MARMARA UNIVERSITY

INSTITUTE FOR GRADUATE STUDIES IN PURE AND APPLIED SCIENCES

DEVELOPMENT OF A LEARNING BASED TRAJECTORY TRACKING CONTROLLER FOR AUTONOMOUS VEHICLES

HASAN ŞENER

MASTER THESIS

Department of Mechanical Engineering

Thesis Supervisor

Assist. Prof. Dr. Uğur TÜMERDEM

Thesis Co-Supervisor

Dr. İ. Hakkı SAVCI

ISTANBUL, 2022

MARMARA UNIVERSITY

INSTITUTE FOR GRADUATE STUDIES IN PURE AND APPLIED SCIENCES

DEVELOPMENT OF A LEARNING BASED TRAJECTORY TRACKING CONTROLLER FOR AUTONOMOUS VEHICLES

HASAN ŞENER

(524617038)

MASTER THESIS

Department of Mechanical Engineering

Thesis Supervisor

Assist. Prof. Dr. Uğur TÜMERDEM

Thesis Co-Supervisor

Dr. İ. Hakkı SAVCI

ISTANBUL, 2022

ACKNOWLEDGEMENT

Firstly, I would like thank my wife for her unbelievable support during this thesis.

I would also thank especially my supervisor, Dr. Uğur Tümerdem for his encouragement, continuous guidance and support.

I would like to thank my second supervisor İsmail Hakkı Savcı for his support during the time we've worked together in Ford Otomotiv San. A.Ş.

Hasan ŞENER

Istanbul, January 2022

TABLE OF CONTENTS

ÖZET

OTONOM ARAÇLAR İÇİN ÖĞRENME TABANLI YÖRÜNGE TAKİBİ KONTROLCÜSÜ GELİŞTİRİLMESİ

Araç teknolojilerindeki son gelişmeler ve ileri sürüş destek sistemlerinin hızla gelişmesiyle otonom araçlar gün yüzüne çıkmaktadır. Bir otonom araç sürücüsüz olarak bir noktadan başka bir noktaya gidebilme kabiliyetine sahip olan bir kara taşıtıdır. Bunu yaparken temel olarak; çevreyi algılama, karar verme, planlama ve planlanan yörünge ve hız takibi için kontrol yeteneklerine sahiptir.

Bu tezde bir otonom yarış aracının kontrol yeteneğinde temel olan yörünge ve hız takibi problemini çözmek amacıyla, aracın dinamiği "bisiklet dinamiği" kullanılarak modellenmiş, bu dinamik model "çift şerit değiştirme" testlerinde doğrulanmıştır. Modellenen sistem doğrusal zaman değişen model öngörülü kontrol yöntemlerinde kullanılmıştır.

Yörünge takibi için tasarlanan model öngörülü kontrolcüde ölçülen gürültü olarak belirlenen aracın sapma oranı, yapay sinir ağları kullanılarak modellenmiş olup böylelikle araç anlık olarak sapma oranını daima hesaplayabilir duruma gelmiştir. Sapma oranının belirlenmesi iki yöntem kullanılmış olup; ilkinde araç yarış pisti üzerinde defalarca sürülmüş, aracın anlık konumuna karşılık sahip olması gereken sapma oranı ileri beslemeli sinir ağları ile modellenmiştir. İkincil olarak ise, otonom yarış aracının yanal dinamiği ileri beslemeli sinir ağları ve tekrarlayan yapay sinir ağları ile birleştirilerek derin sinir ağları oluşturulmuş, elde edilen derin sinir ağı modeli aracın sapma oranını yüksek doğrulukta hesaplayabilmiştir.

Son olarak, tasarlanan optimal kontrol algoritmaları yarış pisti üzerinde referans hız ve yörünge yarış profiline karşılık test edilmiş, kontrolcülerin performansları ve otonom yarış aracının her bir kontrolcüdeki tur zamanları karşılaştırılarak en iyi performansın öğrenme tabanlı model öngörülü kontrolcüler olduğu ortaya konmuştur.

ABSTRACT

DEVELOPMENT OF LEARNING BASED TRAJECTORY TRACKING CONTROLLER FOR AUTONOMOUS VEHICLES

With the recent advances in vehicle technologies and Advanced Driving Assistance Systems (ADAS), autonomous vehicles are emerging day by day. An autonomous vehicle is a land vehicle capable of driving from one point to another without a driver. While doing this, basically; it is capable of perception, decision making, planning and control for the planned trajectory.

In this thesis, in order to solve the trajectory tracking problem, the dynamics of the vehicle was modeled using “bicycle dynamics” and validated in “double lane change” tests. This modeled system was used in linear time-varying model predictive control algorithms.

The yaw rate of the vehicle, which is determined as the measured disturbance in the model predictive controller designed for trajectory tracking, was modeled using artificial neural networks, so that the vehicle can always calculate its yaw rate instantaneously. Two methods were used to determine the angular velocity. At first, the vehicle was driven many times along the race track, and the yaw rate that the vehicle should have in response to its instantaneous position was modeled with feedforward neural networks. Secondly, deep neural networks were created to estimate the lateral dynamics of the autonomous racing vehicle by combining feedforward neural networks and recurrent neural networks. Resulting deep neural network model was able to calculate the yaw rate of the vehicle with high accuracy.

Finally; the designed optimal control algorithms were tested on the racetrack against the race trajectory. Performances of the controllers and the lap times of the autonomous racing vehicle in each controller were compared, and it was revealed that the best performance was on the learning based model predictive controllers.

SYMBOLS

Symbol		Description	Unit
x	:	Vehicle longitudinal axis	m
y	:	Vehicle lateral axis	m
$\dot{x}, V_x, v_x$	:	Vehicle longitudinal velocity	m/s
$\dot{y}$	:	Vehicle lateral velocity	m/s
X	:	Global X axis	m
Y	:	Global Y axis	m
ψ	:	Vehicle yaw angle	rad
$\dot{X}$	:	Vehicle speed in global X direction	m/s
$\dot{Y}$	:	Vehicle speed in global Y direction	m/s
$\dot{\psi}, r$	:	Vehicle yaw rate	rad/s
β	:	Vehicle slip angle	rad
δ	:	Vehicle steering angle	rad
l_f	:	Front axle distance from center of gravity	m
l_r	:	Rear axle distance from center of gravity	m
v	:	Vehicle velocity vector	m/s
$\dot{v}, a$	:	Vehicle total acceleration	m/s^2
$\ddot{x}$	:	Vehicle longitudinal acceleration	m/s^2
$\ddot{y}$	:	Vehicle lateral acceleration	m/s^2

$\ddot{\psi}$	:	Vehicle angular acceleration	rad/s^2
θ_f	:	Direction of the velocity vector at the front tire contact patch	rad
θ_r	:	Direction of the velocity vector at the front tire contact patch	rad
α_f	:	Front tire slip angle	rad
α_r	:	Rear tire slip angle	rad
$C_{\alpha,f}$	:	Front tire cornering stiffness	N/rad
$C_{\alpha,r}$	:	Rear tire cornering stiffness	N/rad
F_x	:	Tire force at longitudinal direction	N
F_y	:	Tire force at lateral direction	N
m	:	Vehicle mass	kg
I_z	:	Vehicle yaw moment of inertia	kg.m^2
K_{pt}	:	Total available traction force	N
L	:	Preview distance for longitudinal controller	m
a_r	:	Rolling resistance coefficient	N
b_r	:	Rolling and driveline resistance coefficient	N.s/m
c_r	:	Aerodynamic drag coefficient	N.s^2/m^2
g	:	Gravitational constant	m/s^2
T_s	:	Sampling time	s

ABBREVIATIONS

AGV	**:**	Autonomous Ground Vehicle
ANN	**:**	Artificial Neural Network
COG	**:**	Center of Gravity
DARPA	**:**	Defense Advanced Research Projects Agency
DLC	**:**	Double Lane Change
DNN	**:**	Deep Neural Network
DOB	**:**	Disturbance Observer
GPL	**:**	Gaussian Process Learning
IMU	**:**	Inertial Measurement Unit
ISO	**:**	International Organization for Standardization
LPV	**:**	Linear Parameter Varying
LQR	**:**	Linear Quadratic Regulator
LTI-MPC	**:**	Linear Time Invariant Model Predictive Control
LTV-MPC	**:**	Linear Time Varying Model Predictive Control
MPC	**:**	Model Predictive Control
NMPC	**:**	Nonlinear Model Predictive Control
NN	**:**	Neural Network
P	**:**	Proportional
PD	**:**	Proportional Derivative

PID : Proportional Integral Derivative

RHC : Receding Horizon Control

RMS : Root Mean Square

RNN : Recurrent Neural Network

RRT : Rapidly Exploring Random Tree

SAE : Society of Automotive Engineers

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Autonomous vehicles, which are one of the popular topics of today, have gradually started to take place in our lives with ongoing studies. Even the automatic vacuum cleaners in our homes, which can be considered autonomous vehicles at the simplest level, use high-end autonomous vehicle technologies. In addition, autonomous ground vehicles (AGVs) provide unmanned logistics services in factories.

High-end autonomous vehicles are divided into 6 levels by Society of Automotive Engineers (SAE) [1] which are shown in Figure 1.1.

Figure 1.1. SAE levels of driving automation

Level 5 autonomous vehicles are still not fully used, and development work continues under the leadership of companies such as Tesla, Ford, Aptiv, ArgoAI. In addition to these development efforts, competitions for autonomous vehicles have been organized and universities and industry provide gains to the literature thanks to these competitions. Formula Student Driverless [2], F1/10TH Autonomous Racing [3], Indy Autonomous Challenge [4] and Roborace [5] are competitions on the development of autonomous racing car algorithms.

Autonomous mobile robots or vehicles should basically have three characteristics according to the system architecture shown in Figure 1.2.

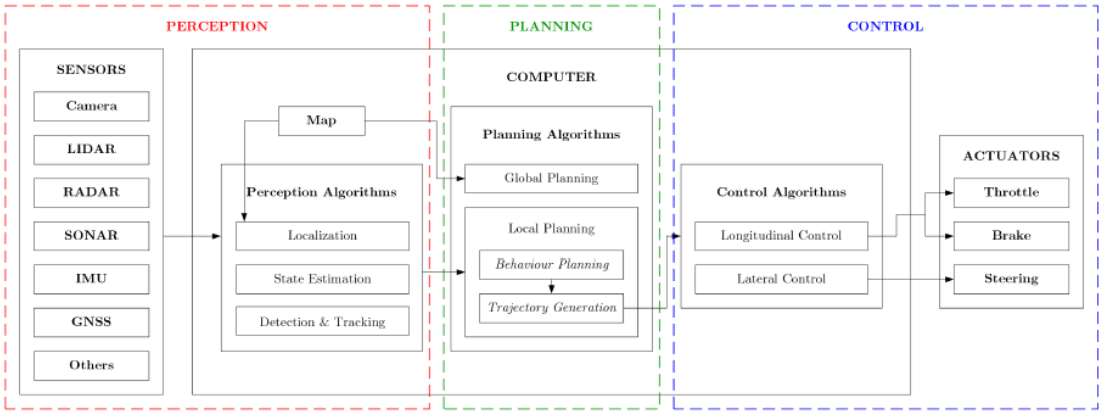


Figure 1.2. Autonomous vehicle system architecture

In the perception module, the vehicle interacts with the world it is in and performs the task of sensing. Vehicle collects data with sensors such as stereo cameras, ultrasonic sensors, radars, laser scanners, GPS and etc.

In the planning module, the data received from the sensors are used to create a local or global trajectory [6] for the navigation of the vehicle. This trajectory planning can be algorithms such as A*, RRT, Djisktra Algorithm [7]–[9] , or Minimum Time Optimal Trajectory Planning [10] when minimizing the lap time for autonomous racing vehicles is the goal.

In the control module, points or states coming from the planning module are used as references. To reach reference states or points necessary commands given at this stage to actuators of the vehicle, which are powertrain and steering system elements. Control stage is more critical when compared to other stages, because vehicle this stage provides the movement of the vehicle directly. Although control stage looks independent from the previous stages, it affects them recursively.

This study, started in Ford Otomotiv San. A.Ş. with a development of a mobile robot and with the assumption of a vehicle has the Ackermann driving kinematics [11], trajectory tracking controllers are developed for an autonomous racing vehicle. Co-simulation environments MATLAB/Simulink [12] and high fidelity vehicle dynamics simulator Carsim [13] were used.

1.1. Problem Statement

The primary task of this thesis is to follow a pre-defined velocity profile generated by Carsim for autonomous racing within its driver model. Secondary task is to follow optimal racing line combining with the reference velocity.

The difficulty occurs because of the non-linearity of the tire-road interactions and the coupling between longitudinal and lateral dynamics of the vehicle. Because of velocity dependency in vehicle dynamics the problem becomes difficult to overcome.

To overcome both problems, two trajectory tracking controllers are developed by using de-coupled control method for longitudinal and lateral dynamics of the vehicle. Longitudinal controller controls the speed of the vehicle with throttle and breaking signals, additionally lateral controller tries to make zero lateral deviation to the racing line by using steering wheel.

1.2. Related Work

Over the past decades many scientists and engineers have been working on autonomous vehicles' control problem which may be called as "navigation problem". To navigate a vehicle along a predefined trajectory, vehicles' steering angle and velocity must be controlled well.

The most basic trajectory tracking model can be told as "Follow the Carrot" model. The model's name came from the analogy of riding donkey and guiding the donkey using a carrot directed to desired orientation [14].

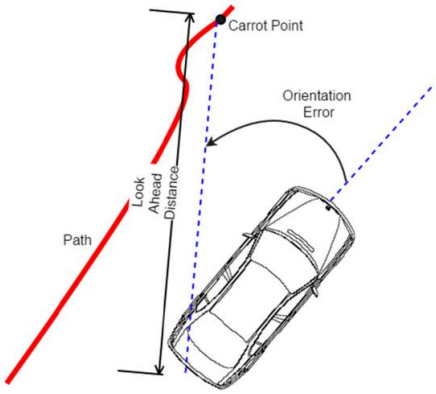


Figure 1.3. Follow the carrot model

In [15], motion controller for an AGV is developed by using double closed loop PID controllers. A constant curvature, circular road profile used as reference trajectory in this study, also simulation is made within Matlab/Simulink only.

In [16] a proportional integral derivative (PID) controllers are used to follow a desired path and velocity profile. Desired steering angle used as reference input so no road curvature estimation needed to be used in this study.

In [17] Snider investigated the automatic steering methods for an autonomous vehicle in Carsim environment. The investigated control methods are; Pure Pursuit, Stanley and Linear Quadratic Regulator (LQR). These control algorithms applied at different velocities and compared the lateral tracking performance with each other, no road information estimation is used in this study also.

Double closed loop controllers are used again in [18]. Steering control was accomplished with P and PD controllers, and non-linear kinematic controller Stanley. Different from the above studies this study used the curvature information came from camera sensor on board. Dynamical vehicle model was not used this study and predictive control method was not taken into account.

The first milestone in the racing of autonomous vehicles was achieved by the Defense Advanced Research Projects Agency (DARPA) in 2005. The autonomous vehicle named Stanley, shown in Figure 1.4, won the race by controlling it with the PID controller, which is one of the classical control methods, and the kinematic model of the vehicle was used as the mathematical model. Later, this control method entered the literature as Stanley Controller [19].

Figure 1.4. The robot “Stanley”

Second and third place vehicles in the competition, Sandstorm and H1ghlander, used pure pursuit algorithm [20], which is one of the geometric control methods. Considering the control algorithms of the winning vehicle and other finalists, the kinematic model of the vehicle was used, not the dynamic model, road information estimation was not taken into account and no dynamic vehicle models were used.

Jan Filip [21] investigated the model predictive control (MPC) of an autonomous vehicle on a driving course with Matlab/Simulink and IPG CarMaker [22]. Composition of his study has three main subjects. Generating the minimum time trajectory, longitudinal control and lateral control of the vehicle along driving course. To control the vehicle, Model Predictive Control (MPC) method was used. However, no online estimation of road profile was used his study and no algorithms were used to estimate measured disturbances.

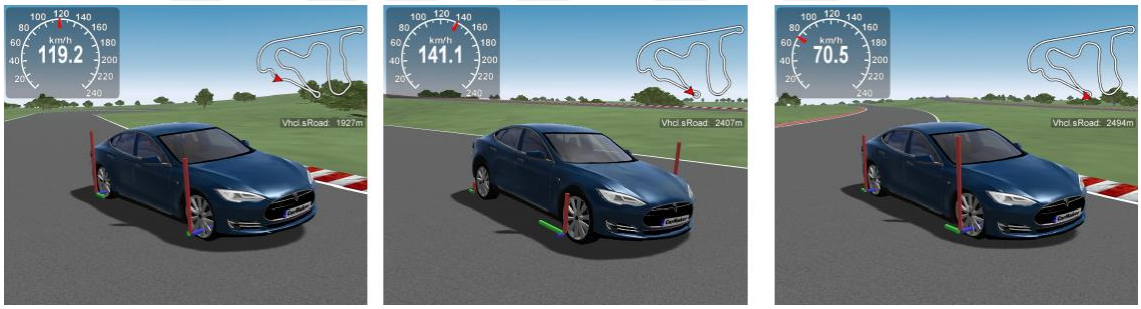


Figure 1.5. Vehicle simulation in CarMaker

In [23] an Audi TTS, was controlled with de-coupled control method and, used references with minimum time trajectory profile. However, designed controllers were using classical control methods with cascade architecture and performance of the controllers were heavily tuning dependent.

Behnaz Ahmadi [24] created MPC for lateral control with combined single track model with switching. Switching logic based on switching between the kinematic model and dynamic model depending on the vehicle velocities. Created lateral controllers were tested on double lane change maneuvers with different constant vehicle speeds. Consequently, she compared the tracking performance of her controller with different speeds but no measured disturbance input used and there was no longitudinal controller to track desired speed in the study.

Florian Curinga [25] created 2 layers MPC. First layer finds the optimal racing line by using optimization algorithm and second layer ensures the tracking of the found racing line. Illustration of the study is shown in Figure 1.6. Created simulation only validated in Matlab environment and as reference track only an S-curve is used. Since simulation is made within Matlab only no disturbances occurred at the simulation stage, additionally no online road profile estimation was made.

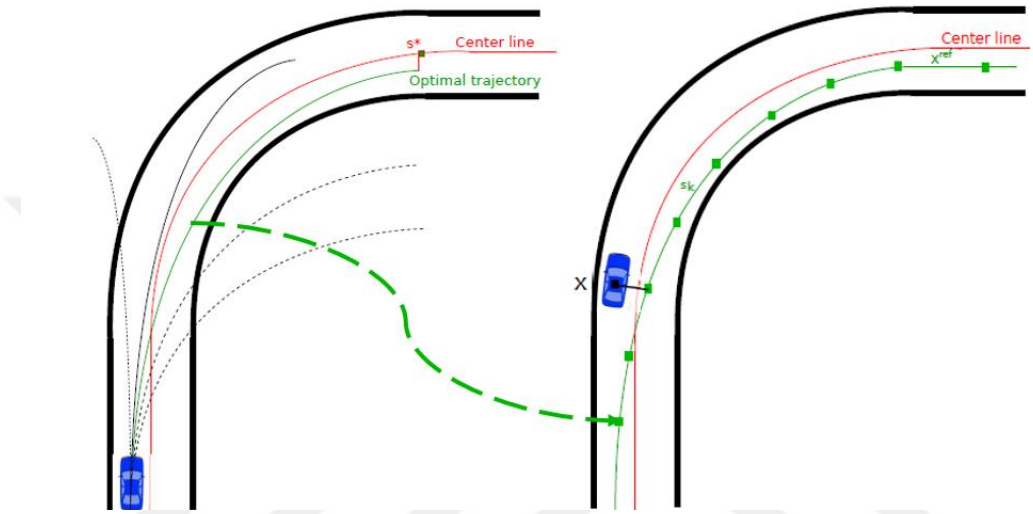


Figure 1.6. 1st and 2nd layer illustration

In 2011, Muhammad A. Abbas [26], applied Nonlinear MPC (NMPC) algorithm on a passenger vehicle in Carsim environment then he compared the nonlinear controller's performance with double lane change maneuvers with different constant vehicle velocities. The studies [26] and [24] were looking similar except; Abbas added obstacle avoidance maneuvers to his study. Yet again in both studies, vehicles weren't aware of the road that were driven. This means that no road information is estimated by vehicles, and so no measured disturbance was estimated for the controller development.

Xu Y. et al. [27] created an MPC controller for lane keeping system. Future road information used in this study, but this future information was not coming from an estimation result. An offline trajectory generator, generated the reference trajectory with 5 preview points. The proposed architecture for lane keeping system is given in Figure 1.7.

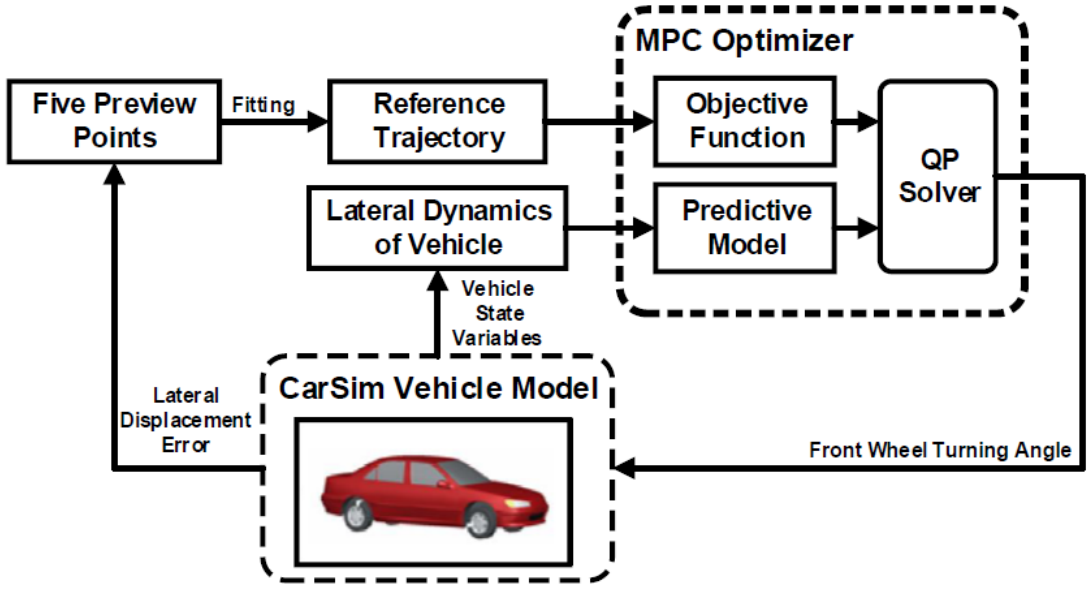


Figure 1.7. Proposed lane keeping system by Xu Y.

In 2016, Xiangjun Qian investigated a problem as known as “Platooning”. Platooning is a method for driving a group of vehicles together. With this technology automated highway systems, reduced fuel consumptions, low traffic collisions become possible. Qian designed an MPC controllers for each individual vehicle in the platoon. The platoon or convoy supervisor vehicle manages the formation of the platoon. Vehicles in the platoon were tested in different obstacle avoidance and lane change maneuvers, which can be seen in Figure 1.8. However, road profile information was not estimated by vehicles, so vehicles weren’t aware of the curvature profile of the road that they drove.

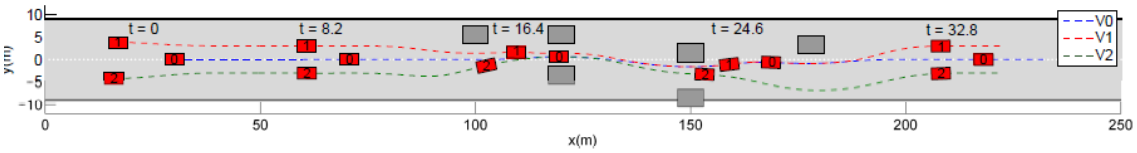


Figure 1.8. Platooning of vehicles in obstacle avoidance scenario with MPC

In 2018, Haoan Wang designed robust PID controllers to maneuver the vehicle for vehicle path following and collision avoidance [28]. After validation of robustness of the controllers, disturbance observers (DOB) were used to estimate the output disturbances to compensate the control errors. DOB used vehicle control system is given in Figure 1.9.

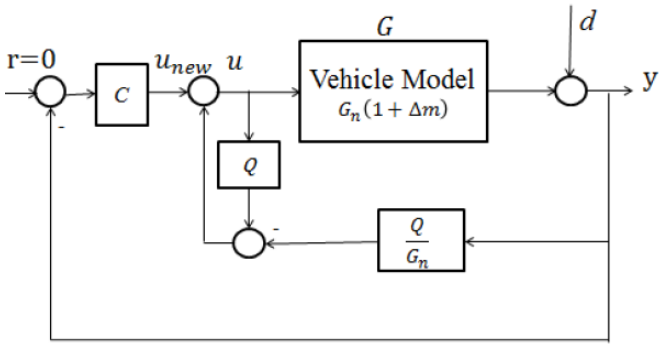


Figure 1.9. Disturbance observer compensated control system

Verschueren et al. studied on time optimal MPC and online trajectory generation in both simulation and real world experiments with small remote controlled cars [29]. In this study, online trajectory is generated during vehicle motion and time optimal driving profile are shown in Figure 1.10. However, in this study, tire road interactions were not taken into vehicle modelling, and slip free bicycle dynamics model was used. In addition, no future road information or curvature profile were estimated as well.

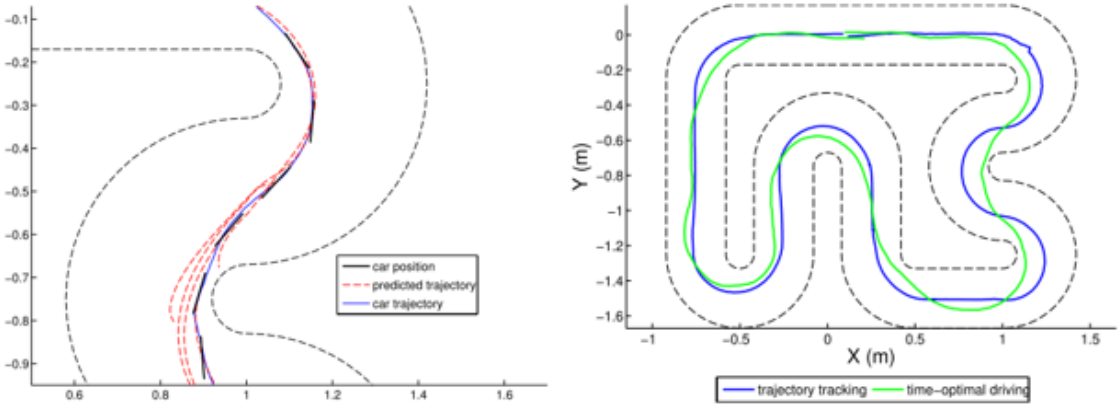


Figure 1.10. Predicted trajectory and time optimal driving profile

Kabzan et al. developed a learning based MPC for autonomous race vehicle named "gotthard" in ETH Zurich. Since race vehicle dynamics are highly nonlinear and complex, this study used a nominal bicycle dynamics model and model is updated with online Gaussian Process Learning (GPL) to predict state and uncertainties in the vehicle. This updated model used as the basis model of the predictive controller, then velocity improvement observed along the race track.

Figure 1.11. Autonomous race vehicle “gotthard”

1.3. Contributions

In this study Linear Time Varying Model Predictive Controllers (LTV-MPC) were developed for trajectory tracking of an autonomous race vehicle. Tracking control problem was separated into two individual areas; longitudinal tracking and lateral tracking along the race track.

The problem of trajectory tracking supported with Neural Networks (NN). An artificial NN was trained to teach the race track to the vehicle, then learned track curvature was given to control algorithm as measured disturbance to improve tracking performance.

In extension to predictive control study, lateral dynamics of the vehicle was modeled by using deep neural networks. NN model used as an estimator for the road curvature since the curvature is proportional with yaw rate and the longitudinal velocity.

Finally, the performance of two LTV-MPCs were verified and compared in high fidelity simulation software in Carsim along a driving course.

1.4. Outline

The main purpose of this thesis to develop a predictive controller supported with artificial neural networks. The proposed system architecture can be found in Figure 1.12. Chapter 2 describes the vehicle modelling by using *bicycle dynamics* in the scope of trajectory tracking. Chapter 3 introduces the model validation and experiments made to validate the developed analytical model. Chapter 4 describes the Model Predictive Control (MPC),

MPC types and quadratic programming with its solvers. Chapter 5 introduces the velocity trajectory and desired path used in this thesis and how they were generated. Chapter 6 and Chapter 7 explains the controllers used for longitudinal and lateral controllers respectively. In Chapter 8 the lateral dynamics of the vehicle was modeled by using deep neural networks. Chapter 9 introduces the control algorithm and compares the controller performances with or without road curvature information. Chapter 10 summarizes this thesis, gives information about the future work may be accomplished. At last section Appendix is given.

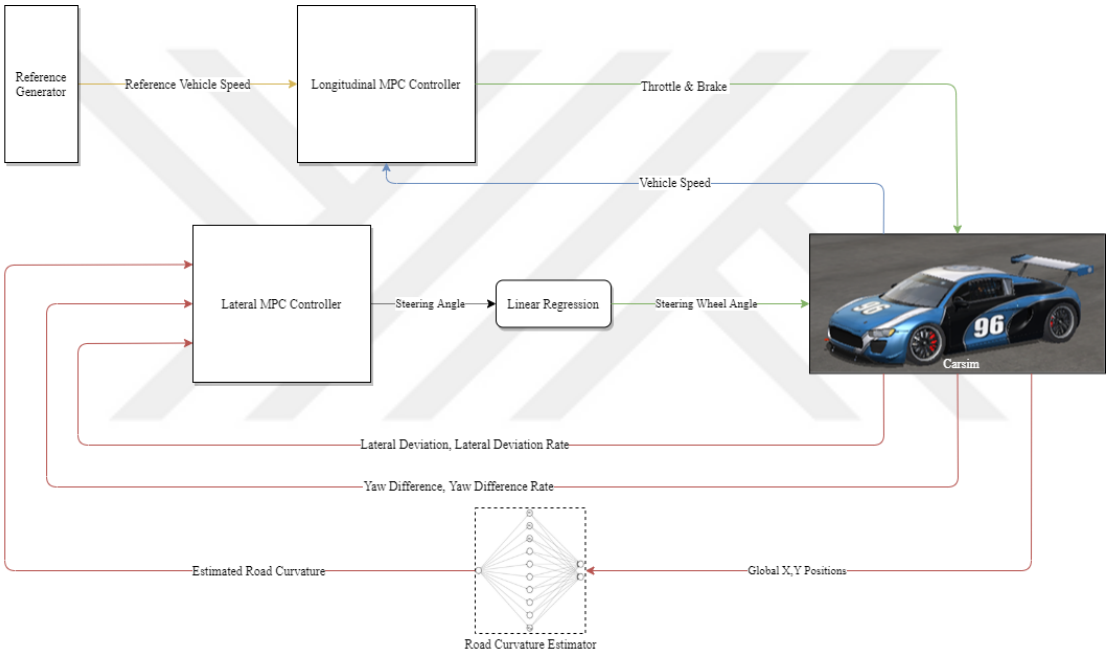


Figure 1.12. Proposed system architecture

2. VEHICLE MODELLING

This section introduces the modelling of vehicle dynamics based on “bicycle model” of the vehicle [30]. The bicycle dynamics make assumption of front and rear tires are lumped on the central axis as shown in Figure 2.1.

2.1. Coordinate Frames

Vehicle dynamics and robotics have many aspects in common. Because a vehicle can be treated as car-like mobile robot, coordinate frames are generally similar. In general, there are two coordinate frames for ground vehicle modelling. These are global and local coordinate frames. Global coordinate frame explains the vehicle motion in a world such as race track or an enclosed area. In contrast, local coordinate frame explains vehicle motion on its body.

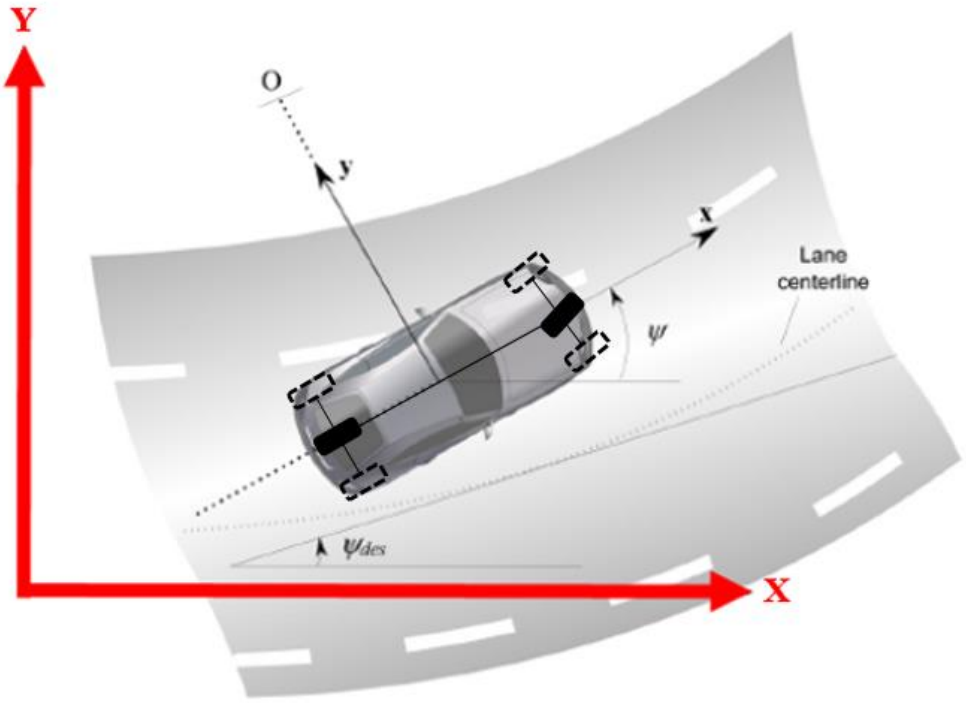


Figure 2.1. Bicycle model of the vehicle

Red lines in Figure 2.1 shows the global coordinate frame and it defines the global pose of the vehicle which are “X, Y” coordinates of the vehicle and “Yaw” angle.

Longitudinal black “x” axis on vehicle body shows local “x” coordinate and lateral “y” axis shows the local “y” axis as well. These coordinate frames are used modelling the dynamics of the vehicle, then results are transformed to global coordinate system via coordinate transformation for trajectory/path following purposes.

ISO and SAE standards have being used for the coordinate frame conventions. The difference is the lateral axis on local coordinate system. The difference is given in more detail in below.

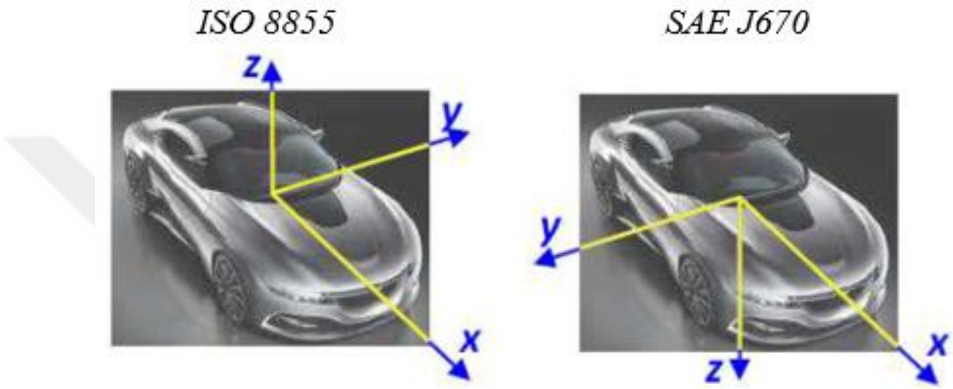


Figure 2.2. Difference in coordinate system in ISO & SAE standards

2.2. Kinematic Model of Vehicle Motion

Under certain assumptions kinematic model of the vehicle may be developed. These assumptions might be summarized as:

1. No slip occurs on the tires
2. Each velocity vector of the tires is aligned with the direction of the tires
3. Vehicle motion is assumed planar
4. No tire forces occur because of tire-ground interactions

The given assumptions are acceptable when acceleration of the vehicle motion relatively low, because when the acceleration becomes higher the non-linear structure of the tires create slip with the ground. The kinematic model is a suitable model for autonomous parking algorithms [31], [32] and path following of car-like mobile robots [33].

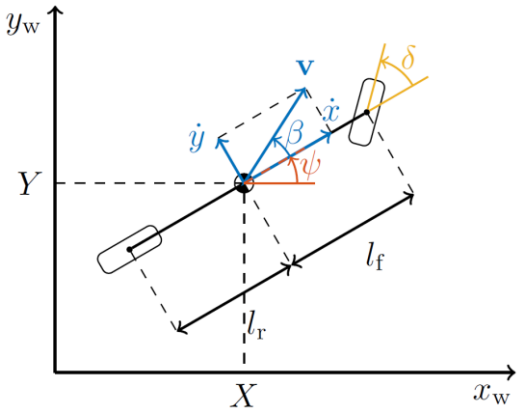


Figure 2.3. Kinematic model of bicycle model of vehicle dynamics

The nonlinear kinematic equations of motion of bicycle model given in Figure 2.3 are:

$$\dot{X} = vcos(\psi + \beta) \tag{2.1}$$

$$\dot{Y} = vsin(\psi + \beta) \tag{2.2}$$

$$\dot{\psi} = \frac{vcos(\beta)}{l_f + l_r}(\tan(\delta)) \tag{2.3}$$

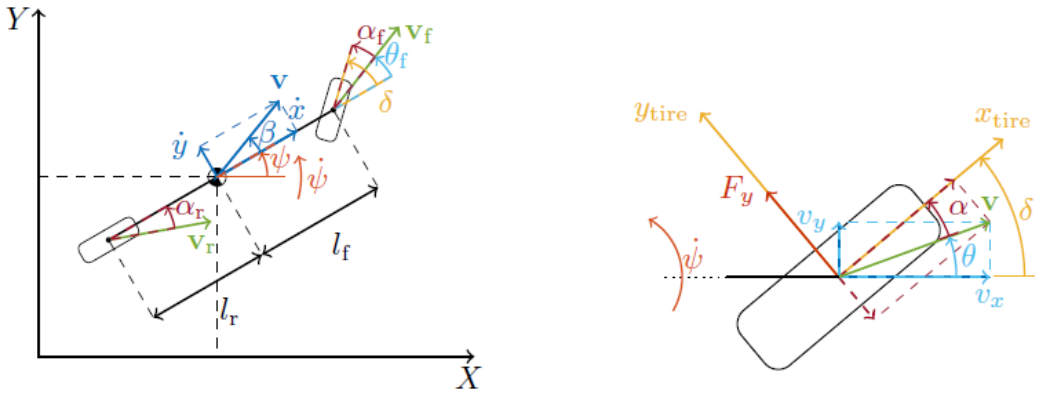
$$\beta = \tan^{-1}\left(\frac{l_r}{(l_f + l_r)}\tan(\delta)\right) \tag{2.4}$$

$$\dot{v} = a \tag{2.5}$$

Where; X and Y shows the vehicle coordinates in global coordinate system and can be found by integrating $\dot{X}, \dot{Y}$. ψ is the heading direction global coordinate system, v is the velocity vector of the vehicle located on center of gravity, a is acceleration, β is the vehicle slip angle, δ is the steering angle with respect to longitudinal axis, l_f and l_r shows the front and rear axle distances from center of gravity respectively.

2.3. Dynamic Model of Vehicle Motion

At higher velocities and higher accelerations, the nonlinear ground – tire interactions occur on the tires, then vehicle dynamics become highly complex. In this case, addition to the kinematic model tire forces have to be included in the dynamics equations.



(a) **(b)**

Figure 2.4. Dynamic model of bicycle model of vehicle dynamics

The nonlinear dynamic equations of motion of bicycle model given in Figure 2.4.a are:

$$\ddot{x} = m^{-1}\left(F_{x,r} + F_{x,f}\,cos(\delta) - F_{y,f}\,sin(\delta)\right) + \dot{\psi}\dot{y} \tag{2.6}$$

$$\ddot{y} = m^{-1}\left(F_{y,r} + F_{x,f}\,sin(\delta) - F_{y,f}\,cos(\delta)\right) - \dot{\psi}\dot{y} \tag{2.7}$$

$$\ddot{\psi} = I_z^{-1}\left(l_f F_{x,f}\,sin(\delta) + l_f F_{y,f}\,cos(\delta) - l_r F_{y,r}\right) \tag{2.8}$$

$$v = \sqrt{\dot{x}^2 + \dot{y}^2},\ \beta = \arctan\left(\frac{\dot{y}}{\dot{x}}\right) \tag{2.9}$$

$$\dot{X} = v\,cos(\psi + \beta), \dot{Y} = v\,sin(\psi + \beta) \tag{2.10}$$

$$a_y = \ddot{y} + \dot{\psi}\dot{x} \tag{2.11}$$

Where; $\dot{x}, \dot{y}, \dot{\psi}$ are longitudinal, lateral and yaw velocity in the vehicle frame, $\ddot{x}, \ddot{y}, \ddot{\psi}$ are longitudinal, lateral and yaw acceleration in the vehicle frame. m is the mass of the vehicle, I_z is the yaw moment of inertia of vehicle body, v is the magnitude of the velocity vector which combines the lateral velocity $\dot{y}$ and longitudinal velocity $\dot{x}$. a_y is the total lateral acceleration, ψ is the heading direction in global frame which is obtained by integrating the (2.8) twice. β is the slip angle of the vehicle, α denotes the tire slip angle. δ denotes steering angle of the front wheel, l_r, l_f denotes distance from center of gravity of the rear and front axles respectively.

X, Y are the position of the vehicle in global coordinate frame which can be obtained by $\dot{X}, \dot{Y}$.

Occurring tire forces from the tire – road interactions shown in Figure 2.4.b $F_{i,j}$ denotes $(where, i = \{x, y\}, j = \{f, r\})$ in (2.6) - (2.8) tire forces which provide vehicle motion.

There are several methods to model the tire forces, and these are empirical methods. Estimation of tire forces and modelling tires are explained in Section 2.4 in depth.

2.4. Tire Modelling

Tires are highly complex and nonlinear systems because of their nature and the material which they made of. Many of the tires consists of rubber materials and radial steel fibers. This complexity causes the tire characteristic to be measured only by experimental methods. The measured values then fitted to mathematical methods, to validate tire models.

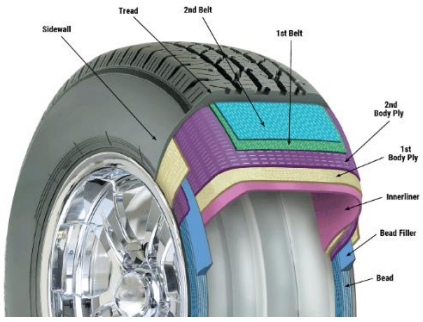


Figure 2.5. Composition of a tire

Not only the structural complexity of the tires made difficult model them but also the ground interactions, vertical load, inflation pressure etc.

During cornering an angle exists between the tire's pointing direction and the velocity of the tire contact patch. This angle is denoted as "tire slip angle, α" as given in Figure 2.4.b.

For the bicycle dynamics of the vehicle front tire slip angle is defined as:

$$\alpha_f = \delta - \theta_f = \delta - \arctan\left(\frac{\dot{y} + l_f\dot{\psi}}{v_x}\right) \tag{2.12}$$

$$\alpha_r = -\theta_r = -\arctan\left(\frac{\dot{y} - l_r\dot{\psi}}{v_x}\right) \tag{2.13}$$

Where θ_f and θ_r denote the direction of the velocity vector at the tire contact patch and δ denotes the wheel pointing direction with respect to body fixed coordinate system.

Since our vehicle's rear wheel can't turn in this study θ_r is taken as zero. δ is being known as "steering angle" in bicycle dynamics of vehicle motion.

There are three ranges of operation in tire dynamics in literature as known as "elastic, transitional" and "frictional" [34]. These operation ranges are determined by slip angle of the tire. Generally, if the tire slip angle is smaller than 3 degrees tire dynamics might be assumed linear, otherwise it becomes nonlinear. In this thesis 305/30/19 and 235/35/19 racing tire models in Carsim were used. Detailed lateral force distribution is defined with varying vertical load is given in Figure 2.6.

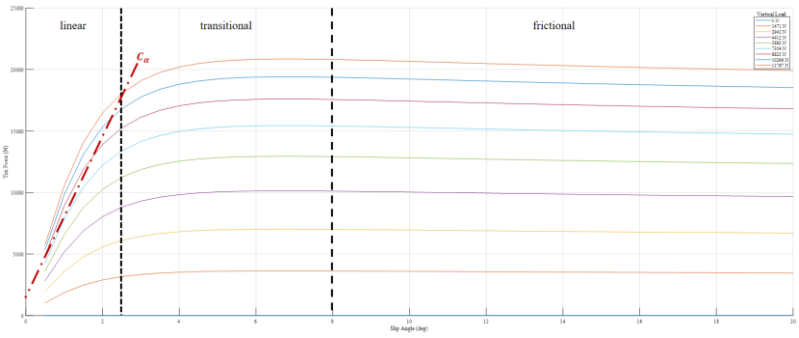


Figure 2.6. Lateral tire forces for varying vertical tire load

2.4.1. Linear tire model

Characterizing tire forces is one of most important cases to do when modelling the dynamics of vehicle motion. For simplicity and low usage computation sources many researchers have been using linear tire models [35], [36].

Using the small angle approximation for steering angle δ, $\sin(\delta) = \delta, \cos(\delta) = 1$ can be approximated similarly in (2.12).

$$\alpha_f = \delta - \left(\frac{\dot{y} + l_f\dot{\psi}}{v_x}\right) \tag{2.14}$$

$$\alpha_r = -\left(\frac{\dot{y} - l_r\dot{\psi}}{v_x}\right) \tag{2.15}$$

The lateral forces occurring on the vehicle can be estimated with proportionality constant C_α, called "cornering stiffness" given in Figure 2.6. Therefore, the tire forces become:

$$F_{y,f} = C_{\alpha,f} \times \alpha_f \tag{2.16}$$

$$F_{y,r} = C_{\alpha,r} \times \alpha_r \tag{2.17}$$

When (2.16) - (2.17) plugged into (2.6) - (2.8), vehicle model becomes linearized. This linearized model is being known as “linear bicycle dynamics model of the vehicle motion” in the literature. Estimation of tire cornering stiffness is given in more detail in Section 3.1.2.

2.4.2. Fiala tire model

When lateral and longitudinal forces occurred on the tires total forces is limited by vertical load (F_z) and friction coefficient(μ). This relation is explained with *Kamm’s Circle of Forces.*

$$\mu F_z \geq \sqrt{F_x^2 + F_y^2} \tag{2.18}$$

Explanation of combined tire forces can be found in [11], [34].

Fiala tire model [37] describes the lateral force occurring on the tires as below:

$$F_y = \begin{cases} C_\alpha \tan(\alpha) - \left(\frac{C_\alpha^2}{3F_{y,max}}\right) |\tan(\alpha)| \tan(\alpha) + \frac{C_\alpha^3}{27F_{y,max}^2} \tan(\alpha)^3 , |\alpha| < \alpha_{sl} \\ sgn(\alpha) F_{y,max}, |\alpha| \geq \alpha_{sl} \end{cases} \tag{2.19}$$

Where C_α is the cornering stiffness, α is the tire slip angle and α_{sl} is the maximum slip angle at the exerted maximum tire force. α_{sl} is described as:

$$\alpha_{sl} = \frac{3F_{y,max}}{C_\alpha} \tag{2.20}$$

Where maximum lateral force $F_{y,max}$ is described in (2.21).

$$F_{y,max} = \rho \mu F_z \tag{2.21}$$

Maximum lateral force in (2.21) is reduced with a constant ρ. To find coefficient ρ for lateral direction, restrictions from (2.18) (Kamm’s Circle) should be integrated. Assuming $|F_x| \leq \mu F\_z$, ρ can be found:

$$\rho(F_x, F_z) = \sqrt{(\mu^2 F_z^2 - F_x^2)/\, \mu F_z} \in \langle 0,1 \rangle \tag{2.22}$$

2.4.3. Pacejka tire model

Hans Pacejka introduced the magic formula in tire dynamics modelling [34]. He provided an analytical formula for the estimation of lateral tire force with high accuracy.

$$F_y(\alpha) = Dsin\big[Carctan\big(B\alpha - E(B\alpha - \arctan(B\alpha))\big)\big] \tag{2.23}$$

Where, B denotes stiffness factor, C denotes shape factor, D denotes peak value, E denotes curvature factor and determines the curvature shape [36]. In this model the cornering stiffness (C_α) is given with production of BCD.

2.4.4. Tire parameter estimation

In this study linear tire model is used as previously explained in Section 2.4.1. The estimation methods for tire cornering stiffness can be divided to two sections: "Time-domain methods [38], [39] and transfer function based methods [40]. In [41] both time domain and transfer functions estimation methods are compared and analyzed.

"Direct method" technique was applied in this thesis to estimate tire cornering stiffness values. According to (2.16) and (2.17) lateral tire force is directly production of *cornering stiffness* and *tire slip angle*. So, cornering stiffness can be calculated by dividing F_y to α.

$$C_{\alpha,f} = \frac{(l_r\ddot{y}m + l_r rm\dot{x} + I_z\dot{r})/L}{(-\dot{y} - rl_f + \delta\dot{x})/\dot{x}} = \frac{F_{y,f}}{\alpha_f} \tag{2.24}$$

$$C_{\alpha,r} = \frac{(l_f rm\dot{x} + l_f\ddot{y}m - I_z\dot{r})/L}{(rl_r - \dot{y})/\dot{x}} = \frac{F_{y,r}}{\alpha_r} \tag{2.25}$$

Where; $C_{\alpha,f}$ and $C_{\alpha,r}$ denotes cornering stiffness of front and rear tires respectively. r is the yaw rate ($\dot{\psi}$), and $L = l_f + l_r$ which is the distance from front axle to rear axle. Since it's difficult to measure lateral acceleration $\ddot{y}$, from (2.11) ($\ddot{y} = a_y - \dot{\psi}\dot{x}$) might be used. Other parameters are explained in Section 2.3 previously.

To measure cornering stiffness in Carsim;

1. Constant speed double lane change scenarios were driven within Carsim
2. Lateral tire forces and tire slip angles were exported; then a linear regression algorithm was run to find cornering stiffness without using an offset term.

3. MODEL VALIDATION

First of all, to create a successive model predictive controller, a robust and fine-tuned model of the process is needed. For this purpose, several tests were made.

3.1. Model Validation Experiments

3.1.1. Steering angle estimation

At first, there are two input parameters affecting the system dynamics of a vehicle when a linear model is used. These parameters are “steering angle of the consolidated front wheel” and “longitudinal velocity of the vehicle”. In a real vehicle it’s not possible to direct measure the steering angle of the vehicle, because there is no wheel on the front axle center. In order to find this consolidated angle, left and right tire angles must be averaged. These angles can be measured with wheel position sensors in real [42].

Figure 3.1. Wheel position sensor

Secondly, to control steer angle; the interactions from steering wheel position to vehicle steering angle should be known. This phenomenon is described as “steering ratio” in the vehicle dynamics literature since there is a constant coefficient between the steering wheel position and the steering angle of the vehicle. In this study, rather than using just a constant gain to model steering system a linear regression model was created with offset term. The found offset term in the linear regression model provides better accuracy for calculation of steering angle.

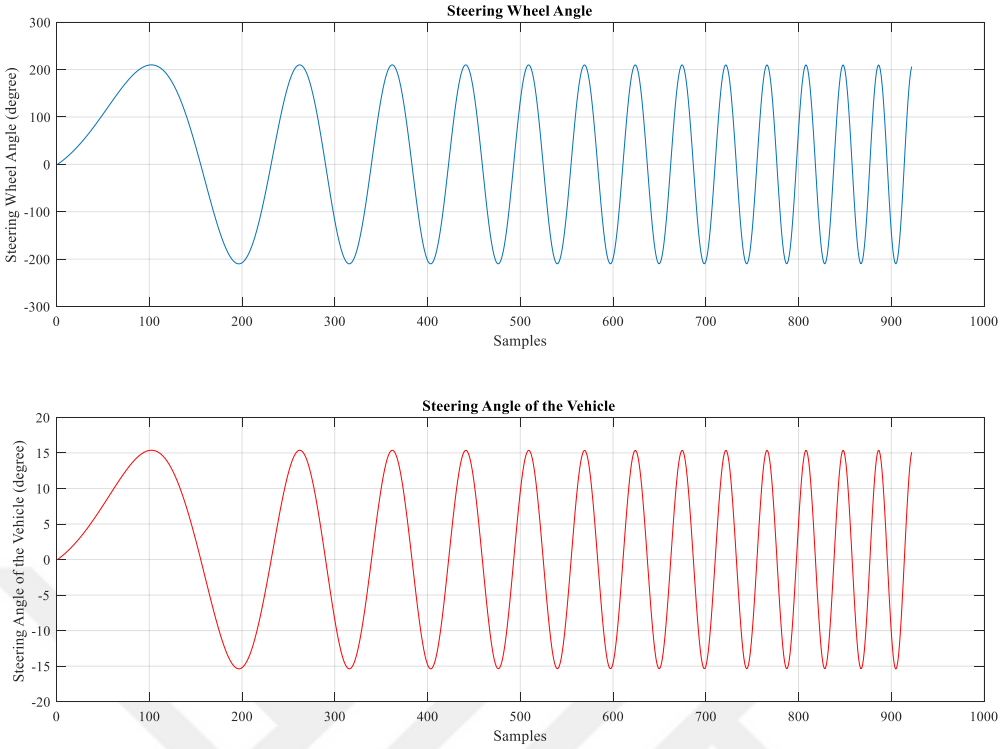


Figure 3.2. Steering wheel input & steering angle of the vehicle

A sine-sweep with increasing frequency and constant amplitude with 200 degrees of steering wheel angle applied to vehicle, then average of steering angle of the front left ($\delta_{f,l}$), front right ($\delta_{f,r}$) tires were recorded.

Where,

$$\delta = \frac{(\delta_{f,l} + \delta_{f,r})}{2} \tag{3.1}$$

At last linear regression was applied to find relation from steering angle (δ) to steering wheel angle.

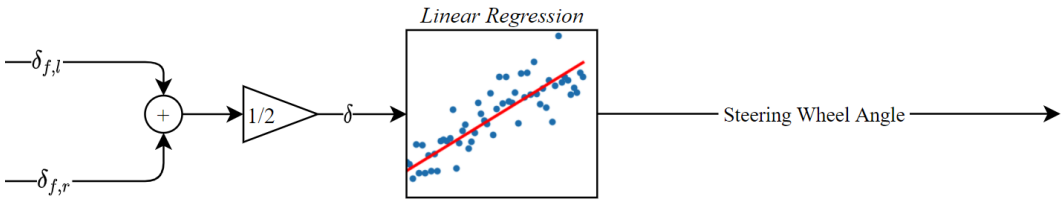


Figure 3.3. Steering angle estimation algorithm

The performance of the linear regression model is given in Figure 3.4.

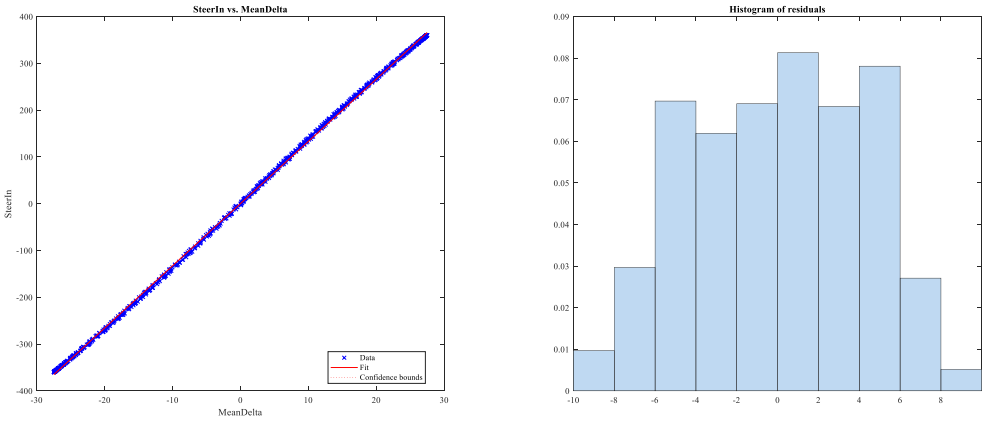


Figure 3.4. Linear regression results steering angle to wheel angle

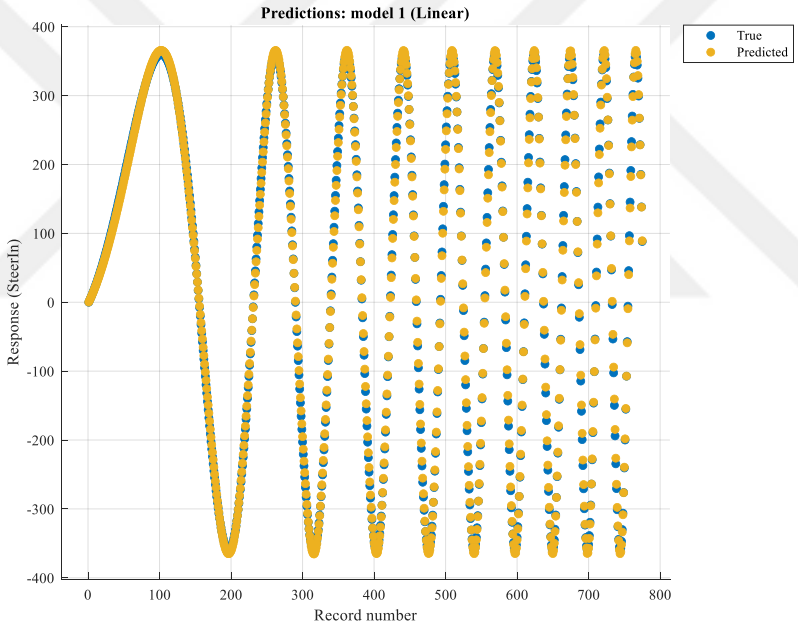


Figure 3.5. Linear regression response plot

The equation for linear model found as:

$$Steering\ Wheel\ Angle\ (rad) = 13.307 \times \delta(rad)\ + 0.25211 \qquad (3.2)$$

3.1.2. Tire model validation

Motion of the vehicle is created by the forces between the tires and ground. To model a vehicle accurately, forces occurring because of interactions have to be modelled near perfect.

As mentioned before tire modeling based on linear tire models and the estimation method “direct method” was used. In this section double lane change maneuvers introduced to vehicle and tire forces were estimated to find cornering stiffness of both front & rear consolidated tires.

Equations (2.24) and (2.25) were used to estimate cornering stiffness of both tires, the resulting graphs in show that, given values in Table 3.1 covers the behavior of the tires.

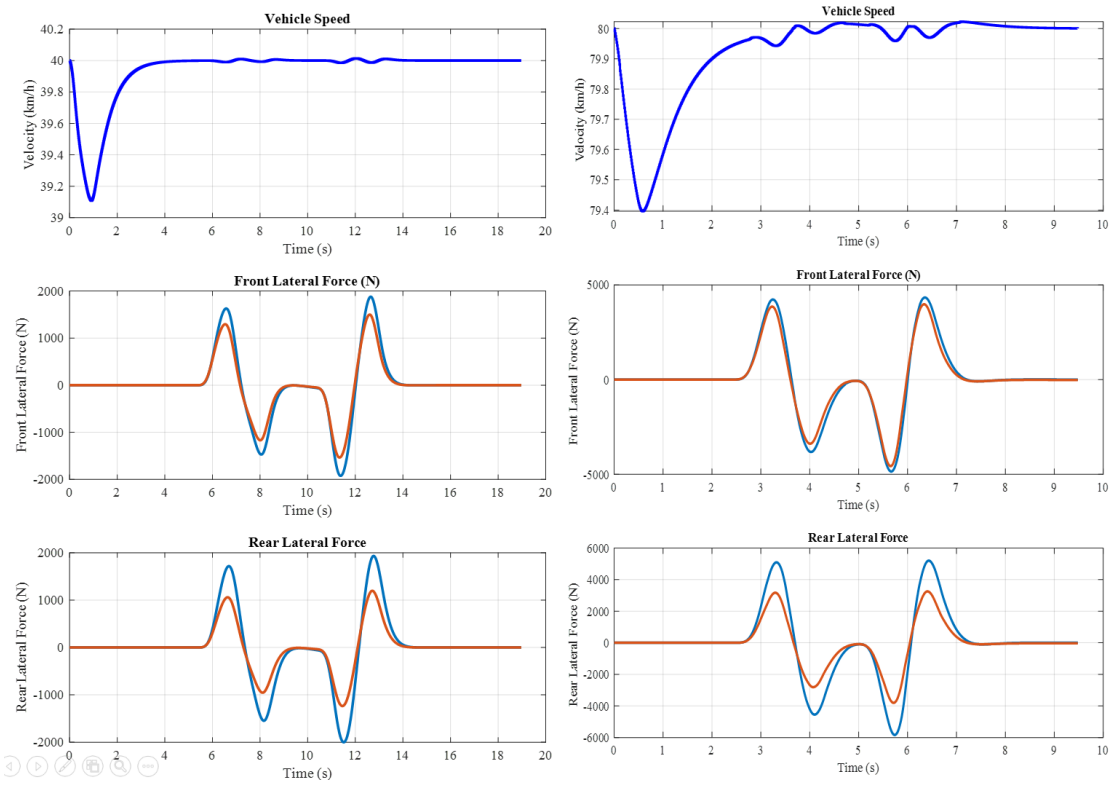
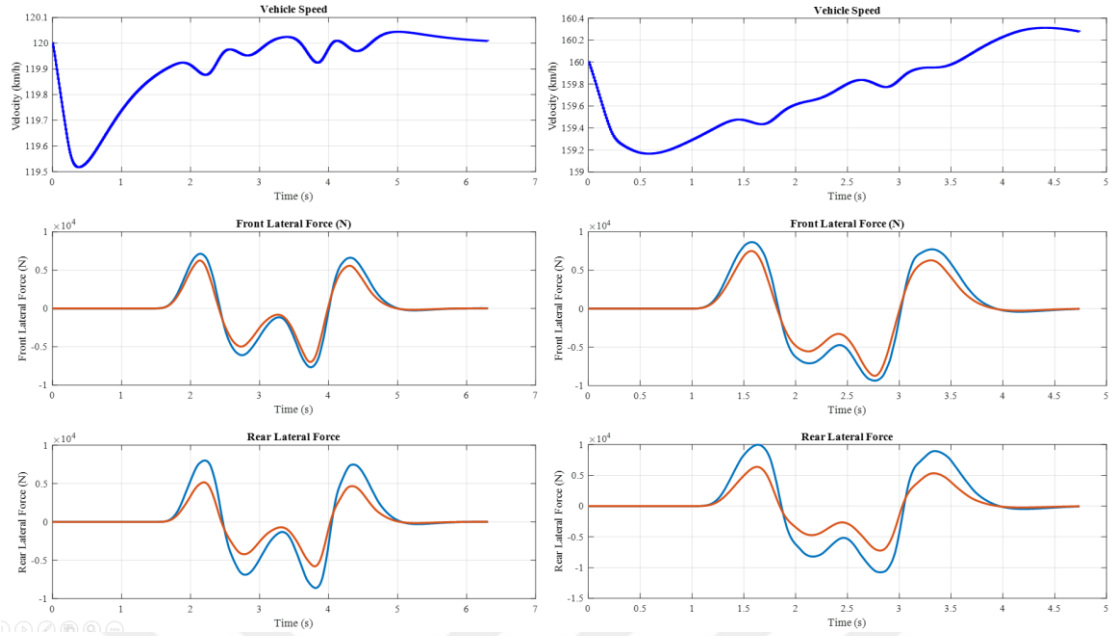


Figure 3.6. Tire force estimation with (a) 40 km/h (b) 80 km/h



(a) **(b)**

Figure 3.7. Tire force estimation with (a) 120 km/h (b) 160 km/h

Table 3.1. Cornering stiffness values for front & rear tires

Front Tire Cornering Stiffness ($C_{\alpha,f}$)	335431.802941654 (N/rad)
Rear Tire Cornering Stiffness ($C_{\alpha,r}$)	556485.037312196 (N/rad)

3.1.3. Vehicle model validation

The nonlinearity in vehicle motion is observed from the tires at the first stage. Making tire models linear ensures that the dynamics of the vehicle motion becomes linear. The linearized equations of nonlinear equations given in (2.6) - (2.8) using small angle assumptions [34] creates system's state space linear model in time-domain as:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{3.3}$$

$$y(t) = Cx(t) + Du(t) \tag{3.4}$$

Where, $\dot{x}$ represents the system dynamics, A represents system matrix, x represents the state matrix, B is the input matrix, u is the input. y is output, C is output matrix and D is feedthrough matrix.

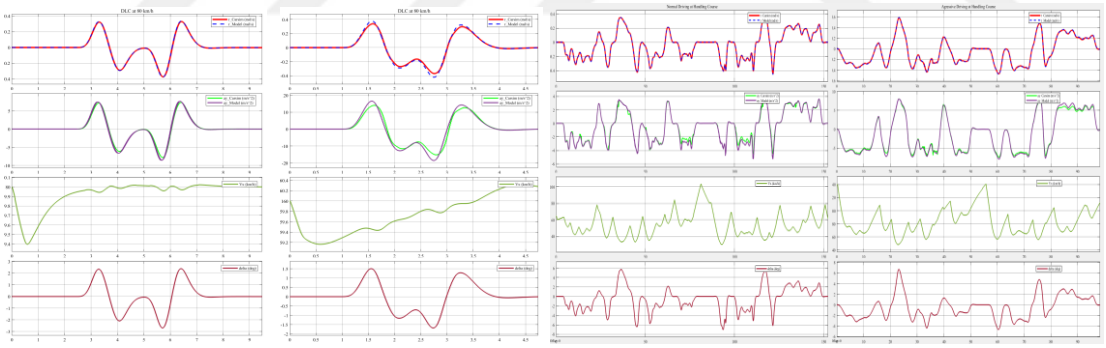
Re-writing Equations (2.6) - (2.8) as state space representation gives:

$$\begin{Bmatrix} \dot{y} \\ \ddot{y} \\ \dot{\psi} \\ \ddot{\psi} \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\left(\frac{2C_{\alpha,f}+2C_{\alpha,r}}{mV_x}\right) & 0 & -V_x-\left(\frac{2C_{\alpha,f}l_f-2C_{\alpha,r}l_r}{mV_x}\right) \\ 0 & 0 & 0 & 1 \\ 0 & -\left(\frac{2l_fC_{\alpha,f}-2l_rC_{\alpha,r}}{I_zV_x}\right) & 0 & -\left(\frac{2l_f^2C_{\alpha,f}+2l_r^2C_{\alpha,r}}{I_zV_x}\right) \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \\ \psi \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2C_{\alpha,f}}{m} \\ 0 \\ \frac{2l_fC_{\alpha,f}}{I_z} \end{bmatrix} [\delta] \tag{3.5}$$

Where, V_x is longitudinal velocity changing with time, $[y, \dot{y}, \psi, \dot{\psi}]^T$ is the state vector, δ is input. Changing longitudinal velocity makes the system Linear Time Varying (LTV) since system's A matrix change with time. Equations (3.3) - (3.4) become:

$$\dot{x} = A(t)x(t) + Bu(t) \tag{3.6}$$

Several tests applied to check LTV model's validity on Carsim with Simulink.



(a) **(b)**

Figure 3.8. Validation tests (a) Double lane change (DLC) (b) Handling course tests

It can clearly be concluded from Figure 3.8 developed model follows the high fidelity Carsim model's dynamics. This proves that found cornering stiffness values are true for prediction of states.

4. MODEL PREDICTIVE CONTROL

Model Predictive Control (MPC) has its roots since 1970s with the process industry where the dynamics of system change relatively slow in petrochemical industry. Slowly changing system dynamics allowed to solve optimization problems within a certain sampling period. Industrial applications to the MPC application investigated in [43], and summarized in [44]. With the increasing power of the processors and computers researchers are able develop nonlinear and linear model predictive controllers for highly changing dynamics systems like autonomous mobile robots, construction machines etc.

MPC is a state feedback controller first of all like Linear Quadratic Regulator (LQR). Different from the LQR; MPC consists of three main elements as shown in block diagram Figure 4.1. The *Optimizer* block finds the optimal control input for the plant by calculating the *Cost Function (J)*. This optimization is done within the defined *Constraints*.

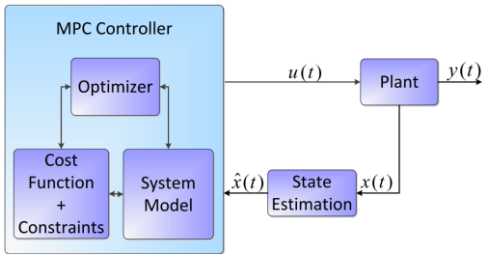


Figure 4.1. Model predictive control block diagram

MPC is based on the discrete system model. The system model shown in Figure 4.1 is used to predict system response within the "prediction horizon". The prediction output of the internal plant is compared with the real plant's output ($y(t)$) then using tracking error and control input ($u(t)$) an optimization problem is solved by calculating a cost function within desired constraints.

The very first term in optimal control input sequence found by the optimization results is applied to the real plant ($u(t)$), at the next interval prediction horizon is shifted and the optimization is repeated. Such control scheme is known as "Receding Horizon Control (RHC)" in control theory.

4.1. Model Predictive Control Types

There are several model predictive controllers depending on the plant model.

4.1.1. Linear time – invariant model predictive controller (LTI-MPC)

This type of MPC is the most simple and accurate since plant model does not change with time and constraints are assumed linear. The cost of function of MPC is quadratic, this satisfies the convex optimization problem in a single global minimum point.

4.1.2. Linear time variant model predictive controller (LTV-MPC)

If the plant model is nonlinear and changes with time, plant model must be linearized within several operating conditions. In this thesis linear time variant model predictive controller was used since the plant dynamics change with velocity, as given in Equations (2.6) - (2.11). There are two types of LTV-MPC in general:

- **Adaptive MPC:** If the optimization problem remains same across all operating conditions at time step k, we can simply approximate a linear model and use adaptive MPC which updates plant model recursively at time step k, when is operating conditions changed.
- **Gain Scheduled MPC:** When plant model is nonlinear and change at some operating conditions, this type of MPC switches the plant model used in optimization problem.

4.1.3. Nonlinear model predictive controller

If somehow it's not possible the linearize the plant model, the nonlinear equations of the model may be implemented in MPC directly. Although this type MPC is so powerful, it needs very high computational power in real time since it's difficult solve nonlinear optimization problem and finding the global minima of the convex optimization problem.

4.2. Model Definition

To find the optimum input MPC needs discrete system model. This system model must cover significant dynamics of the system also system should be simple to overcome the optimization in real time.

In this thesis a discrete system model is used. The state and output equations used in the simulations are given in (4.1) - (4.2) respectively.

$$x_{k+1} = A_k x_k + B_k u_k + E_k d_k \tag{4.1}$$

$$y_k = C_k x_k \tag{4.2}$$

Where, the subscript k indicates the discrete time step index, x_k indicates the state matrix at time step k, u_k is the control input, d_k is the disturbances affecting the system. Matrices $A_k, B_k, E_k, C_k$are; system matrix, input matrix, disturbance matrix and output matrix as well. Since there is no feedthrough input term affecting the output of system dynamics there is no D_k matrix in the state space representation.

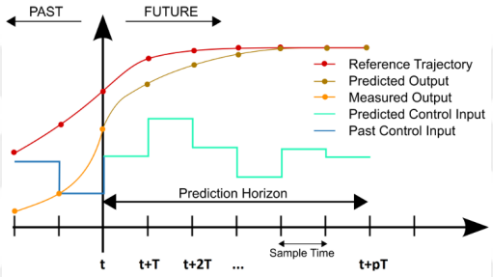


Figure 4.2. Model predictive control strategy

4.3. Quadratic Programming

As defined before MPC solves an optimization problem at each control interval, specifically a quadratic program. The solution determines the input variables to the system to be used in the next interval. To solve the optimization problem a cost function is needed. This cost function, *J*, is solved with quadratic programming under equality and inequality constraints, since solution of a quadratic program is convex the local minimum of the convex shape is selected for the decision of input to be used in the next interval.

A generic quadratic problem in MPC is defined as:

$$\underset{u}{minimize}\ J_N(x_o, u) = \sum_{k=0}^{N-1} \ell(x_k, u_k) \tag{4.3}$$

$$subject\ to:$$

$$
\begin{aligned}
&x_{(0)} = x_0, \\
&u_k \in U, \forall k \in [0, N-1], \\
&x_k \in X, \forall k \in [0, N], \\
&g(x_k, u_k) = 0, \\
&h(x_k, u_k) \geq 0
\end{aligned}
$$

N is the prediction horizon for the receding control, x_k is the state matrix at time step k, u_k is the input matrix at time step k, x_{k+1} equation defines the discrete system dynamics, $x_{(0)}$ is the initial states $g(x_k, u_k)$ is equality and $h(x_k, u_k)$ is inequality constraints as well.

The aim of the MPC is to find an input sequence such:

$$u^* = [u_k, u_{k+1}, u_{k+2}, \dots, u_{k+N-1}] \in R^{n_u N} \tag{4.4}$$

And the state response is:

$$x^* = [x_k, x_{k+1}, x_{k+2}, \dots, u_{k+N}] \in R^{n_x N} \tag{4.5}$$

Where, n_u is the number of inputs, N is the prediction horizon and n_x is the number of states. The quadratic cost is defined as $l(x_k, u_k)$ which is running (stage) cost function might be defined as:

$$l(x_k, u_k) = \frac{1}{2}\begin{bmatrix} x_k \\ u_k \end{bmatrix}\begin{bmatrix} Q_k & S_k^T \\ S_k & R_k \end{bmatrix}\begin{bmatrix} x_k \\ u_k \end{bmatrix} = \frac{1}{2}x_k^T Q_k x_k + \frac{1}{2}u_k^T R_k u_k \tag{4.6}$$

In (4.6); Q_k is positive semi definite weight matrix for states, R_k is positive semi definite weight matrix for inputs, S_k is cross weight matrix which is taken zero in this study.

4.4. State Predictions

After defining the cost function in (4.6) state predictions must be known via integrating the system dynamics given in (4.1) - (4.2). There are two approaches for predicting the states. These are “sparse” and “primal dense” formulations.

4.4.1. Sparse formulation

The state prediction with increasing time steps like $k, k+1, k+2, \dots, k+N$ can be written using system dynamics of the plant;

$$\begin{aligned} x_{k+1} &= A_k x_k + B_k u_k + E_k d_k \\ x_{k+2} &= A_{k+1} x_{k+1} + B_{k+1} u_{k+1} + E_{k+1} d_{k+1} \\ &= A_{k+1}(A_k x_k + B_k u_k + E_k d_k) + B_{k+1} u_{k+1} + E_{k+1} d_{k+1} \\ x_{k+3} &= A_{k+2} x_{k+2} + B_{k+2} u_{k+2} + E_{k+2} d_{k+2} \\ &= A_{k+2}(A_{k+1}(A_k x_k + B_k u_k + E_k d_k) + B_{k+1} u_{k+1} + E_{k+1} d_{k+1}) \\ &\quad + B_{k+2} u_{k+2} + E_{k+2} d_{k+2} \end{aligned} \tag{4.7}$$

Equation (4.7) can be summarized as follows:

$$x_{k+n} = A_{k+(n-1)} x_{k+(n-1)} + B_k u_{k+(n-1)} + E_k d_{k+(n-1)} \tag{4.8}$$

In the closed matrix form (4.8) can be re-written as;

$$\bar{A}\bar{x} + \bar{B}\bar{u} - \bar{b} = 0 \tag{4.9}$$

Where $\bar{A}$ and $\bar{B}$ describe the system dynamics $\bar{b}$ describes the influence of the initial states.

Writing (4.9) in detail:

$$\bar{A} = \begin{bmatrix} -I & & & \\ A_{k+1} & -I & & \\ & A_{k+2} & -I & \\ & & \ddots & \ddots \end{bmatrix}, \bar{B} = \begin{bmatrix} B_k & & & \\ & B_{k+1} & & \\ & & B_{k+2} & \\ & & & \ddots \end{bmatrix}$$
$$\bar{b} = \begin{bmatrix} -A_k x_k - E_k d_k \\ -E_{k+1} d_{k+1} \\ -E_{k+2} d_{k+2} \\ \dots \end{bmatrix} \tag{4.10}$$

Bringing the cost function in (4.3) together with (4.9) sparse formulation becomes complete with (4.11).

$$\begin{aligned} &\min_{\bar{x},\bar{u}} \left(\frac{1}{2}\bar{x}^T \bar{Q}\bar{x} + \bar{u}^T \bar{R}\bar{u}\right) \\ &subject\ to: \bar{A}\bar{x} + \bar{B}\bar{u} - \bar{b} = 0\ , \\ &b_{lb} \le A_{iq}[\bar{x}^T, \bar{u}^T] \le b_{ub}, \\ &\bar{x}_{lb} \le \bar{x} \le \bar{x}_{ub}, \\ &\bar{u}_{lb} \le \bar{u} \le \bar{u}_{ub} \end{aligned} \tag{4.11}$$

The subscript lb denotes lower bounds, ub denotes upper bound of the constraints. Subscript iq denotes inequality.

4.4.2. Primal dense formulation

In this formulation sequence of states are expressed and state variables are eliminated in (4.7).

$$\begin{aligned} \begin{bmatrix} x_{k+1} \\ x_{k+2} \\ x_{k+3} \\ \vdots \end{bmatrix} &= \begin{bmatrix} A_k \\ A_{k+1}A_k \\ A_{k+2}A_{k+1}A_k \\ \vdots \end{bmatrix} [x_k] \\ &+ \begin{bmatrix} B_k & & & \\ A_{k+1}B_k & B_{k+1} & & \\ A_{k+2}A_{k+1}B_k & A_{k+2}B_{k+1} & B_{k+2} & \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} u_k \\ u_{k+1} \\ u_{k+2} \\ \vdots \end{bmatrix} \\ &+ \begin{bmatrix} E_k & & & \\ A_{k+1}E_k & E_{k+1} & & \\ A_{k+2}A_{k+1}E_k & A_{k+2}E_{k+1} & E_{k+2} & \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} d_k \\ d_{k+1} \\ d_{k+2} \\ \vdots \end{bmatrix} \end{aligned} \tag{4.12}$$

Closed form of (4.12) can be written as:

$$[x^*] = A^* x_k + B^* u^* + E^* d^* \tag{4.13}$$

Where A^*, B^* and E^* defines the dynamics of the system and supplies predictions. Diagonal of B^* matrix, shows the input matrix from B_k to $B_{k+(N-1)}$, where $k \in \{1,2,3, \dots, N\}$. The elements under the diagonal of B^* matrix can be obtained recursively. Since the disturbance can be treated as input to the system matrix E^* can be treated as

same as matrix B^*. Advance of initial conditions in A matrix can be obtained by recursive left multiplication, starting with A_k on the first row and left multiplying the row above A_{k+j-1} at the row j.

Plugging (4.12) for the states into (4.3) gives the final form of cost function becomes:

$$J(u^*) = \frac{1}{2}(B^*u^* + A^*x_k + E^*d^*)Q^*(B^*u^* + A^*x_k + E^*d^*) + \frac{1}{2}u^*R^*u^* \qquad (4.14)$$

The final quadratic programming problem becomes:

$$\begin{aligned} &\min_{u^*}\left(\frac{1}{2}u^{*T}H^*u^* + g^{*T}u^*\right) \\ &subject\ to: \\ &b_{lb} \le B^*u^* \le b_{ub}, \\ &u^*_{lb} \le u^* \le u^*_{ub} \end{aligned} \qquad (4.15)$$

Where,

$$H^* = B^{*T}Q^*B^*, g^* = B^{*T}Q^*[A^*E^*]\begin{bmatrix} x_k \\ d^* \end{bmatrix} \qquad (4.16)$$

4.5. Algorithms for Solving Constrained MPC

There are three main methods for solving quadratic problem defined in (4.3). These methods are known as "Active Set Method, Interior Point Method, Gradient Projection Method" in the literature [45] and [46].

4.5.1. Interior Point Method

A barrier function exists in the objective function when the interior point method selected as optimization solving algorithm to prevent constraint violation, in general a logarithmic barrier function is used [47].

4.5.2. Gradient Descent Method

As the name suggests, the idea behind gradient descent algorithm is to take repeated steps in the opposite direction of the gradient to find local minima of the optimization function.

Algorithm's working principle and detailed information can be found in [48] and its references.

4.5.3. Active Set Method

Active set method solves a sequence of equality constrained quadratic sub problems [49]. In this method the goal is predicting the active set which is the equality constraints at each iteration.

4.6. Quadratic Problem Solvers

This section gives brief introductory information about solvers for model predictive control.

In this thesis, "active set" solvers were selected since it is directly integrated within Matlab/Simulink with Model Predictive Controller Toolbox [50]. One might choose FORCESPRO solver instead the default solvers for lower solving times of optimization problem but since FORCESPRO is a commercial software, Matlab's default solvers were selected.

- **FORCESPRO:** A high speed solver for; model predictive control, embedded optimization, decision making, path planning and fully autonomous system problems. Integrated in Matlab but, it needs paid license for usage [51].
- **FiOrdOs:** A Matlab toolbox written in C, for the parametric convex programs with quadratic cost functions [52].
- **CasADi**: An algorithmic differentiation tool for nonlinear optimization. It facilitates rapid and efficient code. Supports Matlab, C and Python, also can be plugged into Simulink by writing Matlab System Functions. The symbolic representation makes coding and preparing the optimization problem easier [53].
- **ACADO:** A toolkit for automatic control and dynamic optimization. It's used in model predictive control, state and parameter estimation and robust optimization in general. Supports C++ and Matlab [54].
- **qpOASES:** This solver extends the capabilities of active set algorithm with parametrization then implements parametric active set method for solving quadratic problems [55].

5. TRAJECTORY GENERATION

In this thesis, racing line trajectory tracking of a race car along a race track is taken into account. Selected race car parameters are given in Appendix A.

To follow a racing line trajectory there are two main parameters to follow:

- Optimum vehicle velocity along the race track ($V_x(t)$) for one lap, shown in Figure 5.1.

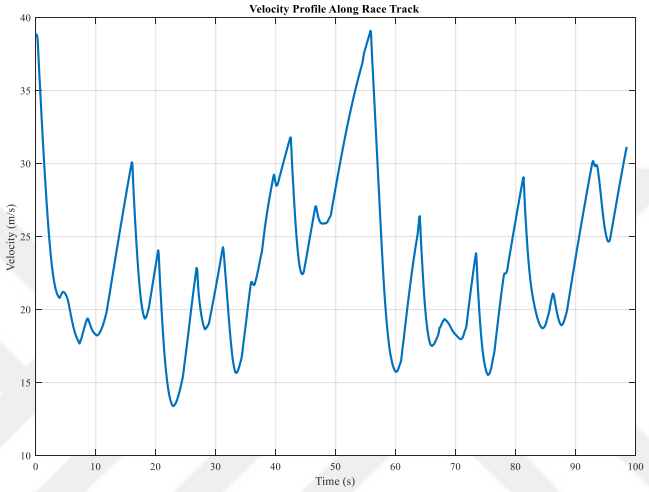


Figure 5.1. Desired velocity profile along the race track

- Race track curvature profile with time ($X(t), Y(t)\ or\ \kappa(t)$) where $\kappa(t)$ is curvature, $X(t), Y(t)$ are the reference coordinates.

Figure 5.2. Race track with desired racing line

For a given subset of X and Y coordinates curvature profile can be determined if this subset is a continuous curve.

Let P_i is the position coordinates consisting X_i, Y_i coordinates and N is the length of coordinate subset. Subscript i shows the i^{th} index of the subset.

$$P_i = [X_i \quad Y_i],$$

$$A_k = \frac{[(X_i - X_{i-1}) * (Y_{i+1} - Y_{i-1}) - (Y_i - Y_{i-1}) * (X_{i+1} - X_{i-1})]}{2} \tag{5.1}$$

$$\kappa_i = \frac{4 * A_k}{\left||P_{i-1} - P_i|\right| \times \left||P_i - P_{i+1}|\right| \times \left||P_{i+1} - P_{i-1}|\right|}$$

Where $i = \{2, 3, \dots, N-1\}, k = \{1, 2, \dots, N\}$, N is the subset length and κ is the curvature of the road profile.

For the given subset of X and Y coordinates shown in Figure 5.2 curvature profile of the road was found as;

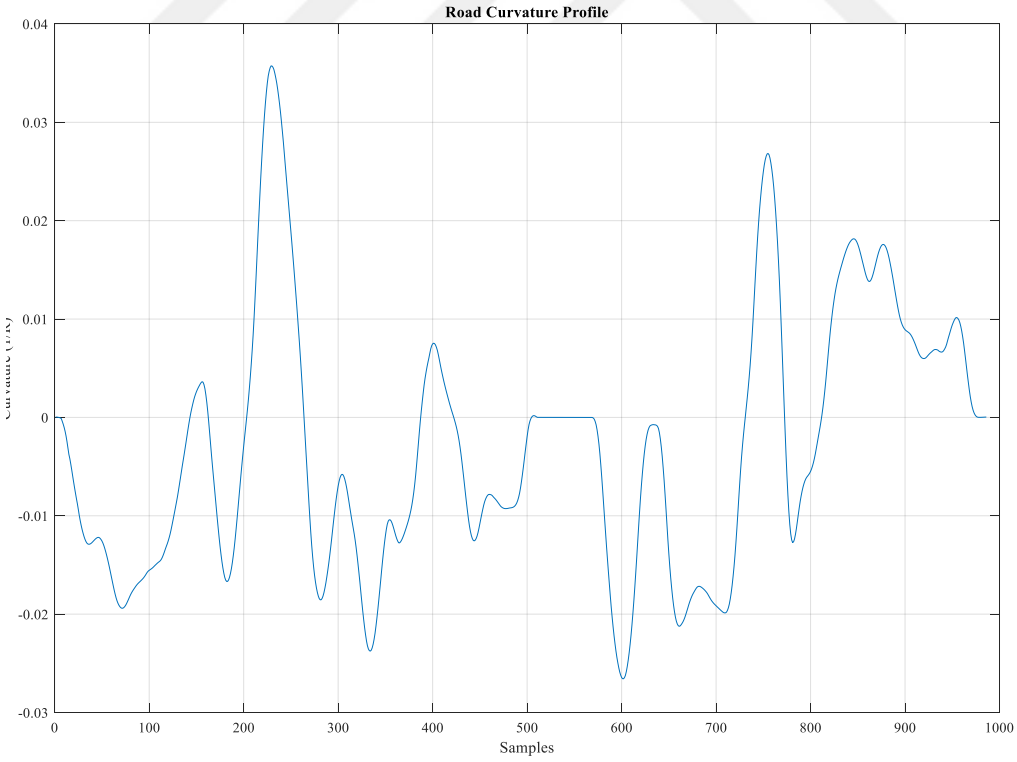


Figure 5.3. Road curvature profile

6. LONGITUDINAL CONTROL

In this chapter a longitudinal driver model is developed. The driver model behaves like the vehicle velocity controller in the control scheme. Developed driver model will be called as "velocity controller or controller" from now on in this chapter.

The controller is based on MacAdam model [56]–[58]. The goal in this section is create a controller that which behaves like a driver in the seat.

6.1. Control Problem Formulation

In the MacAdam model an optimal single point preview, a look ahead is used. The model presents a driver in the seat with normalized throttle and braking signals to achieve velocity tracking.

The controller;

- Represents the driver behavior with a preview point
- Minimizes the previewed error signal at T^*seconds ahead in time.
- Accounts the driver/controller lag occurring from the signal delays [59].

The predictive model used in MacAdam model [57] as follows:

$$\begin{aligned} \dot{x} &= Fx + gu \\ y &= m^T x \end{aligned} \tag{6.1}$$

Where, x is $n \times 1$ state vector, y is scalar output related to the state with the $n \times 1 m^T$, m^T is transpose of the constant observer vector, F is constant $n \times n$ system matrix, g is constant $n \times 1$ control matrix.

To find optimal control u^o, the local cost function J appears as in Equation (6.5) over the current preview interval $(t, t + T^*)$ where T^* is preview ahead in time:

$$J \triangleq \frac{1}{T} \int_t^{t+T^*} \{[f(\eta) - y(\eta)]W(\eta - t)\}^2 d\eta \tag{6.2}$$

W is weight function and f is previewed input for Equation (6.5). The optimal control input, u^o, can be found using:

$$u^o(t) = \frac{f(t+T^*) - m^T\left[I + \sum_{n=1}^{\infty}\left(\frac{F^n(T^*)^n}{n!}\right)\right]}{T^* m^T\left[I + \sum_{n=1}^{\infty}\left(\frac{F^n(T^*)^n}{(n+1)!}\right)\right]g} \tag{6.3}$$

In the compact form of (6.3):

$$u^o(t) = \frac{[f(t+T^*) - y_o(t+T^*)]}{T^* b^*} \tag{6.4}$$

Where $b^* = m^T\left[I + \sum_{n=1}^{\infty}\left(\frac{F^n(T^*)^n}{(n+1)!}\right)\right]g$ in (6.3).

In our application, which is velocity tracking, the following problem formulation is used.

$$\dot{x} = Fx + \bar{g}\bar{u} \tag{6.5}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, x_1 = v, x_2 = \dot{x_1}, F = \begin{bmatrix} 0 & 1 \\ \frac{F_r}{m} & 0 \end{bmatrix}, \bar{g} = \begin{bmatrix} 0 \\ \frac{K_{pt}}{m} \end{bmatrix}$$

$$\bar{u} = u - \frac{m^2}{K_{pt}} g sin(\gamma) \tag{6.6}$$

$$F_r = -[\tanh(x_1)\left(\frac{a_r}{x_1} + c_r x_1\right) + b_r]$$

Equation (6.5) and (6.2) defines the system dynamics with Equation (6.6). x shows the states of the system, v defines the velocity of the vehicle, x_2 defines the acceleration, F is the system matrix and $\bar{g}$ is the input matrix. K_{pt} is the total powertrain force occurred in center of gravity of the vehicle in Newtons (N). g, γ are gravitational acceleration (m/s^2) and road grade in radians (rad). m is the total vehicle mass in kilograms (kg). a_r, b_r and c_r are static rolling & driveline resistance (N), linear rolling and driveline resistance (Ns/m) and aerodynamics driveline resistance (Ns^2/m^2) coefficients respectively.

6.2. Controller Structure

Equation (6.3) behaves like proportional controller, with gain inversely related to preview interval T^* and error between previewed input and output. In this study, T^* is calculated with respect to velocity of the vehicle and preview distance decided by user.

$$T^* = \frac{L}{U}, m^T = [1 \quad 1] \tag{6.7}$$

Where, L is the preview distance and U is the longitudinal velocity of the vehicle and m^T is the constant observer vector as stated before.

To simplify (6.3) can be reduced to;

$$u^o(t) = u(t) + \frac{e(t + T^*)}{a^*} \tag{6.8}$$

Where $a^* = T^* b^*$ and $e(t + T^*) = f(t + T^*) - y_o(t + T^*)$ which is error signal in previewed time.

To find the optimal input in (6.4), following values in the Table 6.1 are used.

Table 6.1. Velocity controller parameters

Parameter	Value	Unit
Vehicle mass, m	1200	kg
Effective vehicle total tractive force, K_{pt}	38500	N
Preview distance, L	2	m
Rolling resistance coefficient, a_r	0.10	N
Rolling and driveline resistance coefficient, b_r	2.25	Ns/m
Aerodynamic drag coefficient, c_r	0.8	Ns^2/m^2
Gravitational constant, g	9.81	m/s^2

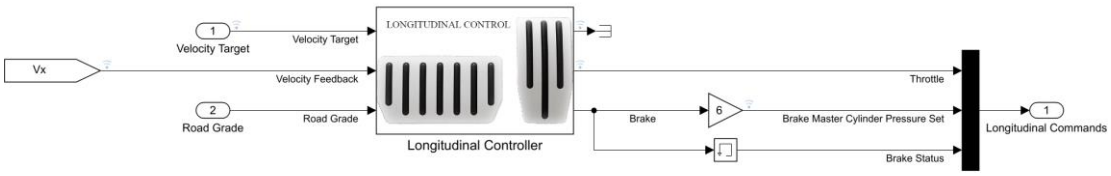


Figure 6.1. Longitudinal control scheme

In Figure 6.1 inputs to the controller are current velocity, target velocity and road grade data. Target velocity was the aggressive driving velocity profile and explained before in Section 5, road grade data was taken from Carsim directly. In reality road grade or road pitch data can be measured by using inertial measurement units (IMU) [60].

Inside the *longitudinal controller* block in Figure 6.1, control algorithm explained in (6.1) - (6.8). In the Figure 6.2, Setup block calculates the F, G, a^* and b^* for calculation of system dynamics model, $\dot{x}$, in real time to find optimal control input $u^o(t)$.

Accel Override and *Decel Override* blocks normalize the incoming signals within [0,1] range and creates a decision logic switch so that acceleration signal and deceleration should not work at the same time to prevent any acceleration command and deceleration command occurring at the same time.

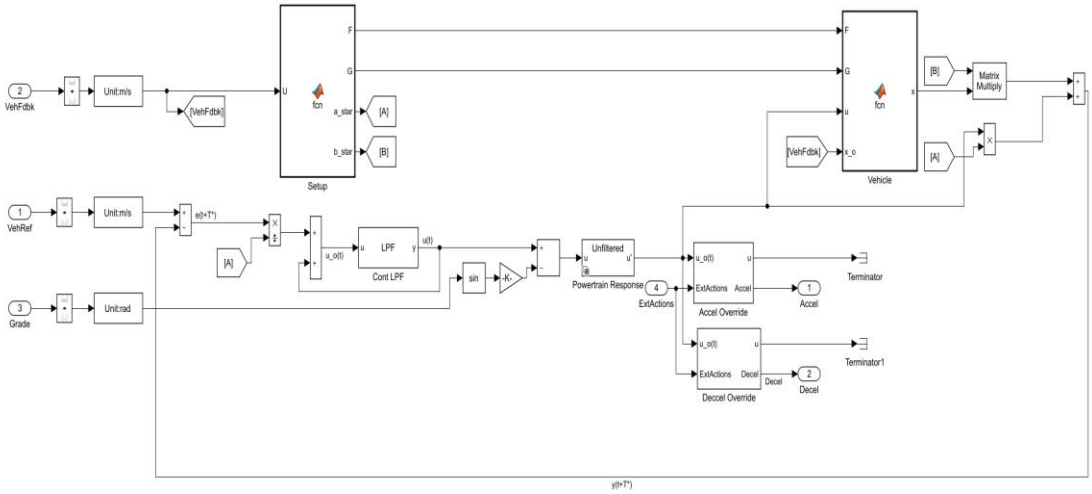
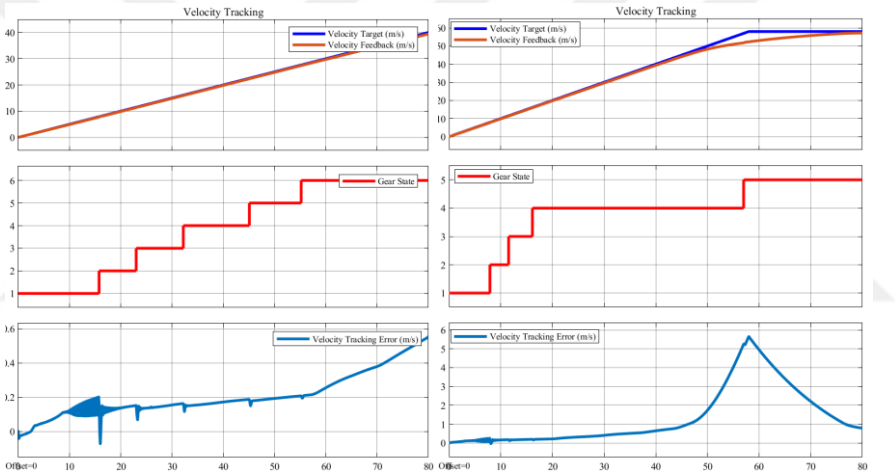


Figure 6.2. Optimal preview controller for longitudinal tracking

6.3. Controller Performance Tests

In this section performance of the longitudinal controller tested within different ramp functions in a straight road shown in and the racing profile mentioned in Section 5.

Figure 6.3. Longitudinal controller tests – Driving area



(a) **(b)**

Figure 6.4. (a) Velocity tracking with 0.5 ramp function, (b) Velocity tracking with 0.5 ramp function

Figure 6.4.a. shows if the velocity of the vehicle under 10 m/s the controller is struggling very little to keep the velocity with the desired value. Figure 6.4.b shows that the vehicle longitudinal velocity saturates at 60 m/s, which means vehicle is at limit velocity at for about 210 km/hr.

In order to see the performance of the controller a combined velocity profile was consisted from linear increasing & decreasing profiles. According to the Figure 6.5 the velocity errors found as between -0.5 and 0.5 levels. The velocity verification profile generated by using Table 6.2.

Table 6.2. Velocity verification profile

Time (s)	Velocity (m/s)	Time (s)	Velocity (m/s)
0	4	**40**	35
10	10	**45**	10
15	20	**50**	4
20	25		

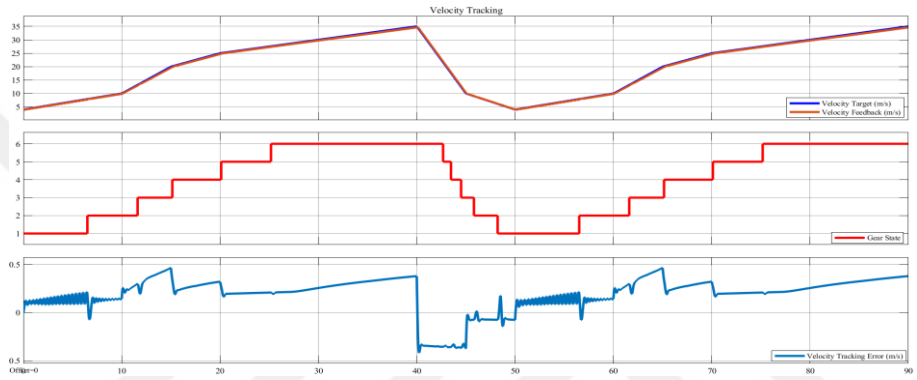


Figure 6.5. Combined velocity profile

To increase complexity of the velocity profile, aggressive driving velocity profile in Section 5 was used.

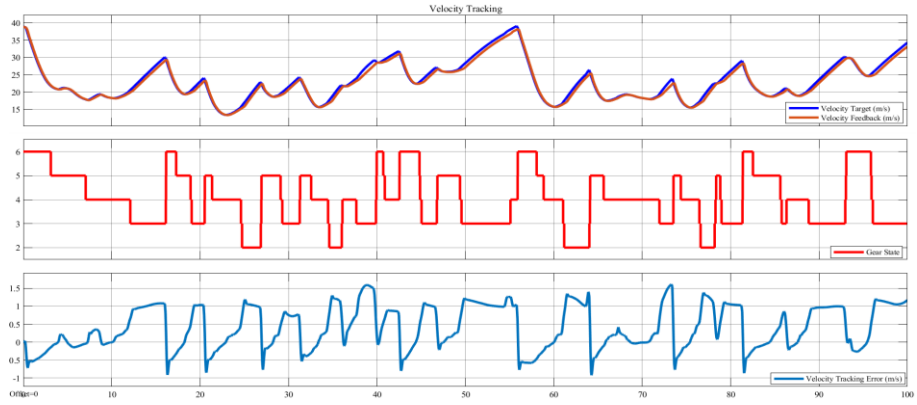


Figure 6.6. Longitudinal controller performance with aggressive driving profile

Finally, Figure 6.6 shows, the error levels are between -1 and +1 during the aggressive driving profile. It can be concluded figures from Figure 6.4 - Figure 6.6 the derived longitudinal driver model/controller follows the given trajectory profile with satisfactory limits.

7. LATERAL CONTROL

In this section dynamics of the system is re-formulated in terms of error with respect to desired path. Developed control system's stability, controllability and observability analyses were taken into account. Lastly, controller definition and tuning of the lateral controller were given.

7.1. Control Problem Formulation

The objective is to develop a steering controller that follows the reference path closely and keeps the velocity vector, v, tangent to the reference profile. When developing this controller ride comfort and jerk (derivative of lateral acceleration, $\frac{d}{dt}\ddot{y}$) was not considered since the vehicle assumed fully autonomous and there is no driver in it.

Consider a vehicle tries to make steady state cornering in Figure 7.1. The orthogonal distance from center of gravity of the vehicle and the path defines the "cross track error". The difference in yaw angle also defines the "yaw error" in a path tracking problem.

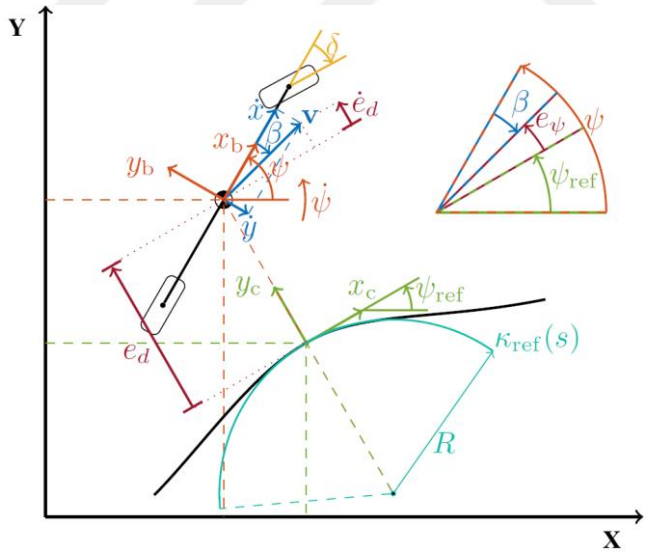


Figure 7.1. Vehicle motion in steady state cornering

$$\dot{\psi}_{des} = \frac{\dot{x}}{R}$$
$$\kappa = 1/R \tag{7.1}$$

In (7.1), ψ_{des} defines the desired yaw rate of a vehicle in steady state cornering, κ defines road curvature and R is assumed constant corner radius.

Defining new two new error variables "cross track error" and "yaw error" lateral with e_d and e_{yaw} gives following equations:

$$\begin{aligned} \ddot{e_d} &= \left(\ddot{y} + \dot{x}\dot{\psi}\right) - \frac{\dot{x}^2}{R}, \\ e_{yaw} &= \psi - \psi_{des} \end{aligned} \tag{7.2}$$

Re-defining e_d with e_1 and e_{yaw} with e_2 for easiness in state space formulation, (7.1) becomes:

$$\begin{aligned} \ddot{e_1} &= \left(\ddot{y} + \dot{x}\dot{\psi}\right) - \frac{\dot{x}^2}{R}, \\ e_2 &= \psi - \psi_{des} \end{aligned} \tag{7.3}$$

Integrating $\ddot{e_1}$ gives:

$$\dot{e_1} = \dot{y} + \dot{x}(\psi - \psi_{des}) \tag{7.4}$$

In (7.4) the longitudinal velocity, $\dot{x}$, assumed constant over time. However, in most general conditions the velocity of the vehicle changes with time. With the changing velocity (7.4) can be re-formulated as:

$$\dot{e_1} = \dot{y} + \int \dot{x}\, e_2 dt \tag{7.5}$$

Equation (7.5) brings non-linearity to control problem formulation and may be not useful in some conditions especially when the computation source is low.

Substituting Equations (7.3) - (7.4) into (2.7) gives:

$$m\dot{e_1} = \dot{e_1}\left[-\frac{2}{\dot{x}}C_{\alpha,f} - -\frac{2}{\dot{x}}C_{\alpha,r}\right] + e_2\left[2C_{\alpha,f} + 2C_{\alpha,r}\right] + \dot{e_2}\left[-\frac{2C_{\alpha,f}l_f}{\dot{x}} + \frac{2C_{\alpha,r}l_r}{\dot{x}}\right] + \dot{\psi_{des}}\left[-\frac{2C_{\alpha,f}l_f}{\dot{x}} + \frac{2C_{\alpha,r}l_r}{\dot{x}}\right] + 2C_{\alpha,f}\delta \tag{7.6}$$

$$I_z\ddot{e_2} = 2C_{\alpha,f}l_f\delta + \dot{e_1}\left[-\frac{2C_{\alpha,f}l_f}{\dot{x}} + \frac{2C_{\alpha,r}l_r}{\dot{x}}\right] + e_2\left[2C_{\alpha,f}l_f + 2C_{\alpha,r}l_r\right] + \dot{e_2}\left[-\frac{2C_{\alpha,f}{l_f}^2}{\dot{x}} - \frac{2C_{\alpha,r}{l_r}^2}{\dot{x}}\right] - I_z\ddot{\psi_{des}} + \dot{\psi_{des}}\left[-\frac{2C_{\alpha,f}{l_f}^2}{\dot{x}} - \frac{2C_{\alpha,r}{l_r}^2}{\dot{x}}\right] \tag{7.7}$$

Finalizing Equations (7.6) and (7.7) with state space representation:

$$\frac{d}{dt}\begin{Bmatrix} e_1 \\ \dot{e_1} \\ e_2 \\ \dot{e_2} \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\left(\frac{2C_{\alpha,f}+2C_{\alpha,r}}{m\dot{x}}\right) & \frac{2C_{\alpha,f}+2C_{\alpha,r}}{m} & \left(\frac{-2C_{\alpha,f}l_f-2C_{\alpha,r}l_r}{m\dot{x}}\right) \\ 0 & 0 & 0 & 1 \\ 0 & -\left(\frac{2l_fC_{\alpha,f}-2l_rC_{\alpha,r}}{I_z\dot{x}}\right) & \left(\frac{2l_fC_{\alpha,f}-2l_rC_{\alpha,r}}{I_z}\right) & -\left(\frac{2l_f^2C_{\alpha,f}+2l_r^2C_{\alpha,r}}{I_z\dot{x}}\right) \end{bmatrix}\begin{bmatrix} e_1 \\ \dot{e_1} \\ e_2 \\ \dot{e_2} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2C_{\alpha,f}}{m} \\ 0 \\ \frac{2l_fC_{\alpha,f}}{I_z} \end{bmatrix}[\delta] + \begin{bmatrix} 0 \\ -\frac{2l_fC_{\alpha,f}-2l_rC_{\alpha,r}}{m\dot{x}} \\ 0 \\ -\frac{2l_f^2C_{\alpha,f}+2l_r^2C_{\alpha,r}}{I_z\dot{x}} \end{bmatrix}\dot{\psi_{des}} \tag{7.8}$$

Equation (7.8) can be written in closed form for better understanding as:

$$\dot{x} = A(t)x(t) + Bu(t) + E(t)d(t), \qquad y = [1 \quad 0 \quad 0 \quad 0] \times x(t) \tag{7.9}$$

Where, $\dot{x}$ shows the error dynamics, $A(t)$ shows the changing system matrix with respect to vehicle velocity, B is input matrix, $u(t)$ is steering angle, $E(t)$ is disturbance matrix depending on velocity, $d(t)$ is the desired yaw rate.

7.2. Stability, Controllability and Observability

This section introduces the stability, controllability and observability assessment of the trajectory tracking control problem defined in (7.8).

7.2.1. Stability

To define stability, there exists three different approaches:

- A system is accepted as stable if the natural response approaches zero as the time goes to infinity,
- A system is accepted as unstable if the natural response grows to infinity as the time goes to infinity,
- A system is marginally stable if the natural response of the system neither decays nor grows, but oscillating within a threshold as the time goes to infinity.

To determine a system is stable or not, one must review the poles of the system if the system defined as transfer functions or modelled in frequency domain. If the system is modeled in time domain like using state space representation as in (7.8) the eigenvalues of the A matrix should be investigated.

To create a decaying natural response as the time goes to infinity, system's poles or eigenvalues must have negative real parts. That is, stable systems have to poles in the left half plane in pole – zero map. Since poles with positive real parts, in right half plane, create growing response. This results the conclusion that the investigated system is unstable.

If a linear dynamic system has the form of,

$$\frac{dx}{dt} = Ax, x(0) = x_0 \tag{7.10}$$

Where $A \in R^{n\times n}$ is a square matrix, corresponding to the system transition in (7.8). The stability can be determined by the eigenvalues of the matrix A as,

$$\lambda(A) = \{s \in C: \det(sI - A) = 0 \tag{7.11}$$

Where λ denotes the eigenvalue subset, λ_j denotes jth eigenvalue of the system.

To investigate stability of the lateral tracking problem defined in (7.8), eigenvalues of the A matrix were found as a function of velocity since A matrix changes with longitudinal velocity with time.

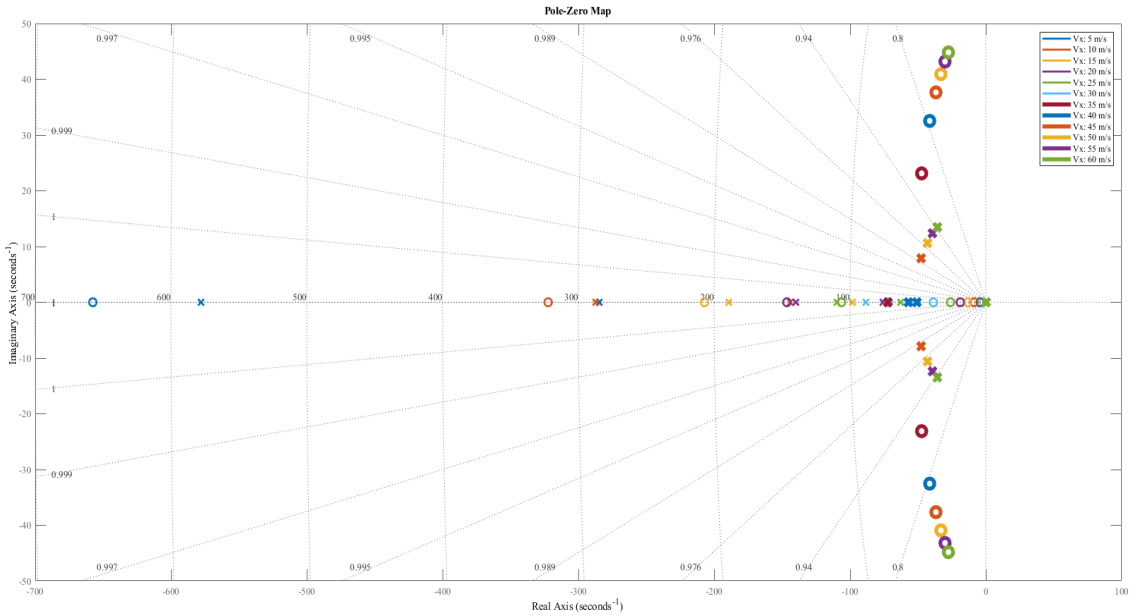


Figure 7.2. Pole - zero map of controlled system

Figure 7.2 shows that with the increasing velocity up to 60 m/s which is defined as the saturation velocity for the vehicle in Section 6, poles become more close to the origin, this results poles become more dominant.

At each velocity level for the vehicle up to 60 m/s, there exists two poles at origin and after 40 m/s, the other two poles for the fourth order system become complex conjugate with negative real parts.

As a conclusion, this system is unstable since there exists two poles at the origin, at each velocity. To see if the system is controllable one must check the controllability matrix.

7.2.2. Controllability

With the feedback control or optimal control as used in this thesis, controllability is an important property of a system dynamics.

A system is controllable if the "controllability matrix", C, has full rank. Defining C as in (7.12):

$$C = [B\ AB\ A^2B\ \dots A^{n-1}B] \tag{7.12}$$

Where, A is the $n \times n$ system matrix, B is the $n \times m$ input matrix, n is the number of rows and m is number of columns.

Since the dynamics of the trajectory tracking problem aforementioned in (7.8) changes with longitudinal velocity, controllability analysis should be done for all applicable velocity ranges. In this case, cluster of the controllability matrices as holds:

$$C_j = [B\ A_jB\ A_j^2B\ \dots A_j^{n-1}B] \tag{7.13}$$

Where subscript j denotes the velocity and $j = (1,2,3,\dots,60)$ since the velocity saturates at 60 m/s.

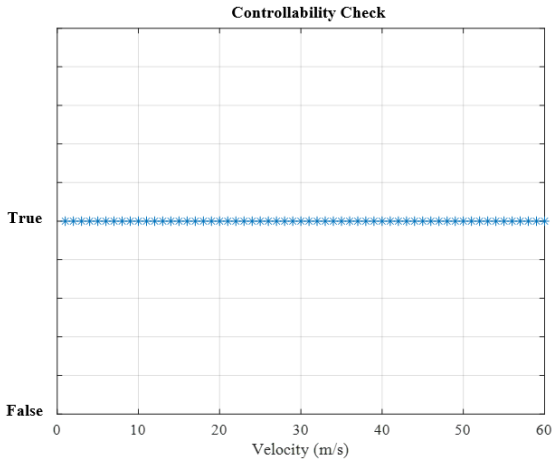


Figure 7.3. Controllability check

Figure 7.3 shows that at all velocities, system is controllable with feedback control laws.

7.2.3. Observability

A system is observable if it is possible to determine the state $x(t_0)$ at initial from observation of the output over a finite time interval.

A system is observable if the "observability matrix", O, has full rank. Defining O as in (7.14):

$$O = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{7.14}$$

Where, A is the $n \times n$ system matrix, C is the $p \times m$ output matrix.

Since the dynamics of the trajectory tracking problem aforementioned in (7.8) changes with longitudinal velocity, controllability analysis should be done for all applicable velocity ranges. In this case, cluster of the observability matrices as holds:

$$O_j = \begin{bmatrix} C \\ CA_j \\ CA_j^2 \\ \vdots \\ CA_j^{n-1} \end{bmatrix} \tag{7.15}$$

Where subscript j denotes the velocity and $j = (1,2,3,\dots,60)$ since the velocity saturates at 60 m/s. When the rank of observability matrix was checked it was seen that controlled system was observable for each velocity as shown in Figure 7.4.

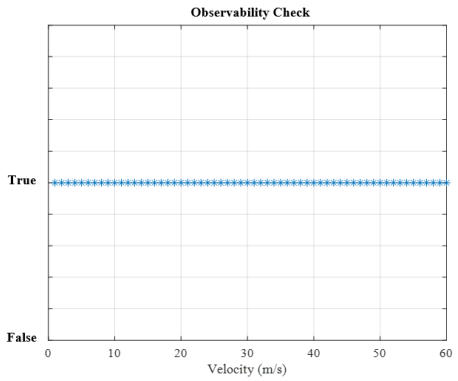


Figure 7.4. Observability check

7.3. Road Curvature Estimation with Feedforward Neural Network

Neural networks are biologically inspired artificially created in any computer, models that takes place of decision mechanisms in brain. Created neural networks may be used in semantic segmentation, feature extraction, lane or sign detection in autonomous vehicles in general.

There exists not only image based neural networks as mentioned above but also neural networks exist, which behaves like a system model. For example in [61] slip angle, β, of the vehicle is estimated by neural networks. Authors of [62] modelled the longitudinal dynamics of the vehicle by using velocity, torque, brake pressure and the grade angle of the vehicle.

In this section the road curvature, κ, in (7.1) estimated with a feedforward neural network, because when re-writing Equation (7.1) as in (7.16) gives the curvature, κ:

$$\dot{\psi}_{des} = \frac{\dot{x}}{R} \rightarrow \dot{\psi}_{des} = \dot{x}\kappa \tag{7.16}$$

In (7.9) state space form shows that the curvature can be used as measured disturbance. Since knowing an external disturbance input would increase the controller performance, a feedforward neural network was created in Matlab with "*nntool*" function to estimate as measured disturbance which consists from 1 input layer $\in R^2$, 2 hidden layers with $\in R^{10}, R^{20}$ and 1 output layer with $\in R^1$. Created neural network scheme is shown in Figure 7.5.

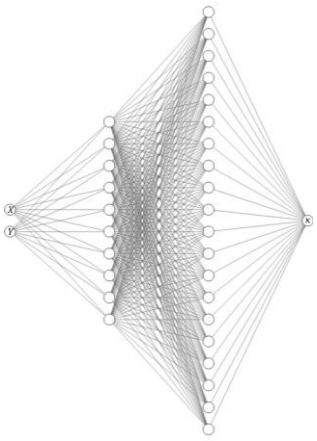


Figure 7.5. Proposed neural network scheme

For the training of network shown in Figure 7.5, properties of the training should be set up very well. In Table 7.1 these properties are given:

Table 7.1. Velocity controller parameters

Property	Description
Input data	X, Y coordinates of the racing line
Output data	Curvature, k @ $[X, Y]$
Training method	Levenberg-Marquardt
Number of layers	3
Layer activation function	Sigmoid, Sigmoid, Pure Linear

After training, best training performance which is the minimum mean squared error, found at epoch 120 and regression results show that neural network output data is in accordance with training data as shown in Figure 7.6.

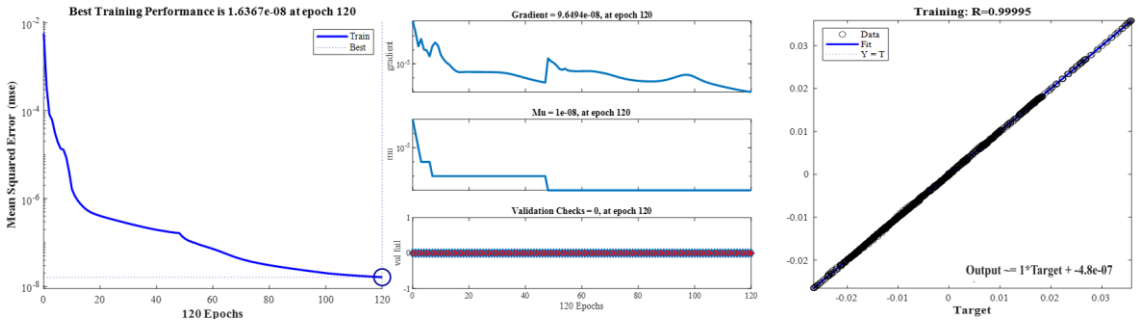


Figure 7.6. Curvature estimation neural network performance

Neural network estimated curvature and the real values with calculated errors are given in Figure 7.7.

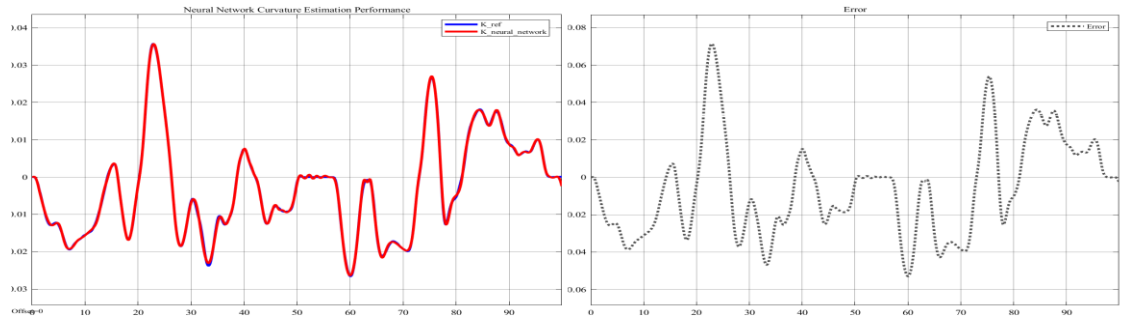


Figure 7.7. Neural network performance in real time

7.4. Controller Definition & Tuning

Firstly, the proposed LTV-MPC controller algorithm shown in Figure 7.8. The algorithm below here describes the lateral model predictive controller. The turquoise describes the model update function, since our model changes within time and orange block describes the linear time varying model predictive controller. The reference value for the controller was given as “0” since desired value for the lateral deviation is “0”.

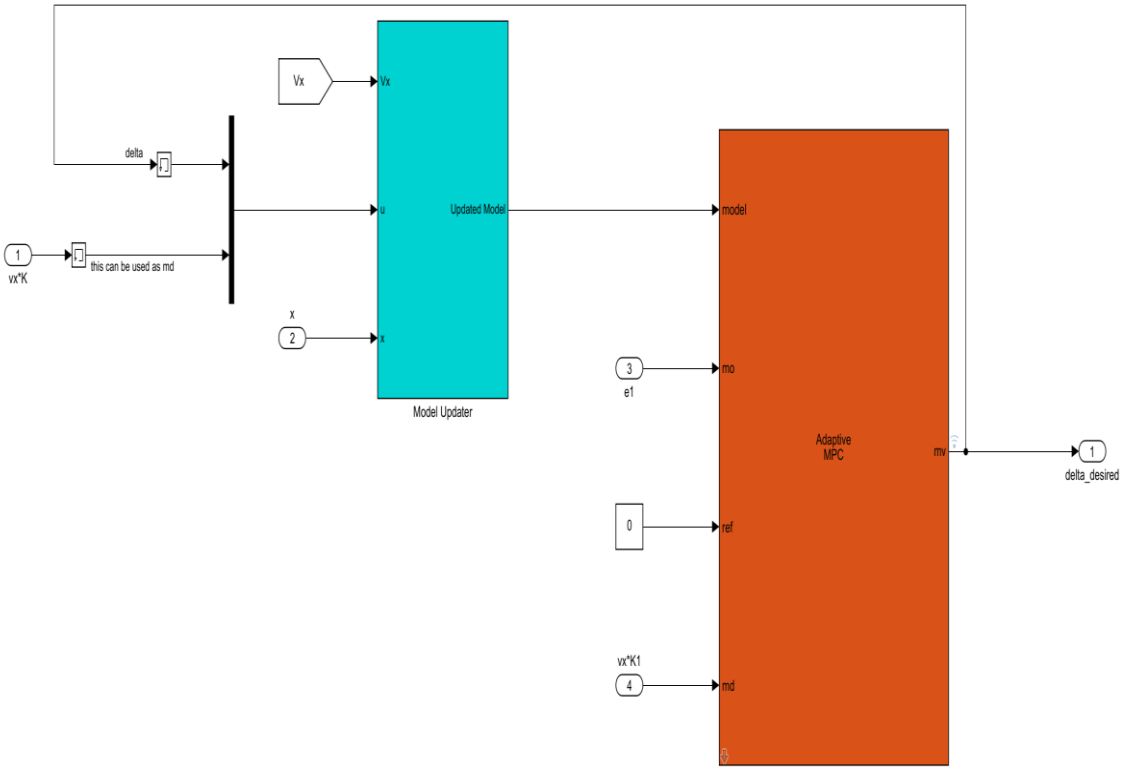


Figure 7.8. Lateral control algorithm

Secondly the changing dynamics of the system given in Section 7.1, internal plant model of the MPC controller has to be updated in each sample time. To do so, the model has to be discretized with a desired sampling time (T_s) and a constant longitudinal velocity ($\dot{x}$).

Continues time domain system with velocity of the vehicle held constant at 80 km/h becomes:

$$\frac{d}{dt}\begin{Bmatrix} e_1 \\ \dot{e_1} \\ e_2 \\ \dot{e_2} \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -66.89 & 1487 & 13.27 \\ 0 & 0 & 0 & 1 \\ 0 & 14.95 & -332.2 & -127.1 \end{bmatrix} \begin{bmatrix} e_1 \\ \dot{e_1} \\ e_2 \\ \dot{e_2} \end{bmatrix} + \begin{bmatrix} 0 \\ 559.1 \\ 0 \\ 916.4 \end{bmatrix} [\delta] + \begin{bmatrix} 0 \\ -7.272 \\ 0 \\ -127.1 \end{bmatrix} \dot{\psi_{des}} \tag{7.17}$$

Discretizing the system in (7.17), with 0.1 sampling time in Matlab with "*c2d*" function gives (7.18).

$$\begin{Bmatrix} e_1 \\ \dot{e_1} \\ e_2 \\ \dot{e_2} \end{Bmatrix}_{k+1} = \begin{bmatrix} 1 & 0.01769 & 1.829 & 0.01488 \\ 0 & 0.0394 & 21.35 & 0.1719 \\ 0 & 0.001727 & 0.9616 & 0.007744 \\ 0 & 0.0002644 & -0.005875 & -3.705e-05 \end{bmatrix} \begin{bmatrix} e_1 \\ \dot{e_1} \\ e_2 \\ \dot{e_2} \end{bmatrix}_k + \begin{bmatrix} 1.375 \\ 23.52 \\ 0.7313 \\ 916.4 \end{bmatrix} [\delta]_k + \begin{bmatrix} -0.09392 \\ -2.021 \\ -0.09203 \\ 8.062 \end{bmatrix} \dot{\psi}_{des_k} \tag{7.18}$$

Since the predictive controller has its initial internal plant model, to convert it LTV-MPC as detailed in Section 4.1.2, plant model has to be updated in each sampling time in a discrete way. To accomplish this a successive linearizer was modeled in order to complete plant model update objective.

Figure 7.9 shows the "successive linearizer", in Figure 7.8 the turquoise block, "Model Updater", represented in Figure 7.8 updates system model and finds "A, B, C, D, U, X, Y, DX" signals at each sampling time, where "A" describes the system matrix, "B" describes the input matrix including the disturbances, "C" corresponds to output matrix, "D" is zero matrix since there is no feedthrough input affecting the system output, "X" is system states, "Y" is current output and "DX" is the state difference matrix as defined in Equation (7.19).

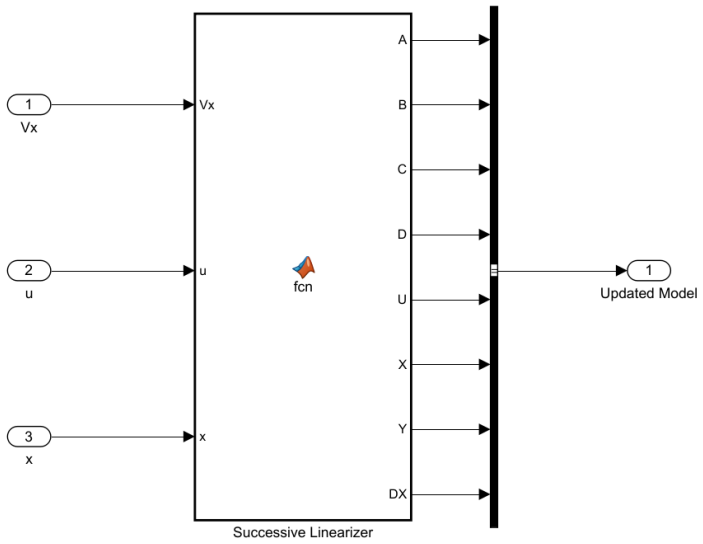


Figure 7.9. Succesive linearizer

$$x_{k+1} = x_k + DX \rightarrow x_{k+1} = f(x_k, u_k)$$
$$DX = f(x_k, u_k) - x_k \tag{7.19}$$

With four states of the internal plant discrete model, predictive controller was designed with the prediction horizon $N = 10$, control horizon $N_c = 3$, steering angle contstraints with $\delta_{max} = 1\ rad, \delta_{min} = -1\ rad, \dot{\delta}_{max} = 0.2\ rad, \dot{\delta}_{min} = -0.2\ rad$, steering input rate of change weight $R_{rate} = 0.1$, output state variable weight with $Q = 10$ initially.

"Short GPS Path" profile in Carsim used to test performance of the controller as shown in Figure 7.10.b. Controller follows the path with fixed velocity at 80 km/h, with 50 cm steady state error shown in Figure 7.10.a. and peaky data at time ≈ 0.53 seconds goes up to 1 meter, which is an unwanted situation. So the controller parameters needed to be tuned, from the first perspective of the results.

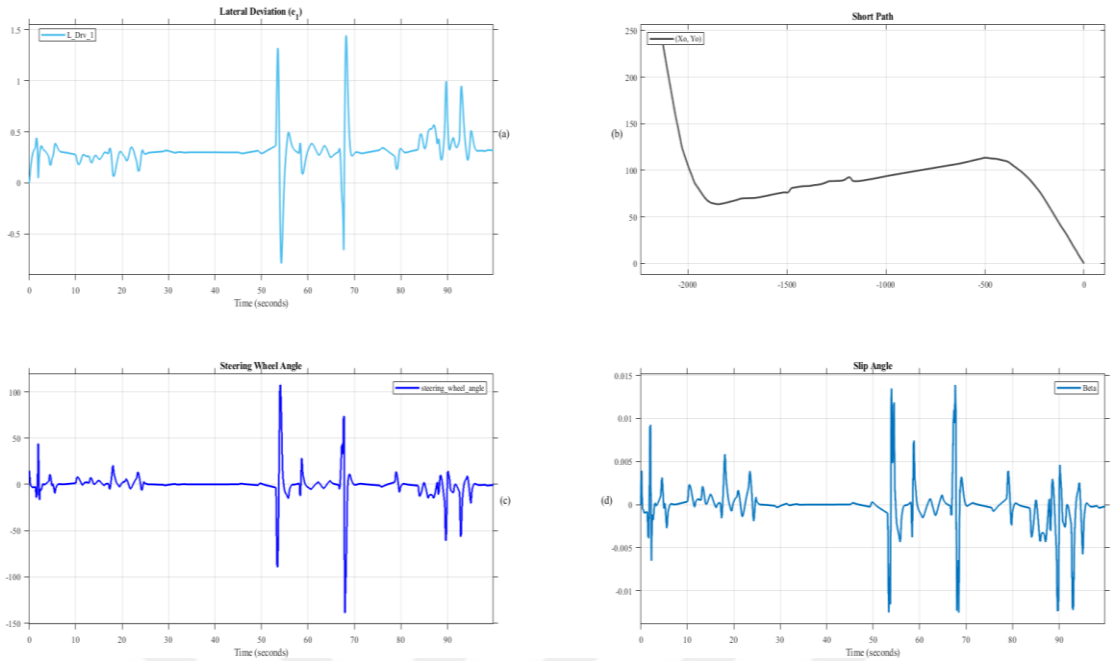


Figure 7.10. Controller performance with 80 km/h

For the second case of the controller, vehicle velocity held constant again at 120 km/h. From the Figure 7.11 it can be seen that controller behaves more aggressively. At time ≈ 38 seconds the lateral deviation from the path increases up to 3 meters, in return this caused a sudden steering wheel change, which brought a slip about 0.08 radians.

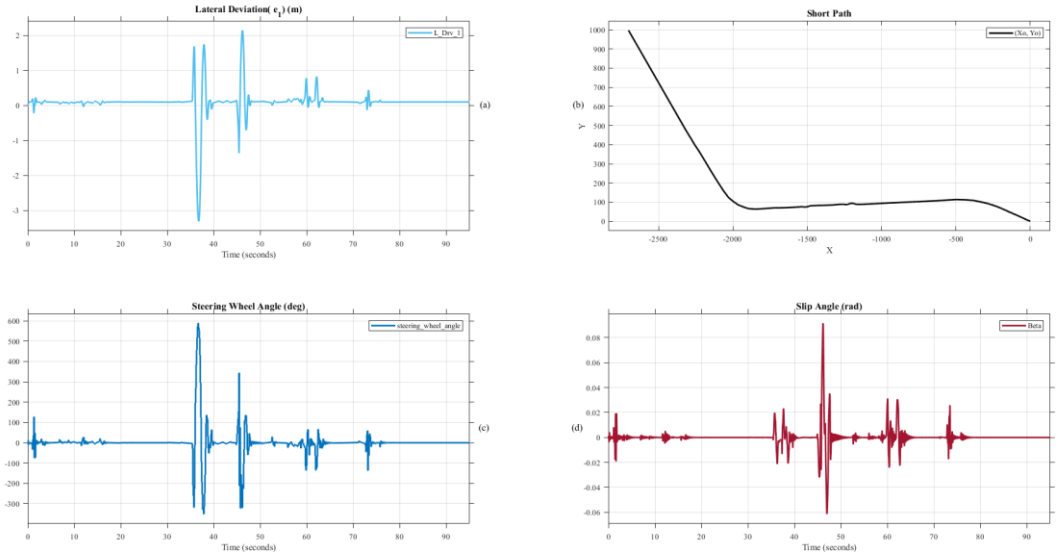


Figure 7.11. Controller performance with 120 km/h velocity

To prevent aggressiveness in controller shown in Figure 7.10 and Figure 7.11 sampling rate of the controller was increased to 100 Hz therefore the internal discrete plant model's update rate was increased to 100 Hz also, to reduce computational burden of the computer used in co-simulations prediction and control horizons lowered a little bit since the rate of controller and "model updater" was increased to 100 Hz and to penalize the lateral deviation weight Q was increased a little bit.

Several test were made to test the performance of the controller. The test parameters of the controller were as follows; vehicle velocity held at 80 km/h, prediction horizon $N = 30,20,10$ control horizon $N_c = 2$, steering angle constraints with $\delta_{max} = 1\,rad, \delta_{min} = -1\,rad$, $\dot{\delta}_{max} = 0.2\,rad, \dot{\delta}_{min} = -0.2\,rad$, steering input rate of change weight $R_{rate} = 0.1$, output state variable weight with $Q = 12$ for each test of the controller.

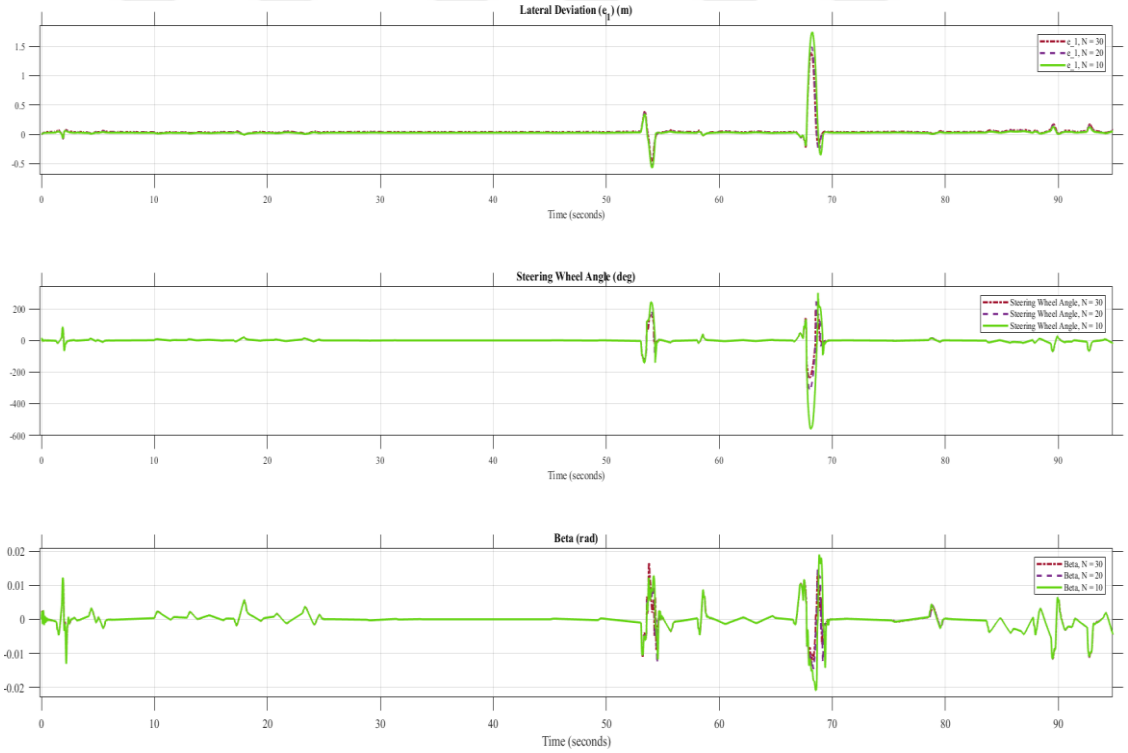


Figure 7.12. Controller performance with changing prediction horizons

Lastly, Figure 7.12 shows the lowest lateral deviation and steady state error occurred when the prediction horizon of the controller were set to 10, about 0.027 meter which is 2.7 centimeters. This steady state error was believed so low, and the controller parameters were set its final configuration at this stage.

8. VEHICLE LATERAL DYNAMICS ESTIMATION WITH DEEP NEURAL NETWORKS

In this section the curvature of the vehicle was estimated without the knowledge of the road by using the dynamics of the vehicle. Since the yaw rate is directly proportional to the vehicle, curvature information can be extracted by using longitudinal velocity by using (7.16).

To estimate the lateral dynamics of the so few researches have been made. In [63], a generalized regression network is used for estimation of longitudinal acceleration, lateral acceleration and yaw angle using a separate model for each DOF. In [64], lateral motion of the vehicle is modeled with feedforward neural networks. Authors of [65] compared the open loop training and closed loop training methods.

In this thesis, differently from the literature both feedforward NN and recurrent NN models are combined in cascade to obtain a deep neural network which is then used to identify vehicle lateral dynamics with only 2 inputs: vehicle speed and steering wheel angle. This will satisfy that without sensor information the states of the vehicle will be possible to measure.

Proposed DNN consisted of two stages. First stage used feedforward NN to determine right and left tire angles, which are then fed to the second stage recurrent NN (RNN). Second stage was also composed of two different recurrent NNs to determine lateral velocity and angular velocity of the vehicle [66].

Since a vehicle or car-like mobile robot uses Ackermann steering [11], front left and right tires create different lateral forces therefore the forces in Equation (2.6) - (2.8) will take place differently from each other.

First stage was composed of 1 input layer $\in R^1$, 1 hidden layer $\in R^{10}$ with 10 neurons with *sigmoid* activation functions and 1 output layer $\in R^2$ with two output neurons with *pure linear* activation functions as shown in Figure 8.1. In this stage, steering angle wheel was used as input to the neural network then front left and right tire angles, $\delta_{f,l}$, $\delta_{f,r}$ were estimated.

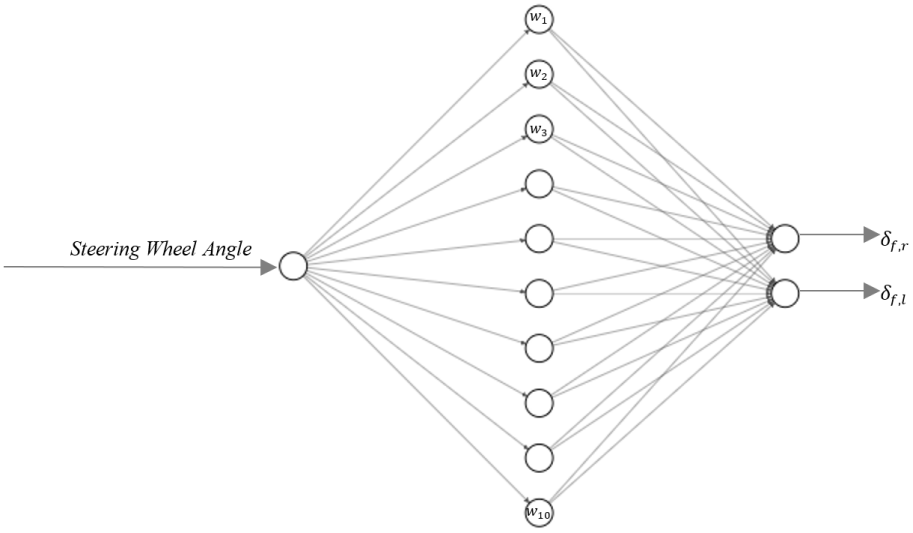
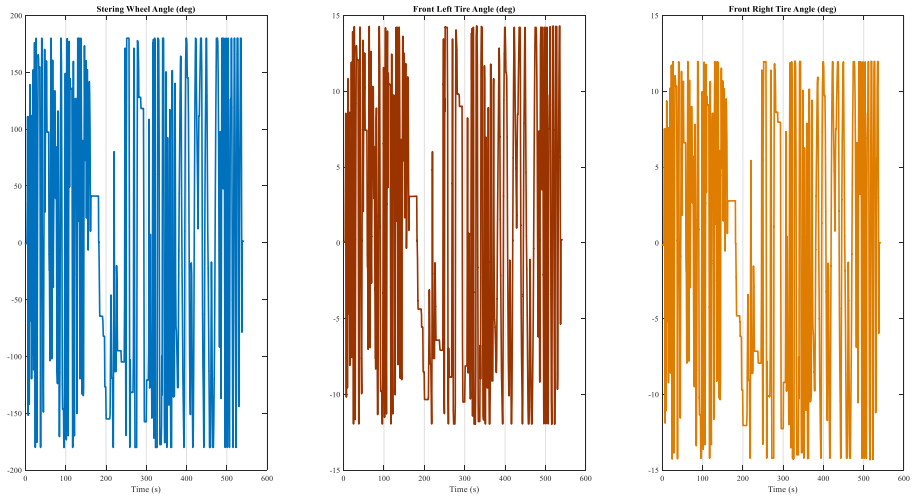


Figure 8.1. Front Tire Angle Estimation Network

Input, steering wheel angle created with Logitech Momo steering wheel & pedal set [67]. Outputs for front left & right tires were taken from Carsim as given in Figure 8.3.

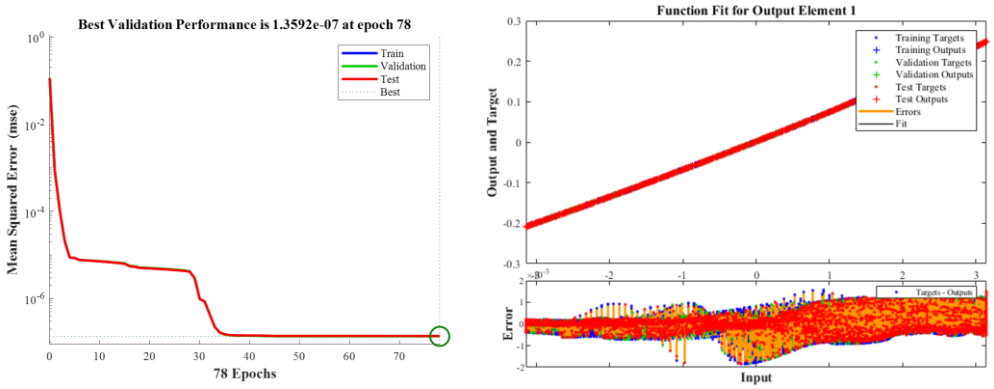
Figure 8.2. Experimental setup

As shown in the given figure sudden steering wheel angle changes to see the changing steering dynamics, also from the second and third plot it can be seen the angles of the two front wheels differ from each other.



(a) **(b)** **(c)**

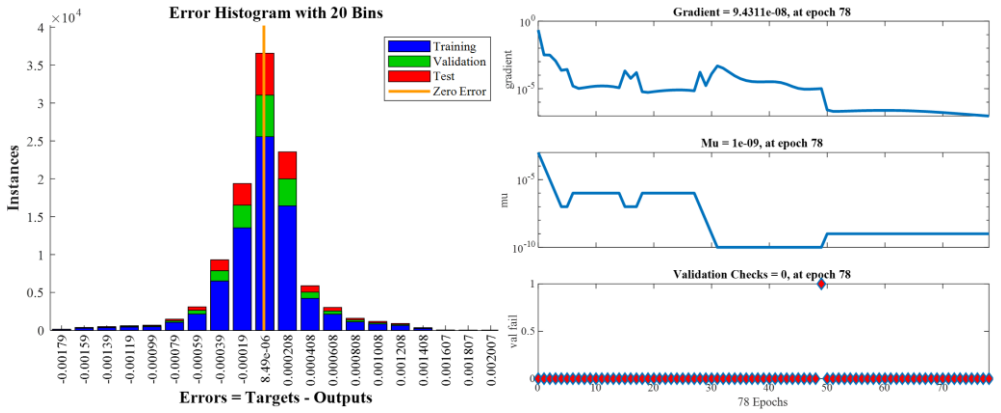
Figure 8.3. Training data for tire angle estimation (a) Steering wheel angle (b) Front left tire angle (c) Front right tire angle



(a) **(b)**

Figure 8.4. (a) Validation performance of tire angle estimation NN (b) Function fit performance of tire angle estimation NN

The feedforward neural network trained with Levenberg-Marquardt method, with 78 epochs. Figure 8.4 and Figure 8.5 shows that the performance of the created neural network almost perfect since the error histogram and mean squared error figures show nearly one in a million error on the instances.



(a) **(b)**

Figure 8.5. (a) Error histogram of tire angle estimation NN (b) Training history of tire angle estimation NN

Real time performance of the created neural network when compared to Carsim data is given in Figure 8.6 and the performance error were found near 0.001 radians level.

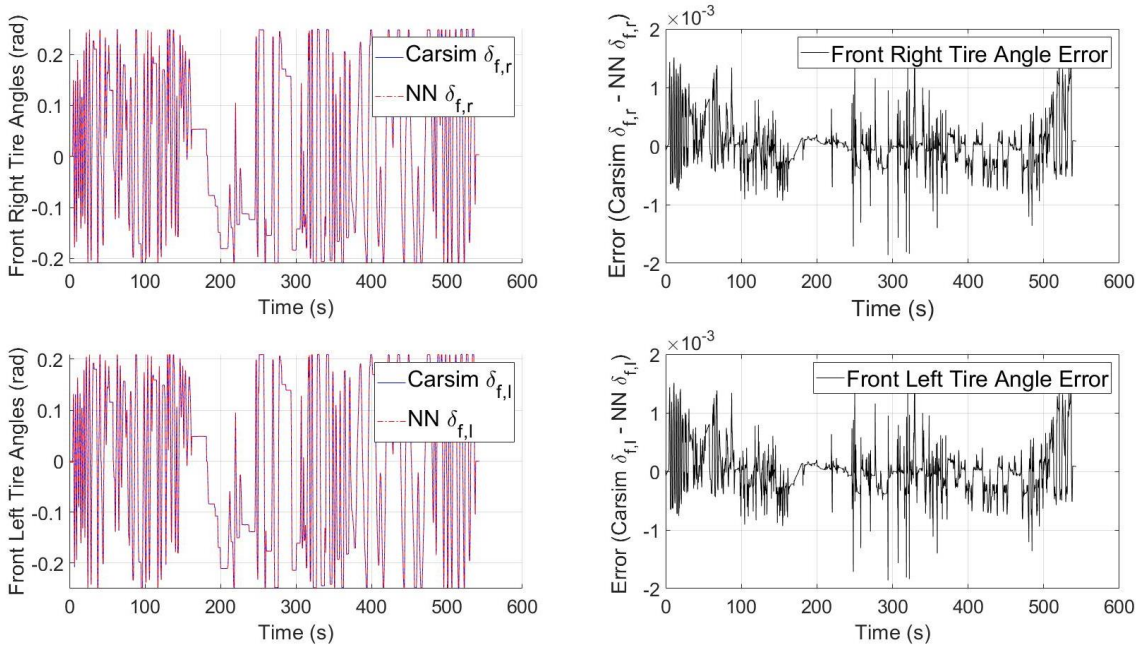


Figure 8.6. Real time performance of tire angle estimation NN

At the second stage, results from the created tire angle estimation network were fed to two recurrent neural networks for estimation of lateral velocity, $\dot{y}$, and yaw rate $\dot{\psi}$. In this way, the final form of deep neural network (DNN) constructed as shown in Figure 8.7.

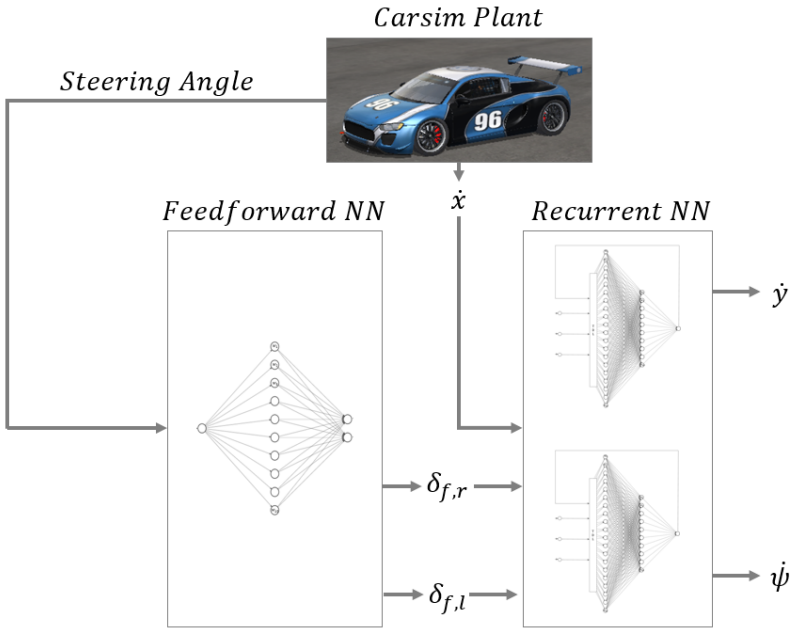


Figure 8.7. Constructed DNN for lateral vehicle dynamics estimation

Two recurrent neural networks were created with; 1 input layer $\in R^3$ ($\dot{x}$, $\delta_{f,r}$, $\delta_{f,l}$) and time delay layer (TDL) with $Z^{-1} \times 10$, 2 hidden layers $\in$ R^{20}, $\in$ R^{10} and 1 output layer $\in$ R^1 ($\dot{y}, \dot{\psi}$). Activation function of the hidden layers and output layer were both *sigmoid* functions.

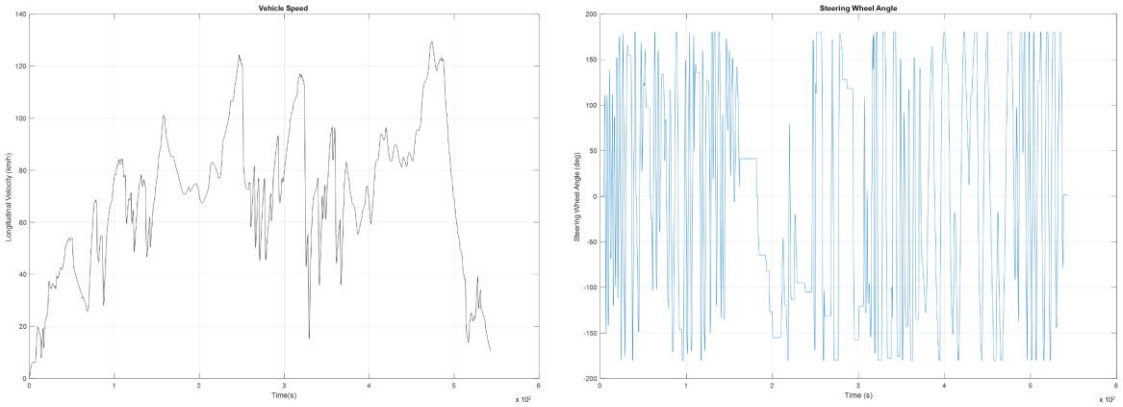


Figure 8.8. Training data for the created DNN

Again the training data shown in Figure 8.8 created by using experimental setup shown in Figure 8.2. Performance of the created DNN can be seen in Figure 8.9 and quantification of the performance is given in Table 8.1.

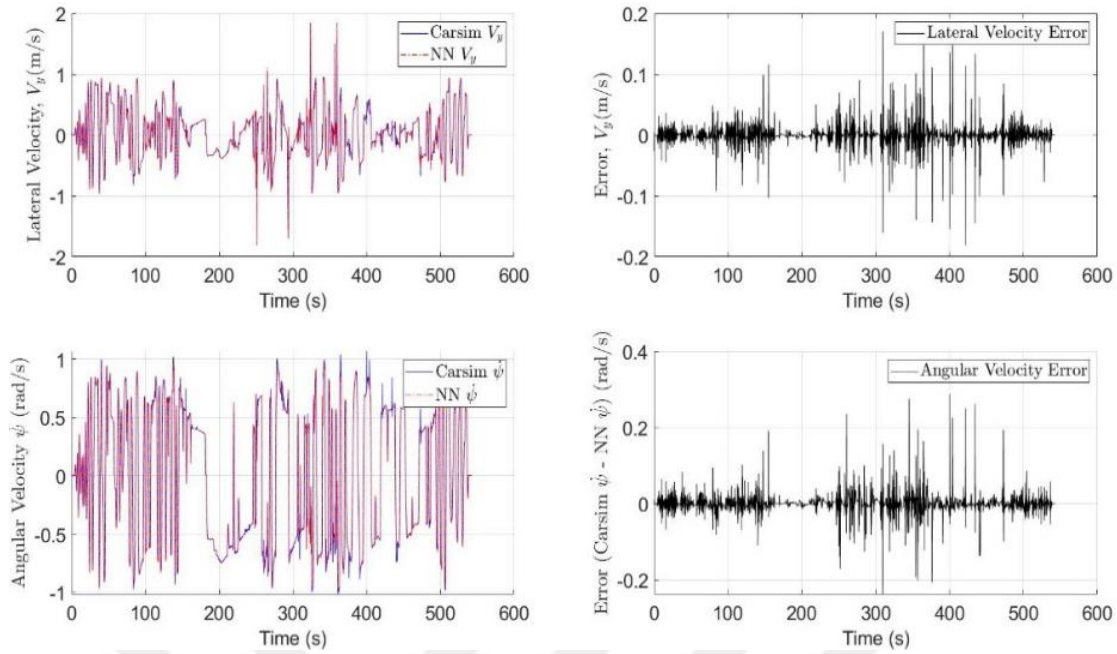


Figure 8.9. NN identification results for $\dot{y}, \dot{\psi}$

At last stage, since the lateral dynamics estimated with neural networks; using yaw rate the curvature can be estimated in real time.

Table 8.1. Quantification of neural networks

Estimation	RMS Error	Network Type
Front Right Tire Angle, $\delta_{f,r}$	3.83e-4 rad	Feedforward NN
Front Left Tire Angle, $\delta_{f,l}$	3.58e-4 rad	Feedforward NN
Lateral Velocity, $\dot{y}$	0.0148 m/s	Recurrent NN
Yaw Rate, $\dot{\psi}$	0.0214 rad/s	Recurrent NN

9. SIMULATIONS & RESULTS

For the created controllers with based on neural networks in Section 7.2 and Section 8, three kinds of tests were performed in Carsim along these three tracks: “Custom Road, S-Turn and Race Track defined in Section 5 with controllers created in Section 6 and Section 7.

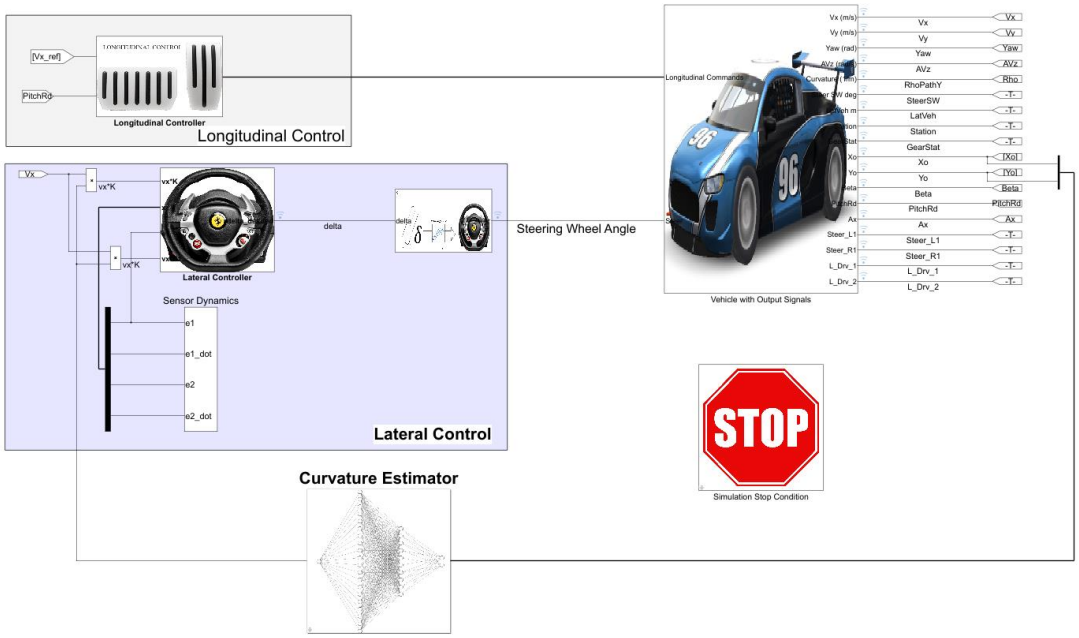


Figure 9.1. Complete decoupled control algorithm

First of all general control scheme shown in Figure 9.1 is explained in detail. The control algorithm was split into three main subjects:

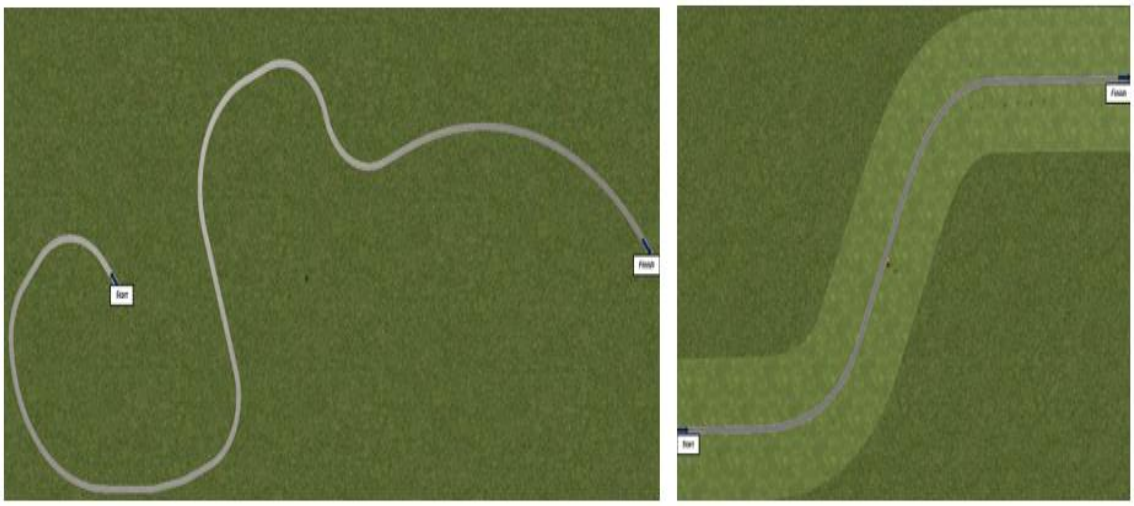
- **Longitudinal Control:** Shown in gray area, responsible from velocity tracking of the vehicle. Inputs are reference velocity and current grade angle of the road.
- **Lateral Control:** Shown in violet area, responsible from path tracking of the vehicle, inputs to the controller are calculated curvature (disturbance), states as sensor measurements.
    - **Sensor Dynamics:** Responsible from calculation controller state estimation, lateral deviation and yaw error measured from Carsim, and their rates were calculated.

- **Delta → Steering Wheel Angle:** Converts the calculated front wheel angle coming from controller via linear regression to steering wheel angle to steer vehicle to desired lateral position.

- **Curvature Estimator:** Located in the feedback line of the control system, responsible from estimation of curvature of the vehicle with respect to its position.

Secondly, simulations were performed in Section 9.1 – Section 9.3 with the created controllers in Section 6 and Section 7 in more detail.

Customized Road and S-Turn profiles can be seen in Figure 9.2.a and Figure 9.2.b.



(a) **(b)**

Figure 9.2. Road test tracks (a) Custom road (b) S-turn

Three different tests were made in this these test courses:

- Controller test without any curvature information
- Controller test with curvature information from lateral vehicle dynamics estimator
- Controller test with curvature information from trained NN in Section 7.3.

9.1. Custom Road Tests

The vehicle and lateral controller are tested in the custom road created in Carsim as shown in Figure 9.2.a, the longitudinal velocity of the vehicle held constant at 35 m/s ~ 126 km/hr with longitudinal controller developed in Section 6 in order to observe the changes when the estimation algorithm changes.

At first for the no curvature information case, E(t) in (7.9) was taken to be zero to omit curvature information for the controller. If there would be no curvature information from the environment or somehow vehicle can't detect the path it would go in future; vehicle's lateral deviation from desired trajectory should be more when it's compared to curvature estimated ones since curvature was defined as measured disturbance in control problem formulation.

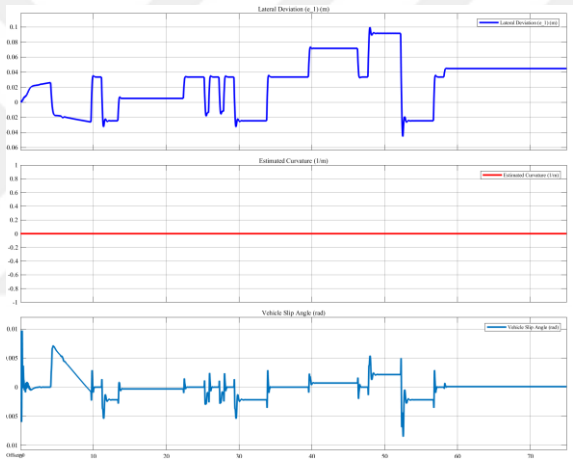


Figure 9.3. Lateral controller performance without any curvature estimation

Figure 9.3 shows that RMS error for the lateral deviation without any curvature estimation was found as 0.042346 meters nearly 4 centimeters, the maximum deviation was found as nearly 10 centimeters for this case.

For the second case, curvature estimation made with the lateral vehicle dynamics estimator developed in Section 8. From Figure 9.4 it can clearly be seen that lateral deviation from the desired path reduced, and the RMS error found 0.035304 meters for this case.

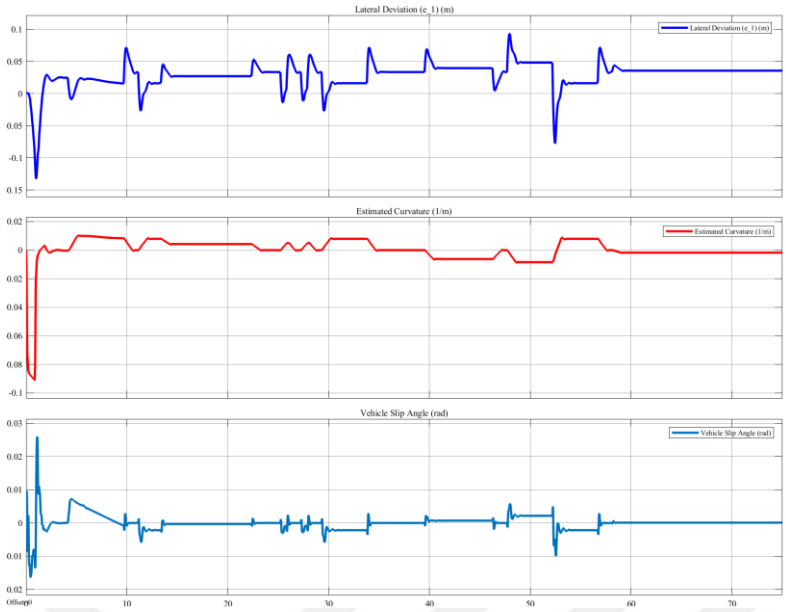


Figure 9.4. Lateral controller performance with lateral dynamics estimator

It's known that knowing the measured disturbance, curvature (κ), increases the controller performance. To supply this hypothesis, curvature estimation performance of the lateral vehicle dynamics estimator and the curvature of the custom road were compared in Figure 9.5. It can clearly be seen that estimation profile follows the original one with minor errors.

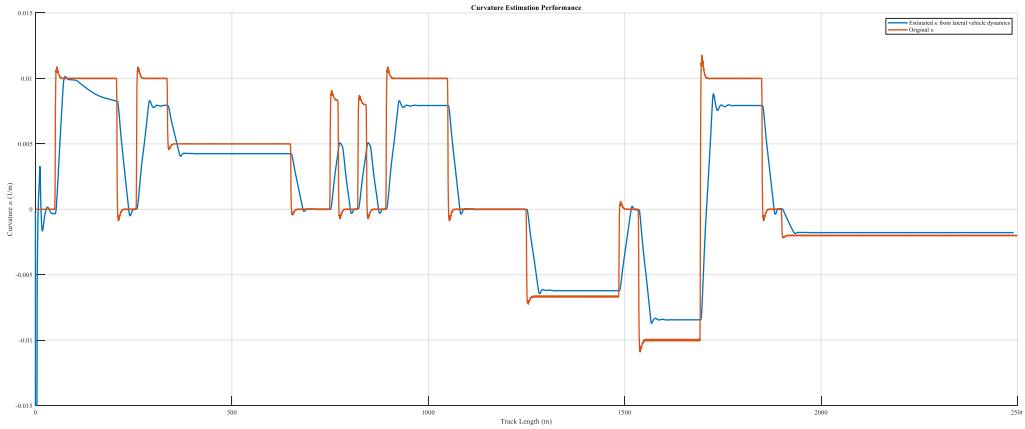


Figure 9.5. Curvature estimation performance of lateral vehicle dynamics estimator

For the third case, neural network curvature estimator developed in Section 7.3 was used for trajectory tracking problem for this custom road.

It’s expected that since the training of feedforward neural network explained in Section 5 depends on X and Y coordinates of the desired path; controller performance should be slightly worse than each other two. To prove this tests were made on different tracks.

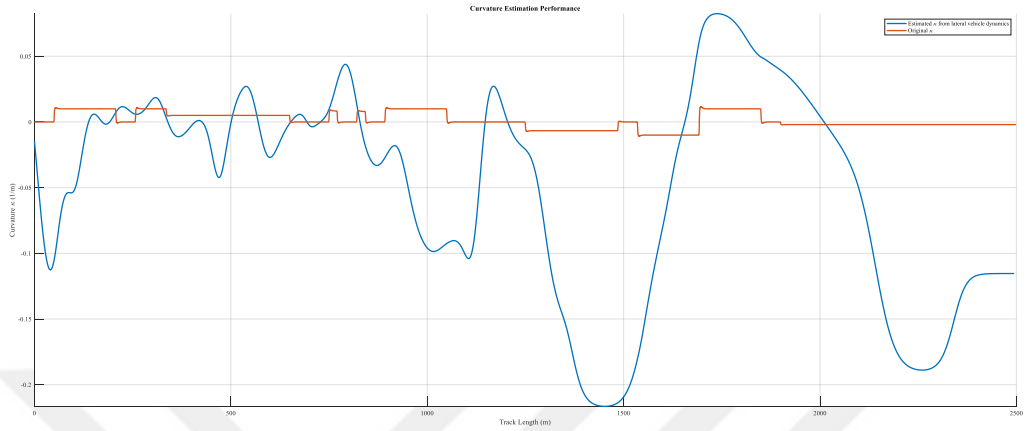


Figure 9.6. Curvature estimation performance of feedforward neural network estimator

It’s clear that Figure 9.6 proves this idea, by showing that estimated curvature and the real curvature is far from each other.

When assessing the lateral controller performance of the controller based on this feedforward neural network, RMS error found as 0.925728 meters which is nearly up to 1 meters, also the maximum deviation from the path found up to 2 meters from Figure 9.7.

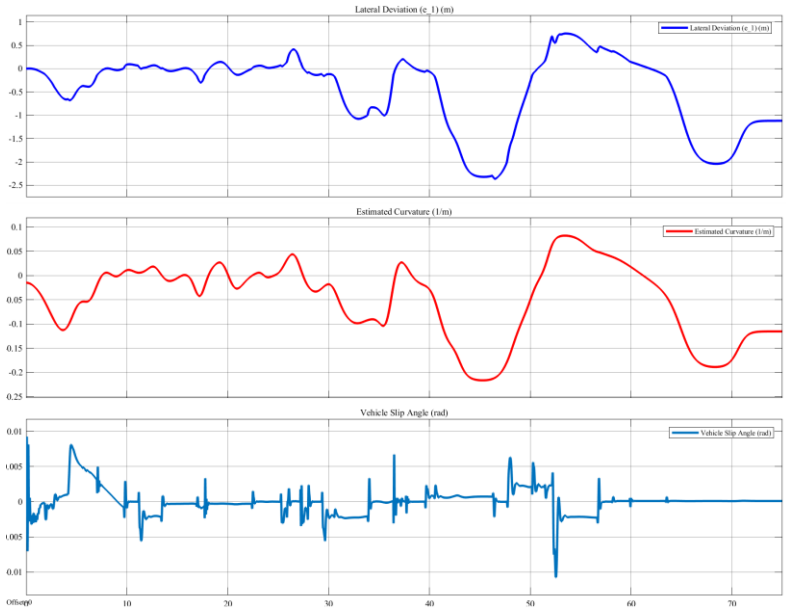


Figure 9.7. Lateral controller performance with feedforward neural network estimator

9.2. S-Turn Road Tests

The vehicle and lateral controller are tested in the custom road created in Carsim as shown in Figure 9.2.b, the longitudinal velocity of the vehicle held constant at 35 m/s ~ 126 km/hr with longitudinal controller developed in Section 6 in order to observe the changes when the estimation algorithm changes.

At first for the no curvature information case, RMS error found as 0.0369 meters and maximum lateral deviation 0.06 meters.

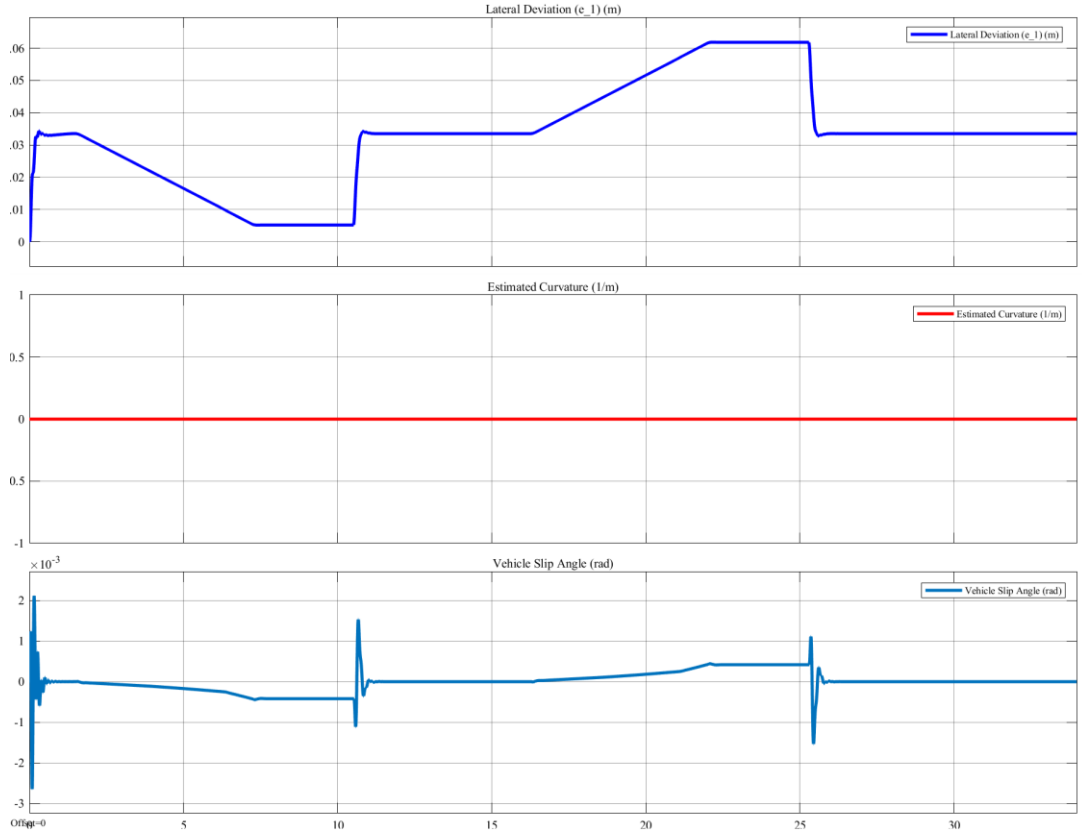


Figure 9.8. Lateral controller performance without any curvature estimation

For the second case, curvature estimation was performed with the lateral vehicle dynamics estimator developed in Section 9. From Figure 9.10 it can clearly be seen that lateral deviation from the desired path was reduced, and the RMS error found 0.033412 meters for this case.

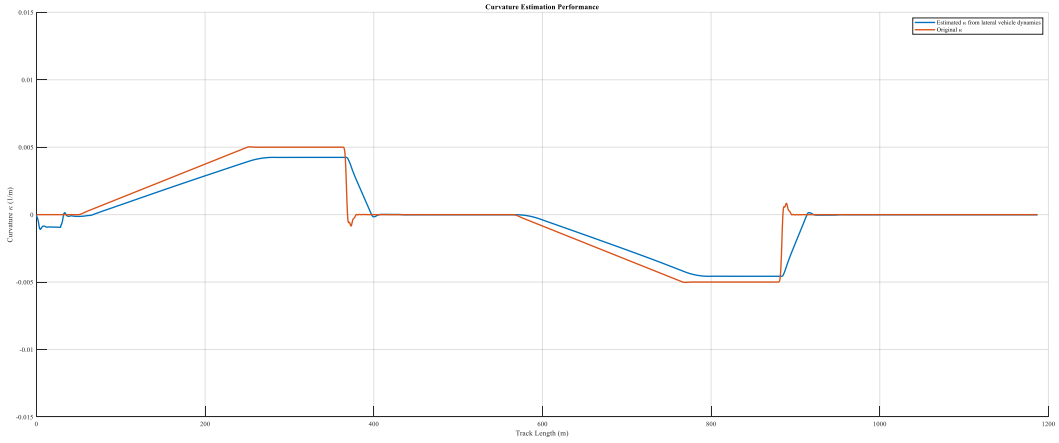


Figure 9.9. Curvature estimation performance of lateral vehicle dynamics estimator

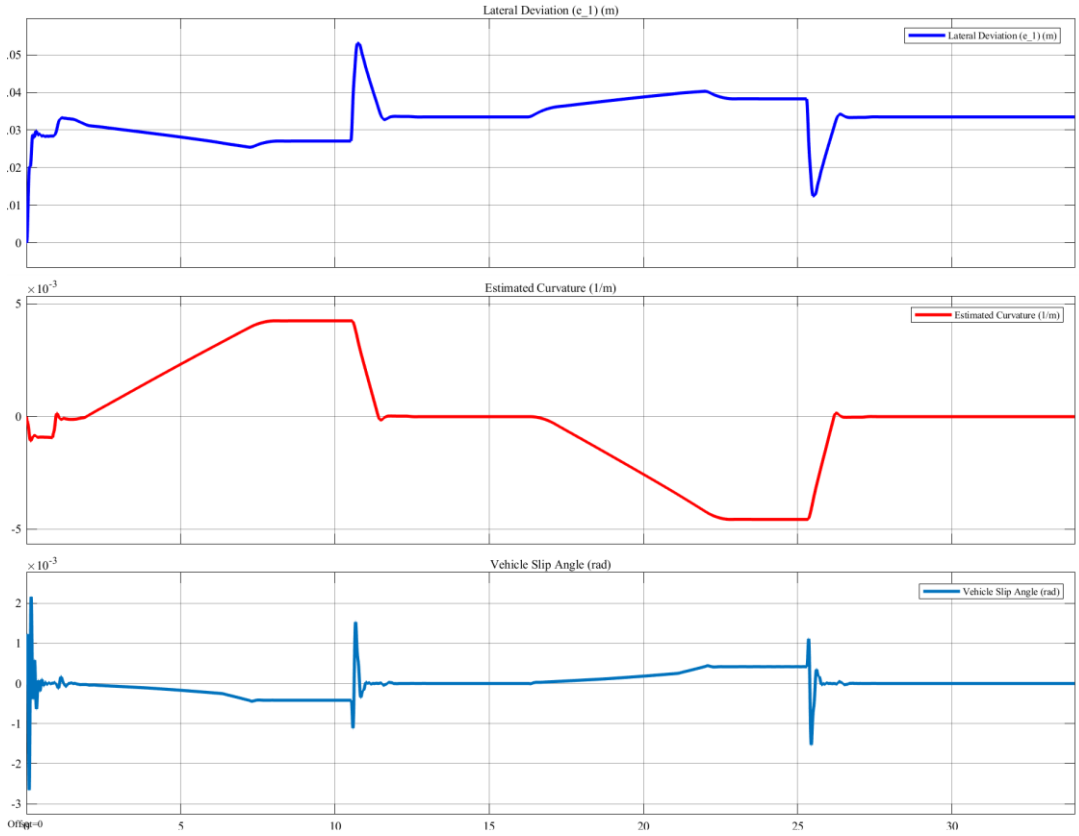


Figure 9.10. Lateral controller performance with lateral dynamics estimator

Curvature estimation performance of the lateral vehicle dynamics estimator and the curvature of the custom road were compared in Figure 9.9. It can clearly be seen that estimation profile follows the original one with minor errors.

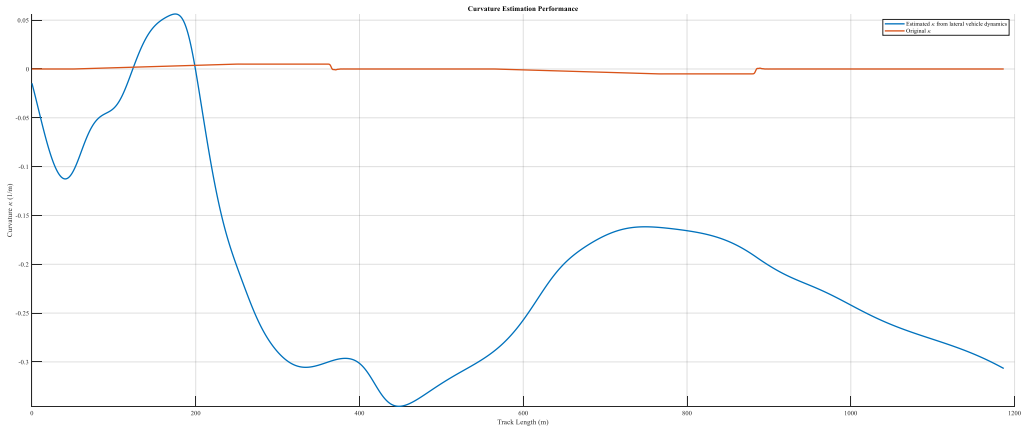


Figure 9.11. Curvature estimation performance of feedforward neural network estimator

It's clear from Figure 9.11 that feedforward neural network's result should be far from the road curvature since training made within another track depending on X and Y coordinates only.

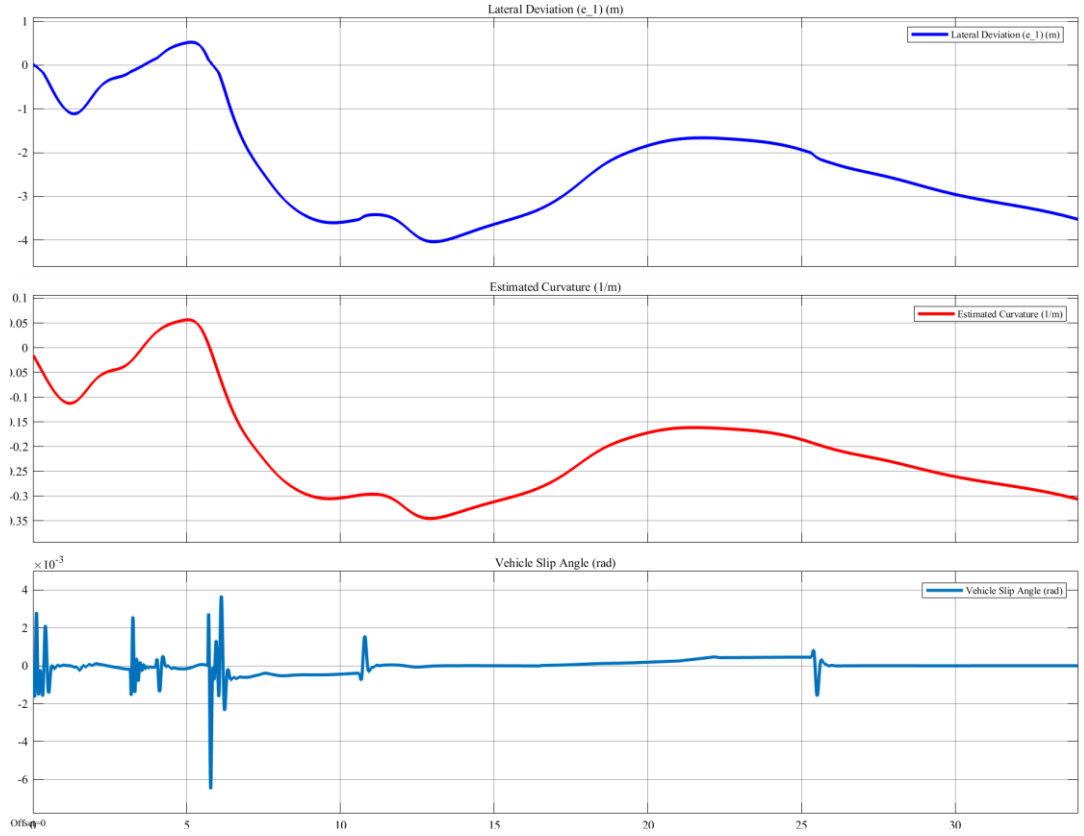


Figure 9.12. Lateral controller performance with feedforward neural network estimator

When assessing the lateral controller performance of the controller based on feedforward neural network, RMS error found as 2.60908 meters, also the maximum deviation from the path found up to 4 meters from Figure 9.12.

9.3. Race Track Tests

In this test, found curvature was calculated from the vehicle dynamics and smoothed with 0.85s windowed moving average filter. Since the Simulink's simulation sampling time was 0.001, with a 0.85s averaging window 850 samples were averaged each time step.

The vehicle and lateral controller are tested in the race track in Carsim as described in Section 5.

Lateral tracking performance seemed quite well at first with staying 15 centimeters threshold for both negative and maximum values even without any curvature information for the controller, lately the RMS error for lateral deviation was found 0.065918 meters which converts to 6.592 centimeters.

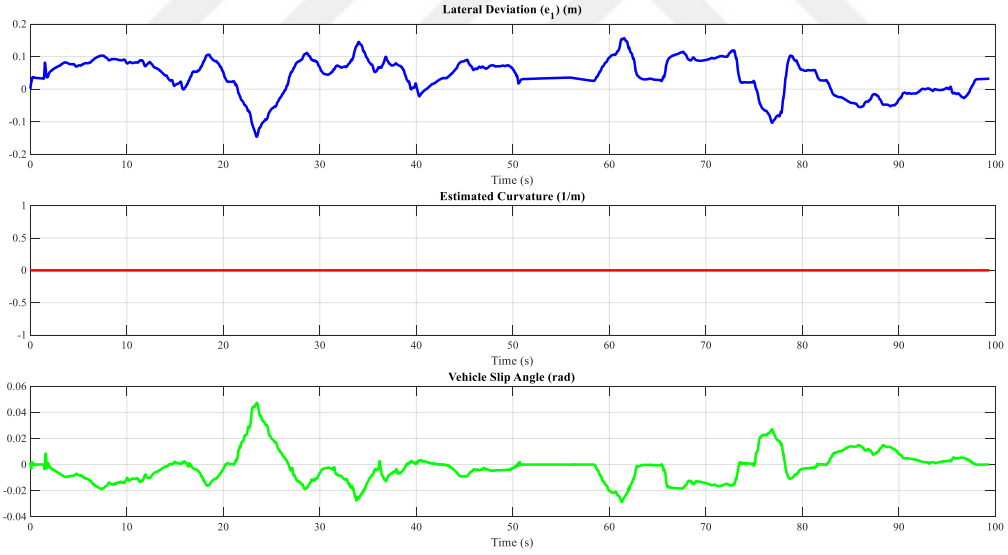


Figure 9.13. Lateral controller performance without any curvature estimation

For the second case, curvature information was supplied by using vehicle lateral dynamics estimator as its performance can be seen in Figure 9.14. For this controller the RMS error of lateral deviation was found 0.034860 meters, which is ~3.5 centimeters from Figure 9.15. When compared to first test, RMS error was found nearly two times lower.

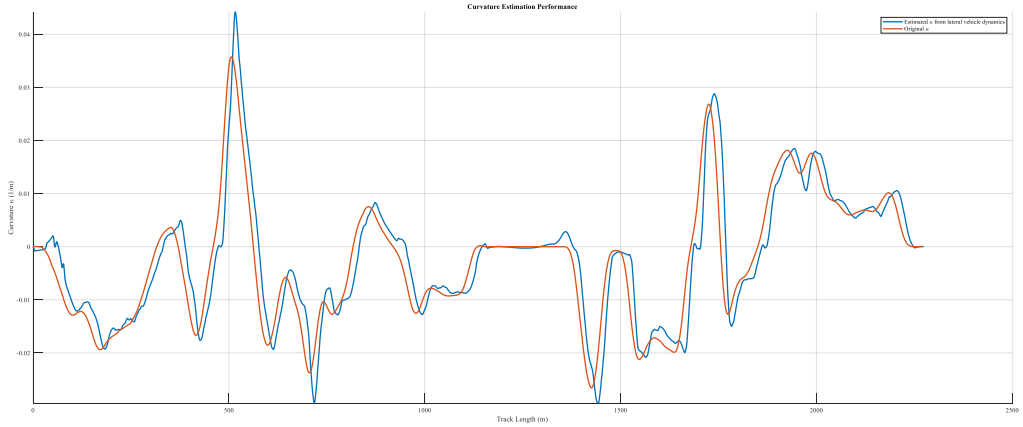


Figure 9.14. Curvature estimation performance of lateral vehicle dynamics estimator

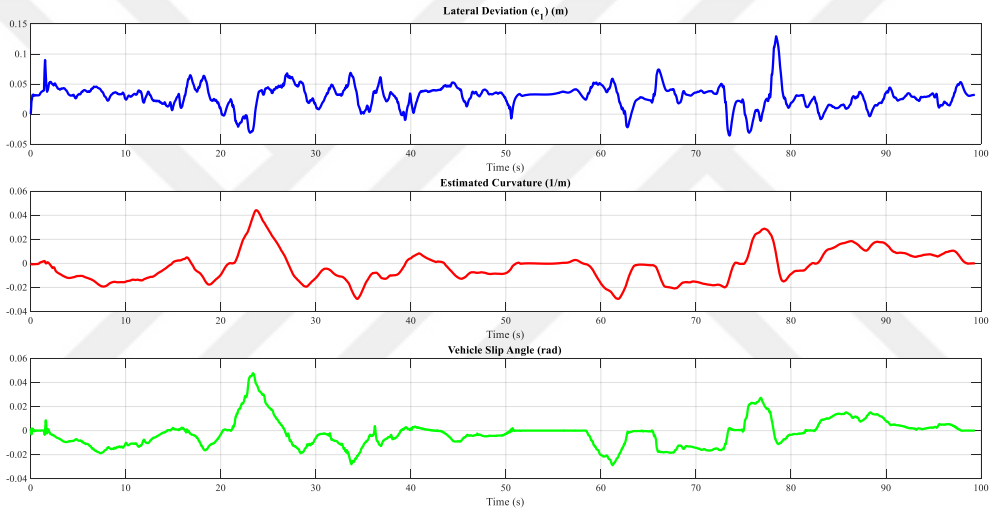


Figure 9.15. Lateral tracking performance with vehicle lateral dynamics estimator

At the third stage, the disturbance information was supplied to the controller with a feedforward neural network estimator, the real time performance of the estimator was given in Figure 7.7.

The most noticeable improvement happened on the lateral tracking performance on this test because vehicle was knowing the path it would go since it was trained within this race track with X and Y coordinates.

The lateral deviation was between ±0.1 meter levels, but in this case the maximum lateral deviation found as 0.06 meters. The difference in the mean of lateral deviation between two tests was 4 centimeters improvement.

Path tracking performance increased with disturbance estimators. The RMS error of lateral deviation was found 0.029998 meters, which is ~3 centimeters. When compared to first test, RMS error was found two times lower.

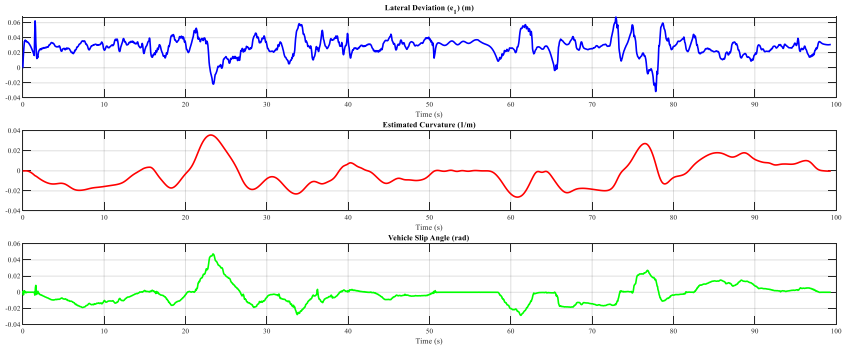


Figure 9.16. Lateral controller performance with feedforward neural network estimator

As in result, both longitudinal and lateral controllers were robust enough to follow the desired path and velocity profiles against the changing road conditions such as; road angle, road bank angle, changing friction on different surfaces and etc.

9.4. Results

Evaluation of the tests completed in Section 9.1 through Section 9.3, are summarized in this section. For the comparison of lateral controller performance for each type, Table 9.1 summarizes the results as follows:

Table 9.1. Lateral controller performance comparison

Controller / Test Track	**RMS Error (cm)**		
	Custom Road	**S-Turn**	**Race Track**
MPC with no κ information	4.235	3.690	6.591
MPC with κ information from vehicle dynamics curvature estimator	3.530	3.341	3.486
MPC with κ information from curvature estimator NN	92.573	260.908	2.999

For custom road track scenario, the best lateral controller performance found with lateral vehicle dynamics estimator controller. The RMS errors were lowered 16.63% when compared to without curvature information one. Since the developed feedforward neural network for the curvature estimation was trained in the race track only, its performance was very low with 92 centimeters RMS errors.

In the second scenario, which is S-Turn Track, the best performance was found with lateral vehicle dynamics estimator controller again. The controller performances of no curvature information one and vehicle dynamics estimator was quite close; but when we looked at the improvement with RMS error levels, decrease in error was 9.45%.

At last, vehicle was driven within a race track with the racing profile given in Section 5. Now the best performance was found with feedforward neural network estimator one, since the vehicle knew the path it would go on the track with the support of neural network training. The RMS error was found ~3 centimeters for this case and the worst performance was found with no curvature information one. From worst case to best case the RMS error decrease was 47.12%.

These results show that, while the neural network curvature estimator from path position data works best on a given track, it does not generalize to other paths. On the other hand vehicle dynamics based estimator is generalizable and provides performance improvements on different tracks.

Lastly according to Table 9.1 and the simulation results show that, with the estimation of outer disturbance that affects the system such as road curvature via neural networks lateral tracking performance increase in a considerable amount.

10. CONCLUSIONS & FUTURE WORK

10.1. Accomplishments and Conclusions

In this study, linear time varying & learning based model predictive controllers were developed for trajectory tracking controller of a race vehicle along three different tracks. These developed controllers based on two different neural network algorithms were used to estimate measured disturbance (road curvature) for the trajectory tracking problem.

A new approach was developed to trajectory tracking problem with both changing velocity and online curvature estimation, as most of the literature solves path tracking problem with constant velocity. Through the developed neural networks vehicle was able to estimate the path it would go and when this information supplied to the lateral controller in this study controller performance improvement was observed. However when the same controller is used on different paths, either the curvature estimator has to be updated with a tour on the path, or the vehicle dynamics based curvature estimator approach proposed in this paper should be used as it is more generalizable to different paths.

In conclusion, it is shown that knowing the disturbances affecting the dynamics of the system such as road curvature, road bank angle and lateral load transfer etc. integrating with the controllers can improve controller performance. With the disturbance estimation algorithms such as disturbance observers, neural networks, unknown input observers etc. creating much improved controllers would be possible for autonomous race vehicle development.

10.2. Future Work

In the future;

- Vehicles degree of freedom can be increased and suspension dynamics can be taken into account for vehicle modelling.
- Proposed neural network which estimates the lateral dynamics, may be improved with different training scenarios and might be used as internal plant model of the lateral model predictive controller.
- Longitudinal dynamics of the vehicle might be modelled with whole powertrain models and can also be controlled with model predictive control. Additionally, it might be modelled with using neural networks to improve modelling accuracy.
- With iterative learning vehicle may learn the velocities, positions and orientations it should possess along the race track then lap time improvement problem may be taken into account.
- Continuous time Linear Parameter Varying (LPV) controllers might be developed to investigate the effect of parameter uncertainties on robustness of the controllers which will be developed.
- Disturbance observers might be created to estimate real time disturbances occurring on the vehicle to compensate them.

11. REFERENCES

[1] SAE. (2021). Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. SAE International.

[2] Formula Student | AMZ Racing. (2021). AMZ Driverless. Retrieved November 14, 2021, from https://driverless.amzracing.ch/en/formula-student

[3] F1TENTH. (2021). F1TENTH Web Site Retrieved November 14, 2021, from https://f1tenth.org/

[4] Indy Autonomous Challenge. Retrieved November 14, 2021, from https://www.indyautonomouschallenge.com/

[5] Roborace. (2021). Roborace Website. Retrieved November 14, 2021, from https://roborace.com/

[6] Marin-Plaza, P., Hussein, A., Martin, D., & Escalera, A. D. L. (2018). Global and Local Path Planning Study in a ROS-Based Research Platform for Autonomous Vehicles. Journal of Advanced Transportation, 2018, 1–10. https://doi.org/10.1155/2018/6392697

[7] Hart, P., Nilsson, N., & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics, 4(2), 100–107. https://doi.org/10.1109/tssc.1968.300136

[8] LaValle, S. M., & Kuffner, J. J. (2001). Randomized Kinodynamic Planning. The International Journal of Robotics Research, 20(5), 378–400. https://doi.org/10.1177/02783640122067453

[9] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1(1), 269–271. https://doi.org/10.1007/bf01386390

[10] Montani, M., Ronchi, L., Capitani, R., & Annicchiarico, C. (2021). A Hierarchical Autonomous Driver for a Racing Car: Real-Time Planning and Tracking of the Trajectory. Energies, 14(19), 6008. https://doi.org/10.3390/en14196008

[11] Milliken, M. F. D. W. L. (1994). Race Car Vehicle Dynamics. SAE International.

[12] MATLAB - MathWorks. (2021). MATLAB & Simulink. Retrieved November 14, 2021, from https://www.mathworks.com/products/matlab.html

[13] Mechanical Simulation. (2021). Carsim Website. Retrieved November 14, 2021, from https://www.carsim.com

[14] Amer, N. H., Zamzuri, H., Hudha, K., & Kadir, Z. A. (2016). Modelling and Control Strategies in Path Tracking Control for Autonomous Ground Vehicles: A Review of State of the Art and Challenges. Journal of Intelligent & Robotic Systems, 86(2), 225–254. https://doi.org/10.1007/s10846-016-0442-0

[15] Zhang, H., Wu, D., & Yao, T. (2018). Research on AGV trajectory tracking control based on double closed-loop and PID control. Journal of Physics: Conference Series, 1074, 012136. https://doi.org/10.1088/1742-6596/1074/1/012136

[16] Park, M., Lee, S., & Han, W. (2015). Development of Steering Control System for Autonomous Vehicle Using Geometry-Based Path Tracking Algorithm. ETRI Journal, 37(3), 617–625. https://doi.org/10.4218/etrij.15.0114.0123

[17] Snider, J. M. (2009). Automatic Steering Methods for Autonomous Automobile Path Tracking. Master's Thesis, Carnegie Mellon University, Robotics Institue, Pittsburgh, Pennsylvania, USA

[18] Törő, O., Bécsi, T., & Aradi, S. (2016). Design of Lane Keeping Algorithm of Autonomous Vehicle. Periodica Polytechnica Transportation Engineering, 44(1), 60–68. https://doi.org/10.3311/pptr.8177

[19] Hoffmann, G. M., Tomlin, C. J., Montemerlo, M., & Thrun, S. (2007). Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing. 2007 American Control Conference. https://doi.org/10.1109/acc.2007.4282788

[20] Coulter, R. C. (1992). Implementation of the Pure Pursuit Path Tracking Algorithm. Robot. Carneige Mellon University Robotics Institue.

[21] Filip, J. (2018). Trajectory Tracking for Autonomous Vehicles. Master's Thesis, Czech Technical University in Prague, Prague, Czech

[22] CarMaker. (2021). Carmaker Website. Retrieved November 14, 2021, from https://ipg-automotive.com/products-services/simulation-software/carmaker

[23] Kritayakirana, K. (2012). Autonomous Vehicle Control at The Limits Of Handling. Doctorate Thesis, Stanford University, CA, USA

[24] Ahmadi, B. (2021). Design a Reliable Model Predictive Control for Path Following with Application to the Autonomous Vehicle and Considering Different Vehicle Models. Master's Thesis, University of Waterloo Mechanical and Mechatronics Engineering, ON, Canada

[25] Curinga, F. (2017). Autonomous racing using model predictive control. Master's Thesis. Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-222801

[26] Abbas, M. (2011). Non-Linear Model Predictive Control For Autonomous Vehicles. Master's Thesis. University of Ontario Institute Of Technology, ON, Canada

[27] Xu, Y., Chen, B., Shan, X., Jia, W., Lu, Z., & Xu, G. (2017). Model predictive control for lane keeping system in autonomous vehicle. 2017 7th International Conference on Power Electronics Systems and Applications - Smart Mobility, Power Transfer & Security (PESA). https://doi.org/10.1109/pesa.2017.8277758

[28] Wang, H. (2018). Control System Design for Autonomous Vehicle Path Following and Collision Avoidance. Doctorate Thesis. The Ohio State University, OH, USA

[29] Verschueren, R., de Bruyne, S., Zanon, M., Frasch, J. V., & Diehl, M. (2014). Towards time-optimal race car driving using nonlinear MPC in real-time. 53rd IEEE Conference on Decision and Control. https://doi.org/10.1109/cdc.2014.7039771

[30] Rajamani, R. (2011). Vehicle Dynamics and Control (Mechanical Engineering Series) (2nd ed. 2012 ed.). Springer.

[31] Ye, H., Jiang, H., Ma, S., Tang, B., & Wahab, L. (2019). Linear model predictive control of automatic parking path tracking with soft constraints. International Journal of Advanced Robotic Systems, 16(3), 172988141985220. https://doi.org/10.1177/1729881419852201

[32] Jhang, J. H., & Lian, F. L. (2020). An Autonomous Parking System of Optimally Integrating Bidirectional Rapidly-Exploring Random Trees* and Parking-Oriented Model Predictive Control. IEEE Access, 8, 163502–163523. https://doi.org/10.1109/access.2020.3020859

[33] Emam, M. & Fakharian, A. (2017). Solving Path Following Problem for Car-Like Robot in the Presence of Sliding Effect via LMI Formulation. Journal of Computer & Robotics. 10. 11 - 22.

[34] Pacejka, H., & Besselink, I. J. M. (2012). Tire and Vehicle Dynamics (3rd ed.). Butterworth-Heinemann.

[35] Kim, G., Yoon, J., & Kim, B. (2016). Estimation of the Steering Angle Based on Extended Kalman-Filter. International Journal of Multimedia and Ubiquitous Engineering, 11(12), 295–306. https://doi.org/10.14257/ijmue.2016.11.12.27

[36] Kong, J., Pfeiffer, M., Schildbach, G., & Borrelli, F. (2015). Kinematic and dynamic vehicle models for autonomous driving control design. 2015 IEEE Intelligent Vehicles Symposium (IV). https://doi.org/10.1109/ivs.2015.7225830

[37] Fiala, E. (1987). Kraftfahrzeugtechnik. Dubbel, 941–966. https://doi.org/10.1007/978-3-662-06778-9_16

[38] Jin-Oh H., Rajamani, R., & Alexander, L. (2002). GPS-based real-time identification of tire-road friction coefficient. IEEE Transactions on Control Systems Technology, 10(3), 331–343. https://doi.org/10.1109/87.998016

[39] Arndt, M., Ding, E., & Massel, T. (2004). Identification of cornering stiffness during lane change maneuvers. Proceedings of the 2004 IEEE International Conference on Control Applications, 2004. https://doi.org/10.1109/cca.2004.1387235

[40] Börner, M., & Isermann, R. (2002). Adaptive One Track Model for Critical Lateral Driving Situation. Proc. of 6th International Symposium on Advanced Vehicle Control, 2002, 393–400. https://ci.nii.ac.jp/naid/10027977500

[41] Sierra, C., Tseng, E., Jain, A., & Peng, H. (2006). Cornering stiffness estimation based on vehicle lateral dynamics. Vehicle System Dynamics, 44(sup1), 24–38. https://doi.org/10.1080/00423110600867259

[42] Wheel Position Sensor (WPS) | A&D. (2020, August 17). A&D Company. Retrieved November 14, 2021, from https://aanddtech.com/2018/06/04/wps/

[43] Qin, S., & Badgwell, T. A. (2003). A survey of industrial model predictive control technology. Control Engineering Practice, 11(7), 733–764. https://doi.org/10.1016/s0967-0661(02)00186-7

[44] Mayne, D. Q. (2014). Model predictive control: Recent developments and future promise. Automatica, 50(12), 2967–2986. https://doi.org/10.1016/j.automatica.2014.10.128

[45] Boyd, S. (2004). Convex Optimization (1st ed.). Cambridge University Press.

[46] Nocedal, J., & Wright, S. (2006). Numerical Optimization (Springer Series in Operations Research and Financial Engineering) (2nd ed.). Springer.

[47] Wächter, A., & Biegler, L. T. (2005). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming, 106(1), 25–57. https://doi.org/10.1007/s10107-004-0559-y

[48] Ruder, S. (2016). An overview of gradient descent optimization algorithms. ArXiv, abs/1609.04747.

[49] Wong, E. (2011). Active-Set Methods for Quadratic Programming. Doctorate Thesis. University of California, Mathematics, San Diego, USA

[50] Model predictive controller - MATLAB. (2021). MPC Toolbox Matlab. Retrieved November 11, 2021, from https://www.mathworks.com/help/mpc/ref/mpc.html

[51] embotech. (2021, April 6). Forcespro Website. Embotech.Com. Retrieved October 11, 2021, from https://www.embotech.com/products/forcespro/overview/

[52] FiOrdOs - Mathematical software - swMATH. (2021). FiOrdOs. Retrieved November 11, 2021, from https://www.swmath.org/software/6277

[53] CasADi. (2021). CasADi. Retrieved November 14, 2021, from https://web.casadi.org

[54] ACADO Toolkit. (2021). ACADO Toolkit. Retrieved November 14, 2021, from https://acado.github.io

[55] Ferreau, H. J., Kirches, C., Potschka, A., Bock, H. G., & Diehl, M. (2014). qpOASES: a parametric active-set algorithm for quadratic programming. Mathematical Programming Computation, 6(4), 327–363. https://doi.org/10.1007/s12532-014-0071-1

[56] MacAdam, C. C. (1986). Development of driver/vehicle steering interaction models for dynamic analysis. Interim report. http://deepblue.lib.umich.edu/handle/2027.42/99

[57] MacAdam, C. C. (1981). Application of an Optimal Preview Control for Simulation of Closed-Loop Automobile Driving. IEEE Transactions on Systems, Man, and Cybernetics, 11(6), 393–399. https://doi.org/10.1109/tsmc.1981.4308705

[58] MacAdam, C. C. (1980). An Optimal Preview Control for Linear Systems. Journal of Dynamic Systems, Measurement, and Control, 102(3), 188–190. https://doi.org/10.1115/1.3139632

[59] Longitudinal speed-tracking controller - Simulink. (2021). Mathworks Longitudinal Driver. Retrieved October 5, 2021, from https://www.mathworks.com/help/vdynblks/ref/longitudinaldriver.html

[60] Inertial Measurement Unit. (2021). Xsens IMU. Retrieved November 10, 2021, from https://www.xsens.com/imu

[61] Chindamo, D., & Gadola, M. (2018). Estimation of Vehicle Side-Slip Angle Using an Artificial Neural Network. MATEC Web of Conferences, 166, 02001. https://doi.org/10.1051/matecconf/201816602001

[62] Da Lio, M., Bortoluzzi, D., & Rosati Papini, G. P. (2019). Modelling longitudinal vehicle dynamics with neural networks. Vehicle System Dynamics, 58(11), 1675–1693. https://doi.org/10.1080/00423114.2019.1638947

[63] Karri, V., & Butler, D. (2002). Using artificial neural networks to predict vehicle acceleration and yaw angles. Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP '02. https://doi.org/10.1109/iconip.2002.1199007

[64] Assi, A. D. (2015). Modeling Lateral Motion of a Vehicle Using Neural Networks technique. Journal of Kerbala University, Vol 13., No 4.

[65] Yim, Y., & Oh, S. Y. (2004). Modeling of Vehicle Dynamics From Real Vehicle Measurements Using a Neural Network With Two-Stage Hybrid Learning for Accurate Long-Term Prediction. IEEE Transactions on Vehicular Technology, 53(4), 1076–1084. https://doi.org/10.1109/tvt.2004.830145

[66] Şener, H., Savci, İ. H., and Tümerdem, U. (2021). Identification of Vehicle Lateral Dynamics in Carsim with a Deep Neural Network. 1st International Congress on Artificial Intelligence and Data Science, 2021 ICADA 2021, https://icada2021.com

[67] Logitech MOMO Racing Force Feedback Wheel. (2021). Logitech MOMO. Retrieved October 11, 2021, from https://support.logi.com/hc/en-au/articles/360024322313

APPENDIX A: Selected Race Vehicle Parameters

Selected race vehicle shown in were as follows in Table A.**1**:

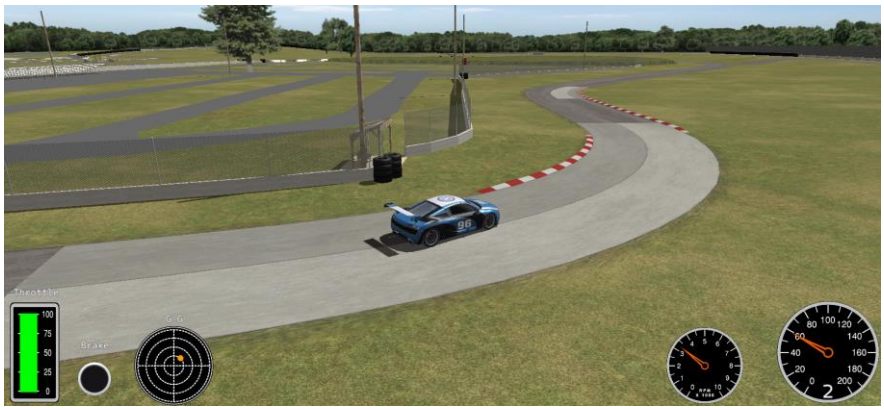


Figure A.1. Selected race vehicle in the race track

Table A.1. Race vehicle parameters

Parameter	Value
Front axle distance from COG, l_f	1.455 m
Rear axle distance from COG, l_r	1.195 m
Yaw moment of inertia, I_z	1065.2 kg.m^2
Mass of the vehicle	1200 kg
Effective vehicle total tractive force, K_{pt}	38500 N
Rolling resistance coefficient, a_r	0.10 N
Rolling and driveline resistance coefficient, b_r	2.25 Ns/m
Aerodynamic drag coefficient, c_r	0.8 Ns^2/m^2
Gear Type	Automatic 6 Speed
Rear Tire Types	305/30-19
Front Tire Types	235/35-19

CURRICULUM VITAE

PERSONAL INFORMATION

Name & Surname : Hasan Şener

EDUCATION

January 2022	MSc
	Faculty of Engineering, Department of Mechanical Engineering
	Marmara University, Istanbul, Turkey
June 2017	BSc
	Faculty of Engineering, Department of Mechanical Engineering
	Marmara University, Istanbul, Turkey