_w}, \quad \dot{\omega} = \alpha => \omega = \frac{T}{J_w \cdot s} \tag{2.21}$$

The ratio between the velocity of the vehicle and its wheels is called slip ratio. The longitudinal slip ratio, s, is defined in the direction of the wheel velocity $\overline{v_i}$. The slip ratio is always between -1 and 1, where 0 means no wheel slip. The formula for the longitudinal slip ratio is as follows:

$$s_i = \frac{\omega_i \cdot R_w \cdot \cos\alpha_i - v_i}{max(w_i \cdot R_w \cdot \cos\alpha_i, v_i)} \tag{2.22}$$

where R_w is the wheel radius.

2.6 Position and Rotation

The yaw (or heading) angle (ψ) is given by the time integration of the yaw rate (ψ) where ψ_0 is the initial yaw angle:

$$\psi = \psi_0 + \int (\dot{\psi}) dt \tag{2.23}$$

To keep track of the trajectory of the vehicle, its position has to be computed in the global frame, rather than the body frame, as is done with Equation (2.24):

$$\overline{p} = \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \int (v_x cos\psi - v_y sin\psi) dt \\ \int (v_x sin\psi + v_y cos\psi) dt \end{bmatrix} \tag{2.24}$$

where $\overline{p}$ denote the position of the car in the world frame. Evidently, the planar position coordinates in the body frame are null, since the body reference frame is coupled with the car.

2.7 Tyre Model

A cartesian coordinate system is attached to a tyre in Figure 2.12, where the three forces and three moments that act on a tyre are shown. The longitudinal force (F_x) appears on driving wheels while accelerating, and on all wheels while braking. The lateral force (F_y), that makes the car turn, is generated when there is a slip angle α_i, i.e. the wheel velocity direction is different from the actual wheel direction. The wheel load (F_z) is the weight of the car on that wheel (including aerodynamic forces). The roll moment (M_ϕ) contradicts the camber angle (γ) and can also generate lateral forces. Pitch moment (M_θ), also called rolling resistance, counteracts wheel rotation. The yaw moment (M_ψ), also called self-aligning moment, makes the wheels tend to point straight if no steering force is applied.

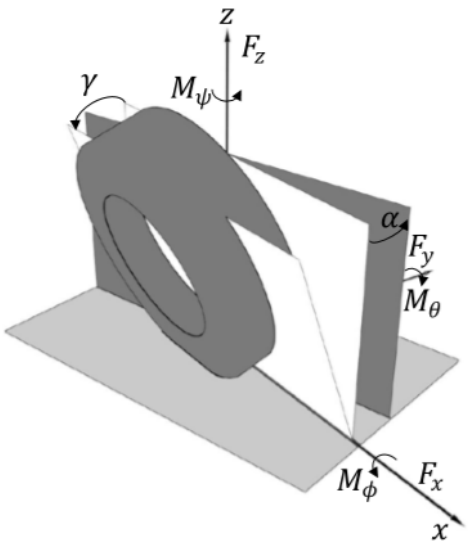

Figure 2.12: Tyre coordinate system [15]

Several models have been presented for the difficult endeavour of simulating tyre behaviour, such as Lumped Models [22], Brush Model, Tread Simulation Model [23], TMeazy [24], Burckhardt [25], and the most known and accepted, Pacejka's Magic Formula [23]. Most of these use non-linear experimental equations with coefficients that have to be adjusted to real tyre data.

The Pacejka's Magic Formula has been used by FST Lisboa in recent years for simulation purposes and for selecting the tyres to be used in the car, based on real tyre data supplied by the FSAE TTC (Formula SAE Tyre Testing Consortium) [26]. This data is also used to adjust the multiple coefficients in the equations of Pacejka's Magic Formula. Since the team has already some experience with this method, it was the one chosen to model the tyre behaviour in the vehicle simulation.

The forces F_x and F_y and the aligning torque M_z result from the input slip components and the wheel load and are presented in Equation (2.25):

$$F_x = f_{F_x}(s, \alpha, \gamma, Fz) \qquad F_y = f_{F_y}(s, \alpha, \gamma, Fz) \qquad M_z = f_{M_z}(s, \alpha, \gamma, Fz) \tag{2.25}$$

where f represents the non-linear force and moment functions which contain the equations of the Pacekja's Magic Formula, s denotes the longitudinal slip ratio, α is the slip angle, γ is the camber angle and F_z is the vertical load on the tyre. In Figure 2.13 it can be seen, as an example, how the lateral force depends on the slip angle produced by the tyre for three different vertical loads.

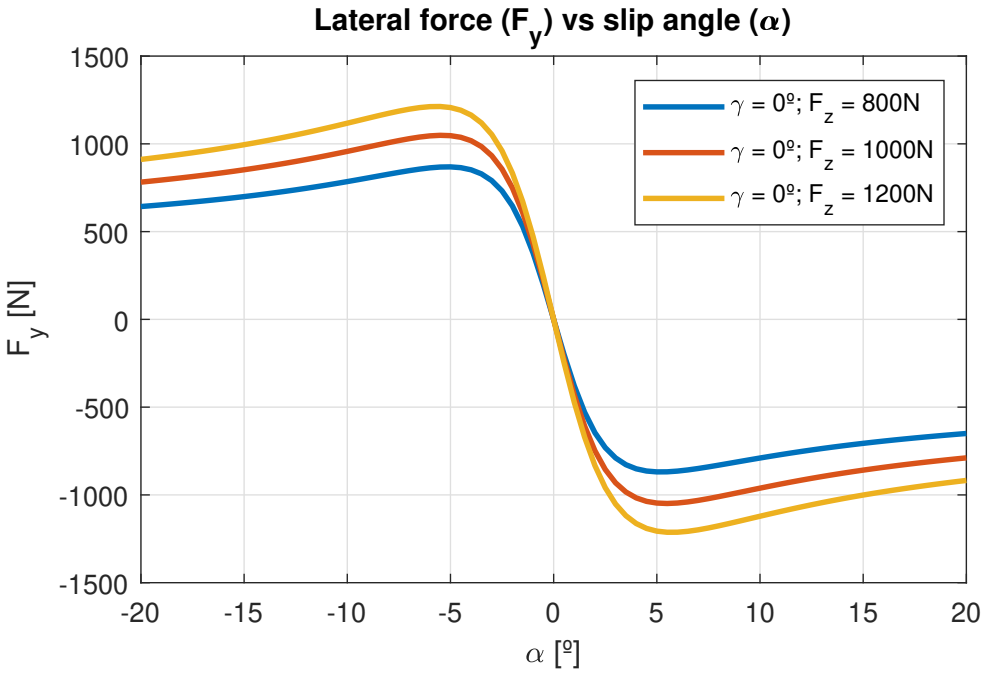

Figure 2.13: Tyre lateral force as a function of slip angle for different vertical loads, using Pacejka's Magic Formula

2.8 Controllers Model

In order to abide by the competition regulations, the battery power output must not be higher than 80 kW. The car is capable of using more power than that, as such a power limiter controller is used in FST09e and it has been modeled on Simulink® as well.

The efficiency map in Figure 2.14 has to be taken into consideration when limiting the power. The power limiter controller simply takes as input the angular velocity of the wheel (ω), measured in RPM, and the requested torque from each wheel and outputs a torque value for that wheel respecting the maximum power of 80 kW, according to Equation (2.26):

$$P = T \times \omega \tag{2.26}$$

where P denotes the power and T the torque on each motor. Equation (2.26) is implemented in the model by means of a look-up table where each value corresponds to the maximum torque of the wheel for a certain value of ω on that wheel.

Field weakening 600VDC		speed [rpm]									
Current [Arms]	Torque [Nm]	500	1000	2000	3000	4000	6000	10000	12000	15000	19000
5	1,3	64,37	71,33	73,64	74,70	75,43	76,57	77,00	77,08	77,56	78,14
10	2,7	58,42	70,48	77,57	80,40	82,01	83,92	85,16	85,44	85,97	86,50
20	5,4	44,94	60,81	73,35	78,82	81,94	85,43	88,20	88,88	89,71	90,44
30	7,9	35,59	51,90	67,02	74,26	78,54	83,42	87,58	88,65	89,84	90,86
40	10,4	29,14	44,78	61,01	69,41	74,57	80,62	85,93	87,34	88,86	90,16
50	12,5	24,17	38,71	55,22	64,39	70,24	77,30	83,73	85,48	87,37	88,98
60	14,4	20,41	33,76	50,04	59,65	65,99	73,88	81,33	83,42	85,66	87,59
70	16,0	17,31	29,40	45,10	54,87	61,55	70,10	78,56	80,97	83,56	85,81
80	17,4	14,82	25,75	40,67	50,41	57,28	66,34	75,70	78,40	81,34	82,71
90	18,5	12,81	22,67	36,72	46,30	53,25	62,67	72,77	75,75	79,02	75,96
100	19,6	11,17	20,05	33,21	42,51	49,44	59,09	69,82	73,06	67,66	69,28

Figure 2.14: Efficiency map provided by the motor manufacturer [16]

The car also has an electronic differential which distributes the torque on each wheel so that cornering performance is increased. This differential works with a look-up table as well, on which each value corresponds to the torque that needs to be subtracted from wheel i, according to the steering angle that the driver inputs. In the real car, this steering angle is measured with a steering rotary potentiometer, as will be explained in Section 2.10.

2.9 Driver Model

The ability of a car to follow a path comes from the driver inputs, when autonomous technology is not present. For the moment, a driver is assumed to exist, but it can easily be replaced in the model by an autonomous controller. On this subsystem the driver commands - throttle, brake and steering angle - are generated. Throttle and brake can take any continuous value from zero to one. If one of these commands has a non-zero value, i.e. it is activated, the other must be zero, because it is assumed the driver either brakes or accelerates at a given moment. The steering angle, measured in degrees, is

denoted by the symbol δ and it is one of the inputs of this subsystem. The other input of this subsystem is the reference velocity, i.e. the desired velocity at each instant, measured in km/h.

The reference velocity is then compared to the current velocity, and the resulting tracking error feeds a proportional-integral (PI) controller. This is a simplification of the driver's response, since it does not take into consideration physical and psychological factors, but it is sufficient for a reliable closed-loop vehicle-driver simulation [25]. The PI controller attributes a value for the throttle or brake variable, depending on whether the signal is positive or negative, respectively. When accelerating forward, the throttle value is between 0 and 1 and when braking, the absolute value of the signal is taken, so that the brake value also ends up to lie between 0 and 1.

2.9.1 Throttle Pedal

The requested torque, for each wheel, is defined according to Equation (2.27):

$$T_{req} = T_{max} \times throttle \tag{2.27}$$

where T_{max} is the maximum torque limitation, $throttle$ is the throttle pedal value between 0 and 1 (0 means no throttle and 1 means full throttle) and T_{req} is the requested torque, which acts as an input to the controllers discussed in Section 2.8. The maximum torque limitation can be whatever value as long as it is lower or equal to the maximum 21Nm the motors can provide. During testing with FST09e, different values for the maximum torque limitation are experimented, and then defined for each type of dynamic event (skidpad, autocross, etc.).

2.9.2 Brake Pedal

When braking is requested, the $brake$ pedal variable, between 0 and 1, is multiplied by the maximum braking pressure, $p_{bmax} = 80$ bar, to find the actual brake pressure. Then, the brake pressure is distributed to the front and the rear brake lines according to the brake bias, as presented in Equation (2.28). The brake bias, $bias_{F,R}$ is adjustable in the vehicle itself, but usually after some testing it is kept constant. Equation (2.29) shows the calculation of the brake torque.

$$p_{b_{F,R}} = brake \times p_{bmax} \times bias_{F,R} \tag{2.28}$$

$$T_{b_{F,R}} = -p_{b_{F,R}} \times A_{b_{F,R}} \times R_{b_{F,R}} \tag{2.29}$$

where the subscripts F and R indicate if the values are for the front or rear axles, $T_{bF,R}$ denotes the brake torque, $A_{b_{F,R}}$ denotes the contact area of the brake pads and $R_{bF,R}$ the effective brake radius, which is the distance between the rotor centre and the centre of pressure of the caliper pistons.

2.10 Validation

The inputs to simulate the driver were obtained through data from 2019 Formula Student East competition with FST09e. For this, it was necessary to obtain data from the steering angle and the longitudinal velocity of the car. To use the same model in another 4-wheel vehicle, it is sufficient to change the relevant specifications.

The steering rotary potentiometer measures how much the steering turns to the left and to the right, with the specifications detailed in Table 2.1. The range refers to the maximum rotation for both sides.

Steering rotary potentiometer	
Range	4096°
Resolution	0.1 °
Accuracy	2 °
Update rate	50 Hz

Table 2.1: Steering rotary potentiometer datasheet values

Unfortunately, data from the GPS of FST09e was not available, but there was data available from the GoPro camera which is usually mounted near the driver. The GoPro camera contains GPS, accelerometer and gyroscope, giving us information about the position, linear and angular velocities and linear accelerations. Specifications pertaining the accuracy of its sensors are considered proprietary information, but when comparing the trajectories and the velocity from the GoPro with the video footage the values proved coherent. The sampling time of this velocity data is 0.055 s. A script in MATLAB®was created to read the data both from the steering rotary potentiometer and from the GoPro GPS and to resample it to a sampling time of 0.055 s in the case of the steering data.

In Figure 2.15, the inputs and main outputs of the dynamical model (seen with all subsystems in Figure 2.2) are shown. The inputs are the steering angle δ and the reference longitudinal velocity v_{ref}, defined by the driver. The outputs that are going to be analysed are the car Cartesian trajectory (X, Y), lateral acceleration (a_y) and yaw rate $(\dot{\psi})$. Note that the longitudinal velocity is an input (v_{ref}) as well as an output (v_x) that corresponds to the tracking of v_{ref}.

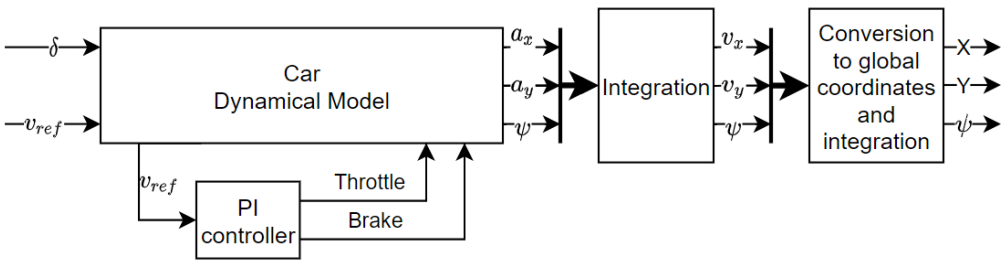

Figure 2.15: Dynamical Model Structure

The skidpad track is chosen for validation purposes, consisting of a circular track with a defined radius used to test the lateral acceleration of the vehicle as well as its handling on the limit of grip. The track surface is smooth and levelled. Figure 2.16 shows the skidpad configuration, where the driver enters the 8-shape circuit in the middle and does two full laps on the right circle followed by two full laps on the left circle, crossing the finish line in the middle, after which he goes straight untill the stop area.

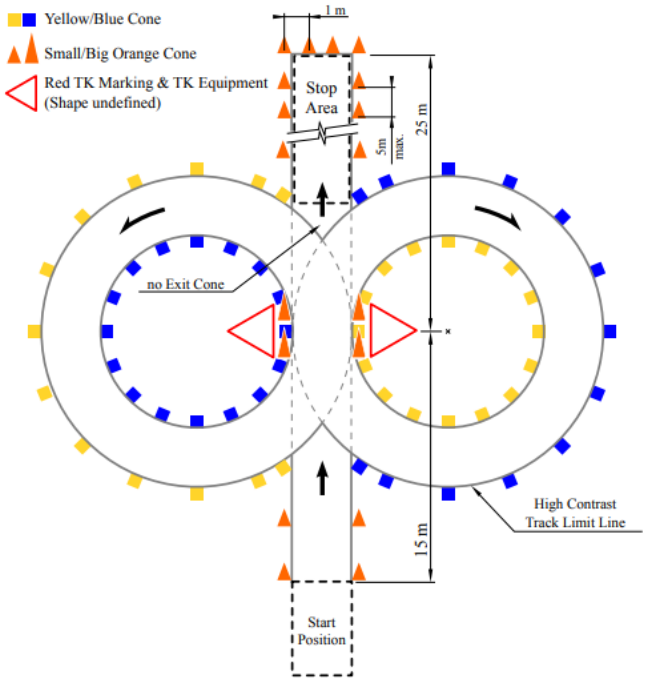

Figure 2.16: Skidpad base configuration according to competition rules [27]

2.10.1 Inputs for the skidpad track

Steering Angle - δ

In Figures 2.17 to 2.20 recorded data of the real prototype performing a skidpad is illustrated. Negative values of steering angle represent the car cornering to the right (clockwise) while positive values represent the car cornering to the left (anticlockwise).

It can be seen from Figure 2.17 that the driver is turning the steering wheel with a steering angle close to -75° up until 15 s, while doing the 2 laps to the right, after which the steering angle is the symmetric, 75° . After 25 s, it can be seen that the driver turned the steering wheel more aggressively to the right to correct the heading of the vehicle, likely due to the acceleration peak in Figure 2.18 when exiting the left circle.

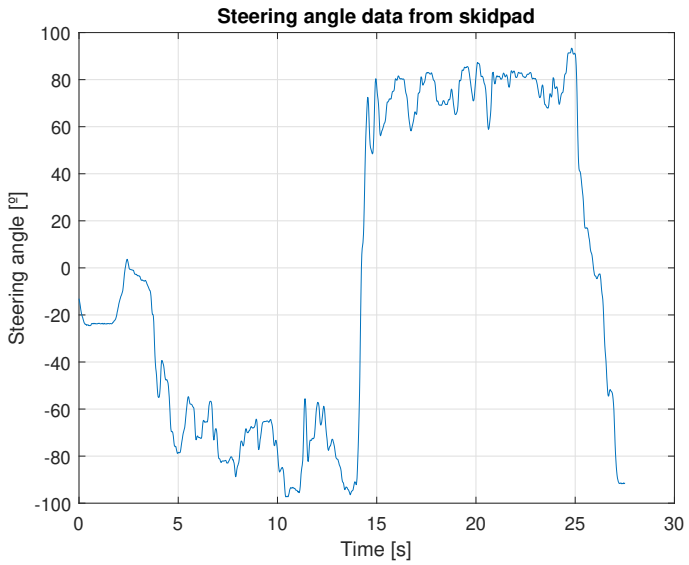

Figure 2.17: Steering angle real data recorded during a skidpad

Reference Velocity - v_{ref}

In Figure 2.18 it can be seen that after the initial acceleration the car maintains an approximate longitudinal velocity of 10 m/s. The higher velocity slightly before 25 s happens because the driver typically goes full throttle when crossing the finish line. The reference velocity, which is the longitudinal velocity data from the real prototype, is an input to the model, which then outputs the simulated velocity, also shown in Figure 2.18. The simulated velocity is very close to the reference velocity given.

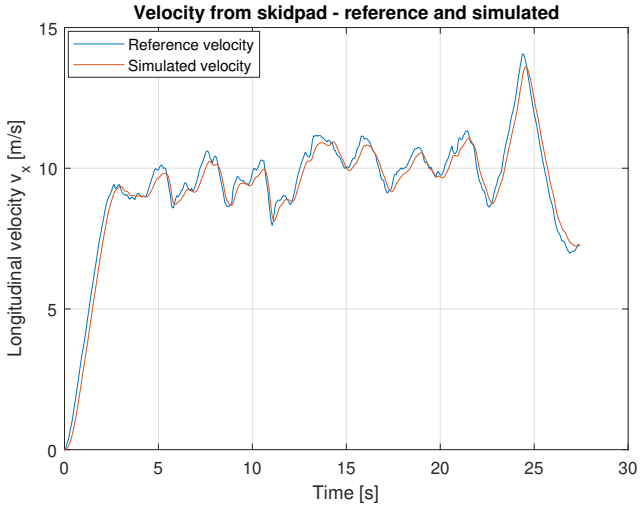

Figure 2.18: Velocity data from skidpad - real and simulated

2.10.2 Outputs for the skidpad track

The next step is to give these inputs to the model and compare the simulation outputs with the real data.

Lateral acceleration - a_y

To remove noise from the sensors, both for lateral acceleration and yaw rate, a smoothing filter with a gaussian-weighted moving average has been used. Figure 2.19 shows the lateral acceleration, where negative values represent the car cornering to the right and positive values to the left.

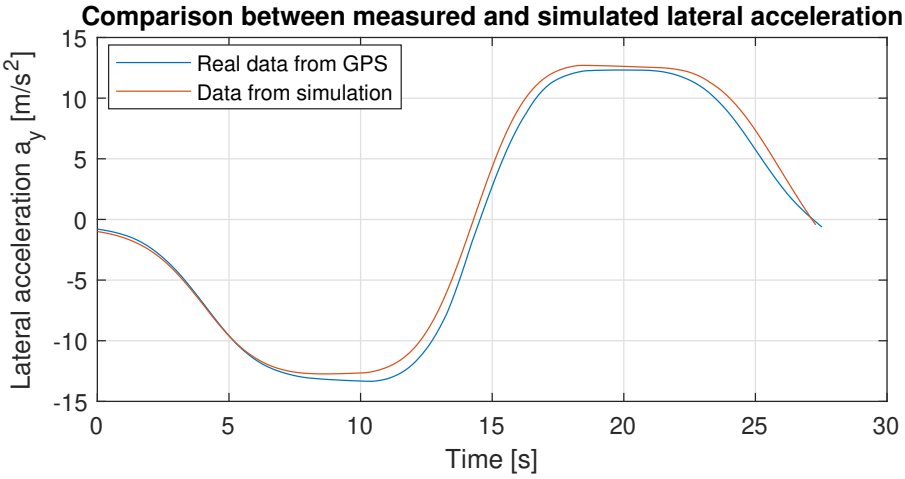

Figure 2.19: Comparison between measured and simulated lateral acceleration

Yaw rate - $\dot{\psi}$

Figure 2.20 shows that the simulated model follows closely the yaw rate of the real car. Like the steering angle, negative values correspond to the car cornering to the right and positive values to the left.

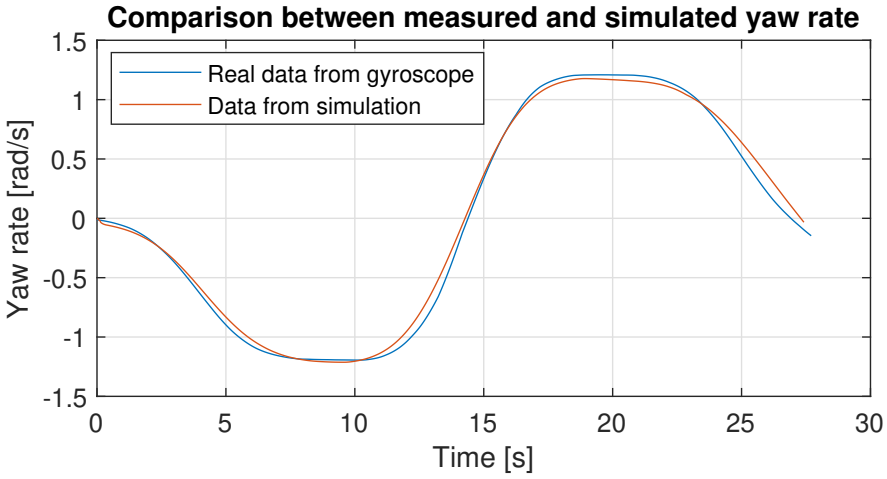

Figure 2.20: Comparison between measured and simulated yaw rate

Trajectory - (X, Y)

The trajectories obtained by the simulator for full tracks are not completely accurate, as is evidenced in Figure 2.21. The accumulation of errors along a lap means that in a more complex track the simulated car will not end its lap in exactly the same place as the real car. Despite the differences in the trajectory over time, Table 2.2 displays the average error between measured vehicle data and simulated data and shows that the dynamics of the simulation are realistic enough to test different control techniques. In order to obtain a simulator that could recreate perfectly the trajectory of the real prototype only by inputting the steering angle and the reference velocity, a huge amount of time and testing, both on simulator and with the real prototype, would be required in order to model everything as accurate as possible. Modelling the tyres with full accuracy is the biggest obstacle, due to the large amount of parameters present in the Pacejka's Magic Formula [23]. The tyre parameters were adjusted to obtain a correct trajectory for the skidpad track, but unfortunately, in other tracks it's not possible to maintain the same trajectory accuracy, as will be seen in Figure 4.4.

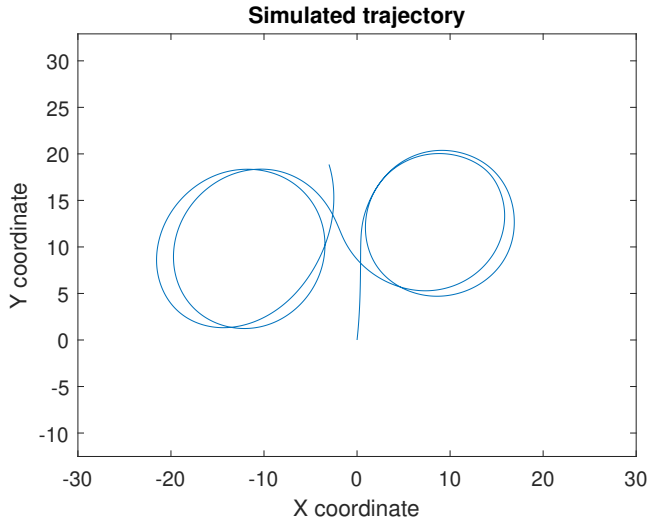

Figure 2.21: Skidpad trajectory simulated

Average relative error		
v_x	a_y	$\dot{\psi}$
1.4%	3.4%	1.7%

Table 2.2: Average error between vehicle data and simulation

After having the vehicle dynamical model validated, it can be used for simulation purposes throughout the rest of the work. The next step is to attempt to learn this model with ANN.

Chapter 3

Artificial Neural Networks Model

Similarly to biological nervous systems, artificial neural networks (ANN) are structures consisting of a collection of parametric nodes called artificial neurons connected through directional links defining a causal relationship between them. The values of the parameters of the nodes are adaptive, which means the outputs of these nodes depend on modifiable parameters. The learning rule specifies how these parameters should be updated to minimise a defined error measure between the output of the network and the target output [28].

The capacity of abstraction of ANN is due to its parallel structure and to its learning ability. The parallel structure comes from the many interconnections of the artificial neurons, which makes the network fault tolerant because if one or some neurons fail, the performance of the network is not significantly impacted, since many connection paths between the neurons exist.

The goal is to identify the vehicle model of Chapter 2 with an artificial neural network, using an adequate network architecture and a set of parameters which best model the vehicle system, described by an appropriate set of input-output data. This chapter describes the method used and the results obtained with the ANN.

3.1 Artificial Neurons

Artificial neural networks require an information processing unit, called artificial neuron, which is the basic component of ANN. Similarly to the biological neuron, the artificial neuron has one or more input signals and only one output signal. The input signals (stimuli) reach the neuron simultaneously [29].

Figure 3.1 represents the neuronal model, where three basic elements are present:

- A set of synapses or connecting links, each one characterized by its own weight. Specifically, an input signal x_j of synapse j, connected to neuron k, is multiplied by the synaptic weight w_{kj}.

- An adder for summing the input signals, weighted by the respective synapses of the neuron, making up a linear combiner.
- An activation function through which the neuron decides whether or not to activate based on the sum of the inputs, limiting the amplitude of the output signal of a neuron. Typically, the normalized amplitude range of the output of a neuron is [0,1] or [-1,1].

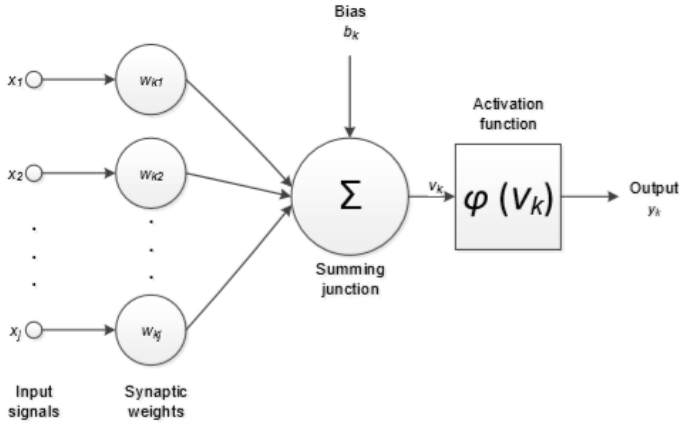

Figure 3.1: Nonlinear model of a neuron [29]

The neuronal model shown in Figure 3.1 also includes an external bias, represented by b_k. This bias can increase or decrease the net input of the activation function, depending on whether it is positive or negative, respectively.

The neuron k can be described mathematically by Equations (3.1) and (3.2).

$$u_k = \sum_{j=0}^{m} w_{kj} x_j \quad , \quad v_k = u_k + b_k \tag{3.1}$$

$$y_k = \phi(v_k) \tag{3.2}$$

The activation functions can have several forms such as threshold, linear and sigmoid, as represented in Figure 3.2.

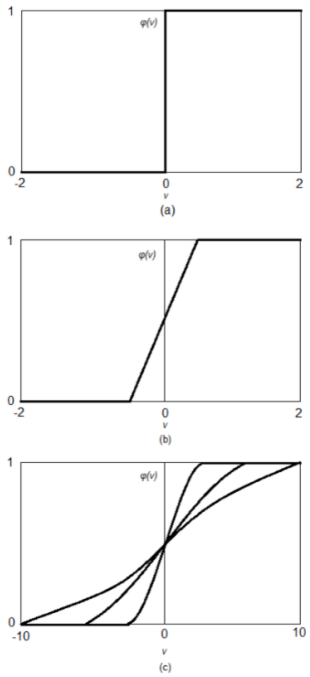

Figure 3.2: (a) Threshold function; (b) Linear function; (c) Sigmoid function with different slopes [29]

3.2 Network Architecture

In a neural network, neurons are organized in layers, with each layer having inputs and outputs. Usually, layers are classified in 3 groups:

1. Input layer - where information from the outside world is provided.
2. Hidden layer - This layer has no direct connection with the outside world (hence the name ”hidden”). Computations are performed and information is transfered from the input layer to the output layer.
3. Output layer - responsible for computations and transferring information from the network to the outside world.

A neural network can have one or multiple hidden layers, as shown in Figure 3.3. There is a direct relationship between the number of layers and respective neurons and the complexity of the neural network [29].

Neural networks can also be classified relatively to the way neurons are connected between the several layers:

- Feedforward network - the data circulates in a single direction, starting from the input layer and going to the output layer. These networks are a static mapping between inputs and outputs, and this mapping can be made through linear or nonlinear relationships. A typical application for feedforward networks is the development of nonlinear models used for pattern recognition and classification.

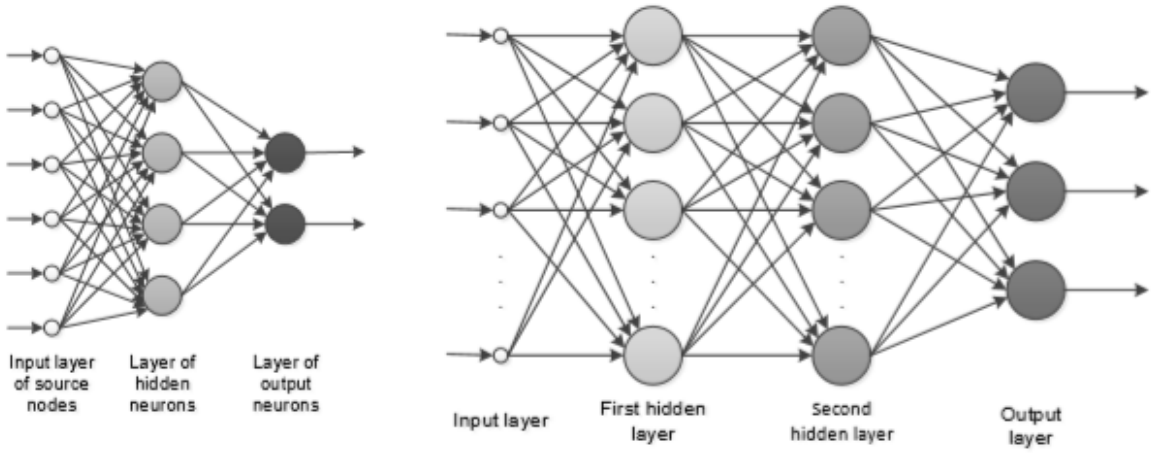

Figure 3.3: Single-layer network (left) and 2 layer network (right) [29]

- Feedback or recurrent network - the network receives as input the feedback of its outputs. This is useful for modelling dynamic behaviour, particularly when the network addresses problems involving time series or pattern recognition that require an internal memory to reinforce the learning process.

The training algorithm for neural network can be of two different kinds: incremental learning and batch learning. In incremental learning the weights of the network are updated once a new input is available, while on batch learning the weights are updated only after all inputs have been made available [30].

3.3 Learning Paradigms

ANN have the ability to learn with examples and are able to extract basic rules from real data. The learning process is finished once a generalized solution for a class of problems is found. The learning process can be divided into two categories:

- Supervised learning - the network is trained by providing input and matching output signals. The training is complete when the neural network reaches a certain precision estimating the outputs for a given sequence of inputs. The network parameters are adjusted under the combined influence of the training vector and the error signal. These adjustments are made with backpropagation algorithms.

- Unsupervised learning - the neural network learns without requiring previous training, as such, the target is obtained through repeated inputs until the ANN retains the knowledge. In this model, the network does not receive external influences to adjust the weight of each variable, having only internal information on how to organize itself. The network is supposed to discover statistically salient features on the input population, similar to the clustering approach.

The backpropagation algorithm is one of the most common learning methods in multilayer neural networks. It computes the gradient of the cost function with respect to the weights of the network for

an input and the matching output. Its efficiency makes it feasible to use gradient methods for training multilayer networks, updating the weights to minimize the cost. The backpropagation algorithm works by calculating the gradient of the cost function relatively to each weight by the chain rule, calculating the gradient one layer at a time, iterating backwards from the last layer to avoid redundant calculations of intermediate terms in the chain rule.

3.4 Nonlinear System Identification

In system identification, the goal is to find the parameters of a given mathematical model such that the difference between the system response and its mathematical model is as little as possible. Generally, linear processes can be represented by an ARX (Auto Regressive eXogenous) model, while nonlinear processes can generally be identified with a NARX (Nonlinear Auto Regressive eXogenous) model. The NARX model structure enables the application of neural networks, fuzzy sytems and neuro-fuzzy systems for approximation of the nonlinear function [31].

Neural networks have been applied to the identification of nonlinear dynamical systems, mostly using multilayer feedforward neural networks with backpropagation learning algorithms, due to the simpler training algorithms when compared to feedback neural networks.

When it comes to nonlinear models, two different models will be used in this work:

- NFIR (Nonlinear Finite Impulse Response) model:

$$\mathbf{y}(k) = F[\mathbf{u}(k-1), \mathbf{u}(k-2), ..., \mathbf{u}(k-n_{\mathbf{u}})] \tag{3.3}$$

  where $F[.]$ is the nonlinear function, k is the current sampling instant, $\mathbf{u}(k)$ is the input vector, $\mathbf{y}(k)$ is the output vector and $n_{\mathbf{u}}$ is the maximum lag of the input.

- NARX (Nonlinear AutoRegressive with eXogenous input) model:

$$\mathbf{y}(k) = F[\mathbf{u}(k-1), \mathbf{u}(k-2), ..., \mathbf{u}(k-n_{\mathbf{u}})), \mathbf{y}(k-1), \mathbf{y}(k-2), ..., \mathbf{y}(k-n_{\mathbf{y}})] \tag{3.4}$$

  where $n_{\mathbf{y}}$ is the maximum lag of the output.

Neural network based models corresponding to the NFIR and NARX models may be obtained by adjusting the weights of a multi-layer perceptron architecture with adequately delayed inputs.

3.5 Data

The vector of inputs ($\mathbf{u}(k)$) of the neural network contains the steering angle (δ) and the reference longitudinal velocity (v_{ref}), the two variables that the driver commands, according to the dynamical model

in Chapter 2:

$$\mathbf{u}(k) = [\delta(k) \quad v_{ref}(k)] \tag{3.5}$$

The vector of outputs ($\mathbf{y}(k)$) contains the velocities of the vehicle (v_x, v_y and $\dot{\psi}$):

$$\mathbf{y}(k) = [v_x(k) \quad v_y(k) \quad \dot{\psi}(k)] \tag{3.6}$$

The outputs are sufficient to describe the trajectory of the vehicle and are also given as feedback input. The dynamical model can be formulated as the following NARX model, which has 2 delays due to the fact that the system being modelled is of second order. The NARX model is also illustrated in Figure 3.4:

$$\mathbf{y}(k) = F[\mathbf{u}(k-1), \mathbf{u}(k-2), \mathbf{y}(k-1), \mathbf{y}(k-2)] \tag{3.7}$$

This ANN is to be used as a prediction model for Model Predictive Control (MPC). As such, for each time step, past simulated values from v_x, v_y and $\dot{\psi}$ are available for the feedback delays in the network, but during the prediction horizon H_p these values are obtained through feedback from the ANN outputs.

Figure 3.4 shows the inputs and outputs of the neural network with the respective delays.

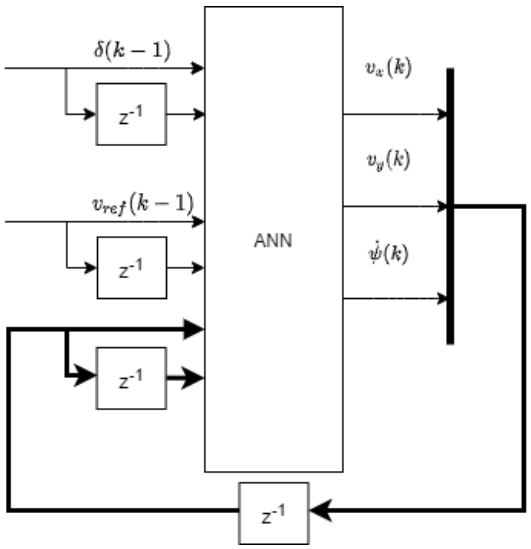

Figure 3.4: Closed loop neural network structure

3.5.1 Data Selection

The data used to construct the neural network model has to be divided in three different categories: training, validation and testing. The training data subset is used to directly estimate the weights and biases of the ANN, which means that performance estimates relative to the training dataset are biased. The validation dataset is used to rank multiple designs and to determine when overtraining and overfitting begin to occur. Overtraining occurs when the performance on the training data is increased at the expense of deteriorating the performance on the nontraining data. Overfitting occurs when more weights

and biases then necessary are used. Validation data is thus used to measure network generalization and to halt training when generalization stops improving. Performance estimates on validation data are significantly less biased than training data estimates. Finally, the testing dataset is used to obtain unbiased estimates of performance on nontraining data [30].

For the selection of the training, validation and testing groups, several data from different tracks was available. All the data was recorded between July and August of 2019 with FST09e, from the following events:

- FS East 2019 endurance event;
- FS East 2019 skidpad event;
- FSG 2019 endurance event;
- Stuttgart practice track.

While the first three events are from FS competitions, the Stuttgart practice track was used to test the car after FSG, in Germany, and before Formula Student Spain (FSS), the last competition of the 2019 season for FST Lisboa. This practice track was the same one used by GreenTeam Uni Stuttgart e.V., the FS team from University of Stuttgart, which also kindly provided a workshop for FST Lisboa during the week between FSG and FSS. The available data was compared between each other in order to provide more insight to which tracks covered the situations the dynamical model can experience, as shown on Figures 3.5 to 3.8.

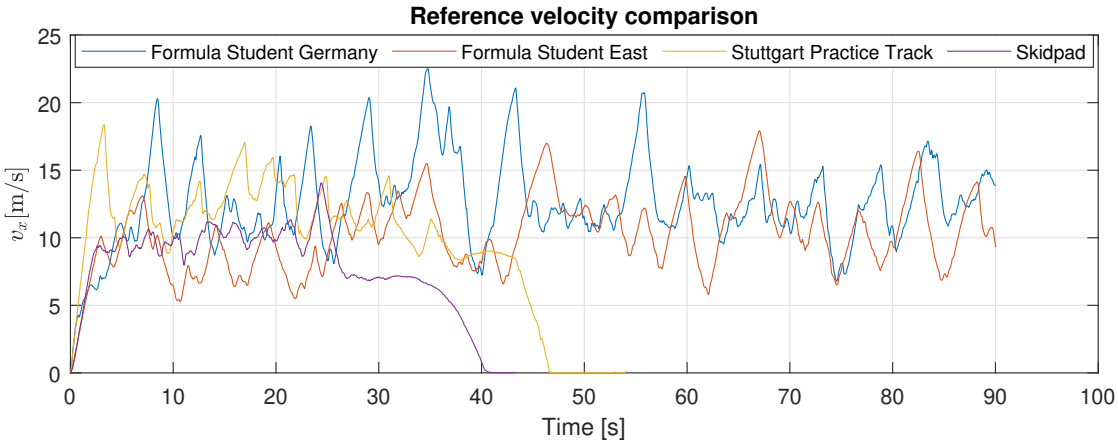

Figure 3.5: Comparison of reference velocity on different tracks

In Figure 3.9 the frequency analysis of the available data is shown, with the single sided power spectrum across the frequency domain, $|P1(f)|$. This was accomplished using a Fast Fourier Transform (FFT) algorithm. This analysis shows that the skidpad has the biggest amount of low frequencies, which makes sense since it is mostly a steady state cornering event. In the remaining events, the spectres are similar between each other.

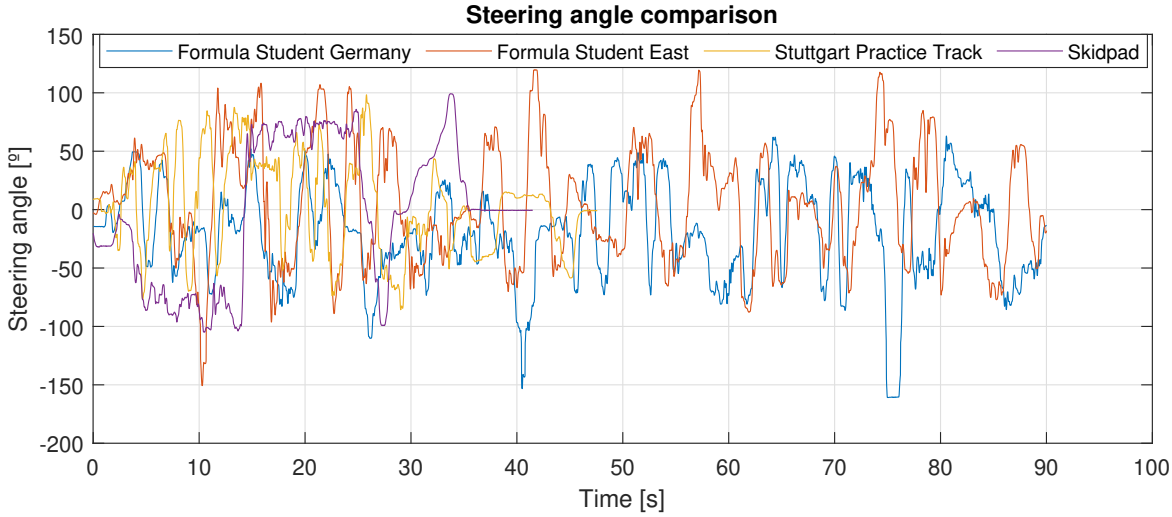

Figure 3.6: Comparison of steering angle on different tracks

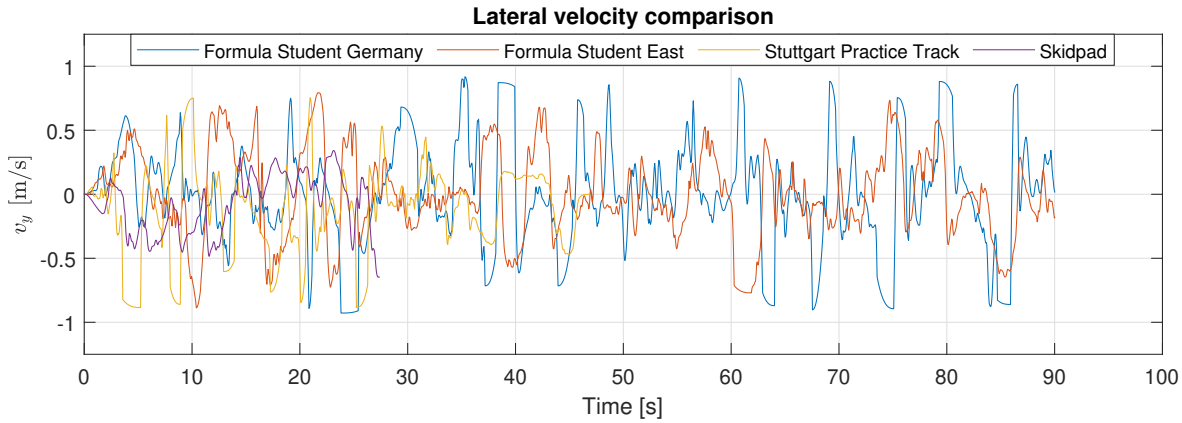

Figure 3.7: Comparison of lateral velocity on different tracks

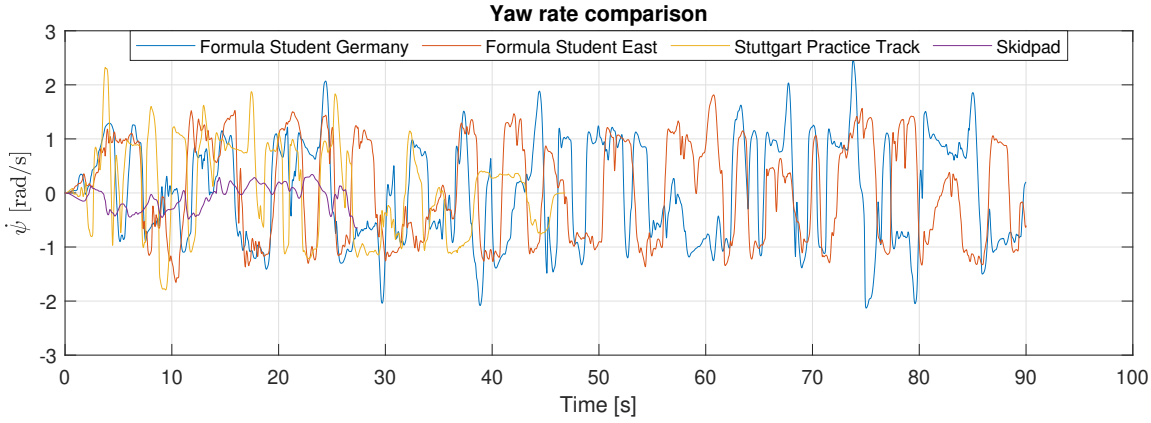

Figure 3.8: Comparison of yaw rate on different tracks

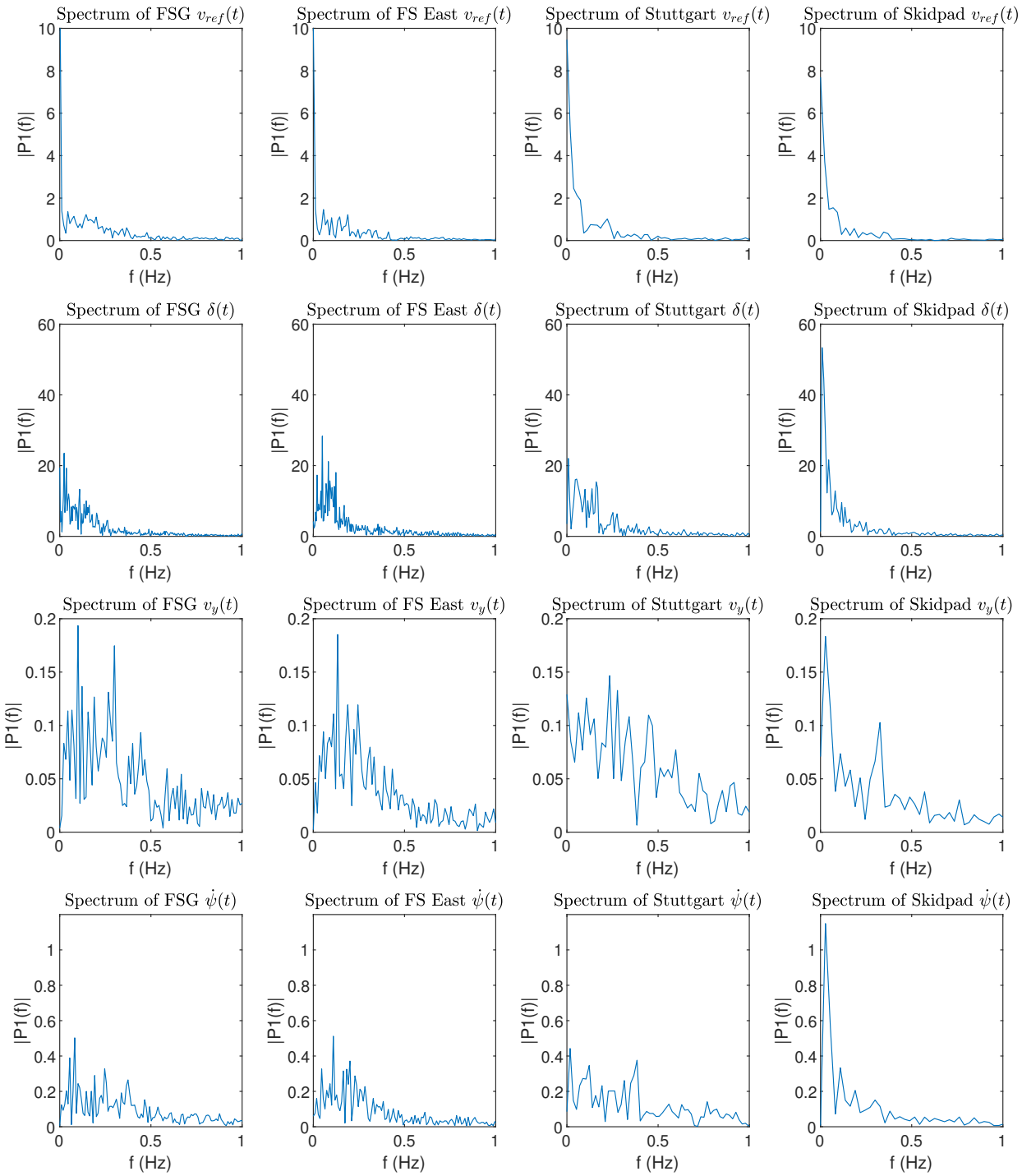

Figure 3.9: Comparison of single-sided amplitude spectrum on different tracks for reference velocity, steering angle, lateral acceleration and yaw rate

The layout of Formula Student tracks from all competitions, for the autocross and endurance events, is built according to the following guidelines [32]:

- Straights: No longer than 80 m;
- Constant Turns: up to 50 m diameter;
- Hairpin Turns: Minimum of 9m outside diameter (of the turn);
- Slaloms: Cones in a straight line with 7.5 m to 12 m spacing;
- Miscellaneous: Chicanes, multiple turns, decreasing radius turns, etc. The minimum track width is 3 m.

It can be seen that the minimum track width of 3 m is relatively small when compared to the width of the vehicle itself, which is approximately 1.2 m, which makes it challenging for a human driver or a controller to maintain the car on its limits while staying inside the track. The track limits are marked with cones on large asphalt areas, so that if cars go offtrack, no major damage is caused.

The presence of certain elements on all FS tracks increases the chance of a neural network successfully learning the dynamics and generalizing well. Since all tracks have relatively similar features, only one track was chosen for training. FS East endurance and FSG endurance both have a larger spectrum of the vehicle dynamics, but since FS East endurance has tight corners for both left and right it was chosen for training. The FSG endurance was left for validation, since it was the second richest event in terms of dynamics captured. For testing, the skidpad was chosen because despite being relatively simple it is a completely different track, which should be sufficient to assess if the neural networks is generalizing well.

3.5.2 Data Preprocessing

Since the steering encoder and the used GPS have different sampling times (0.02 s and 0.055 s, as seen in Section 2.10), the data from these sensors must be resampled before being learned by a neural network. As such, a sampling time of 0.055 s was chosen so that some data points from the steering encoder were discarded, instead of having to interpolate new data points.

Artificial neural networks models learn a mapping between input and output variables. As such, the scale and distribution of the data may be different for each variable. In Figures 3.6 and 3.7, it can be seen that the steering angle varies approximately between -160° and 120° , while the lateral velocity varies between -1 and 1 [m/s]. This sparsity of values makes modelling harder, because smaller weights have a reduced influence in the learning process through backpropagation.

The method chosen to rescale the data was standardization, which rescales the distribution of values so that the mean of observed values is 0 and the standard deviation is 1. A value $u(k)$ is standardized to $u(k)_{stdd}$ by the following equation:

$$u(k)_{stdd} = \frac{u(k) - \overline{u}}{\sigma_u} \tag{3.8}$$

where $\overline{u}$ represents the sample mean and σ_u represents the standard deviation of the sample.

In Figure 3.10 it can be visualized how standardization rescales the data. On the left side of the figure, the non standardized data has very different scales, while the standardized data on the right side has similar scales and is more adequate for an ANN to learn.

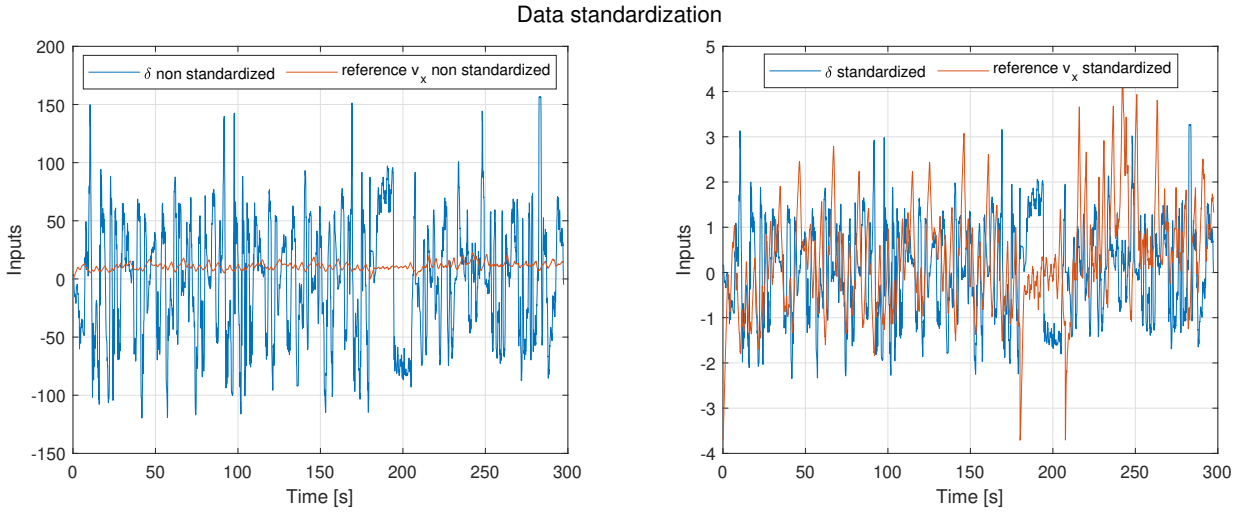

Figure 3.10: Non standardized data of the inputs (steering angle and reference velocity) on the left and standardized data of the same inputs on the right

3.6 Application and Results

Supervised batch learning was used since all the data of the inputs and target outputs is available *a priori*. The learning algorithm used was the Levenberg-Marquardt, which is relatively quick and deals with most situations. The performance of the ANN is measured by the mean squared error (MSE), which is the sum of the squared difference between the real target outputs ($\mathbf{y}(k)$) and the ones calculated by the ANN ($\hat{\mathbf{y}}(k)$), divided by the total number of samples N, according to Equation (3.9):

$$MSE = \frac{1}{N}\sum_{k=0}^{i=N}(\mathbf{y}(k) - \hat{\mathbf{y}}(k))^2 \tag{3.9}$$

Training is done with approximately two laps of FS East endurance, validation is done with approximately one lap of FSG endurance and the test is done with a skidpad run (2 circles to the right and 2 circles to the left).

Training is performed in two stages. In the first stage the network is created and trained in open loop form, as shown in Figure 3.11. This allows the network to be supplied with the correct past outputs during training to produce the correct current outputs. Afterwards, the network is converted to closed loop, as in Figure 3.12, which is the way it is intended to be used, and it is retrained in closed loop, to further improve its performance, using the network trained in open loop as a starting point. In both figures it can be seen that the hidden layer activation function is a hyperbolic tangent sigmoid function while the output layer activation function is linear.

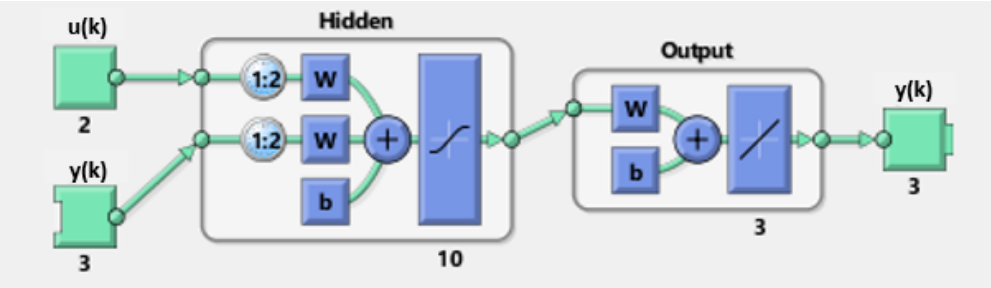

Figure 3.11: Neural network structure in open loop during training [30]

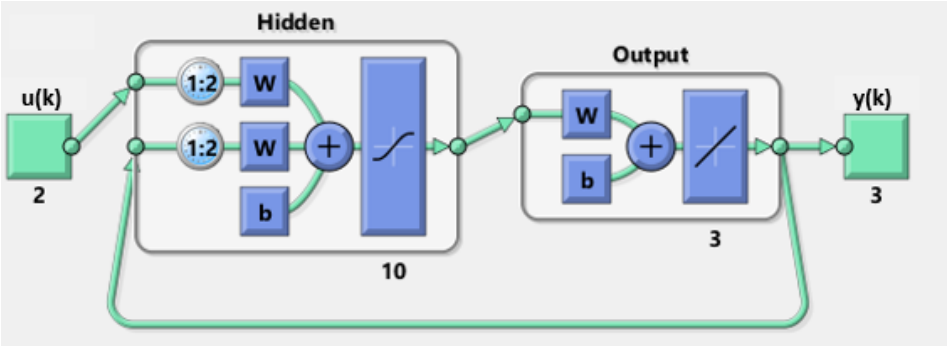

Figure 3.12: Neural network structure in closed loop [30]

Several parameters were varied and the best results were obtained with one hidden layer with 10 neurons, with 2 delays for both input and output feedback.

In Figures 3.13, 3.14 and 3.15 the results in closed loop are presented, i.e. the neural network used its own outputs as feedback, and not the target outputs. Training, validation and testing are shown continuously but during training, each dataset had its separate initialization values on the output feedback (e.g. it wouldn't make sense to start the test with output feedback from validation).

In Figures 3.13 to 3.15, it can be seen that good approximations were achieved on all datasets. This indicates that the obtained neural network may be used as a reliable dynamical model for the vehicle.

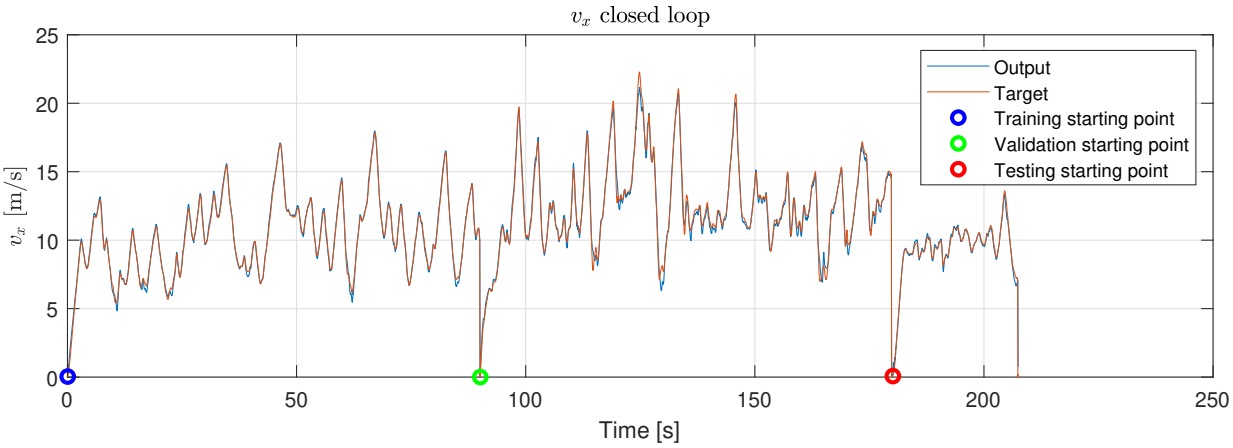

Figure 3.13: Comparison between the target v_x and the output of the ANN

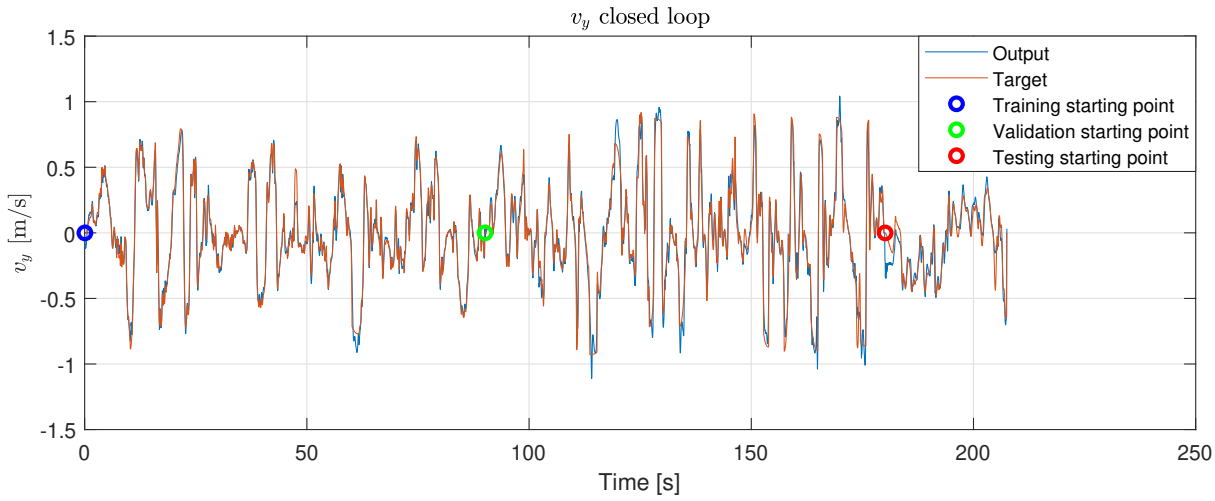

Figure 3.14: Comparison between the target v_y and the output of the ANN

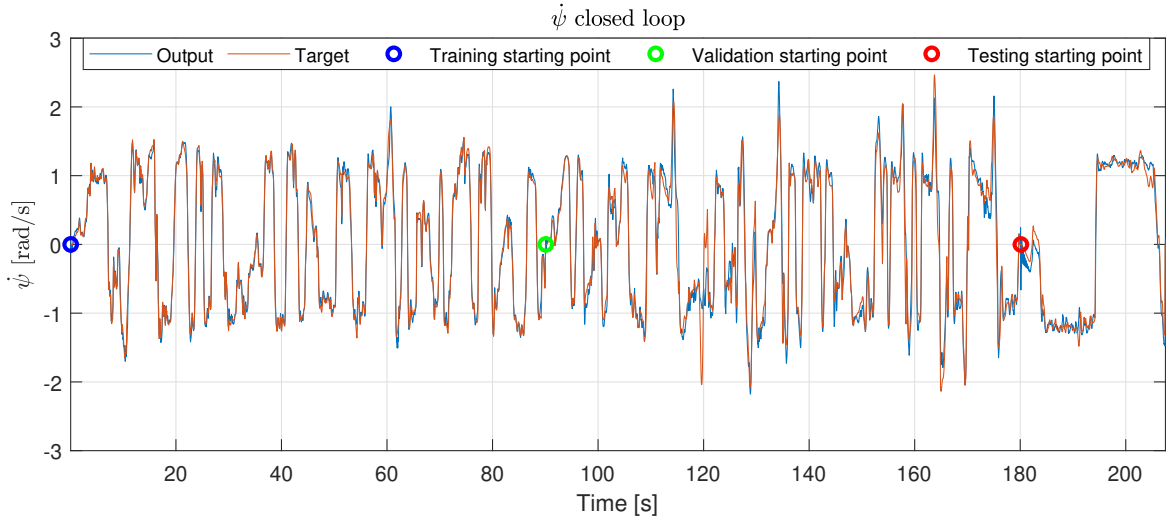

Figure 3.15: Comparison between the target $\dot{\psi}$ and the output of the ANN

In Figure 3.16 the MSE is shown with respect to the number of iterations. The test curve shows that the neural network performed well in the skidpad, a track with a different layout.

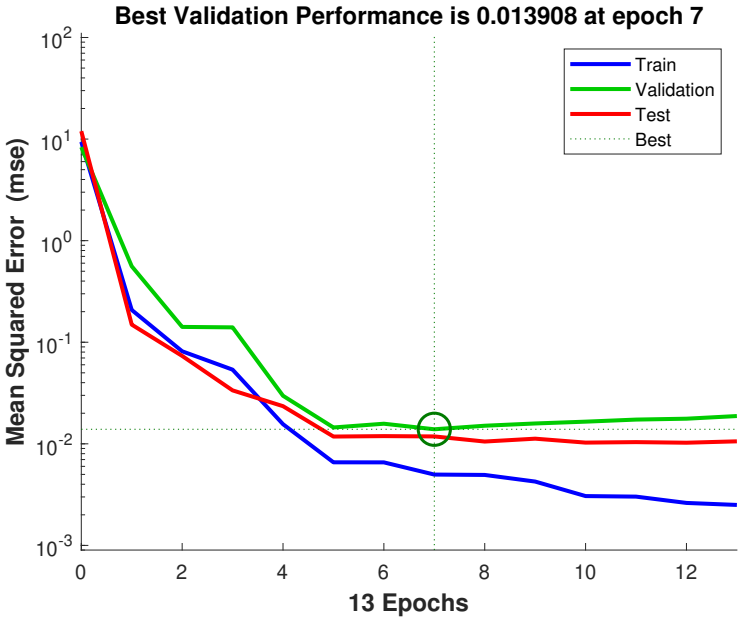

Figure 3.16: Performance of the ANN

The regression plots in Figure 3.17 also allow to evaluate the quality of the neural network by analysing the relationship between the outputs and the targets. The dotted line represents, in each plot, the ideal result: outputs equal to targets. The solid line represents the best linear regression line that adjusts the outputs and the targets. If R = 1, there is a linear relationship between outputs and targets and if R is close to zero, then there is not a linear relationship between the data [30]. Analysing the obtained plots, it can be seen that there is an almost linear relationship between the targets and the outputs, which means the results of the ANN are close to the targets, indicating a good learning result of the vehicle dynamical model.

The obtained ANN can now be used to predict the evolution of the states of the system, as will be seen in Chapter 4.

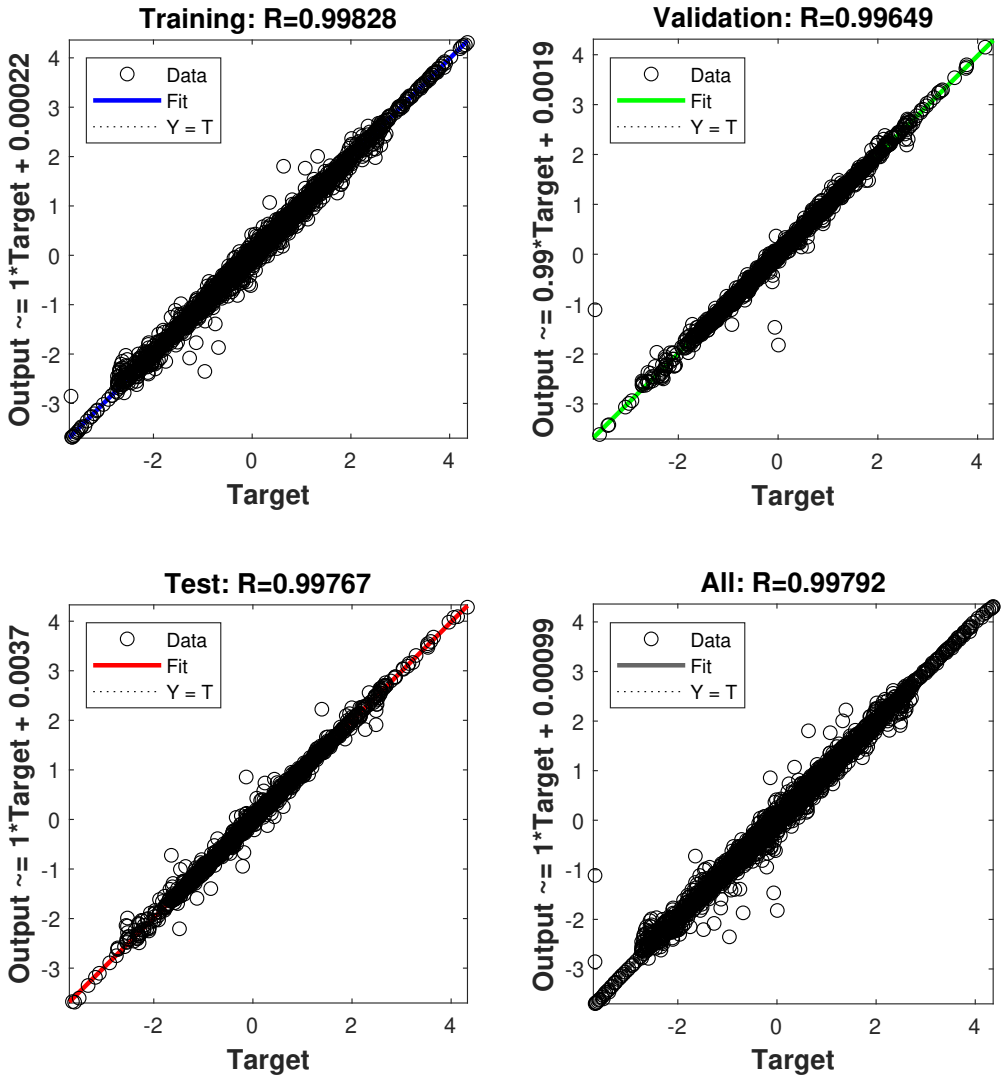

Figure 3.17: Regressions for training, validation, testing and combined

Chapter 4

Nonlinear Model Predictive Control

In the present chapter it is assumed that there is available *a priori* knowledge of the track layout, and the objective is to design a controller capable of generating the appropriate references for the actuators to follow the path. It is assumed that the perception subsystems of the car retrieve visual information from the track limits by detecting cones, and this information is used to generate a trajectory, which will be the reference for the car to follow.

To follow the reference trajectory, a controller based on nonlinear Model Predictive Control (MPC) is proposed. This solution has the advantage of not needing any predetermined logic, only the track layout and the vehicle model. It is a design objective that the proposed controller works on any Formula Student track.

4.1 Introduction

MPC can handle multiple-input multiple-output (MIMO) systems with interactions between their inputs and outputs, as is the case of the vehicle dynamics model used. Because of these interactions it is often challenging to design controllers for MIMO systems using traditional approaches such as PID [33]. An advantage of MPC is its ability to deal with constraints that affect the evolution of the states of the system. MPC also has preview capabilities, which is useful when the reference is known in advance because the controller can better react to those changes and improve its performance.

The principle of MPC lies in repeating an optimization of a cost function, defined in a finite prediction horizon. The optimization uses an explicit model to predict the response of the system with a certain control action. The result of this optimization is a sequence of control actions that minimizes the cost function. The first control action of the sequence is used until the next sampling instant Ts, when the optimization is repeated over the receding horizon with the updated information about the states of the system [34]. This means there is a need for online calculations, which can be costly in terms of computation, one of the disadvantages of MPC type controllers.

A hypothetical single-input single-output (SISO) MPC system that has been operating for several sampling instants is illustrated in Figure 4.1. At each time instant k, the best control sequence over the prediction horizon H_p is found, such that the output prediction of the model approaches the reference trajectory over the horizon H_p, while respecting the constraints given. After finding the best control sequence, only the first control move, $\mathbf{u}(k)$, is applied, after which new updated measurements are obtained and the horizon shifts one sampling instant forward, repeating the optimisation for instant $k+1$, and the process is repeated for every sampling instant.

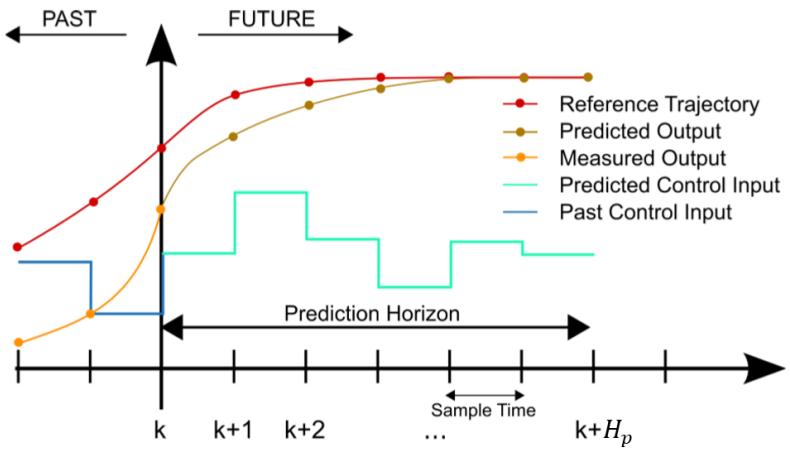

Figure 4.1: Receding horizon control principle [35]

4.2 Contouring Formulation

The objective of the contouring formulation is to follow a reference path as fast as possible. The contouring formulation from [36] is adapted and implemented in the present case. The reference paths are the trajectories obtained in the vehicle model shown in Chapter 2, using as inputs real data from steering angle and reference velocity. The reference path is then parameterized by its arc length (t). For this, a third order spline is used, since it offers a fast way to evaluate any point along the contour $(X_{ref}(t), Y_{ref}(t))$. In order to follow the path, the position of the car (X, Y) has to be connected to the position on the path, i.e. the arc length. This arc length is represented by t and it can be computed by projecting the position of the car (X, Y) onto the reference path.

4.2.1 Parameterization of the Reference Trajectory

The arc length parameter is represented by $t \in [0, L]$, where L is the total length. The splines used for this parameterization are obtained by an offline fitting of the trajectory. Using this parameterization, any point $X_{ref}(t), Y_{ref}(t)$ on the path can be obtained by evaluating a third order polynomial for its argument

t. The angle of the tangent to the path at the reference point, $\Phi(t)$, with respect to the x-axis can be calculated by:

$$\Phi(t) \triangleq arctan\Big(\frac{\partial Y_{ref}(t)}{\partial X_{ref}(t)}\Big) \tag{4.1}$$

4.2.2 Trajectory Error Calculation

Error measures define the deviation of the current position of the car X, Y from the desired reference point $X_{ref}(t), Y_{ref}(t)$ and this measure is needed to formulate the MPC problem. This deviation is the contouring error, which can be visualized in Figure 4.2:

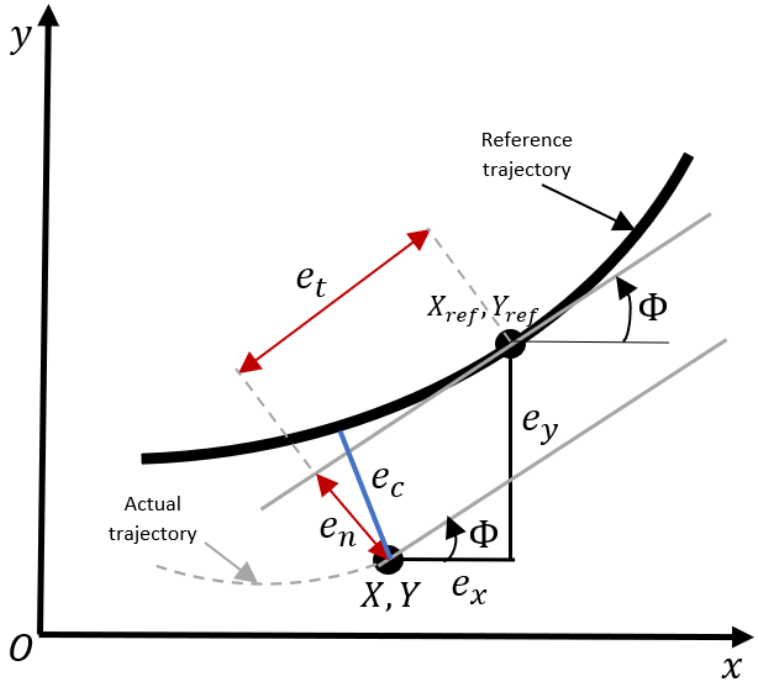

Figure 4.2: Contouring error of the vehicle (X, Y) relative to the reference trajectory (X_{ref}, Y_{ref})

Let $t_{min} : \mathbb{R}^2 \longrightarrow [0, L]$ be a projection operator on the reference trajectory defined by:

$$t_{min} \triangleq arg\min_{t} \Big(X - X_{ref}(t)\Big)^2 + \Big(Y - Y_{ref}(t)\Big)^2 \tag{4.2}$$

where t_{min} is the arc length that corresponds to the closest point in the reference path in relation to the current car position. The orthogonal distance from the car to the reference path is given by the contouring error e_c, which can be approximated by its normal component e_n:

$$e_n(X, Y, t_{min}) \triangleq sin\Big(\Phi(t_{min})\Big) * \Big(X - X_{ref}(t_{min})\Big) - cos\Big(\Phi(t_{min})\Big) * \Big(Y - Y_{ref}(t_{min})\Big) \tag{4.3}$$

Note that this is not the same as calculating the Euclidean distance because of the tracking error, represented in Figure 4.2 by e_t, which inevitably exists because the values of t are discretized. The

presented formulation remains accurate even if tracking error is present.

4.3 MPC Problem Formulation

Figure 4.3 shows the overview of the proposed control architecture. Since all simulations are to be made in Simulink®, the nonlinear MPC block is used, customizing it to use the cost function in Equation (4.4a) with the respective constraints and the neural network state prediction model described in Section 3.6. The function *fmincon* is used to minimize the cost function while respecting the constraints. The MPC outputs two control actions: steering angle (δ) and reference velocity (v_{ref}), which are the inputs given to the car dynamical model detailed in Chapter 2. This model then outputs the velocities v_x, v_y and $\dot{\psi}$ which are converted to global coordinates and feedback to the MPC.

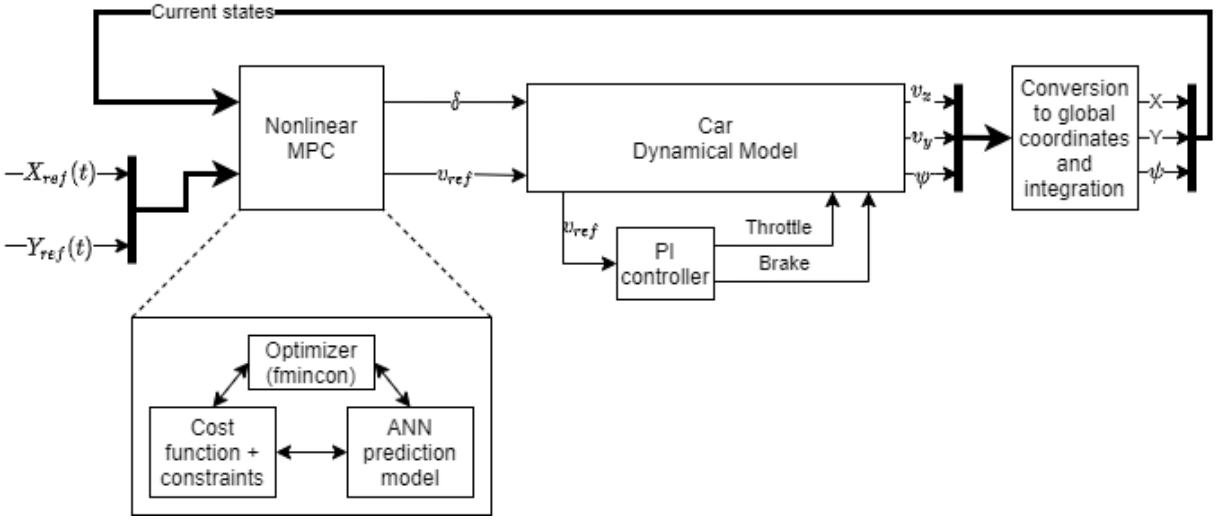

Figure 4.3: Proposed Nonlinear MPC Control Architecture

4.3.1 Cost Function

After defining the error measures, the MPC problem can now be formulated. The formulation shown is based on the one used in [36] with some adaptations. The goal is to minimize the contouring error while maximizing the progress along the track over a finite horizon of H_p sampling times, while respecting model dynamics and input constraints:

$$J = \min \sum_{k=1}^{H_p} ||e_n(X(k), Y(k), t_{min}(k))||^2 w_n - w_t * t_{min,N} + ||\Delta\delta(k)||^2 w_{\Delta\delta} + ||\Delta v_{ref}(k)||^2 w_{\Delta v_{ref}} + ||\delta|| w_\delta \tag{4.4a}$$

$$s.t. \quad \mathbf{y}(k) = F[\mathbf{y}(k-1), \mathbf{y}(k-2), \mathbf{u}(k-1), \mathbf{u}(k-2)] \tag{4.4b}$$

$$\underline{\mathbf{u}} \leq \mathbf{u}(k) \leq \overline{\mathbf{u}} \tag{4.4c}$$

$$\underline{\Delta \mathbf{u}} \leq \Delta \mathbf{u}(k) \leq \overline{\Delta \mathbf{u}} \tag{4.4d}$$

where $X(k), Y(k)$ is the position of the car at time step k, determined by the nonlinear prediction model $F[.]$ in Equation (4.4b), which is based on the ANN obtained in Section 3.6, where $\mathbf{u}(k)$ represents the input vector and $\mathbf{y}(k)$ represents the output vector. The term e_n is the approximation to the contour error defined in Equation (4.3). By subtracting the term $t_{min,N}$ to the cost function, the arc length parameter t is maximized, thus the car progress over the track is also maximized. The variable δ is the steering angle control action, and v_{ref} is the reference velocity control action. This cost function minimizes the variations of the control actions, and it also minimizes the module of the steering wheel angle. Constraints (4.4c) and (4.4d) limit the inputs **u** to physically admissible values. Tuning weights (w_i) exist for each parameter.

The imposed constraints are shown in Equations (4.5) to (4.8). These constraints limit the rotation of the steering wheel and its maximum rotation speed. The maximum rotation depends on the steering geometry and the maximum rotation speed depends on the actuators used for the steering wheel in the autonomous vehicle. A minimum vehicle velocity of 5 km/h was defined to avoid trivial solutions and a maximum velocity of 30 km/h was defined. This value can be increased once the algorithm is successfully tested in the real prototype. The variable Δv_{ref} represents the longitudinal acceleration of the car and the upper limit for the constraint is the maximum longitudinal acceleration of the car obtained during simulations, while the lower limit was set considerably lower than the maximum negative acceleration simulated, which was 19.4 m/s^2. This is to prevent solutions with the car braking on the limit of tyre grip while also turning the steering wheel, which leads to unstable behaviour. A human driver has the ability to brake and turn at the same time while balancing the limits of grip, but this is harder to reproduce with a controller.

$$-150 \leq \delta(k) \leq 150 \text{ [º]} \tag{4.5}$$

$$-3000 \leq \Delta\delta(k) \leq 3000 \text{ [RPM]} \tag{4.6}$$

$$5 \leq v_{ref}(k) \leq 30 \text{ [km/h]} \tag{4.7}$$

$$-10 \leq \Delta v_{ref}(k) \leq 14.1 \text{ } [m/s^2] \tag{4.8}$$

The nonlinear prediction model ($F[.]$) used to predict the states $\mathbf{y}(k)$ during the prediction horizon receives as input the steering angle and the reference velocity, similarly to the ANN. However, the outputs of the ANN are simply the velocities of the car (v_x, v_y and $\dot{\psi}$), which need to be converted to world coordinates. This is done with a similar approach to the one used in Section 2.6, except now the integrals will be approximated by assuming constant acceleration between each time step, so that calculations are simplified. The time step (T_s) of the MPC is the same as the ANN: 0.055 s.

The yaw (or heading) angle (ψ) is given by the approximation of the time integration of the yaw rate ($\dot{\psi}$):

$$\psi(k+1) = \psi(k) + \dot{\psi}(k)T_s \tag{4.9}$$

To keep track of the trajectory of the vehicle, its position has to be computed in the global frame:

$$\overline{p}(k+1) = \begin{bmatrix} X(k+1) \\ Y(k+1) \end{bmatrix} = \begin{bmatrix} X(k) + \Big(v_x(k)cos(\psi(k)) - v_y(k)sin(\psi(k))\Big)T_s \\ Y(k) + \Big(v_x(k)sin(\psi(k)) + v_y(k)cos(\psi(k))\Big)T_s \end{bmatrix} \tag{4.10}$$

where $\overline{p}$ denotes the position of the car in the world frame.

4.4 Simulation Results

In order to test the developed controller, a reference trajectory is given, in this case, using data from FS East endurance. The steering angle and reference velocity are given to the dynamical model described in Chapter 2 and the obtained output trajectory is saved and used as reference for the MPC. Due to errors present in the dynamical model, the trajectory obtained with this model is not exactly the same as the real one, as was also seen in Figures 2.16 and 2.21. If during simulation, the trajectory over one corner in the beginning is deviated from reality, the rest of the trajectory will be affected. As a result, the track does not form a closed circuit, i.e. the finishing coordinates after a lap do not coincide with the starting coordinates. In Figure 4.4 the trajectory obtained in this simulation is shown.

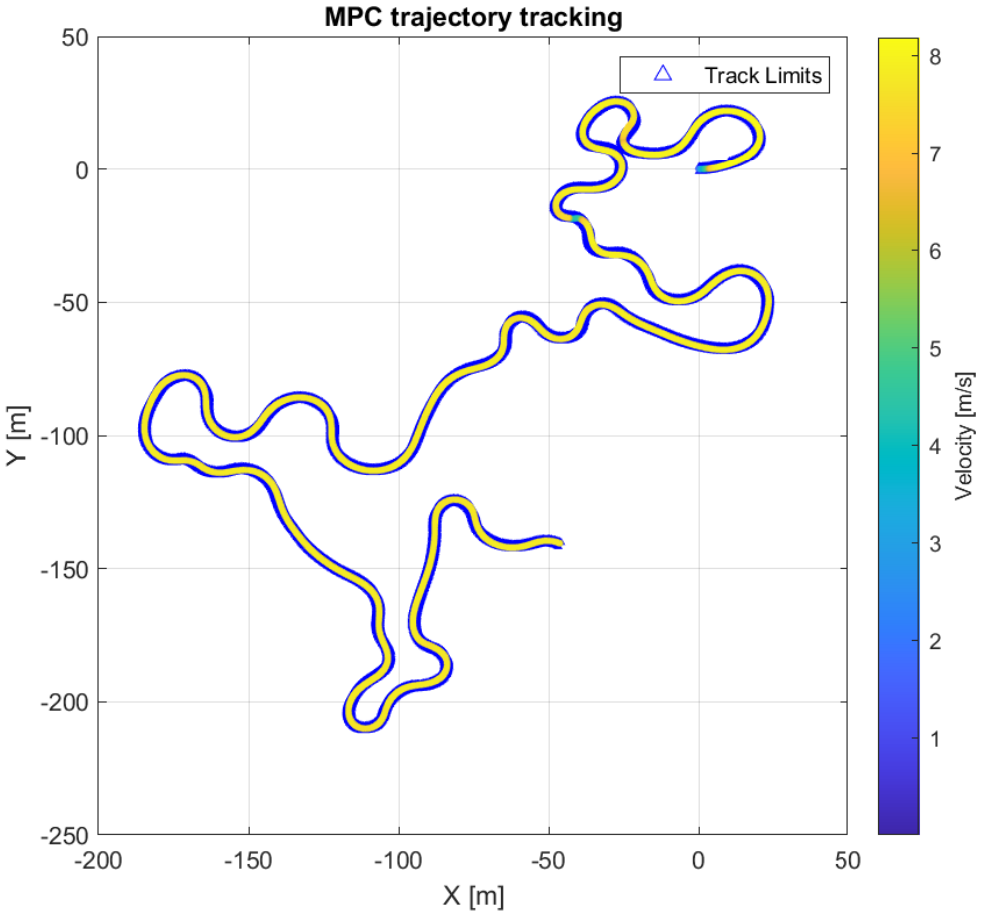

Figure 4.4: Simulated MPC trajectory with the velocity profile for FS East track

Figure 4.5 gives a closer look on how the car follows the reference trajectory, represented by a black line. The blue markers show the track limits, which are marked with cones in the competitions. In Formula Student competitions the track width varies along the track but the minimum width is 3 m, according to competition rules [32]. To take in consideration the fact that the trajectories shown are for the CoG of the car, the track limits represented correspond to the minimum track width subtracting the width of the car $(1.2\ m)$: $3 - 1.2 = 1.8\ m$. Hence if the output trajectory is within the limits of the blue markers, this means that the whole car should stay inside the track.

Figure 4.6 shows the steering wheel angle over time for FS East Endurance. The constraints on the maximum rotation of the steering wheel are almost hit three times in the first 30 s.

The sampling time of the controller is $T_s = 0.055\ s$ and the prediction horizon length is H_p = 15 time steps, which gives an ahead prediction of 0.825 s. The control horizon is $H_c = 3$ time steps. Several combinations of these parameters were used but the closest tracking of the reference trajectory was achieved with this set of parameters.

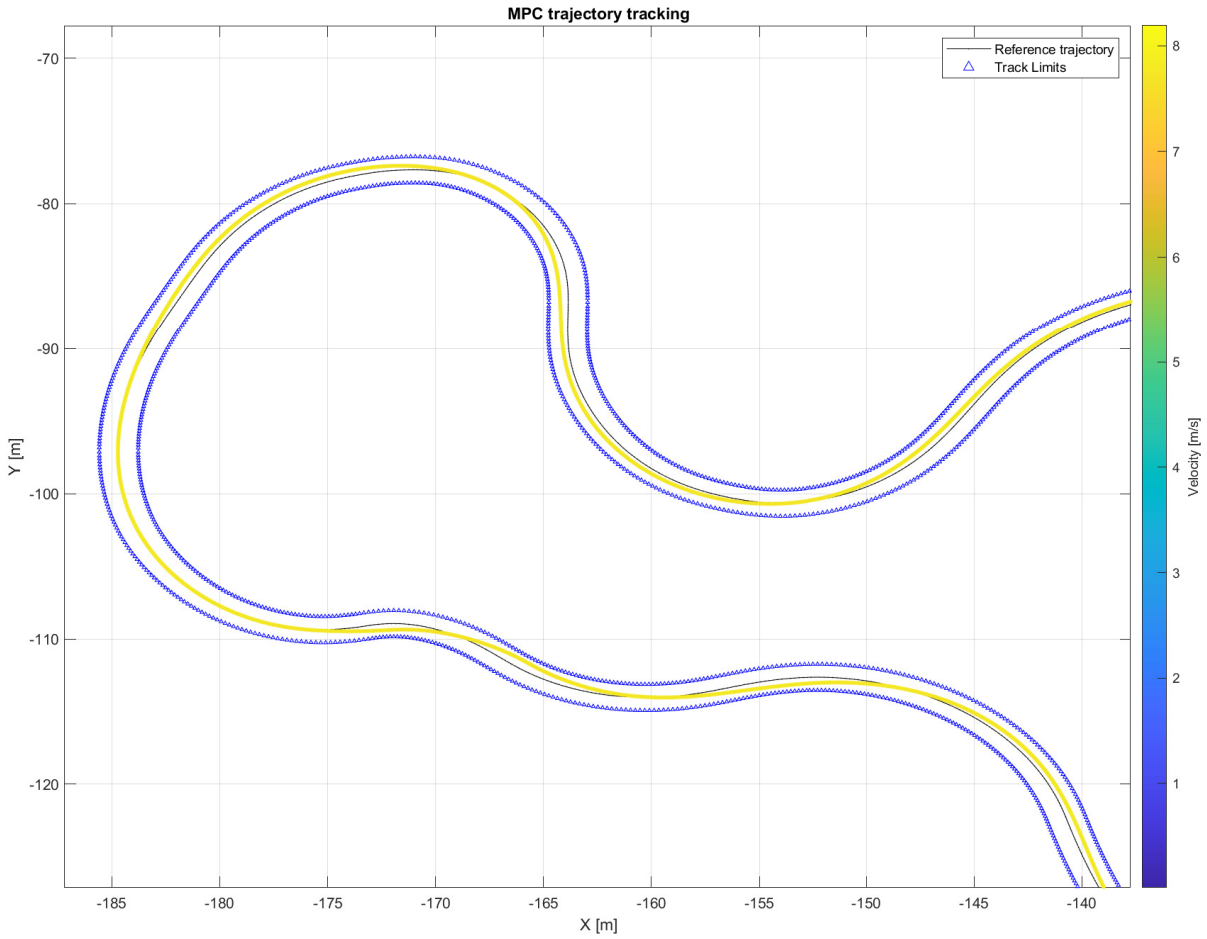

Figure 4.5: Close-up of simulated trajectory with MPC for FS East endurance, after the third corner

It can be verified that the maximum velocity of the car was constrained to 30 km/h, which is approximately 8.3 m/s. The color of the trajectory indicates the velocity of the car, which is constant throughout most of track, as indicated by the yellow line in Figures 4.4 and 4.5. The car stays inside the track limits throughout the entire run.

Figure 4.7 shows the MPC controller following the reference trajectory of the centreline of the skidpad. Unlike the reference trajectory from FS East endurance, which was obtained from simulation, the reference trajectory for the skidpad was defined geometrically so that it is the centreline of the skidpad layout. In Figure 4.7, it can be seen that the car also stays within track limits during the entire simulation. The parameters used for the skidpad were the same as for FS East, except for one change that was done due to the fact that the track intersects itself several times in the middle, which only happens in the skidpad event. Previously, when calculating the contouring error, the definition of the argument t_{min} in Equation (4.2) was used, which is the arc length that corresponds to the reference point closest to the current trajectory point. This is valid when the track does not intersect itself, but if that is not the

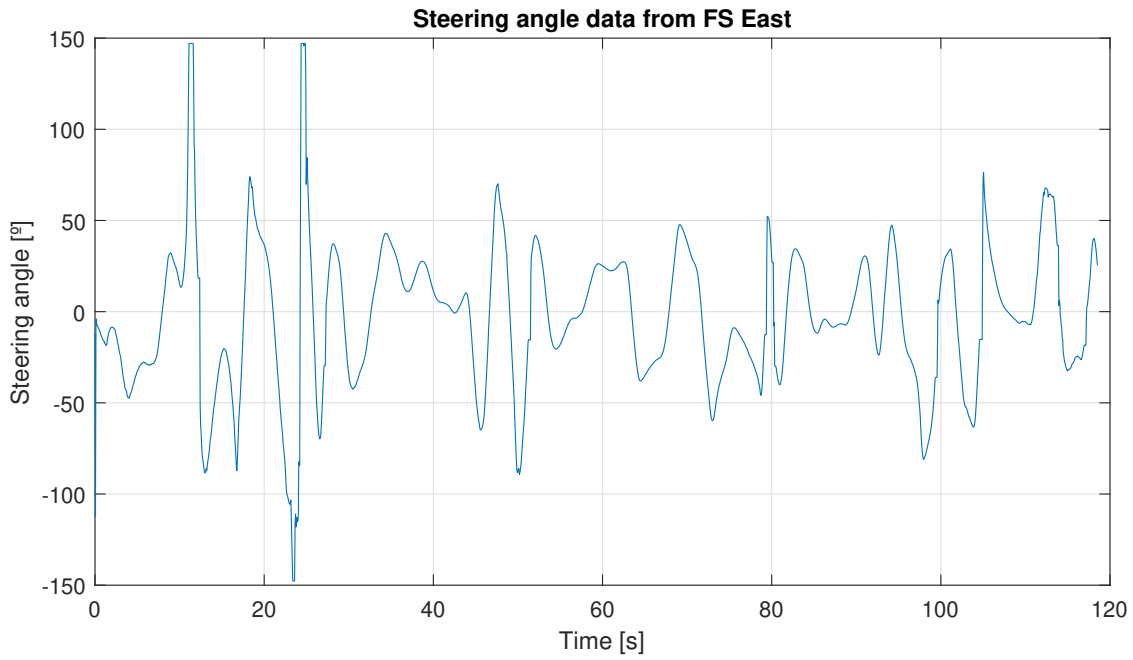

Figure 4.6: Steering wheel angle in the MPC FS East endurance simulation

case, then the closest position point may not be the right one to use as a reference. To solve this issue for the case of the skidpad, the references for the calculation of the contouring error are calculated with the estimation $t_{min}(k+1) = t_{min}(k) + v_x \times T_s$. This means the reference trajectory over the prediction horizon, instead of corresponding to the points closest to the current trajectory, corresponds to the points $X(t), Y(t)$ with the parameter t closest to t_{min}.

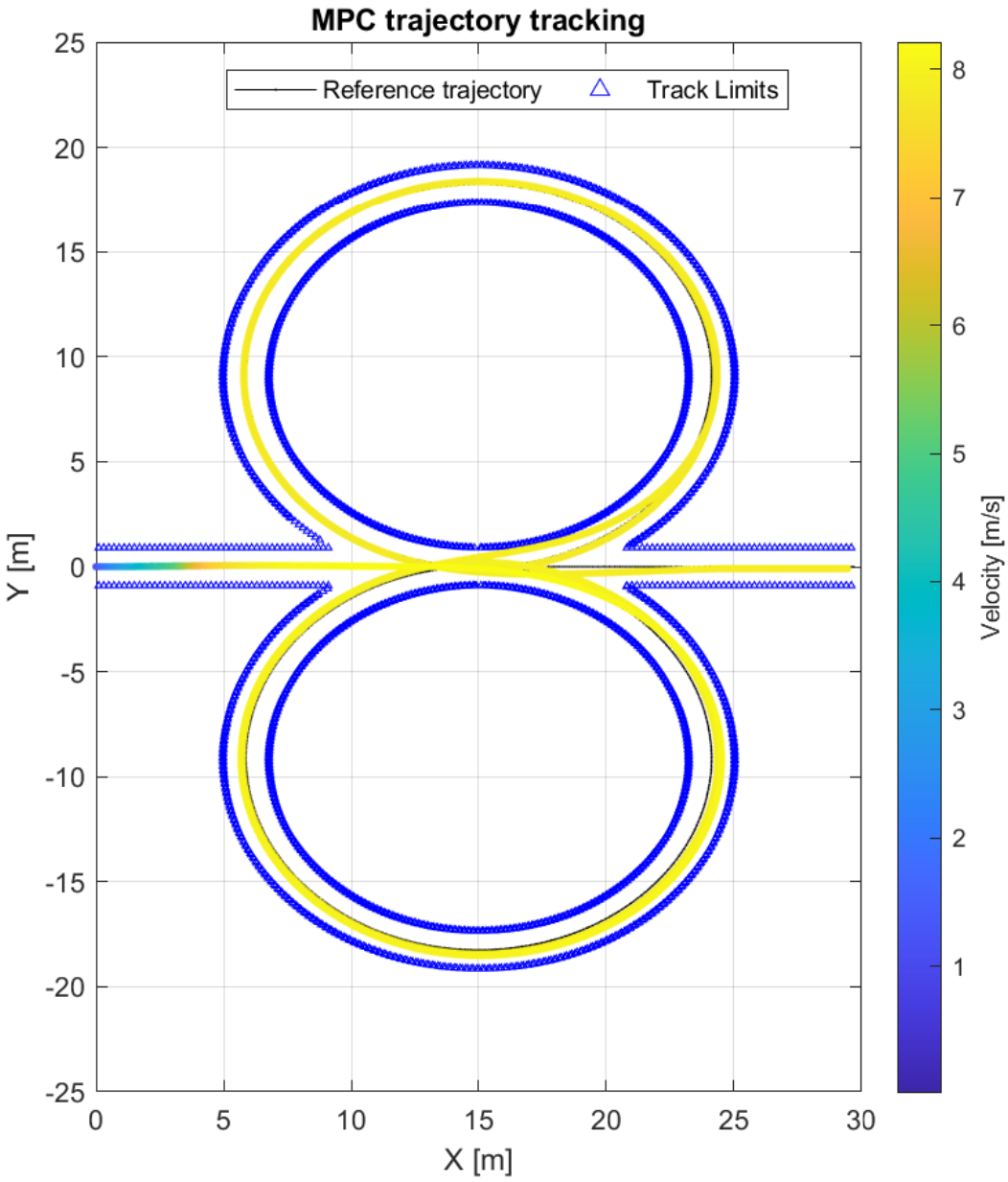

Figure 4.7: Simulated trajectory with MPC with the velocity profile for skidpad track - the car starts on the left, performs two circles to the right followed by two circles to the left and finishes in the middle

Figure 4.8 shows the steering angle for the skidpad over time, where the transition from the right circle to the left circle can be seen around 17.5 s.

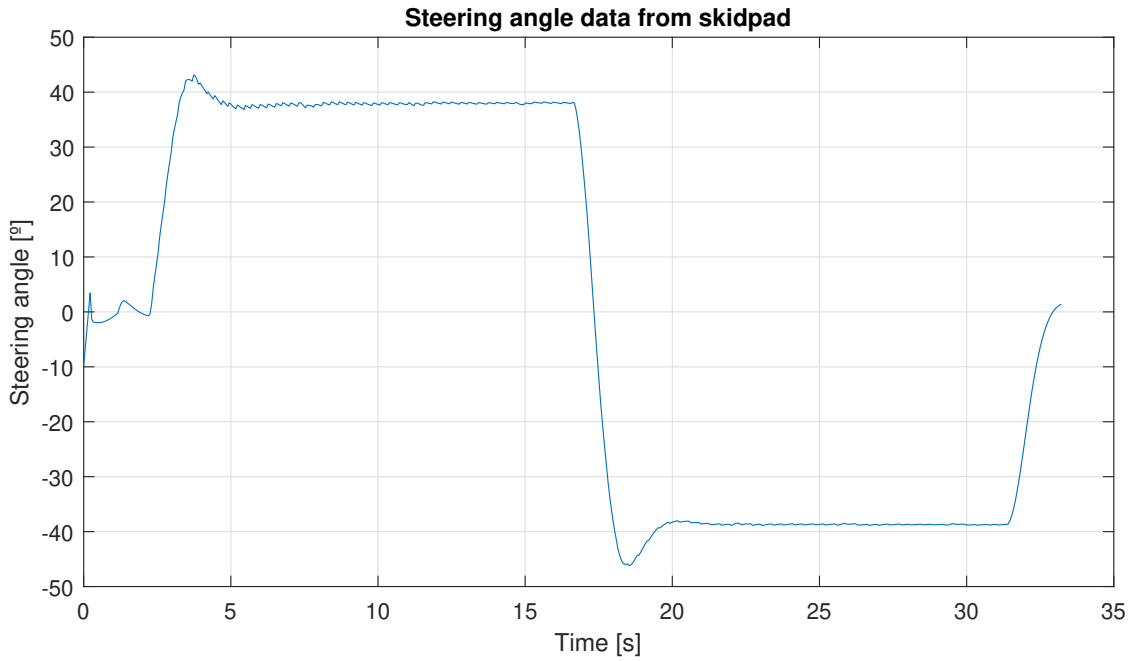

Figure 4.8: Steering wheel angle in the MPC skidpad simulation

To further test the robustness of the MPC, particularly dependant on whether the ANN has indeed learned the dynamics of the system and appropriately generalized the training data, a different reference trajectory was given to the MPC controller, as shown in Figure 4.9. This reference trajectory is from a practice track in Stuttgart, with corners built according to competition rules, except for the fact that this track intersects itself due to the limited physical space available to mount the practice circuit.

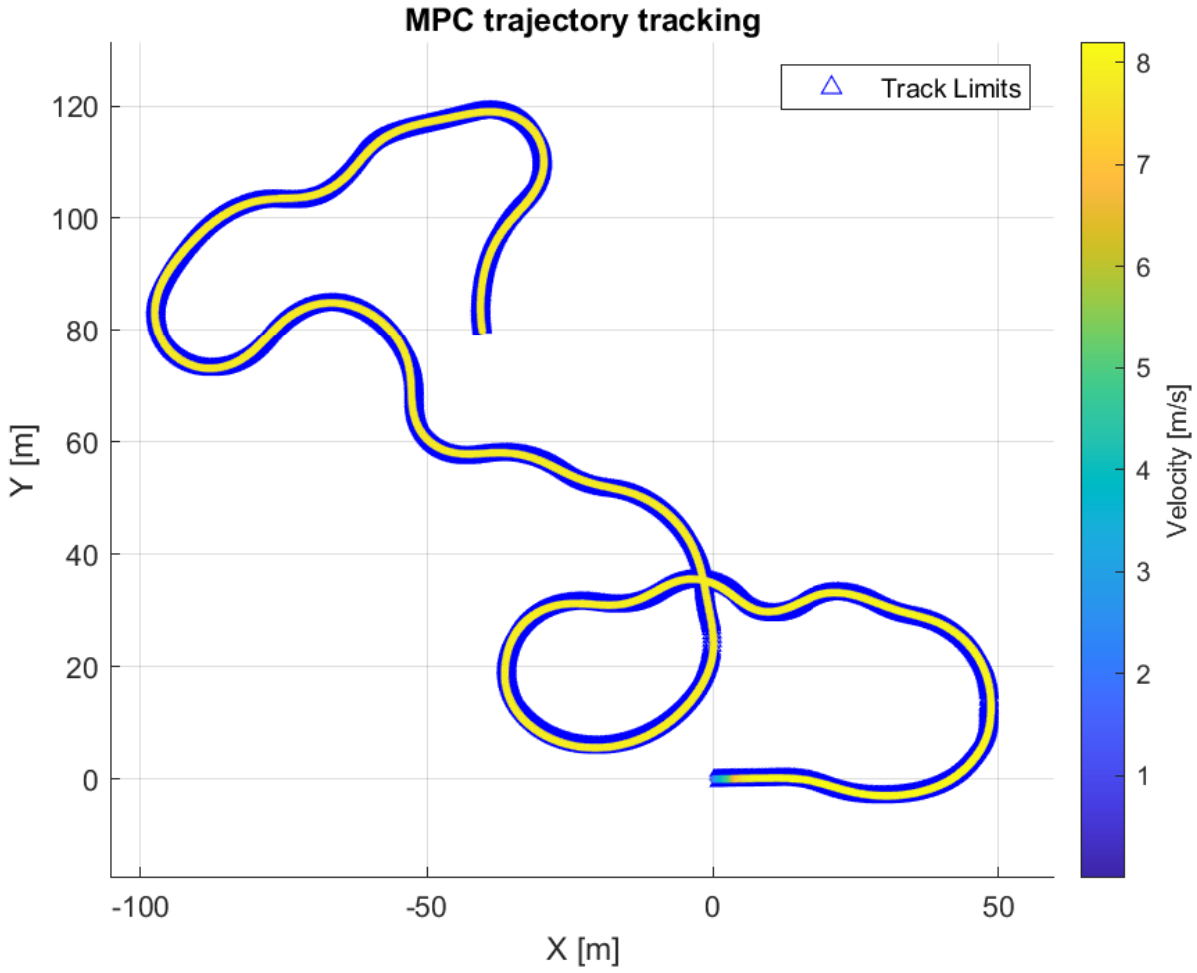

Figure 4.9: Simulated trajectory with MPC with the velocity profile for Stuttgart practice track

Unlike the skidpad which intersects several times in the middle, this track only intersects itself once, and the intersection is orthogonal. For this reason, the initial definition of the Equation 4.2 for the argument t_{min} was used, which is the arc length that corresponds to the reference point closest to the current trajectory point and the results in Figure 4.9 show that the controller still followed the reference through the intersection.

Figure 4.10 shows a closer look of a particular corner, where the black line represents the reference trajectory and the blue markers represent the track limits considering always the minimum track width as well as the width of the car itself.

In Figure 4.11 another close-up of the of the simulated trajectory is shown during a series of corners. Notice that the reference trajectory is always the centreline, but the MPC controller follows the reference while slightly approximating the apexes of the corners, thus shortening the path, similarly to how a human would drive in a race.

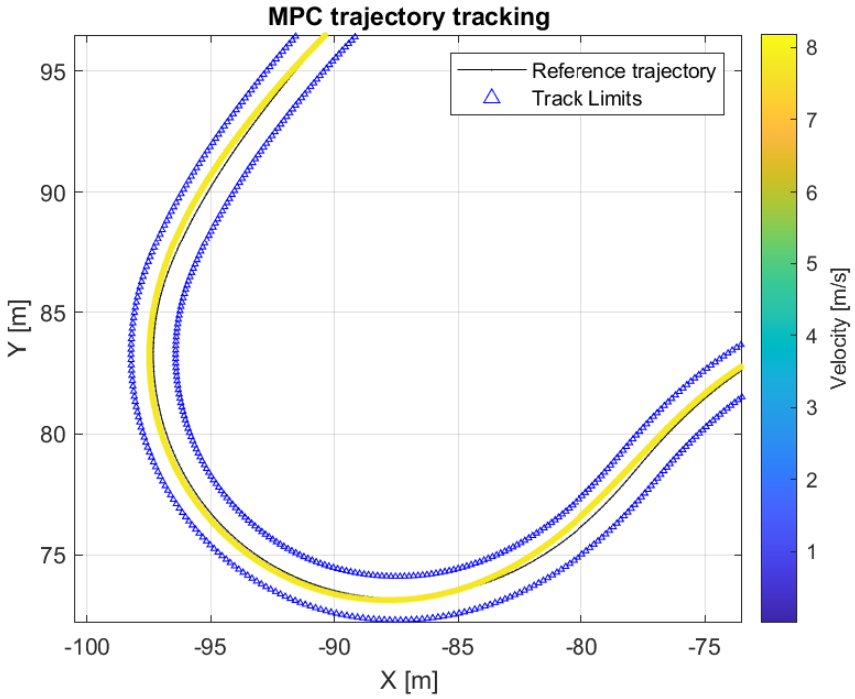

Figure 4.10: Close-up of simulated trajectory with MPC with the velocity profile for Stuttgart practice track

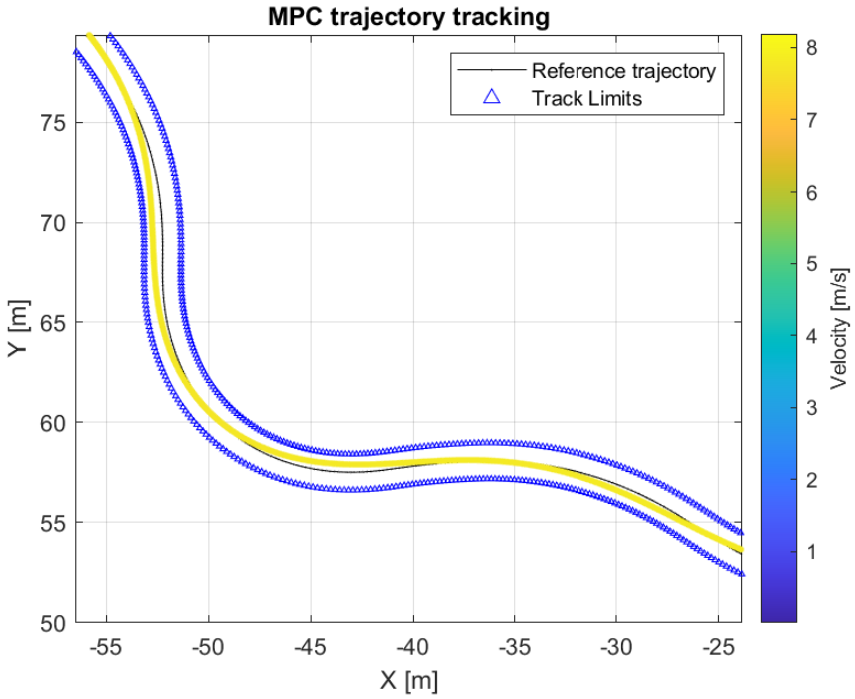

Figure 4.11: Close-up of simulated trajectory with MPC with the velocity profile for Stuttgart practice track

In Figure 4.12 the steering angle over time during the lap in the Stuttgart practice track is displayed.

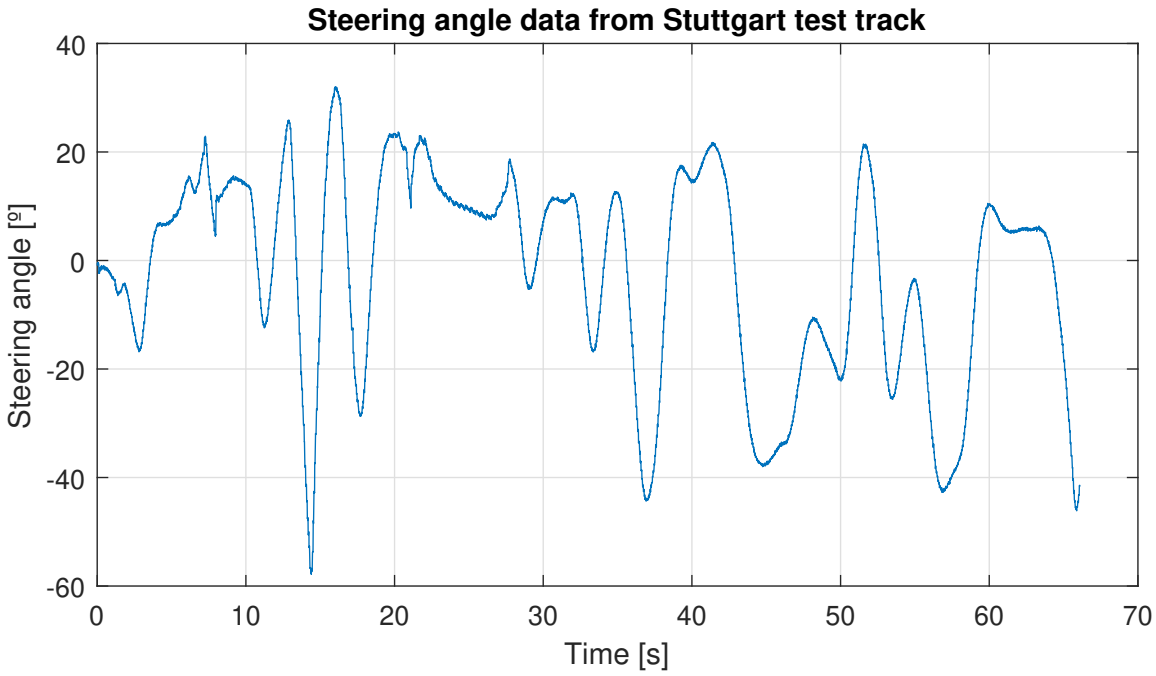

Figure 4.12: Steering wheel angle in the MPC Stuttgart practice track simulation

4.4.1 Runtime

Due to the complexity involved in the nonlinear optimization and in the nonlinear model used, the computation time is longer than the sampling time of 0.055 s by a very significant amount, thus it is not possible to implement this controller in the real prototype with the current technology. For one lap on the FS East endurance track, a simulated lap that should last approximately 2 minutes took 1h34min to actually simulate. Even reducing the prediction and control horizons and increasing the sampling time proved to not be enough. The computer used to run this simulation has an Intel® Core™ i7 5700HQ processor, which has similar capabilities to the hardware currently used in the car for its control algorithms. In order to be able to run the controller in real-time, different aproaches using ANN will be explored in Chapter 5.

Chapter 5

Learning the MPC with an ANN

Due to the long computation times of the controller, a different implementation approach is tested. In Section 3.6 it was seen how a neural network could learn the dynamics of the vehicle (v_x, v_y and $\dot{\psi}$) that result from inputs of steering angle (δ) and reference velocity (v_{ref}). If instead, the neural network learns what the steering angle and the reference velocity should be for a certain progression of reference trajectory coordinates over time ($X_{ref}(k), Y_{ref}(k)$), then it may be possible to control the vehicle using such network without the need for online optimization of the control actions. Two ANN model structures are going to be analysed in this chapter: NARX and NFIR models. Unlike a NARX model, which requires output feedback, a NFIR model does not. This proposed controller is intended to work on any FS track, so while all the information is available on the track used for training, the outputs are not known for other tracks and there is no way to guarantee that those outputs of the ANN will be sufficiently accurate. If the neural network receives wrong outputs through feedback, the errors will propagate. This is a more exigent situation than when the ANN is used as a prediction model in the MPC, since in that case the model predictions are feedback only for a limited H_p horizon.

5.1 Data Preprocessing

Since the trajectory is previously known but the velocity is not, there is no information about the sampling time of each input for training the ANN. Instead of spacing the data with equal sample times, the data was spaced with equal sample distances, which means that between each sample, the car covers 2 cm.

Using third order polynomials just like in Section 4.2.1, it is possible to parameterize the track by its arc length, which is equivalent to the distance traveled along the track. The track was parameterized with small equal arc length distances, and for each data sample, the steering angle and the reference velocity were interpolated from simulation data from the car dynamical model from Chapter 2.

The data used for learning was standardized, rescaling the distribution of values so that the mean of

observed values is 0 and the standard deviation is 1. This is the same method as described in Section 3.5.2.

It is crucial to have the most accurate possible neural network for this controller to work. This ANN outputs the control actions just from knowing the trajectory. This may not be enough information to determine the control actions as accurately as possible, because the MPC optimizes predicted future states of the system while the ANN has to learn the control actions just from the reference trajectory.

To try to improve the learning results, additional data preprocessing is done. The coordinates of the reference trajectory X_{ref}, Y_{ref} are inputs of the ANN. These coordinates can be converted to coordinate variations $(\Delta X_{car}, \Delta Y_{car})$ on the vehicle reference axis, which should prevent the ANN from learning only specific track coordinates. For this, a similar formulation to the trajectory error defined in Section 4.2.2 was used, as shown in Figure 5.1 with two consecutive points of a given trajectory are shown. The variable ΔY_{car} can be seen as the normal reference trajectory variation, relative to a line tangent to the reference point at instant k while ΔX_{car} can be seen as the parallel reference trajectory variation along the tangent to the reference point at instant k, as demonstrated by Equations (5.2) and (5.3), being Φ the angle of the tangent relative to the x-axis:

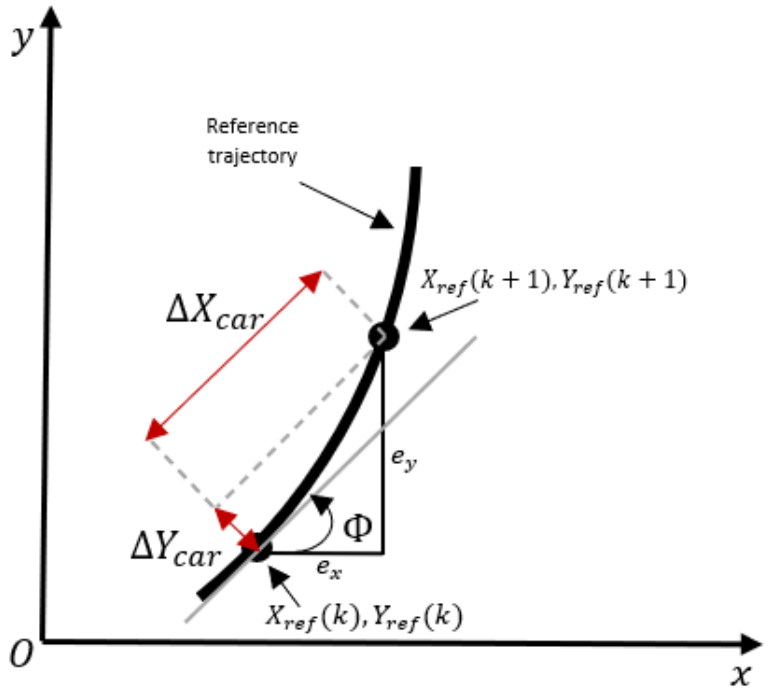

Figure 5.1: Coordinate variations ΔX_{car} and ΔY_{car} relative to vehicle reference frame at instant k

$$\Phi(k) \triangleq arctan\Big(\frac{\partial Y_{ref}(k)(t)}{\partial X_{ref}(k)(t)}\Big) \tag{5.1}$$

$$\Delta X_{car} \triangleq -cos(\Phi(k)) * \Big(X_{ref}(k) - X_{ref}(k+1)\Big) - sin(\Phi(k)) * \Big(Y_{ref}(k) - Y_{ref}(k+1)\Big) \tag{5.2}$$

$$\Delta Y_{car} \triangleq sin(\Phi(k)) * \Big(X_{ref}(k) - X_{ref}(k+1)\Big) - cos(\Phi(k)) * \Big(Y_{ref}(k) - Y_{ref}(k+1)\Big) \tag{5.3}$$

Since the MPC has information about future states when computing the control actions, the same type of information may be useful for the ANN to better learn the control actions. Therefore, additional inputs are given to the ANN when training, as will be shown in Figures 5.4 and 5.5. These inputs correspond to future reference position variations, so that at instant k the ANN receives information about the current and future position variations, which is more similar to how an MPC controller works. This is possible because the complete reference trajectory is known *a priori*. Different values for the number of future reference input signals, n, were experimented but the best results were achieved with $n = 11$. This value is similar to the prediction horizon of the MPC, $H_p = 15$.

5.2 ANN Structure and Training

The training algorithm used was the Bayesian regularization, which works by assuming the weights and biases of the network to be random variables with specified distributions, which are related to the regularization parameters, estimated using statistical techniques. Bayesian regularization usually takes longer but deals better with challenging problems [30]. Only the training dataset is displayed because the Bayesian regularization backpropagation does not use data for validation of the generalization (or regularization) because it has its own form of validation built into the algorithm. No data was preprocessed to create a test dataset, but testing can be done later if this control concept works.

Two main types of neural network architecture are tested: ANN controller 1 - receives as inputs only the coordinates variations of the reference trajectory ($\Delta X_{car}, \Delta Y_{car}$), and ANN controller 2 - which in addition to $\Delta X_{car}, \Delta Y_{car}$ also receives as inputs the current states of the vehicle (X,Y and ψ). Adding the states as inputs makes the ANN controller more similar to an MPC controller, which receives the current states at each time step. For the ANN controller 1, without the vehicle states, the training was done with data from the real prototype on FS East endurance and FSG endurance, while for the ANN controller 2, with the vehicle states, the data used was from simulation with the MPC controller on FS East endurance and FSG endurance as well.

The different types of controller structures tested are schematically shown in Figures 5.2 to 5.5. For simplicity, all structures are represented with only one input and feedback delays. The dashed lines represent the ANN output feedback. In the case of a NFIR model structure these lines are not present.

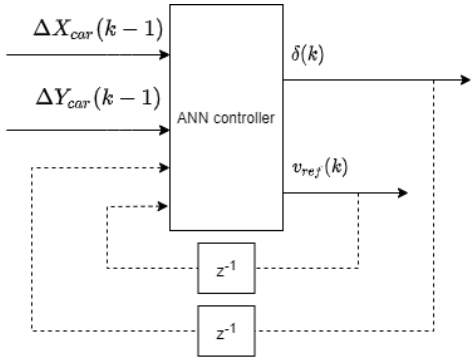

Figure 5.2: ANN controller 1 without future reference input signals

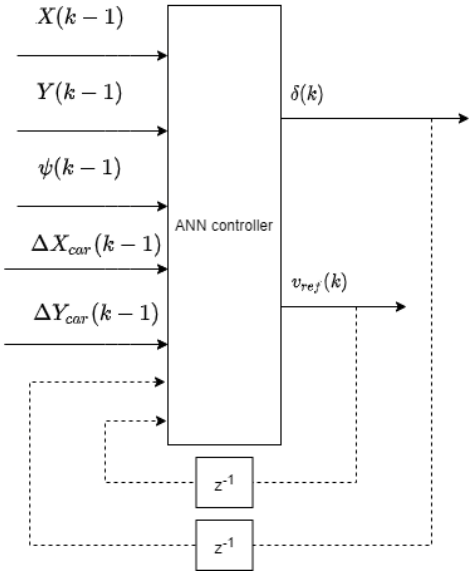

Figure 5.3: ANN controller 2 without future reference input signals

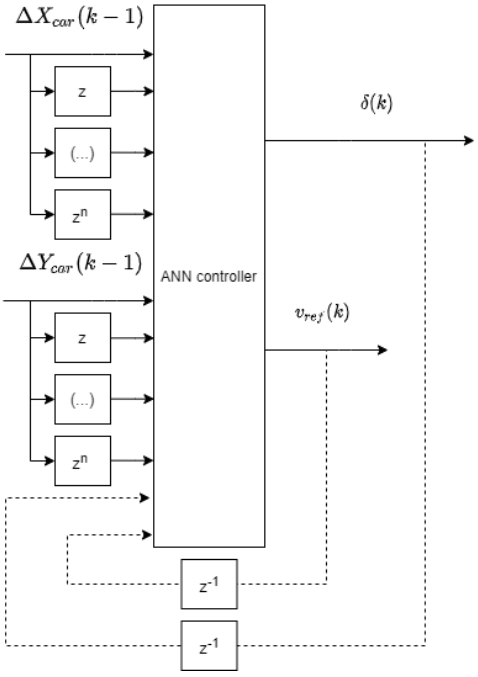

Figure 5.4: ANN controller 1 with future reference input signals

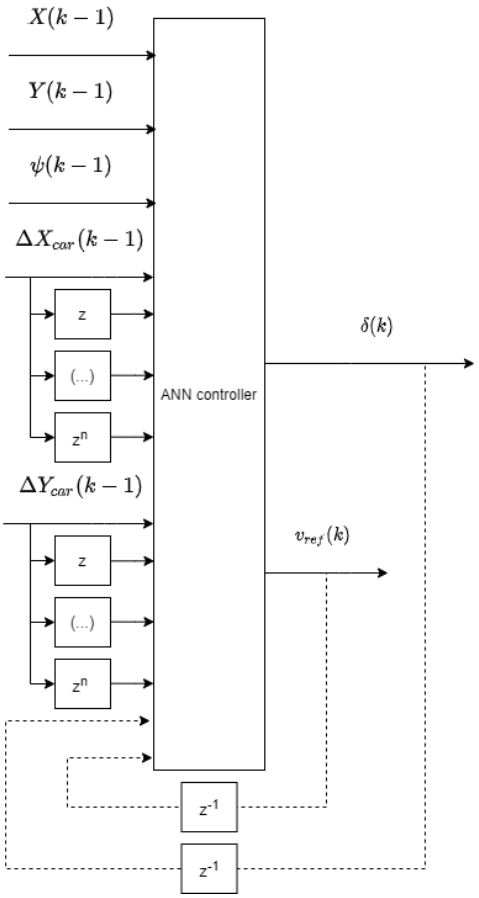

Figure 5.5: ANN controller 2 with future reference input signals

To determine the best architecture for each of the two ANN controllers, different ANN are trained and compared, according to Tables 5.1 and 5.2. The comparison analyses the influence of the following characteristics in the performance of the neural network:

- Model used - either NFIR or NARX model. Number of inputs - total number of inputs, considering all delays.
- Number of delays - input delays in the case of NFIR models and both input and feedback delays in the case of NARX models.
- Number of future reference input signals for the reference trajectory given in each input sample.
- MSE - corresponds to the mean squared error of outputs and targets for the training data.

ANN Controller 1 Performance Comparison				
Model	# Inputs	# Delays	# Future References	MSE
NFIR	2	1	0	2.69
NFIR	4	2	0	1.47
NFIR	24	1	11	1.65
NFIR	**26**	**2**	**11**	**1.02**
NARX	4	1	0	6.93
NARX	8	2	0	10.00
NARX	26	1	11	3.81
NARX	30	2	11	2.42

Table 5.1: Performance comparison between different ANN controller 1 structure parameters

ANN Controller 2 Performance Comparison				
Model	# Inputs	# Delays	# Future References	MSE
NFIR	5	1	0	0.38
NFIR	10	2	0	1.19
NFIR	**27**	**1**	**11**	**0.25**
NFIR	32	2	11	0.45
NARX	7	1	0	8.88
NARX	14	2	0	3.15
NARX	29	1	11	8.29
NARX	36	2	11	2.41

Table 5.2: Performance comparison between different ANN controller 2 structure parameters

The best performance for ANN controller 1 is achieved with a NFIR model with 2 input delays and 11 future reference input signals, while the best performance for ANN controller 2 is achieved with a NFIR model with 1 input delay and 11 future reference input signals.

Several numbers of hidden layers and neurons were experimented and the best results were achieved with 2 hidden layers, with 25 neurons each, as exemplified in Figure 5.6 for the ANN controller 1, where the input vector has 24 elements and 11 future reference input signals.

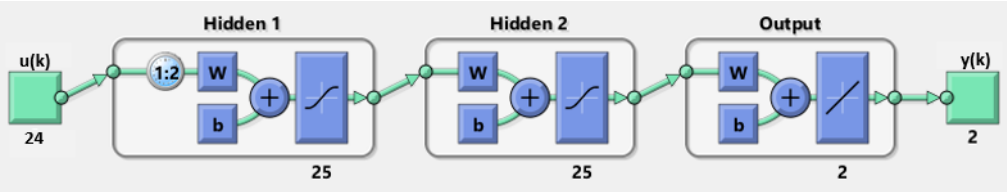

Figure 5.6: NFIR ANN controller 1 structure

5.2.1 ANN Controller 1 Training Results

The learning results for the ANN controller 1 with the best performance can be seen in Figures 5.7 and 5.8, where the targets and the outputs of the ANN are similar throughout the data. The target data corresponds to the recorded data with the real prototype in FS East endurance and FSG endurance. The first part of the data is from FS East and from sampling instant 4585 onwards the data is from FSG.

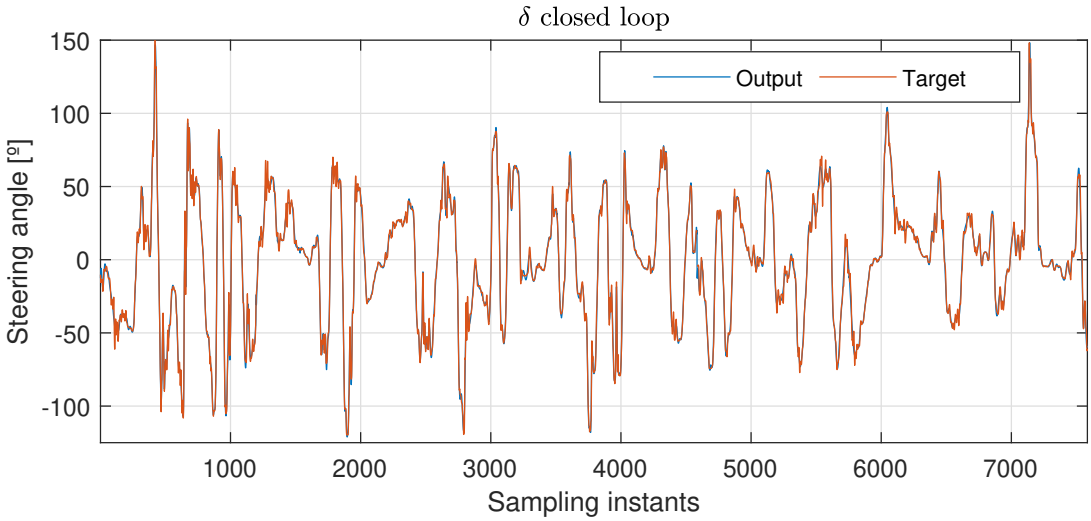

Figure 5.7: Comparison between the target δ and the output of the ANN controller 1 with best performance

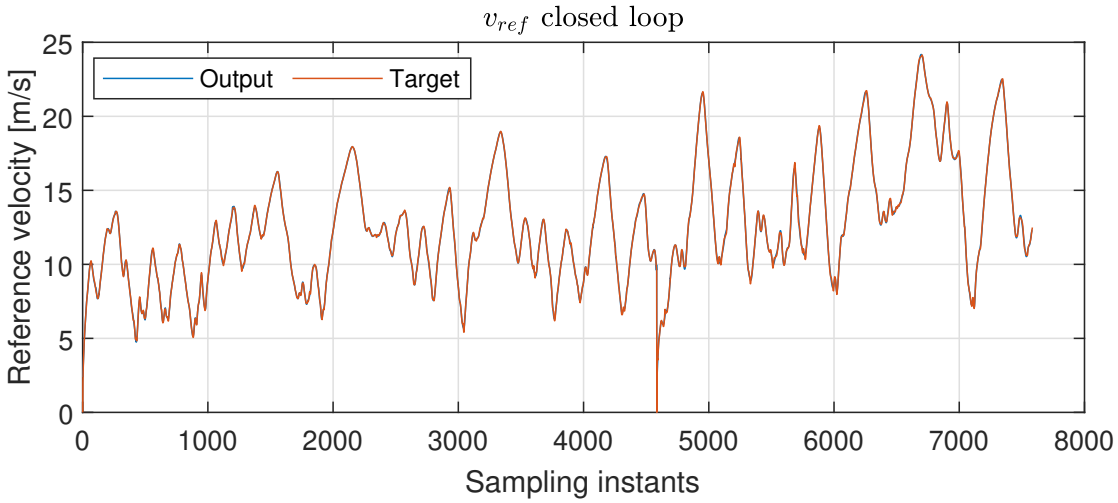

Figure 5.8: Comparison between the target v_{ref} and the output of the ANN controller 1 with best performance

The MSE of the presented ANN with respect to the number of iterations can be seen in Figure 5.9. The linear regression of the outputs and the targets in Figure 5.10 shows an almost linear relationship. Both the MSE and the R value indicate a neural network with good learning results, at least on the training dataset.

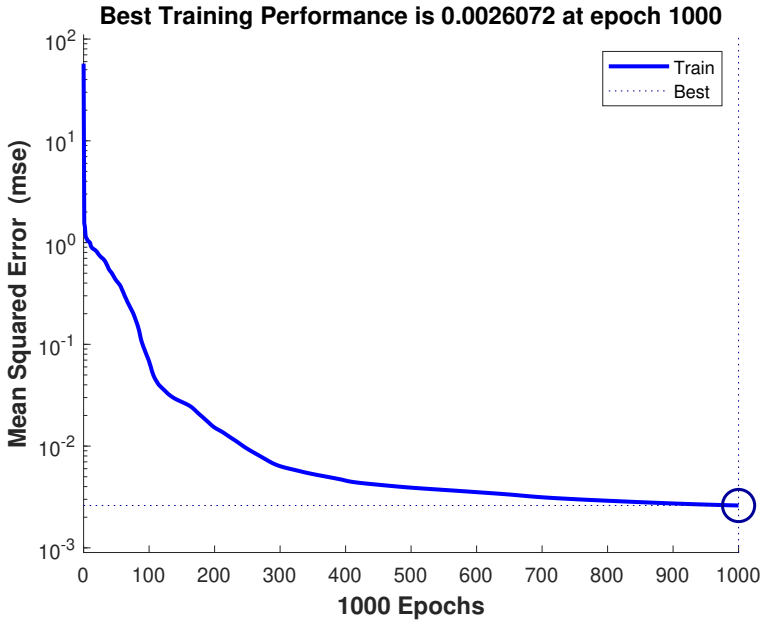

Figure 5.9: Best performance of the ANN controller 1 on the train dataset

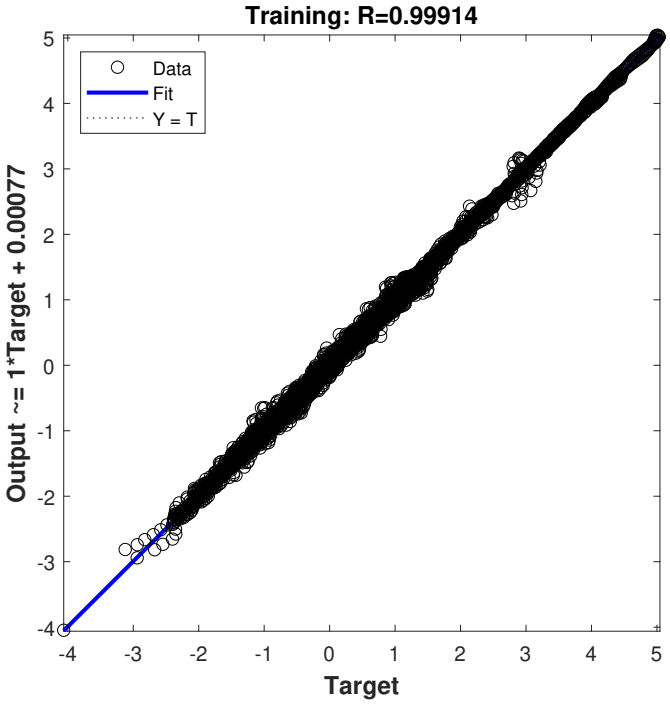

Figure 5.10: Regressions for the training data of ANN controller 1 with best performance

5.2.2 ANN Controller 2 Training Results

The learning results for the ANN controller 2 with the best performance can be seen in Figures 5.11 and 5.12, where the blue line representing the output is overlaid with the orange line representing the targets. The target data is from FS East and FSG endurance tracks simulation with the MPC controller. The first part of the data is from FS East, and from sampling instant 5725 onwards the data is from FSG.

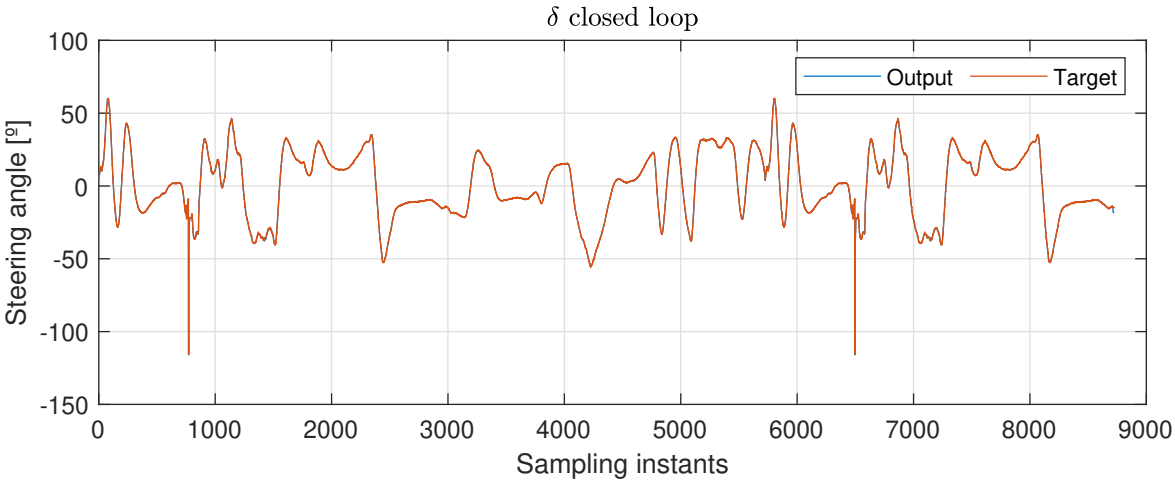

Figure 5.11: Comparison between the target δ and the output of the ANN controller 2 with best performance

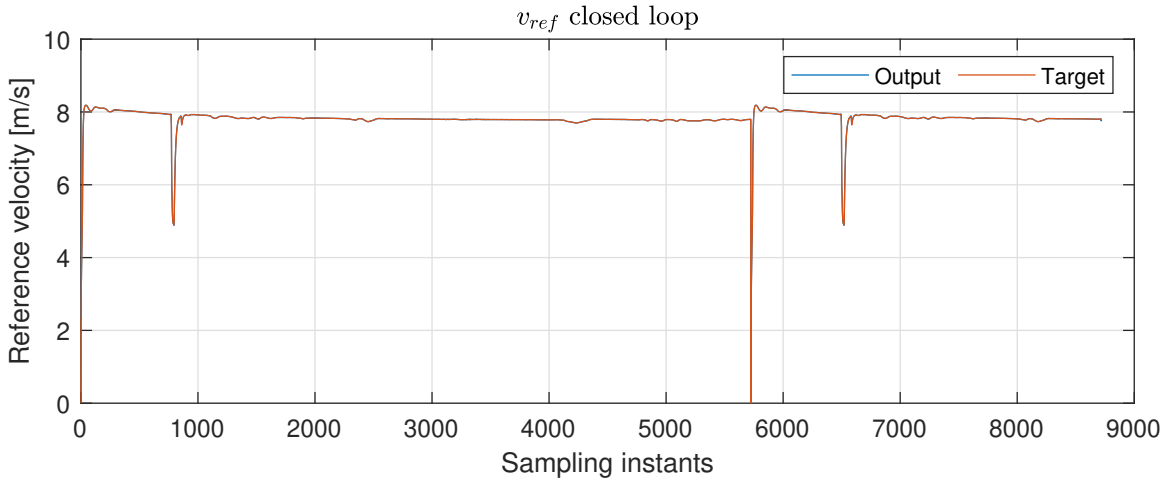

Figure 5.12: Comparison between the target v_{ref} and the output of the ANN controller 2 with best performance

The MSE of the presented ANN with respect to the number of iterations can be seen in Figure 5.13, and it is slightly lower than the MSE achieved with ANN controller 1. The linear regression of the outputs and the targets in Figure 5.14 shows an almost linear relationship. Both the MSE and the R value indicate a neural network with good learning results, at least on the training dataset. It can be noted that most values are on the top right corner, which is due to the fact that the MPC controller simulation had the reference velocity limited, as evidenced in Figure 5.12. It can be concluded that both ANN controller 1 and ANN controller 2 achieved good learning results, with ANN controller 2 achieving a lower MSE.

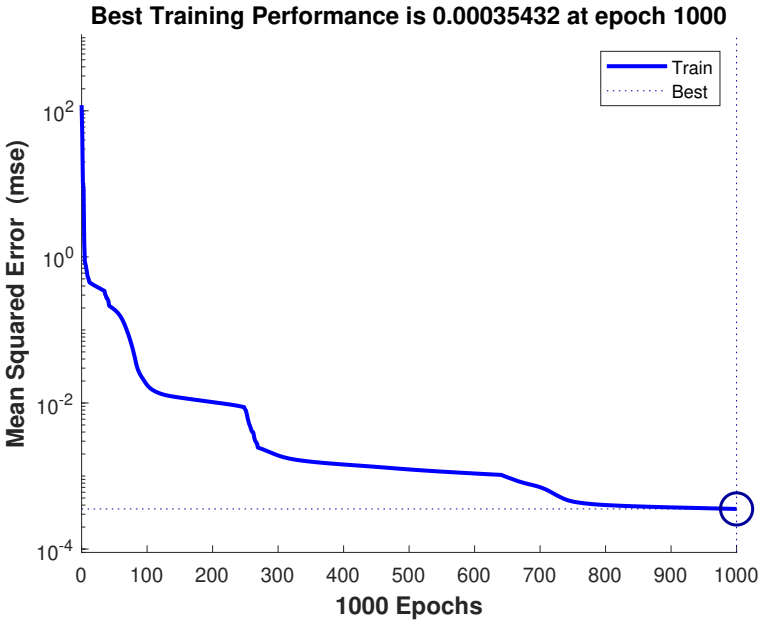

Figure 5.13: Best performance of the ANN controller 2 on the train dataset

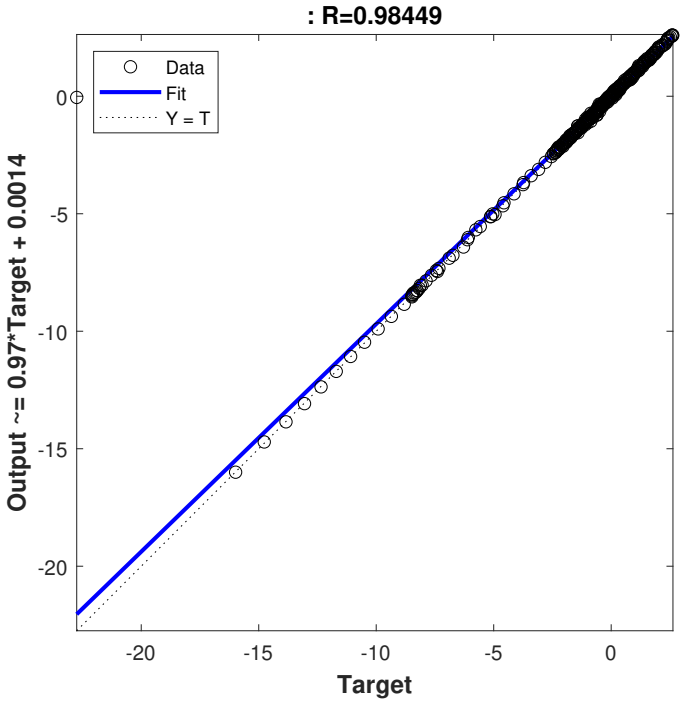

Figure 5.14: Regressions for the training data of ANN controller 2 with best performance

5.3 Simulation Results

Both ANN were simulated replacing the MPC controller in the closed-loop configuration. The trajectory obtained for FS East endurance track with ANN controller 1 is shown in Figure 5.15. During this simulation, the reference trajectory is compared to the simulated one, and the control action corresponding to the closest reference point is given. This controller tracks the reference trajectory, represented with a black line, for some corners until a certain time instant. Even though the MSE between the outputs and the targets of the ANN is relatively small, as seen in Figure 5.9, the accumulation of errors ends up being significant. This brings out the major issue with this type of controller, which is its inability to feedback errors, unlike MPC or even PID controllers.

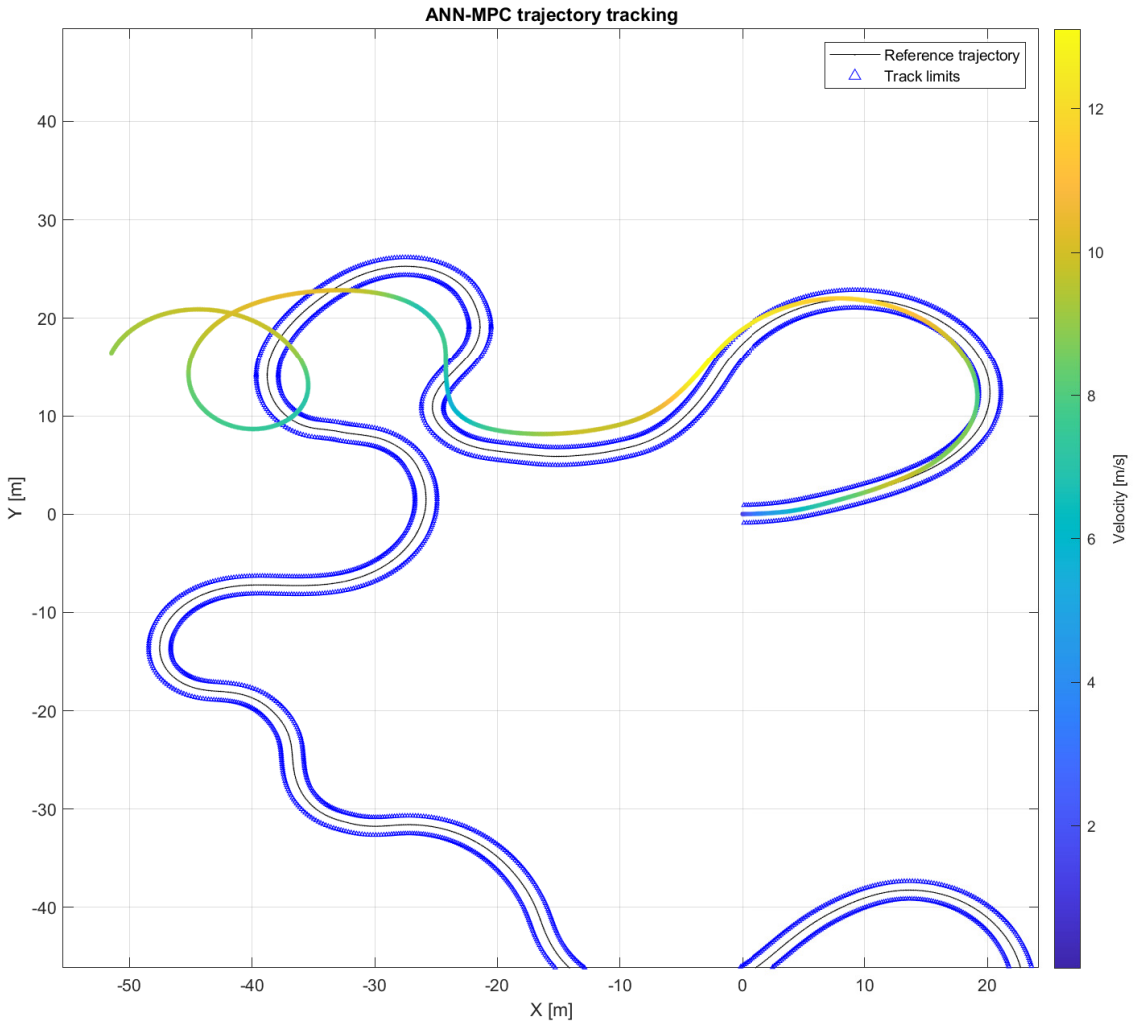

Figure 5.15: Simulated trajectory with ANN controller 1 on FS East track

The trajectory obtained with the ANN controller 2 is shown in Figure 5.16. This controller starts the corner well, but quickly goes offtrack. This may be due to the fact that the data given for the ANN to learn contained states (X, Y and ψ) of a vehicle that followed the reference trajectory closely, as shown in Figure 4.4. When the vehicle follows the trajectory almost perfectly, there is not enough dynamical information for the ANN to learn well.

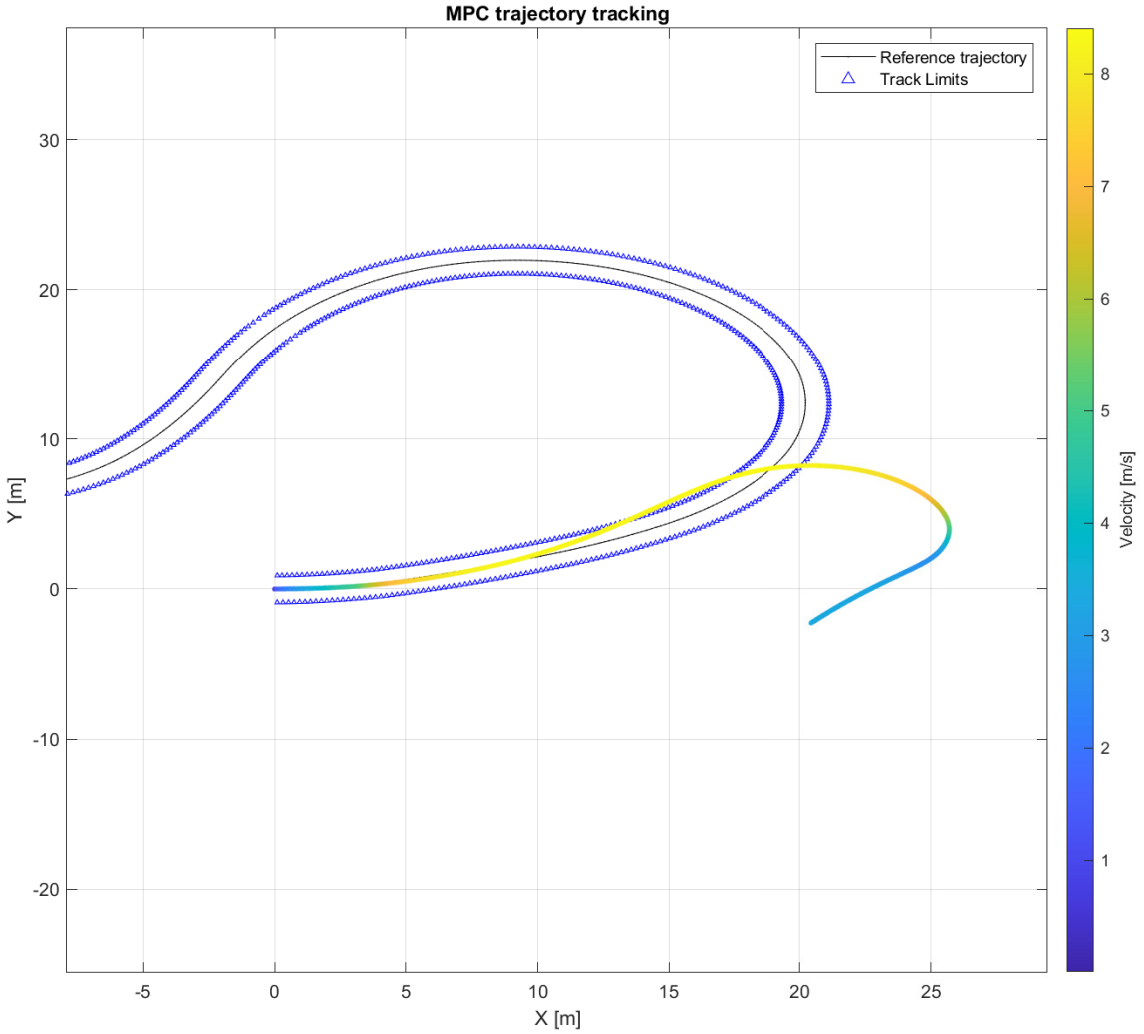

Figure 5.16: Simulated trajectory with ANN controller 2 on FS East track

Chapter 6

Conclusions

6.1 Summary and Conclusions

Using the principles of vehicle dynamics, a car model was obtained and validated by comparing its results to real data. The full model was divided into subsystems comprising an empirical tyre model to calculate the forces on the wheels, steering kinematics that relate the steering wheel angle and the angle of each wheel, motors and aerodynamics which all are necessary for the planar model to describe the vehicle position and orientation.

Data from testing and from competitions was selected and preprocessed in order to train a neural network that is a good approximation to the vehicle model. This was achieved with only a single lap of training and good generalization was also obtained.

In order to design an MPC, the reference trajectory was parameterized with respect to its arc length and the contour error was defined. The cost function is such that the contouring error is minimized while progress along the track is maximized. It is also desired to have small control action variations and a small steering angle. It was proven that it is possible to use a prediction model based on an artificial neural network to control the vehicle in a defined trajectory. Despite the advantages ANN provide of not requiring knowledge about the system due to black box modelling, the nonlinear optimization still takes a significant amount of computation time, which is a barrier for real-time implementation of this control technique in the real prototype.

To attempt to solve the long computation times, several different neural network models were tested to learn the model predictive controller dynamics. In spite of the ANN models providing a good approximation of the simulated output controller signals in the learning phase, they were not able to sustain the same performance in a real-time implementation due to the feedback error accumulation.

6.2 Future Work

The results obtained with the ANN controller seen in Chapter 5 may be improved with an external control loop made up of a PID controller to track the contouring error [37].

An alternative path to facilitate the learning process is to obtain the reference velocity from a point-mass simulator, which FST Lisboa has already developed for their prototypes. A point-mass simulator assumes that the entire vehicle can be approximated by a single point particle, where all properties are concentrated on that point, including mass. This simulator takes little time to run and is able to provide the approximate maximum velocity of the vehicle on each part of the track. The advantage of using this would be that there would be one less variable for the controller to learn, since the reference velocity could be given by the point-mass simulator, remaining only the steering angle to be controlled.

A linear model of the car may replace the ANN prediction model in the MPC and be able to run in real time. Developing a linear model of the car is not trivial because the car dynamics are highly non-linear, so care has to be taken to ensure that the approximations made allow for good predictions. The vehicle cannot be linearized around a single operating point because its velocity is constantly changing throughout a lap, if maximum performance is to be achieved. To cope with the varying velocity, adaptive MPC can be used because the prediction model is linearized for each operating point, and then that linearized model is used for the entire prediction horizon, after which a new linearization is performed in the following time step [38].

Explicit MPC may also be an option, where all possible optimizations are computed beforehand and then in real time the controller only has to find the values of the control actions in look-up tables [39, 40]. The complexity of the model and the number of states may lead to an amount of values on said tables so large that searching for values takes to long to run in real time.

Deep learning algorithms may be used to reduce the computation time by approximating the MPC laws more efficiently, as demonstrated in [41]. Another interesting development is the use of relatively simple and adaptive vehicle model which is improved based on online measurements and other machine learning tools used in [42]. Similarly, Linear Parameter Varying (LPV) theory may be used to model the vehicle dynamics and implement an MPC controller with lower computation costs [43].

After good results and reasonable computation times are obtained in simulation, one of the following steps is to implement the control algorithm in a real Formula Student prototype.

Bibliography

[1] C. F. Kerry and J. Karsten. Gauging investment in self-driving cars, 2017. URL https://www.brookings.edu/research/gauging-investment-in-self-driving-cars/. [Online; accessed on August 2020].

[2] E. Ackerman. Self-Driving Cars Were Just Around the Corner - in 1960, 2016. URL https://spectrum.ieee.org/tech-history/heroic-failures/selfdriving-cars-were-just-around-the-cornerin-1960. [Online; accessed on September 2020].

[3] K. Walch. The Future With Level 5 Autonomous Cars, 2019. [Online; accessed on September 2020].

[4] J. Kabzan, M. I. Valls, V. J. Reijgwart, H. F. Hendrikx, C. Ehmke, M. Prajapat, A. Bühler, N. Gosala, M. Gupta, R. Sivanesan, A. Dhall, E. Chisari, N. Karnchanachari, S. Brits, M. Dangel, I. Sa, R. Dubé, A. Gawel, M. Pfeiffer, A. Liniger, J. Lygeros, and R. Siegwart. AMZ Driverless: The Full Autonomous Racing System. May 2019.

[5] R. Shreyas, A. Bradley, and G. Collier. MPC Controller for Autonomous Formula Student Vehicle. In *SAE Technical Paper*. SAE International, March 2020. doi: 10.4271/2020-01-0089.

[6] J. Richalet, A. Rault, J. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14:413–428, September 1978. doi: 10.1016/0005-1098(78)90001-8.

[7] M. Arnold and G. Andersson. Model Predictive Control of energy storage including uncertain forecasts. In *Proceedings of the 17th Power Systems Computation Conference (PSCC 2011)*, Stockholm, Sweden, August 2011.

[8] T. Geyer. *Model predictive control of high power converters and industrial drives*. Wiley, 2016. ISBN:978-1-119-01090-6.

[9] E. Camacho and C. Bordons. *Model Predictive Control*. Springer, 1999.

[10] M. Lazar and O. Pastravanu. A neural predictive controller for non-linear systems. *Mathematics and Computers in Simulation*, 60:315–324, 09 2002. doi: 10.1016/S0378-4754(02)00023-X.

[11] J. Qin and T. Badgwell. A Survey of Industrial Model Predictive Control Technology. *Control engineering practice*, 11:733–764, 07 2003. doi: 10.1016/S0967-0661(02)00186-7.

[12] D. Schramm, M. Hiller, and R. Bardin. *Vehicle Dynamics: Modelling and Simulation*. Springer, 2^{nd} edition, 2018. ISBN 978-3-662-54483-9.

[13] L. M. M. A. V. Abreu. *Mechanical design of the wheel assembly of an electric Formula Student prototype*. Mechanical Engineering Master's Thesis, Instituto Superior Técnico, 2019.

[14] A. M. P. Antunes. *Sideslip Estimation of Formula Student Prototype*. Mechanical Engineering Master's Thesis, Instituto Superior Técnico, 2017.

[15] R. N. Jazar. *Vehicle Dynamics: Theory and Application*. Springer, 2008. ISBN: 978-0-387-74243-4.

[16] *AMK RACING KIT 4 wheel drive 'Formula Student Electric'*. AMK Arnold Muller GmbH and Co. KG, 2015.

[17] W. Milliken and D. Milliken. *Race Car Vehicle Dynamics*. Society of Automotive Engineers Warrendale, 1995. ISBN:1-56091-529-9.

[18] M. Blundell and D. Harty. *The Multibody Systems Approach to Vehicle Dynamics*. Butterworth-Heinemann, 2004. ISBN:0-7506-5112-1.

[19] E. M. Almas. Notes from "Machine Elements" course. 2007. Aerospace and Mechanical Engineering, Instituto Superior Técnico.

[20] Silberwolf. Topview of a cardan joint according to DIN 808 (August 1984), type E (single), July 2007. URL `https://commons.wikimedia.org/wiki/File:Cardan-joint_DIN808_type-E_topview.png`. [Online; accessed on September 2020].

[21] C. Smith. *Tune to Win: The art and science of race car development and tuning*. Aero Publishers, 1^{st} edition, 1978. ISBN 978-0879380717.

[22] C. Wit and P. Tsiotras. Dynamic tire friction models for vehicle traction control. In *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No.99CH36304)*, volume 4, pages 3746—3751, 1999.

[23] H. Pacejka. *Tire and Vehicle Dynamics*. Butterworth-Heinemann, 2^{nd} edition, 2006. ISBN:980-0-750-66918-4.

[24] W. Hirschberg, G. Rill, and H. Weinfurter. Tire model tmeasy. *Vehicle System Dynamics*, (45:sup1): 101—-119, 2007. doi: 10.1080/00423110701776284.

[25] U. Kiencke and L. Nielsen. *Automotive Control Systems For Engine, Driveline, and Vehicle*. Springer, 2^{nd} edition, 2005. ISBN 3-540-23139-0.

[26] E. Kasprzak and D. Gentz. The Formula SAE Tire Test Consortium-Tire Testing and Data Handling. pages 776–4841, 12 2006. doi: 10.4271/2006-01-3606.

[27] *FSG Competition Handbook 2020*. Formula Student Germany, 2020. URL `https://www.formulastudent.de/fileadmin/user_upload/all/2020/rules/FSG20_Competition_Handbook_v1.0.pdf`. [Online; accessed on September 2020].

[28] J. S. R. Jang., C. T. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice-Hall, Inc., 1^{st} edition, 1997.

[29] S. Haykin. *Neural Networks - A Comprehensive Foundation*. Pearson Education, 9^{th} edition, 1999.

[30] H. Demuth, M. Beale, and M. Hagan. *Neural Network Toolbox^{TM} User's Guide*. The MathWorks, Inc., 2017.

[31] V. Ranković, J. Radulović, N. Grujović, and D. Divac. Neural Network Model Predictive Control of Nonlinear Systems Using Genetic Algorithms. *International Journal of Computers, Communications and Control*, 7:540–549, September 2012.

[32] *Formula Student Rules 2020*. Formula Student Germany, 2020. URL `https://www.formulastudent.de/fileadmin/user_upload/all/2020/rules/FS-Rules_2020_V1.0.pdf`. [Online; accessed on September 2020].

[33] S. Jiang, H. Chen, and X. He. PID controller design for MIMO high-order systems based on internal model control. *Hangkong Xuebao/Acta Aeronautica et Astronautica Sinica*, 30:236–241, February 2009.

[34] M. A. Botto. Notes from "Optimal Control" course. 2007. Mechanical Engineering, Instituto Superior Técnico.

[35] M. Behrendt. A basic working principle of Model Predictive Control, October 2009. URL `https://commons.wikimedia.org/wiki/File:MPC_scheme_basic.svg`. [Online; accessed on August 2020].

[36] A. Liniger, A. Domahidi, and M. Morari. Optimization-based autonomous racing of 1:43 scale RC cars. *Optimal Control Applications and Methods*, April 2014. doi: 10.1002/oca.2123.

[37] S. Ito, T. Sato, N. Araki, and Y. Konishi. Two-loop Design for Dual-rate Cascade System. *Preprints of the 3rd IFAC Conference on Advances in ProportionalIntegral-Derivative Control*, May 2018.

[38] M. Bujarbaruah, X. Zhang, E. Tseng, and F. Borrelli. Adaptive MPC for Autonomous Lane Keeping. In *14th International Symposium on Advanced Vehicle Control (AVEC)*, July 2018.

[39] C. F. Lee, C. Manzie, and C. Line. Explicit Nonlinear MPC of an Automotive Electromechanical Brake. In *Proceedings of the 17th World Congress*. The International Federation of Automatic Control, July 2008.

[40] S. Chen, K. Saulnier, N. Atanasov, D. Lee, V. Kumar, G. Pappas, and M. Morari. Approximating Explicit Model Predictive Control Using Constrained Neural Networks. pages 1520–1527, June 2018. doi: 10.23919/ACC.2018.8431275.

[41] B. Karg and S. Lucia. Efficient Representation and Approximation of Model Predictive Control Laws Via Deep Learning, June 2018.

[42] J. Kabzan, L. Hewing, A. Liniger, and M. Zeilinger. Learning-Based Model Predictive Control for Autonomous Racing. *IEEE Robotics and Automation Letters*, PP:1–1, July 2019. doi: 10.1109/LRA.2019.2926677.

[43] E. Alcalá, V. Puig, J. Quevedo, and U. Rosolia. Autonomous racing using Linear Parameter Varying-Model Predictive Control (LPV-MPC). *Control Engineering Practice*, 95:104270, 2020. ISSN 0967-0661. doi: https://doi.org/10.1016/j.conengprac.2019.104270. URL `http://www.sciencedirect.com/science/article/pii/S0967066119302187`.

Appendix A

Expanded vertical model equations

The vertical model presented in Section 2.3 is defined by a state space system $\dot{x} = Ax + Bu$, where x is a vector with the state variables and u is the vector with input variables.

$$x = [Z, \dot{Z}, \theta.\dot{\theta}, \phi, \dot{\phi}, h_1, \dot{h}_1, h_2, \dot{h}_2, h_3, \dot{h}_3, h_4, \dot{h}_4]^T$$

$$u = [F_z, M_\theta, M_\phi, G_1, G_2, G_3, G_4]$$

- Expansion of Equation 2.7, representing the vertical movement Z of the sprung mass:

$$\begin{aligned} m_{ch}\ddot{Z} = {} & F_z - [K_1 + K_2 + K_3 + K_4]Z - [C_1 + C_2 + C_3 + C_4] + [(K_2 + K_4)d - (K_1 + K_3)c]\phi \\ & + [(C_2 + C_4)d - (C_1 + C_3)c]\dot{\phi} + [(K_1 + K_2)a - (K_3 + k_4)b]\theta + [(C_1 + C_2)a - (C_3 + C_4)b]\dot{\theta} \\ & + K_1 h_1 + C_1\dot{h}_1 + K_2 h_2 + C_2\dot{h}_2 + K_3 h_3 + C_3\dot{h}_3 + K_4 h_4 + C_4\dot{h}_4 \end{aligned} \tag{A.1}$$

- Expansion of Equation 2.8, representing the roll rotation ϕ:

$$\begin{aligned} I_\phi\ddot{\phi} = {} & M_\phi - [(K_2 + K_4)d - (K_1 + K_3)c]Z - [(C_2 + C_4)d - (C_1 + C_3)c] \\ & + [(K_1 + K_3)c^2 - (K_2 + K_4)d^2]\phi - [(C_1 + C_3)c^2 - (C_2 + C_4)d^2]\dot{\phi} \\ & + [(K_1 a - K_3 b)c - (K_4 b + k_2 d)b]\theta + [(C_1 a + C_3 b)c - (C_4 b + C_2 a)d]\dot{\theta} \\ & + K_1 h_1 c + C_1\dot{h}_1 c + K_2 h_2 d + C_2\dot{h}_2 d + K_3 h_3 c + C_3\dot{h}_3 c + K_4 h_4 d + C_4\dot{h}_4 d \end{aligned} \tag{A.2}$$

- Expansion of Equation 2.9, representing the pitch rotation θ:

$$\begin{aligned} I_\theta\ddot{\theta} = {} & M_\theta - [(K_1 + K_2)a - (K_3 + K_4)b]Z + [(C1 + C_2)a - (C_3 + C_4)b]\dot{Z} \\ & + [(K_1 + K_2)a^2 + (K_3 + K_4)b^2]\theta - [(C_1 + C_2)a^2 + (C_3 + C_4)b^2]\dot{\theta} \\ & + [(K_1 c - K_2 d)a - (K_4 d + k_3 c)b]\phi + [(C_1 c - C_2 d)a + (C_4 d + C_3 c)b]\dot{\phi} \\ & - K_1 h_1 a - C_1\dot{h}_1 a - K_2 h_2 a - C_2\dot{h}_2 a + K_3 h_3 b + C_3\dot{h}_3 b + K_4 h_4 b + C_4\dot{h}_4 b \end{aligned} \tag{A.3}$$

- Expansion of Equation 2.10, representing the vertical movement h_1:

$$\begin{aligned} m_1\ddot{h}_1 = {} & K_1 Z + C_1\dot{Z} + K_1\phi c + C_1\dot{\phi}c - K1\theta a - C_1\dot{\theta}a \\ & - [K_1 + K_{p1} + K_{arbF}]h_1 - [C - 1 + C_{p1}\dot{h}_1 + K_{arbF}h1 + K_{p1}G_1 \end{aligned} \tag{A.4}$$

- Expansion of Equation 2.11, representing the vertical movement h_2:

$$m_2\ddot{h}_2 = K_2Z + C_2\dot{Z} + K_2\phi d + C_2\dot{\phi}d - K2\theta a - C_2\dot{\theta}a \\ - [K_2 + K_{p2} + K_{arbF}]h_2 - [C - 2 + C_{p2}\dot{h}_2 + K_{arbF}h2 + K_{p2}G_2 \tag{A.5}$$

- Expansion of Equation 2.12, representing the vertical movement h_3:

$$m_3\ddot{h}_3 = K_3Z + C_3\dot{Z} + K_3\phi c + C_3\dot{\phi}c - K3\theta b - C_3\dot{\theta}b \\ - [K_3 + K_{p3} + K_{arbR}]h_3 - [C - 3 + C_{p3}\dot{h}_3 + K_{arbR}h3 + K_{p3}G_3 \tag{A.6}$$

- Expansion of Equation 2.13, representing the vertical movement h_4:

$$m_4\ddot{h}_4 = K_4Z + C_4\dot{Z} + K_4\phi d + C_4\dot{\phi}d - K4\theta b - C_4\dot{\theta}b \\ - [K_4 + K_{p4} + K_{arbR}]h_4 - [C - 4 + C_{p4}\dot{h}_4 + K_{arbR}h4 + K_{p4}G_4 \tag{A.7}$$