DEGREE PROJECT IN MECHANICAL ENGINEERING
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2021

Trajectory planning and control of an autonomous race vehicle

TOBIAS SJÖLIN

ALEXANDER SUNDBERG

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF INDUSTRIAL ENGINEERING AND MANAGEMENT

KTH Industriell teknik och management

Examensarbete TRITA-ITM-EX 2021:230

Banplanering och kontroll av ett autonomt racingfordon

Tobias Sjölin

Alexander Sundberg

Godkänt	Examinator	Handledare
2021-06-14	Lei Feng	Masoumeh Parseh
	Uppdragsgivare	Kontaktperson
	Cybercom Group AB	Matilda Thorburn

Sammanfattning

Det råder inga tvivel om att autonoma fordon håller på att utvecklas i en snabb takt inom flera olika områden samtidigt. För alla sorters snabba autonoma fordon så är manövrering på friktionsgränsen nödvändig eftersom det ökar fordonets manövreringsförmåga. Ökad manövreringsförmåga tillåter skarpare styrning vid högre hastigheter och därmed förbättrad säkerhet. Autonom racing är ett område där fordon uppmanas till att köra nära friktionsgränsen för att på så vis uppnå lägst varvtid. Samma tekniker som används på racerbanan kan sedan integreras till andra områden så som säkerhetssystem.

Alla autonoma fordon kräver omgivningsuppfattning, lokalisering, kartläggning, banplanering och kontroll. Varje enskild del krävs, däremot när man undersöker autonom racing är banplanering och kontroll mest intressant eftersom de har störst effekt på varvtiden. Därför undersöks två banplanerare med avseende på deras racingförmågor, känslighet mot felmodellerad massa och undvikande av hinder. Den första banplaneraren kallas Feedforward-feedback (FF-FB). Den är uppbyggd via en tre stegs process som består av ruttplanering, hastighetsplanering och kontroll som räknas ut en åt gången. Den andra banplaneraren kallas 2-lagers Model predictive control (MPC). Den är uppbyggd som en två stegs process, bestående av ett optimeringsproblem som beräknar rutt och hastighetsplanering i samma steg och sedan kontrollerar mot det. Banplanerarna är uppbyggda, integrerade och utvärderade i en Robot Operating System (ROS) baserad simuleringsmiljö som kallas Formula Student Simulator (FSSIM).

Båda banplanerana bevisar att det är möjligt att kontrollera ett fordon nära friktionsgränsen på ett kontrollerat sätt. Deras prestanda skiljer sig åt beroende på tillämpningsområde. FF-FB banplaneraren lyckades åstadkomma lägre varvtider på en racerbana förutsatt att alla banans premisser var kända. Å andra sidan, visade 2-lagers MPC:n att det var möjligt att uppnå hyggliga varvtider på en racerbana utan samma krav på kända förutsättningar. När ett hinder var introducerat i racerbanan så klarade 2-lagers MPC:n att undvika hindret vid betydligt lägre detektionsavstånd i jämförelse med FF-FB banplaneraren. Tilläggningsvis genomfördes ett viktändringstest vilket visade att MPC:n är känsligare mot minskad massa i jämförelse med FF-FB banplaneraren och FF-FB banplaneraren är känsligare mot ökad massa.

Eftersom deras prestanda skiljde sig åt beroende på tillämpningsområde så kan vidareutveckling inom ett visst område föredra den ena före den andra. Om målet är att uppnå kortast möjlig varvtid inom autonom racing skulle FF-FB banplaneraren vara att föredra. Om målet istället är att utveckla ett säkerhetssystem i ett fordon så skulle troligtvis MPC:n vara det givna valet.

KTH Industrial Engineering and Management

Master of Science Thesis TRITA-ITM-EX 2021:230

Trajectory planning and control of an autonomous race vehicle

Tobias Sjölin

Alexander Sundberg

Approved	Examiner	Supervisor
2021-06-14	Lei Feng	Masoumeh Parseh
	Commissioner	Contact person
	Cybercom Group AB	Matilda Thorburn

Abstract

There is no doubt that autonomous vehicles are developing in a fast pace in several sectors and markets. For any fast-driving autonomous vehicle the capability of operating close to the limits of traction is essential since it increases manoeuvring capabilities. Increased manoeuvring capabilities allow for increased vehicle handling at higher velocities and in turn improved vehicle safety. Autonomous racing is a field where vehicles are forced to drive close to friction limits to reduce completion times. The same techniques used on a racetrack can further be integrated into other areas such as safety systems.

All autonomous vehicles require perception, localization, mapping, trajectory planning and control. Each part is essential, although when considering autonomous racing planning and control are particularly important. Therefore, this thesis investigates two different trajectory planners and their racing capabilities, sensitivity against mass alteration and obstacle avoidance capabilities. The first trajectory planner is called the Feedforward-feedback (FF-FB) trajectory planner. It is built up with a three-step process consisting of path planning, velocity planning and control one at a time. The second approach is called the 2-layer Model predictive control (MPC). It is built up of a two-step process consisting of an optimization problem solving path planning and velocity planning as one and thereafter, control. The trajectory planners are built, integrated, and evaluated in a Robot Operating System (ROS) based simulation environment called Formula Student Simulator (FFSIM).

Both trajectory planners proves that it is possible to handle a vehicle close to the limits of traction in a satisfactory manner. Their performance differentiates depending on the subject area. The FF-FB trajectory planner proved to achieve lower completion times on a racetrack considering known circumstances. On the other hand, the 2-layer MPC proved to achieve decent completion times on a racetrack without the same requirement on known circumstances. When an obstacle was introduced to a track the 2-layer MPC was able to detect at a significantly shorter detection distance compared to the FF-FB trajectory planner. Additionally, a mass alteration test shown that the 2-layer MPC is more sensitive to reduced mass in comparison to the FF-FB while the FF-FB trajectory planner is more sensitive to increased mass compared to the MPC.

Since the performance differentiated, further development within a specific area could favour one over the other. If the only objective is a short completion time in a race for autonomous vehicles, the FF-FB could be advantageous given known circumstances, while considering use within a safety system, the MPC would probably be preferable.

Foreword

In this chapter the authors acknowledge all people assisting or helping throughout the course of this project.

Thanks to Cybercom AB for giving us this research opportunity, as well as supplying research material and guidance. A special thanks to our Cybercom supervisor Matilda Thorburn for advising us throughout this project and always driving the project forward.

Thanks to KTH, our course supervisor Fredrik Asplund and our examiner Lei Feng for helping us finalize our research question and the scope of the project. A special thanks to our KTH supervisor Masoumeh Parseh for the continuous feedback and discussions making sure the project was correctly directed. An additional thanks to Lars Svensson for providing a code base and thus simplifying the project vastly.

Tobias Sjölin, Alexander Sundberg

Stockholm, June 2021

Abbreviations

FF-FB	*Feedforward-feedback*
MPC	*Model Predictive Control*
ROS	*Robot Operating System*
FSSIM	*Formula Student Simulator*
COG	*Centre of gravity*
QP	*Quadratic Programming*
OSQP	*Operator Splitting Quadratic Program*
ADMM	*Alternating Direction Method of Multipliers*
SLSQP	*Sequential Least-Squares Quadratic Programming*

Notations

Symbol	Description
x, y	Global Cartesian-coordinates (m)
x',y'	Global velocity of COG in Cartesian-coordinates ($\frac{m}{s}$)
V_x, V_y, V	Longitudinal (V_x), lateral (V_y) and resulting (V) velocities ($\frac{m}{s}$)
$V_{x,des}$	Desired longitudinal velocity given by velocity profile ($\frac{m}{s}$)
V_t	Velocity along trajectory ($\frac{m}{s}$)
s	Distance along track in Frenet-coordinates (m)
d	Distance from centreline at distance s in Frenet-coordinates (m)
$\kappa(s)$	Curvature of track at distance s ($\frac{1}{m}$)
m	Vehicle mass (kg)
$w_{vehicle}$	Vehicle width (m)
w_{track}	Track width (m)
m_{wheel}	Wheel mass (kg)
r_{wheel}	Wheel radius (m)
g	Gravitational constant ($\frac{m}{s^2}$)
l_f, l_r	Distance between COG to front and rear wheels (m)
L	Distance between front and rear wheel axle (m)
h	Distance to COG from ground plane (m)

I_z	Vehicle yaw inertia ($\frac{kg}{m^2}$)
I_w	Wheel inertia ($\frac{kg}{m^2}$)
μ	Friction constant between road and tire ($-$)
δ	Steering angle (rad)
δ'	Steering rate ($\frac{rad}{s}$)
φ	Yaw angle (rad)
φ'	Yaw rate ($\frac{rad}{s}$)
φ''	Yaw acceleration ($\frac{rad}{s^2}$)
β	Slip angle at COG (rad)
t	Time (s)
T_s	Sample time (s)
α_f, α_r	Slip angle at front and rear wheel (rad)
$F_{x,f}$, $F_{x,r}$	Longitudinal force at front and rear wheels (N)
$F_{y,f}$, $F_{y,r}$	Lateral force at front and rear wheels (N)
$F_{z,f}$, $F_{z,r}$	Vertical force at front and rear wheels (N)
a_{long}	Longitudinal acceleration of vehicle ($\frac{m}{s^2}$)
$F_{d,f}$	Drag forces at front wheels (N)
φ_e	Angular deviation from the centreline (rad)
C_r, C_f	Tire coefficient at front and rear wheels ($\frac{N}{rad}$)
d_{wind}	A coefficient describing the air resistance ($\frac{Ns^2}{m}$)
N	Prediction horizon ($-$)
S	Planning distance (m)
S_{locked}	Locking distance (m)
$\vec{p}_x$	Vector of x-coordinates along the reference path ($[m, m]$)
$\vec{p}_y$	Vector of y-coordinates along the reference path ($[m, m]$)
$F_{x,max,acc}$	Maximal longitudinal acceleration ($\frac{m}{s^2}$)
$F_{x,max,dec}$	Maximal longitudinal deceleration ($\frac{m}{s^2}$)
F_{acc}	Required acceleration force ($\frac{m}{s^2}$)
F_{wind}	Resulting drag force originating from the air resistance (N)
F_{roll}	Resulting drag force originating from the rolling resistance (N)
B	Tire stiffness coefficient ($-$)
C	Tire shape coefficient ($-$)
D	Tire peek coefficient ($-$)
E	Tire curvature coefficient ($-$)

e_{LA}	Projected lookahead error (m)
x_{LA}	Lookahead distance (m)
$e_{LA,path}$	Lookahead error along path (m)
φ_{LA}	Lookahead error-angle towards path (rad)
$k_{p,lat}$	Proportional gain for lateral feedback controller ($-$)
$k_{p,long}$	Proportional gain for longitudinal feedback controller ($-$)
g_{LA}	Lookahead gain ($-$)
m_{it}	Number of iterations controlled towards for the 2-layer MPC ($-$)

Table of contents

1 Introduction

This chapter describes the background, purpose, research question and the methodology. The delimitations of the project are also presented.

1.1 Background

Cybercom is working on a smart and connected city where technologies linked to autonomous racing could play a key role. In the near future, autonomous cars might become as good or even better than human drivers. Although, for autonomous cars to become reliable and safe, the vehicles must be capable of operating at the limits of traction to increase manoeuvring capabilities. Increased manoeuvring capabilities would increase the safety margin of the vehicle since it would allow for faster deceleration and obstacle avoidance. Autonomous racing is an interesting area since it requires the vehicle to drive at friction limits to reduce completion times. The same techniques used on the racetrack can thus be used when considering collision avoidance. When considering collision avoidance, the path and velocity profiles must be in sync to obtain an achievable trajectory that avoids the obstacle. The trajectory planner should furthermore be able to follow the path and velocity profile generated while avoiding sharp steering or acceleration inputs, resulting in a stable path around the track without exceeding the tractions limits. The goal of this project is to develop and test two different trajectory planners' performance.

1.2 Purpose

For any autonomous vehicle, a few abilities are required. The vehicle must contain perception to generate information about the surroundings. To process the information of the perception and create an understanding of the surroundings, localization and mapping are required. Thereafter, the vehicle requires a trajectory planner describing how to achieve the goal set, which is subsequently followed using a regulator. A trajectory planner is a combination of both a path and a velocity planner.

The project is simulation based. A simulation environment providing built-in perception, localization, and mapping, as well as the ability to create customized racetracks and customized vehicle models is therefore desired. The Robot Operating System (ROS) framework provides several simulation environments of the kind.

On top of the simulation base, path planning, velocity planning and regulation is implemented. The focus of this project is reducing completion times by optimizing two different trajectory planners. One feedforward-feedback trajectory planner (FF-FB) and one 2-layer Model Predictive Controller (MPC). The FF-FB trajectory planner is based on a step-by-step approach that creates a path to follow, thereafter, generates a velocity profile and lastly control towards the planned trajectory. The MPC approach, on the other hand, solves the optimization problem based on both path and velocity simultaneously, afterwards low-level controllers handle following the trajectory generated. A desirable trajectory for low completion times follows a racing line where the mean velocity of the vehicle is the highest throughout the track, without exceeding friction limits.

The goal of the project is to thoroughly investigate the manoeuvring capabilities of two trajectory planners, the FF-FB and the 2-layer MPC, at the limits of traction. The trajectory planners are implemented in a simulation environment to compare and evaluate their performance. A discussion of the safety system applicability of the trajectory planners is furthermore held.

1.3 Research question

A suitable trajectory planner with the right controller can significantly improve racing performance as well as work as a safety system within an autonomous vehicle. The FF-FB trajectory planner and the 2-layer MPC has shown great performance when it comes to autonomous racing [1], [2], [3], [4]. The two trajectory planners have been analysed, in previous research, within different areas of applicability while using different vehicles and tracks, but no clear comparison of the two has been done. A comparison of the trajectory planners can generate precious knowledge of their advantage's respective disadvantage's when acting in different situations. Therefore, the research question of the project was formulated accordingly:

- Considering the control strategies of a feedforward-feedback trajectory planner and a 2-layer MPC, which trajectory planner has a better performance regarding the following:
    - Minimizing completion time.
    - Sensitivity against modelling errors.
    - Minimizing detection distance when an obstacle is introduced to the track.

1.4 Methodology

As mentioned in Section 1.2 , the intention of this thesis is to compare and evaluate two trajectory planners on a simulated race vehicle regarding completion times, sensitivity against modelling errors and ability of obstacle avoidance. A set of interesting driving scenarios are investigated in the scope of this project, since investigating every scenario is unfeasible, which suggests a case study methodology [5]. A case study furthermore supplies an in-depth view of a complex issue in its real-life context, which is suitable for the case at hand. Moreover, Crowe et al. [6] state that a case study approach should be considered when an experimental design is inappropriate. An experimental design requires randomizing design decisions and varying independent system variables, which is an unfeasible task in the scope of the project [7]. The case study methodology is chosen.

The case study generally consists of five stages: defining the case, selecting the case(s), collecting, and analysing the data, interpreting data, and reporting the findings. The first stage assumes deep knowledge of the subject and a carefully formulated research question derived from a thorough literature search. This stage consists of defining the case boundaries (i.e., scope, beginning, end), the environment of interest, the nature of the evidence to be collected, and the preferences when it comes to analysis and data collection [6]. The trajectory planners and vehicle model are set up during this stage.

The second stage includes reflecting around different test cases and selecting unique and interesting cases for a research purpose [6]. The type of case selection used in the scope of this project is typical cases. All the test cases chosen are thus assumed to be representative for all racing tracks [8]. A set of eight test cases are examined in the scope of this project including three different driving scenarios. Two common driving scenarios on a racetrack are the 90-degree turn track and the double hairpin track. These scenarios are chosen because of their challenging nature and because they require the vehicle to drive at its friction limits to achieve a good completion time. The 90-degree turn is fairly straight forward, the vehicle has to enter with the right speed and keep a wide turning radius. The double hairpin track is a bit more complex since the exit from the first turn influences the entry of the second turn. Four test cases are conducted using these driving scenarios, where each driving scenario is run three times with two different curvature radii. An additional set of two test cases are conducted three times each to check the trajectory planners' sensitivity against modelling errors by altering the mass of the vehicle. Most mechanical models are dependent on the mass and a mass alteration is therefore assumed to render effects throughout

the system, which is why it is chosen. The double hairpin track with small curve radius is the most challenging driving scenario, it therefore suffices to investigate the sensitivity for this scenario. The output of all the test cases mentioned above is the completion time. The two remaining test cases examined are not directly linked to racing but to obstacle avoidance. The test cases utilize the last driving scenario, which is a straight track with a sudden obstacle appearing in the middle of the track, to investigate the minimum detection distance of the trajectory planners using two different reference velocities. The minimum detection distance refers to the shortest distance the trajectory planners can detect the obstacle at and successfully avoid it. The test is conducted by lowering the detection distance of the obstacle successively until the trajectory planner fails the test. The test is conducted two times at each detection distance and the test is failed if not both runs are successful. The shortest successful detection distance at each of the reference velocities is noted regarding both trajectory planners. Further explanation of the test cases can be seen in Section 4.1.1, see Table 2.

The third stage consists of collecting the evidence rendered by the tests in stage two, mentioned above, minimum detection distance and completion time [6]. The easiest way to measure detection distances and completion times are with a quantitative approach. Times and distances need to be compared for several test cases to achieve a good understanding of the trajectory planners at hand, which is a difficult task using a qualitative approach. A qualitative visual inspection of the paths and velocities generated from the tests are however employed. Primary data are used in the scope of this project since the trajectory planners are depending on the vehicle model. Usage of secondary data where several factors of the vehicle model differ greatly would invalidate the results. The trajectory planners are furthermore applied in a simulation environment for validity. The simulation environment should supply a simplified yet realistic milieu, showing the capabilities of the trajectory planners.

The fourth and fifth stages of a case study include interpreting, analysing, and reporting of the case study results [6]. This is done by noting the completion times for each test case as well as plot the path and velocity driven by the vehicle. The performance is furthermore evaluated by comparing the completion times as well as analyse the planned and driven path and velocity profiles. The results are observed for statistical significance.

1.5 Delimitations

Controlling an autonomous vehicle by using raw sensor data to control towards a trajectory is a major task, even for the smallest vehicles. Due to the finite time-span and well-defined budget of the project several delimitations are necessary. Firstly, the project is only simulation based because construction of a test vehicle would require adaptation of perception, localization, and mapping, which is not in the scope of the project. Even at 1:10 scale it would require a lot of time and a high budget to acquire the sensors and microcontrollers necessary. Building a simulation environment from scratch is also ruled out due to time restrictions, therefore, a prebuilt simulation environment with inbuilt perception, localization, and mapping is used.

1.6 Ethics

A trajectory planner must make decisions in real time to plan and follow the trajectory generated. Such a system might encounter scenarios lacking a clearly defined solution, forming more of an ethical dilemma. A typical example is the trolley dilemma. The trolley dilemma is essentially a thought experiment consisting of a trolley hurtling down a track with no way of stopping it. Five persons are placed on the main track further ahead and one person a deviating track. The decision maker can choose to either actively change the track of the train and save the people on the main

track but still kill one person or avoid acting and let the five people die. Most people think the best option in this case is to change track and thus saving five persons while killing one. Although, with slight alterations to the formulation of the dilemma people's opinion change [9]. For instance, in case of autonomous driving, an alteration of the formulation could be the dilemma of either killing the passenger of the vehicle or five persons on the vehicle track. Is it still preferable to kill the passenger, thus saving the people on the track? If the answer is yes, riding in an autonomous vehicle might not be as appealing.

One approach of solving this ethical dilemma, in the field of autonomous driving, is using weighting functions. The difficulty then lies in how to weigh different scenarios and lives against each other. A survey published in the Nature magazine investigates how people weigh different ethical dilemmas, always including at least one casualty, depending on several factors such as quantity, age, gender, social status, lawfulness, etc [10]. The survey shows a clear difference in preferences in different regions around the world making the ethical dilemma even more complicated.

Despite the complexity of the problem, it still allows weighing according to the majority's opinion. A human on the other hand, when presented with an ethical dilemma, acts in a more chaotic and situation-based manner.

When developing a trajectory planner for public usage, the weighing of different obstacles is thus essential, where the ethical dilemma lies in the definition of the weight-function. This weight-function will not be implemented in the scope of this project, it is, however, something left for future development when preparing a trajectory planner for public usage. The thesis is furthermore made in collaboration with Cybercom Group AB which may ethically affect the content of the report, due to confidentiality and loyalty to the company.

1.7 Individual contribution

Most of the work accomplished during the project was conducted through collaboration. A few key responsibilities were however divided amongst the authors. Alexander was responsible for creating a vehicle model and implementing the FF-FB trajectory planner. Tobias was responsible for creating the customized tracks and implementing the 2-layer MPC. Close communication between the authors was nonetheless kept throughout the project, resulting in a high involvement of both authors in all design choices.

1.8 Outline

Chapter 2 presents the theoretical background of the trajectory planners which are to be implemented as well as findings of research papers and articles related to the scope of the project. The chapter furthermore presents software and simulation environments applicable for the task at hand.

Chapter 3 presents the implementation process of the trajectory planners, starting with the experiment definition, including a definition of the test cases, vehicle model parameters as well as the software and simulation specifications. A detailed description of the trajectory planner implementation is subsequently presented including trajectory planner specific parameters, software implementation and design choices.

Chapter 4 presents the results of the project when the trajectory planners are applied to using each of the eight test cases.

Chapter 5 discusses the results of the project and evaluates the trajectory planners using the research question defined.

Chapter 6 presents conclusions found in the scope of the project. The chapter furthermore presents the future work recommendations. The future work recommendations discuss the possibility of implementing the trajectory planners on a real 1:10 scale test vehicle as well as the possibility of reducing computations times.

2 Theoretical Background

This chapter describes the theoretical background necessary for full understanding of the project. The chapter introduces concepts and models used throughout the project.

2.1 Racing theory

In a race, the objective is to reach the goal as fast as possible. There are several types of races with varying road conditions and different number of contestants driving simultaneously. Although, common within autonomous racing is that every vehicle drives around the track alone. The completion times are thereafter compared, this is for instance the way it is done within the Formula student driverless competition [11]. The assumption that there are no other vehicles on the track simplifies the objective significantly since the optimal path is not dependent on other vehicles positions. The optimal path becomes the racing line, which is the most time efficient path. The most time efficient path is a combination of a short path and a high average velocity. The friction between the wheels and road are as mentioned limited. This friction limit does not matter as much on the straight sections of the track since the lateral forces are low, although, when turning the lateral forces becomes higher. The lateral force is a function of both velocity and curvature, a high curvature in turn limits the maximum drivable velocity and a high velocity limits the possible drivable curvature. A common representation of an optimal race line is the geometric racing line, which hits the apex in the middle of the curve as shown in Figure 1 by the yellow line. The apex is the point in the inner portion of a corner which the vehicle passes closest to, and it highly influence the racing line of the vehicle.

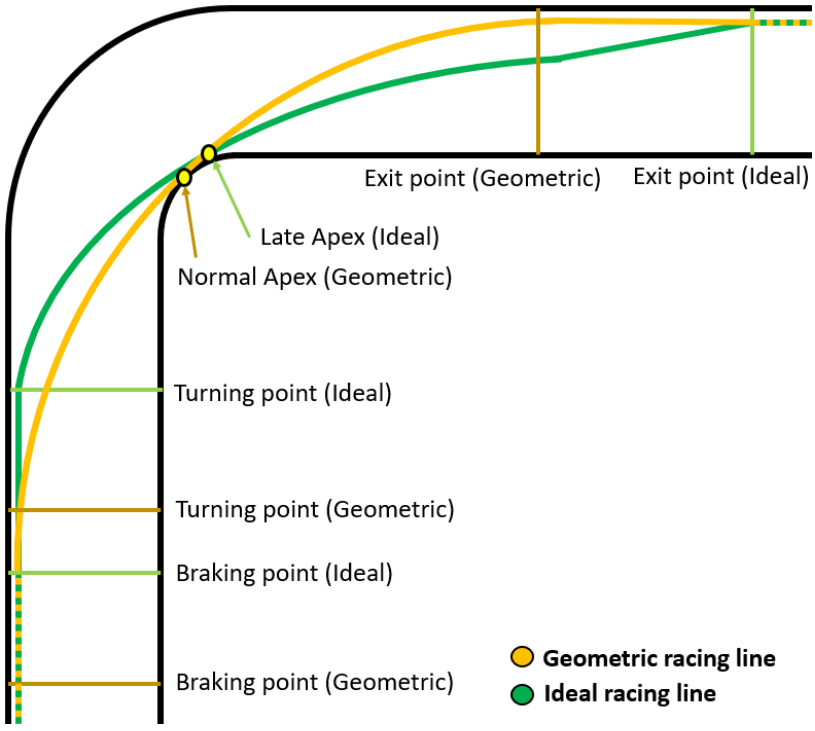

Figure 1. An illustration of the geometric and ideal racing lines.

Generally, the vehicle should brake as late as possible and keep the highest possible velocity within the curve. It should thereafter accelerate as soon and much as possible without exceeding the friction limits to exit the curve with a high velocity. As explained and shown by the yellow line in Figure 1, the geometric racing line is a common representation of an optimal racing line, although it is not always the case. Depending on the vehicle, an optimal racing line is sometimes to hit a

later apex, which is commonly called the ideal racing line. This is especially true if the vehicle at hand has a powerful drivetrain since it results in a possibility to accelerate harder [12]. An example of the ideal racing line hitting a late apex can be seen by the green line in Figure 1. As shown, the braking and turning point is moved forward which in turn allows for higher velocities further into the curve. The disadvantage of moving the apex forward is that the highest curvature will be increased and result in lowered minimum velocity. The exit point is additionally moved forward which require a longer straight section following the curve for the line to be the optimal one. The low curvature of the ideal racing line in the outgoing section of the turn allows for high acceleration before the exit point, which also is favourable.

An optimal racing line quickly becomes harder to find with additional constraints such as following turns. One example of a turn followed up by another is the double hairpin track. A hairpin, also known as a U-turn, is a straight line followed by a 180-degree turn with constant radii followed by another straight line. The double hairpin track is a combination of two hairpin turns following each other, which results in an interesting racing line. Several factors such as entry speed and vehicle parameters can alter the optimal racing line, although it usually consists of a late apex followed by a normal apex [13]. An illustration of a double hairpin track with a common optimal racing line consisting of a mix between an ideal and geometric racing line can be seen in Figure 2.

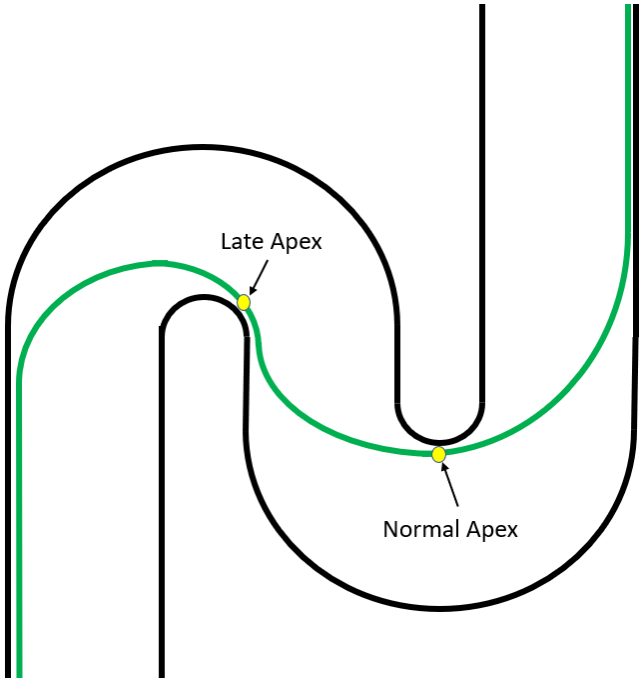

Figure 2. An illustration of a common optimal racing line for double hairpin track.

2.2 Vehicle model

Vehicle model is a collective name for the models describing the physics of the vehicle, often consisting of a car model combined with a tire model. There are several car models available employing various levels of complexity, like for instance the kinematic bicycle model and the dynamic bicycle model. The same goes for the tire models.

A common car model often utilized in the field of autonomous racing is the bicycle model [1], [3], [14], [15]. The bicycle model simplifies the physics of the car by merging both rear and front tires resulting in one tire at the front and rear axle [15]. In turn assuming the same steering angle for both front tires, neglecting the roll dynamics of the vehicle as well as assuming equal forces acting on the right and left tires. An illustration of the bicycle model is given in Figure 3.

2.2.1 Kinematic bicycle model

The kinematic bicycle model makes use of the bicycle model simplification while assuming a kinematic motion study. A kinematic motion study is a study of the motion without regard to the cause of the motion [16]. One representation of the kinematic bicycle model is given below, see Equations (1 – 5).

$$x' = Vcos(\varphi + \beta) \quad (1)$$

$$y' = Vsin(\varphi + \beta) \quad (2)$$

$$V' = a \quad (3)$$

$$\varphi' = \frac{V}{l_r}\sin(\beta) \quad (4)$$

$$\beta = \delta \cdot \tan^{-1}\left(\frac{l_r}{l_f + l_r}\right) \quad (5)$$

The parameters x' and y' are velocities in the global x respective y direction, V is their resulting velocity, φ is the angle of the vehicle, β is the angle between the longitudinal direction and the resulting velocity V, a is the acceleration, l_r and l_f are the distances between the vehicle centre of gravity (COG) and the rear axle respective the front axle of the vehicle and δ is the steering angle [15]. An illustration of the kinematic bicycle model is given in Figure 3.

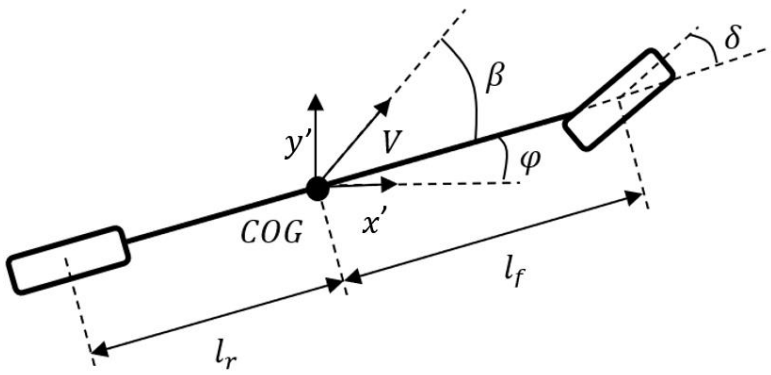

Figure 3. The kinematic bicycle model.

The equations given above is exclusively dependent of the geometry and speed of the vehicle and does not take the forces, masses, and inertias of the vehicle into consideration, resulting in a purely kinematic motion study. The model assumes no slip as well as no skip, producing a model often limited to low-speed applications [15].

2.2.2 Dynamic bicycle model

The dynamic bicycle model makes use of the bicycle model simplification while assuming a dynamic motion study. A dynamic motion study, unlike the kinematic motion study, takes the causes of motion into consideration [16]. One representation of the dynamic bicycle model suggested by Alcalá et al. [3] is given below, see Equation (6-15).

$$V_x' = a_{long} + \frac{-F_{y,f}\sin(\delta) - F_{d,f}}{m} + \varphi' V_y \quad (6)$$

$$V_y' = \frac{F_{y,f}\cos(\delta) - F_{y,r} l_r}{m} - \varphi' V_x \quad (7)$$

$$\varphi'' = \frac{F_{y,f} l_f \cos(\delta) - F_{y,r} l_r}{I_z} \tag{8}$$

$$d' = V_x \sin(\varphi_e) + V_y \cos(\varphi_e) \tag{9}$$

$$\varphi_e{}' = \varphi' - \frac{V_x \cos(\varphi_e) - V_y \sin(\varphi_e)}{1 - d\kappa}\kappa \tag{10}$$

$$F_{y,f} = C_f(\alpha_f)\alpha_f \tag{11}$$

$$F_{y,r} = C_r(\alpha_r)\alpha_r \tag{12}$$

$$F_{d,f} = \mu m g + d_{wind} V_x^2 \tag{13}$$

$$\alpha_f = \delta - tan^{-1}\left(\frac{V_y + l_f \varphi'}{V_x}\right) \tag{14}$$

$$\alpha_r = -tan^{-1}\left(\frac{V_y + l_r \varphi'}{V_x}\right) \tag{15}$$

The variable V_x is the longitudinal velocity of the vehicle, V_y is the lateral velocity of the vehicle, $F_{y,f}$ is the lateral force applied to the front wheels, $F_{y,r}$ is the lateral velocity applied to the rear wheels, φ is the angle of the vehicle, I_z is the jaw inertia of the vehicle, d is lateral vehicle deviation from the centreline and φ_e is the angular deviation of the vehicle with respect to the centreline. κ is the curvature of the centreline at the given point, C_f is the front tire coefficient, C_r is the rear tire coefficient, $F_{d,f}$ is the resulting drag force applied to the vehicle, d_{wind} is a wind resistance constant, α_f is the front tire slip angle, and α_r is the rear tire slip angle. The model presented above is a non-linear dynamic bicycle model utilizing a reformulated version of the simplified Pacejka tire model, different tires models are discussed further below in Section 2.2.3. The equations are dependent of the forces rendered from the motion as well as masses, inertias, slip angles etc. resulting in a well formulated representation of the vehicle [3]. A visual representation of the dynamic bicycle model is given in Figure 4.

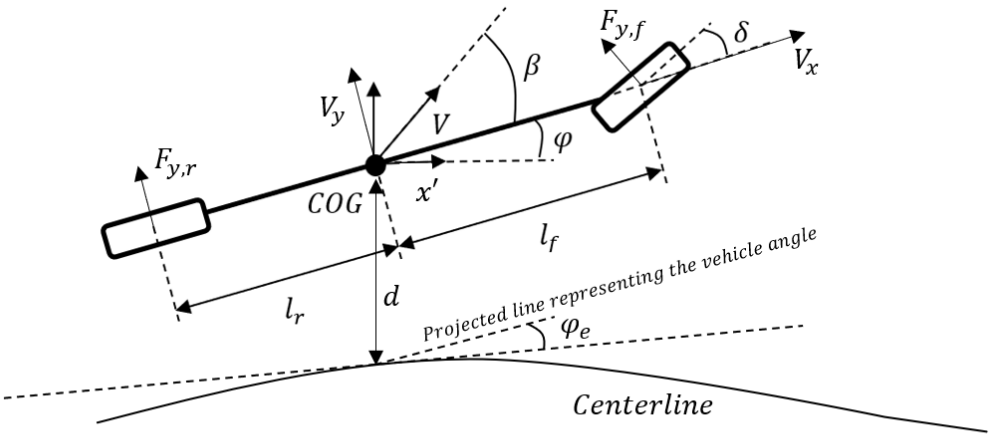

Figure 4. A visual representation of the dynamic bicycle model.

2.2.3 Tire model

The only connection between the vehicle and the road is the tires. A tire model is needed to estimate the lateral and longitudinal forces obtained when altering acceleration and steering angle. The forces are generated by deformation of the tires while cornering, accelerating, or braking. When driving straight the tire only produce longitudinal forces which are quite easy to estimate. However, when a curvature is introduced, the tires begin to slip slightly sideways while rolling which causes a slip angle. The same phenomenon causes a lateral force on the tire which is used for steering [17]. The slip angle is the angular difference between the wheel heading angle and the actual velocity of the wheel as shown in Figure 5.

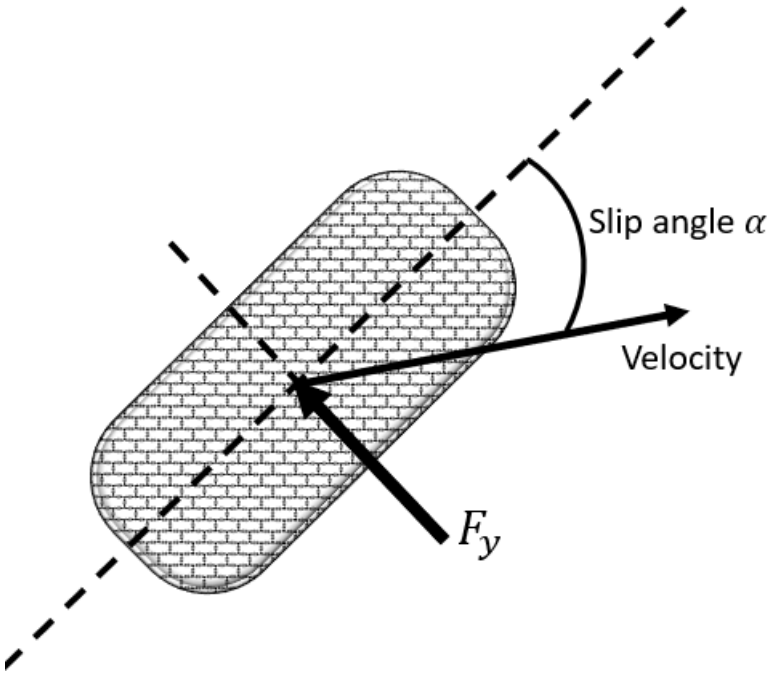

Figure 5. A representation of tire slip angle and resulting lateral force.

Normal cornering manoeuvres result in minimal sliding of the tire and thus low slip angles and lateral forces. With increased magnitude of lateral forces in the system the slip angle must increase, until it eventually begins to slide on the surface. For low-speed applications with low slip angles the tire model can be assumed to be linear. As the speed and curvature increase the lateral forces becomes higher and a more complex non-linear tire model is preferred [17]. Pacejka has created several tire models, where some of the most common are the Pacejka model, Magic formula tire model and Fiala brush tire model [18].

2.3 Frenet coordinates

The traditional Cartesian coordinate system is often used when it comes to describing a point (x,y) on a plane. However, when it comes to for instance representing road networks in a computer environment, the usage of the Cartesian coordinate system may not be suitable. The Frenet coordinate system on the other hand offers a system describing the progress of the vehicle with respect to the road centre line, s, and the lateral deviation from it, d [19].

Conversion from the Cartesian coordinate system to the Frenet coordinate system of a point X along a path, P, can be accomplished using the following equations, see Equation (16) and (17). It is, in the equations, assumed that P is an open path consisting of straight lines of length l_i, see Figure 6.

$$s = \sum_{i=1}^{u-1} l_i + \Delta l + \Delta y \cdot sin\varphi \tag{16}$$

$$d = \Delta y \cdot cos\varphi \tag{17}$$

The parameter u represents the index of the segment in which point X intersects with a line parallel to the y-axis (u=4 in Figure 6), Δl is the distance between the intersection point and the start of the segment, Δy is the distance in the y-direction between the intersection point and the point X and φ is the angle between the y-axis and a line perpendicular to the segment of intersection [20].

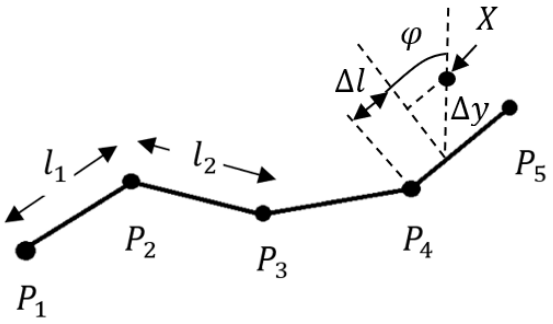

Figure 6. A path describing the transformation of a point in the Cartesian coordinate system to the Frenet coordinate system.

The transformation back to Cartesian coordinates from Frenet coordinates defers slightly from the equations presented above. The method assumes knowledge about each start and end point P_1, P_2, ... of the segments and their respective position in the Cartesian coordinate system, $x_{p,i}$ and $y_{p,i}$. The resulting equation is given below, see Equation (18) and (19).

$$x = x_{p,i} + (\Delta s - dtan\varphi)cos\varphi \tag{18}$$

$$y = y_{p,i} + (\Delta s - dtan\varphi)sin\varphi + \frac{d}{cos\varphi} \tag{19}$$

The variables $x_{p,i}$ and $y_{p,i}$ are the x respective y coordinate of P_i and Δs is the s-distance (Frenet coordinate) between P_i and the point investigated. A figure describing the current transformation of a point X is given below, see Figure 7 [20].

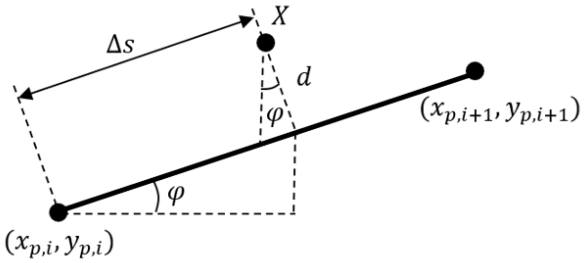

Figure 7. A graph describing the Frenet to Cartesian transformation.

2.4 Software

To properly investigate the performance of the trajectory planners, a simulation environment is required. Within this section two options of simulation environments are investigated. A decision was later made resulting in exclusive usage of the Formula Student Simulator (FSSIM) environment.

2.4.1 Robot Operating System

Robot Operating System (ROS) is a Linux based open-source, meta-operating system made for various robot related tasks. The system is made up of a flexible framework for writing and implementing software with several built-in functionalities. A key part is the communication system built-in within ROS, which allows for both communication on the same system and within the network. A normal ROS system consists of nodes, topics, services, and a master. Each node is an executable code that communicates with the master. Requests can be sent to the master to communicate with other nodes through topics or access services. The system is modular and allow replacements of modules to simplify hardware implementation [21].

2.4.2 Formula Student Simulator

FSSIM is based on a popular Gazebo software package within ROS. It is a vehicle simulator developed specifically for the Formula Student Driverless Competition, utilizing the ROS kinetic API. It uses its own physics engine to simulate how an actual vehicle would handle in a given situation. The ground is simulated as a flat plane with cones marking the outer boundaries of the track. The simulator comes with prebuilt functionalities to assist with perception, localization, and mapping. Due to computation times the perception part is simplified and mimics raw sensor data instead of reading from the environment. The software comes with a functionality allowing for easy track customization. Additionally, the vehicle model including vehicle parameters are easy to access, resulting in easy customization of the vehicle [22].

2.4.3 F1tenth Simulator

The F1tenth Simulator is a 2D simulation environment utilizing the ROS melodic API developed by the F1tenth association. The simulation environment is developed for easy imitation of a real car and prototyping of racing algorithms in a rapid manner. The simulator is furthermore designed to be conveniently interchangeable with a real vehicle, to easily test the algorithm developed in the real world. The F1tenth association moreover supplies a tutorial of how to build one of these real cars, of scale 1:10.

The simulator utilizes a simulated lidar to obtain its surroundings and thus the perception of the vehicle. The lidar data can be accessed by subscribing on one of the topics. The live pose or location of the vehicle is also accessible through one of the topics, which together with the lidar detection serves as environment mapping. The simulation environment is visualized using RVIZ [23].

3 Related Work

This chapter presents work related to the trajectory planners to be developed. The chapter presents prior research affecting the development of the trajectory planners.

3.1 Feedforward-feedback trajectory planner

Kapania presents in his dissertation from 2016 a solution to the problem of controlling an autonomous racing vehicle using a feedforward-feedback (FF-FB) trajectory planner. The trajectory planner is made up of a three-step process, consisting of path planning, velocity profile generation and lastly lateral and longitudinal control [1]. A similar approach utilizing the same three step process is also presented by Heilmer et al. 2019 [2].

For a race vehicle to obtain the shortest possible completion times it must drive a path that minimizes the length of the path while maximizing the velocity. The approach suggested by Kapania is a combination of two different path planners. The first path planner minimizes the distance while the second minimizes the curvature. By minimizing the curvature, the path in turn maximizes the achievable velocity. The completion times are, however, more sensitive to reduced velocity than reduced distance, a weight parameter is therefore used to favour the minimum curvature compared to the shortest distance. The relationship between the two path planners is shown in Equation (20), with the weight parameter value set to $0 \leq \eta \leq 1$. P is the new path, $P_{mincurve}$ is the minimum curvature path and $P_{mindist}$ is the path with the minimum distance. A weight parameter set to $\eta \approx 0$ is preferred for most sections along the track, although a more balanced weight selection is preferred in other sections. Kapania, therefore, uses another approach to obtain a function that alters the weight at different points along the track. The alteration was done using a human race driver racing line as a reference which trained the weighting function [1].

$$P = (1 - \eta)P_{mincurve} + \eta P_{mindist} \tag{20}$$

The second step of the trajectory planner is obtaining a velocity profile. The objective of the velocity profile is to achieve the highest velocities possible without exceeding the friction limit. If the planned velocity is too high the vehicle cannot follow the path and risk driving off the track. At the same time velocities below the maximum drivable velocity would result in sub-optimal completion times.

Lastly, for the vehicle to follow the path and velocity profile generated a controller is required. One approach of controlling the vehicle is with a feedforward-feedback lateral and longitudinal controller. The longitudinal controller alters the engine force while the lateral controller alters the steering input. The layout of the feedforward-feedback controller can be seen in Figure 8.

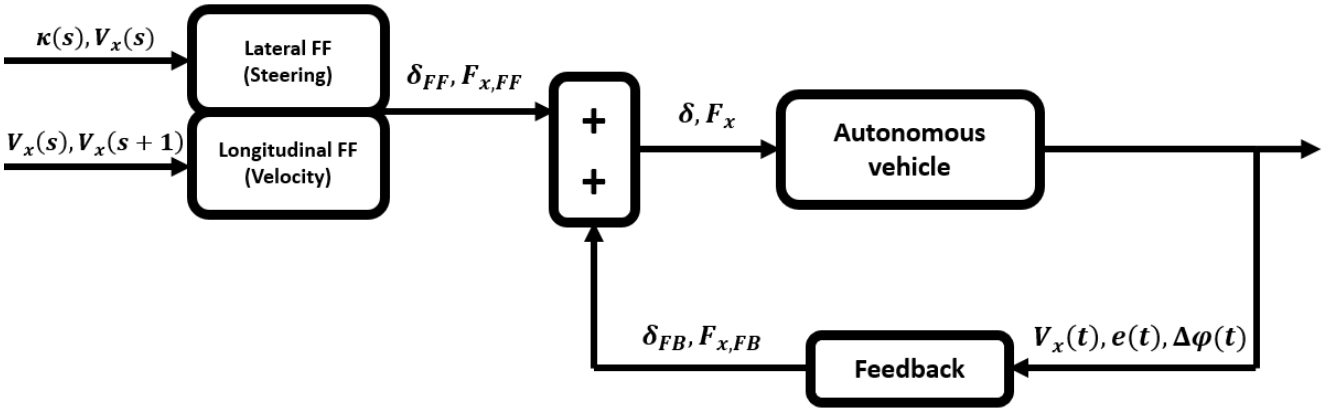

Figure 8. A representation of feedforward-feedback controller

3.1.1 A* path planner

The A* search algorithm is one of the most successful algorithms when it comes to finding the shortest possible path to a destination. The algorithm makes use of three cost functions, $g(n)$, $h(n)$ and $f(n)$, to obtain its path. The function $g(n)$ represents the exact cost of moving from the planners starting point to an arbitrary point n ahead. This cost can be in form of a distance, time, or other expense. The cost of moving from a point (0,0) to (1,1) would for instance result in $\sqrt{1^2 + 1^2}$ if distance were to be used to measure cost. The function $h(n)$ represents the heuristic estimated cost of moving from an arbitrary point n to the goal set. The heuristic to be used must be chosen depending on the application in which the path planner is to be used. One simple heuristic for the application of finding the goal state in a set x- and y-coordinate system is for instance the Euclidean distance, see Equation (21).

$$h(n) = (x_{curr} - x_{goal})^2 + (y_{curr} - y_{goal})^2 \tag{21}$$

The variable x_{curr} is the x-coordinate of the point currently investigated, y_{curr} is the y-coordinate of the point currently investigated, x_{goal} is the x-coordinate of the goal and y_{goal} is the y-coordinate of the goal. The function $f(n)$ represents the complete cost of moving to an arbitrary point n, see Equation (22).

$$f(n) = h(n) + g(n) \tag{22}$$

The algorithm essentially calculates the costs of each point accessible from the starting point respectively and thereafter observes the point with the lowest cost. The algorithm then calculates the cost of each point accessible from the entered point and thereafter observes the point with the lowest cost. This behaviour continues until the goal state is found [24].

A simplified example of how the algorithm works is shown in Figure 9. The green square is the starting position, the red squares are the path generated and the blue is the goal. All the costs are saved and shown in the squares and the grey squares are considered an obstacle.

7	6	5	6	7	8	9	10	11		19	20	21	22
6	5	4	5	6	7	8	9	10		18	19	20	21
5	4	3	4	5	6	7	8	9		17	18	19	20
4	3	2	3	4	5	6	7	8		16	17	18	19
3	2	1	2	3	4	5	6	7		15	16	17	18
2	1	0	1	2	3	4	5	6		14	15	16	17
3	2	1	2	3	4	5	6	7		13	14	15	16
4	3	2	3	4	5	6	7	8		12	13	14	15
5	4	3	4	5	6	7	8	9	10	11	12	13	14
6	5	4	5	6	7	8	9	10	11	12	13	14	15

Figure 9. A simplified example illustrating the functionality of A* [25].

3.1.2 Minimum curvature path planner

The path with minimized curvature will generate a path close to the geometric racing line presented in Section 2.1, thus obtaining a close to optimal racing line. The approach presented by Kapania in his dissertation from 2016 is a way of minimizing the curvature as an iterative process. The planner generates a velocity profile from a path and then updates the path using the velocity profile [1]. This is suboptimal when considering comparison with a Model Predictive Controller since it requires several iterations to render sufficient results. Therefore, an alternative approach presented by Heilmeier et al. that uses an optimization problem to obtain the path with the minimized squared sum of the curvature along the path is considered [2]. The optimization problem considered is shown in Equation (23), where κ is the curvature and d is the displacement of points along the centreline. The track boundaries d_{min} and d_{max} are set to be fixed values since width of the track is constant. The parameter S is the planning distance.

$$\begin{aligned} &\underset{[d_1,\dots,d_S]}{minimize} && \sum_{i=1}^{S} \kappa_i^2(t) \\ &subject\ to && d_i \epsilon [d_{min}, d_{max}] \quad \forall\ 1 \le i \le S \end{aligned} \tag{23}$$

The path planner initially requires the centreline of the track, which has to be built up by sequential points, including the heading angle at each of these points. The heading angle is further used to calculate the normalized normal vector at each point. Thereafter, the points can be moved along the normal vector to obtain a new position that minimizes the curvature. In Figure 10, the centreline is given by the grey points p, the normal vectors by $\vec{n}$ and the minimized curvature by the green points r. Each point r_i can be obtained using Equation (24), where $\vec{p}_i = [x_i, y_i]$ [2].

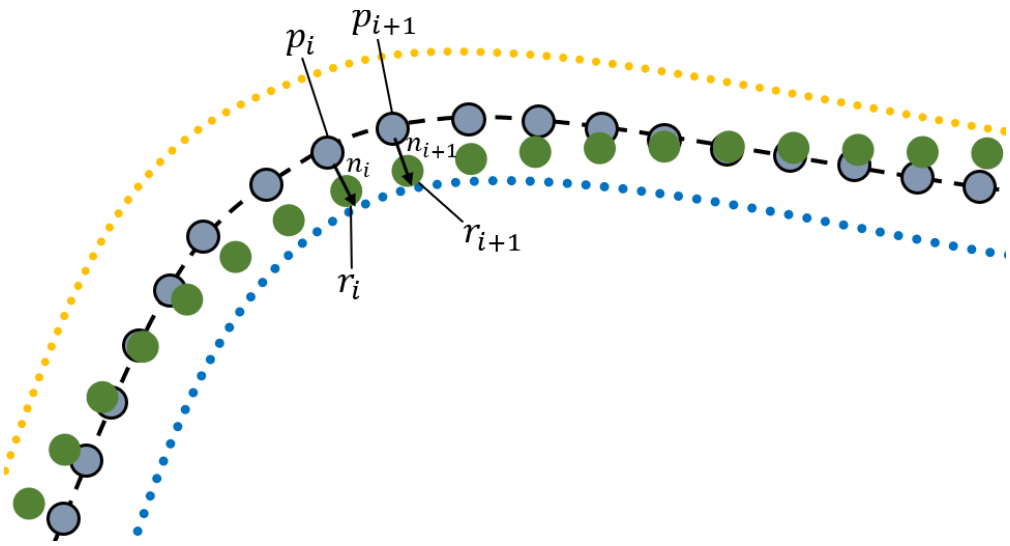

Figure 10. An example of points along centreline and how they are moved to obtain the minimum curvature.

$$\vec{r}_i = \vec{p}_i + d_i\vec{n}_i \tag{24}$$

To obtain the optimal distances at each point, d_i must be calculated which in turn means the squared curvature needs to be obtained. Equation (25) approximates the unsquared curvature at a point i along the track. The squared curvature shown in Equation (26) can thereafter be obtained from Equation (25).

$$\kappa_i = \frac{x_i' y_i'' - y_i' x_i''}{(x_i' + y_i')^{\frac{3}{2}}} \tag{25}$$

$$\kappa_i^2 = \frac{x_i'^2 y_i''^2 - 2x_i' x_i'' y_i' y_i'' + y_i'^2 x_i''^2}{(x_i'^2 + y_i'^2)^3} \tag{26}$$

To calculate the curvature, an approximation of $x_i', x_i'', y_i' \; and \; y_i''$ is required. This can be done using cubic splines, shown in Equation (27). Each distance between a set of two points have a $a_i, b_i, c_i \; and \; d_i$ parameter to represent the curve between the two points. The distance along the path s is replaced by the normalized curvilinear parameter t_i(s) to simplify the equation at the intersection points between splines. At the start of a spline $s = s_{i0}$ which results in $t = 0$, similarly $s = s_{i1}$ at the end of a spline which results in $t = 1$.

$$\begin{gathered} x_i(t) = a_i + b_i t + c_i t^2 + d_i t^3 \\ x_i'(t) = b_i + 2c_i t + 3d_i t^2 \\ x_i''(t) = 2c_i + 6d_i t \\ t_i(s) = \frac{s - s_{i0}}{\Delta s_i} \end{gathered} \tag{27}$$

A set of six conditions is generated to be able to solve the parameter values, the conditions can be observed in Table 1. The conditions shown are for the x-values, although the same conditions can be applied for the y-values and thus the resulting equations shown can be used. The set of conditions can be used to form two matrices A and b so that $A\vec{z} = b$. The solution vector $\vec{z} = [a_0, b_0, c_0, d_0, \dots, a_S, b_S, c_S, d_S]$ represents the resulting splines, from spline 1 to spline S. Condition (1) and (6) are start and end conditions and are therefore only applied once while the intersection conditions (2-5) will repeat once every spline [2].

Table 1. Conditions within cubic splines

Nr	Condition	Resulting equation
1	$x_0''(0) = 0$	$2c_0 = 0$
2	$x_i(0) = a_i$	$a_i = x_i(0), y_i(0)$
3	$x_i(1) = x_{i+1}(0)$	$a_i + b_i + c_i + d_i = x_i(1), y_i(1)$
4	$x_i'(1) = x_{i+1}'(0)$	$b_i + 2c_i + 3d_i = b_{i+1}$
5	$x_i''(1) = x_{i+1}''(0)$	$2c_i + 6d_i = 2c_{i+1}$
6	$x_S''(1) = 0$	$2c_S + 6d_S = 0$

The optimization problem in Equation (23) can be reformulated by inserting Equation (26) into it. The resulting equation can be broken up into three parts, see Equation (28). Each of the three parts can thereafter be transformed into vector and matrix form by extracting the second derivative vectors of x and y. The values of x' and y' can be approximated to be constant since the path heading only changes slightly from the reference path. This in turn allows the P matrices shown in Equation (29) to work as a kind of linearization along the reference path. The modified optimization problem can be observed in Equation (28), where $x'' = [x_1'', x_2'', \dots, x_S'']$ and $y'' = [y_1'', y_2'', \dots, y_S'']$.

$$\begin{aligned} &Minimize \\ &[d_1, \dots, d_S] \end{aligned} \quad \vec{y}''^T P_{yy} y'' + \vec{y}''^T P_{xy} \vec{x}'' + \vec{x}''^T P_{xx} \vec{x}'' \qquad (28)$$
$$subject\ to \quad d_i \epsilon [d_{min}, d_{max}] \quad \forall\ 1 \le i \le S$$

$$P_{yy} = \begin{bmatrix} \frac{x_1'^2}{(x_1'^2 + y_1'^2)^3} & 0 & \cdots & 0 \\ 0 & \frac{x_2'^2}{(x_2'^2 + y_2'^2)^3} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{x_S'^2}{(x_S'^2 + y_S'^2)^3} \end{bmatrix}$$

$$P_{xy} = \begin{bmatrix} \frac{-2x_1'y_1'}{(x_1'^2 + y_1'^2)^3} & 0 & \cdots & 0 \\ 0 & \frac{-2x_2'y_2'}{(x_2'^2 + y_2'^2)^3} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{-2x_S'y_S'}{(x_S'^2 + y_S'^2)^3} \end{bmatrix} \qquad (29)$$

$$P_{xx} = \begin{bmatrix} \frac{y_1'^2}{(x_1'^2 + y_1'^2)^3} & 0 & \cdots & 0 \\ 0 & \frac{y_2'^2}{(x_2'^2 + y_2'^2)^3} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{y_S'^2}{(x_S'^2 + y_S'^2)^3} \end{bmatrix}$$

The values of x_i' and y_i' can be calculated using Equation (30). The matrix A^{-1} is the inverse matrix of A, $A_{ex,b}$ is the extraction matrix and b_x, b_y are the solution vectors. The extraction matrix $A_{ex,b}$ is shown in Equation (31). The extraction matrix is used obtain the spline values b_i, which can be obtained from Equation (27) due to the condition: $x_i'(0) = b_{i,x}$ and $y_i'(0) = b_{i,y}$.

$$\begin{aligned} x_i' &= A_{ex,b}A^{-1}b_x \\ y_i' &= A_{ex,b}A^{-1}b_y \end{aligned} \tag{30}$$

$$A_{ex,b} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & 2 & 3 \end{bmatrix} \tag{31}$$

To obtain knowledge about the values of x_i'' and y_i'' depending on the value of d_i, a set of matrices must be formulated. Equation (32) represents the change of x_i'', although the same equation can be used for y_i'' with a replaced $\vec{p}_x$ and N_x to the corresponding y representation. In Equation (32), $\vec{p}_x$ is the solution vector for the reference path given by the centreline and N_x is a matrix representing the normalized normal vectors. In turn, the previous solution vector b is reformulated to $b_{new} = \vec{p}_x + N_x d$. The extraction matrix $A_{ec,x}$ is further used to extract all values of c_i since $x_i''(0) = 2c_i$, the matrix can be seen in Equation (33).

$$\begin{bmatrix} x_1'' \\ x_2'' \\ \vdots \\ x_{S-1}'' \\ x_S'' \end{bmatrix} = 2A_{ex,c}A^{-1} \left(\underbrace{\begin{bmatrix} 0 \\ p_{1,x} \\ p_{2,x} \\ 0 \\ 0 \\ p_{2,x} \\ p_{S-1,x} \\ 0 \\ 0 \\ \vdots \\ p_{S-1,x} \\ p_{S,x} \\ 0 \end{bmatrix}}_{\vec{p}_x} + \underbrace{\begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ n_{1,x} & 0 & \dots & 0 & 0 \\ 0 & n_{2,x} & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \\ 0 & n_{2,x} & \dots & 0 & 0 \\ 0 & 0 & \dots & n_{S-1,x} & 0 \\ 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & n_{S-1,x} & 0 \\ 0 & 0 & \dots & 0 & n_{S,x} \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}}_{N_x} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{S-1} \\ d_S \end{bmatrix} \right) \tag{32}$$

$$A_{ex,c} = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 2 & 6 \end{bmatrix} \tag{33}$$

Equation (32) reformulated using a vector formation can be seen in Equation (34), where $T_c = 2A_{ex,c}A^{-1}$, $T_{N,x} = 2A_{ex,c}A^{-1}N_x$ and $T_{N,y} = 2A_{ex,c}A^{-1}N_y$.

$$\begin{aligned} \vec{x}'' &= T_c\vec{p}_x + T_{N,x}\vec{d} \\ \vec{y}'' &= T_c\vec{p}_y + T_{N,y}\vec{d} \end{aligned} \tag{34}$$

The complete optimization problem to be solved is, lastly, obtained by inserting Equation (34) into Equation (28) and converting it into a quadratic programming (QP) formulation, see Equation (35). The matrices $H_x, H_{xy}, H_y, f_x, f_{xy}$ and f_y are presented in Equation (36). In the process of converting the optimization problem into a QP formulation a set of constants are removed [2].

$$\begin{aligned} &Minimize \\ &[d_1, \dots, d_S] \end{aligned} \quad \frac{1}{2}\,\vec{d}^T\left(H_x + H_{xy} + H_y\right)\vec{d} + \left(f_x + f_{xy} + f_y\right)^T\vec{d} \qquad (35)$$
$$subject\ to \quad d_i \epsilon [d_{min}, d_{max}] \quad \forall\ 1 \le i \le S$$

$$\begin{aligned} H_x &= T_{N,x}^T P_{xx} T_{N,x} \\ H_{xy} &= T_{N,y}^T P_{xy} T_{N,x} \\ H_y &= T_{N,y}^T P_{yy} T_{N,y} \\ f_x &= 2T_{N,x}^T P_{xx}^T T_c \vec{p}_x \\ f_{xy} &= T_{N,y}^T P_{xy}^T T_c \vec{p}_x + T_{N,x}^T P_{xy}^T T_c \vec{p}_y \\ f_y &= 2T_{N,y}^T P_{yy}^T T_c \vec{p}_y \end{aligned} \qquad (36)$$

3.1.3 Velocity profile generation

Given a fixed reference path, the next step is to determine a minimum-time velocity profile. The speed profile must keep the highest possible velocity without exceeding the friction limit. The solution presented in this section is an algorithm that uses the "three-pass" approach that originated from Velenis and Tsiotras [26] and Griffiths [27], improved upon by Subosits and Gerdes [28], and further described by Kapania [1]. The minimum-time velocity profile is a profile describing the highest possible vehicle velocity at each point along the track without exceeding the available friction limit. The velocity profile considers the approach of reaching the velocity in each point considering the required velocities in the adjacent points. The permissible friction shown in Equation (37) can be used to limit the lateral forces F_x and longitudinal forces F_y. The relationship can be described by the friction circle, which can be applied to the front or rear wheels as a pair or for each wheel one by one [1].

$$\begin{aligned} F_{x,r}^2 + F_{y,r}^2 &\le (\mu F_{z,r})^2 \\ F_{x,f}^2 + F_{y,f}^2 &\le (\mu F_{z,f})^2 \end{aligned} \qquad (37)$$

Equation (38) can be formulated using a dynamic bicycle model. Furthermore, Equation (39) is obtained by taking the time derivative of d' and substituting in Equation (38), d is the lateral deviation of the vehicle from the centreline. By observing steady state cornering condition and assuming small steering angles, the equation can be simplified by setting $s' = V_x$ and $d'' = 0$. Equation (40) is thereafter obtained by inserting the relationship between front and rear tyre forces.

$$\begin{aligned} d' &= V_x(\beta + \varphi_e') \\ \varphi'' &= \frac{l_f F_{y,f} - l_r F_{y,r}}{I_z} \\ \beta' &= \frac{F_{y,f} + F_{y,r}}{mV_x} - \varphi' \\ \varphi_e'' &= r - s'\kappa \end{aligned} \qquad (38)$$

$$d'' = \frac{F_{y,f} + F_{y,r}}{m} - V_x \kappa s' \qquad (39)$$

The first pass out of the three checks the maximum allowed vehicle velocity excluding longitudinal forces shown in Equation (37). Equation (41) describes the maximum allowed vehicle velocity at each distance s along the track and is obtained by inserting Equation (40) into (37). To simplify the velocity profile, the topography and weight transfer is neglected.

$$F_{y,f} = \frac{ml_r}{L} V_x^2 \kappa$$
$$F_{z,f} = \frac{ml_r}{L} \tag{40}$$

$$V_x(s) = \sqrt{\frac{\mu g}{|\kappa(s)|}} \tag{41}$$

The second pass calculates the maximum longitudinal force $F_{x,max,acc}$. In [1], $F_{x,max,acc}$ is calculated by accounting for the maximum engine force and the lateral tire force demand. The second pass is a forward iteration step iterating forward and checking the maximum achievable speed in the upcoming step considering maximum acceleration and lateral force demand on the wheels. The forward iteration step can be seen in Equation (42). The velocities generated at each point are compared where the lowest out of the two is kept for the updated velocity profile.

$$V_x(s + \Delta s) = \sqrt{V_x^2(s) + 2\frac{F_{x,max,acc}}{m}\Delta s} \tag{42}$$

The third and final pass consists of a backward iteration step. It iterates backwards and checks the maximum allowable velocity in the previous step considering maximum deceleration and lateral force demand on wheels. The backward iteration step can be seen in Equation (43). Similarly, to the second step the velocities generated are compared at each point and the lowest are kept and used in the final velocity profile [1].

$$V_x(s - \Delta s) = \sqrt{V_x^2(s) - 2\frac{F_{x,max,dec}}{m}\Delta s} \tag{43}$$

An example showing each of the steps in the velocity profile generation given a fixed path can be seen in Figure 11. Each pass is described in a separate graph where the filled line is the generated velocity profile and dotted line is the velocity profile from the previous iteration.

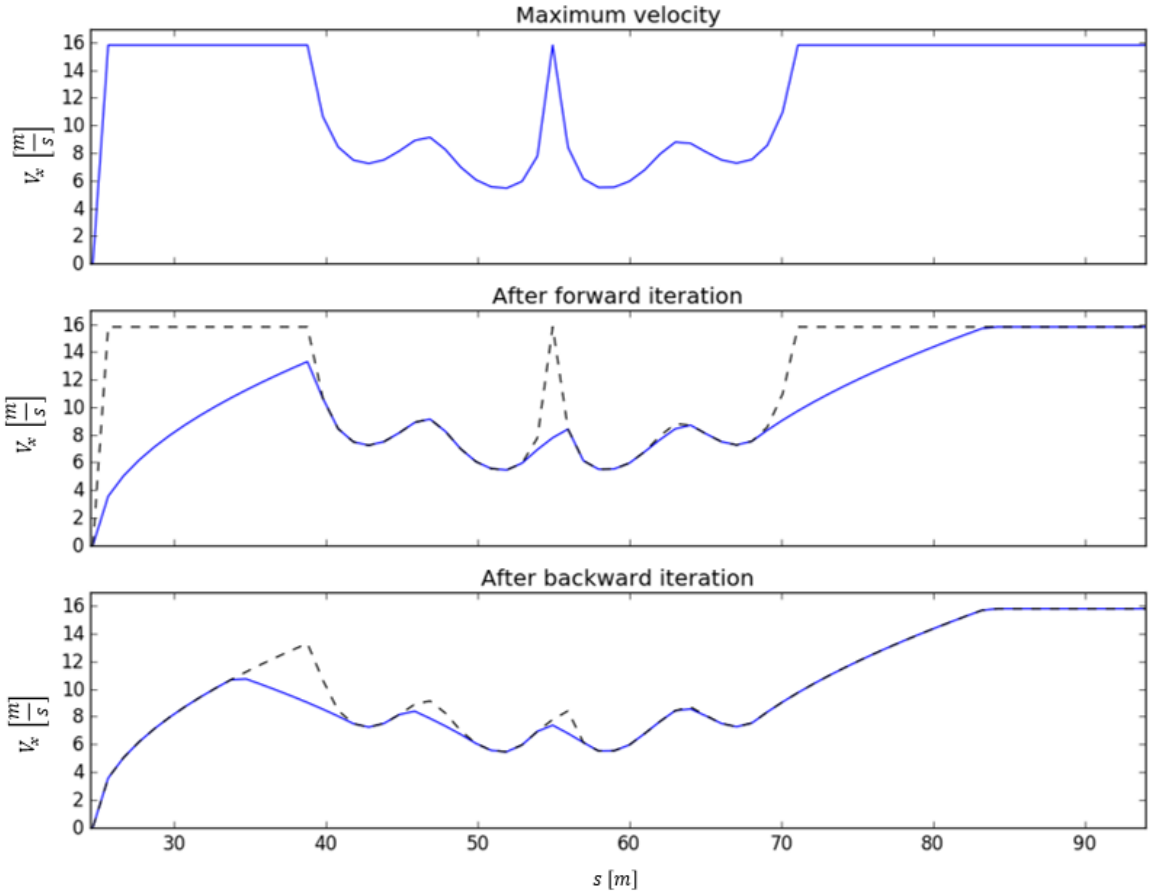

Figure 11. Graphs describing velocity profile generation given a fixed reference path.

3.1.4 Feedforward-feedback longitudinal control

The output of the feedforward-feedback longitudinal control is built up by a feedforward and a feedback part as shown in Equation (44). The feedforward section of the controller simply assumes that the controller is following the velocity profile and estimates the force required in each step along the track to keep the desired velocity given by velocity profile. This is done by accounting for acceleration and drag forces as shown in Equation (45). A complete visualization of the control structure can be seen in Figure 8 presented in Section 3., including both longitudinal and lateral control.

$$F_x = F_{x,FF} + F_{x,FB} \tag{44}$$

$$F_{x,FF} = F_{acc} + F_{wind} + F_{roll} \tag{45}$$

The forces required to compensate for the acceleration and drag can be estimated using Equation (46). The variable $V_{x,des}(s)$ is the desired velocity at the current position, $V_{x,des}(s+1)$ is the desired velocity a distance Δs ahead, Δt is the time required to drive the distance Δs, d_{roll} is a roll resistance estimation and d_{wind} is a combined wind resistance coefficient. In Equation (46), the equation for wind and rolling resistance are common equations used within the automotive sector [29], [30]. The wind and rolling resistance simplification are due to the lack of data accessible regarding the wind and rolling resistance parameters. It is however crucial that the wind resistance is dependent on the velocity squared and the rolling resistance is close to static to capture a valid approximation of the forces. The calculation used for Δt and Δs can be seen in Equation (47), where s is a distance along the path.

$$F_{acc} = \frac{m\big(V_{x,des}(s+1) - V_{x,des}(s)\big)}{\Delta t}$$
$$F_{wind} = \frac{1}{2}\rho A C_D V_{x,des}^2(s) \approx d_{wind} V_{x,des}^2(s) \tag{46}$$
$$F_{roll} = mg\left(C_R + \frac{1}{p}\left(0.01 + 0.0095\left(\frac{V_{x,des}}{100}\right)^2\right)\right) \approx mg \cdot d_{roll}$$

$$\Delta s = s(t+1) - s(t)$$
$$\Delta t = -V_{x,des}(s) + \sqrt{V_{x,des}^2(s) + 2\Delta s} \tag{47}$$

The feedback part of the longitudinal controller is a simple P-controller. It reads the current velocity and compares it with the planned velocity. The aggressiveness of the controller is chosen using a gain $K_{p,long}$. The complete feedback part of the longitudinal controller can be seen in Equation (48).

$$F_{x,FB} = K_{p,long}\big(V_x(s) - V_{x,des}(s)\big) \tag{48}$$

The output of the controller is further limited using the friction circle concept to avoid exceeding the maximum friction. This is done by using the same estimation as in the velocity planner shown in Equation (37). The equation estimates the maximum allowed longitudinal force depending on the current lateral and friction force [1]. A visual representation of the friction circle can be seen in Figure 12, where the outer border of the circle is the maximum allowed friction. This in turn means that if all friction is used for longitudinal acceleration as shown by the red dots there is no friction left for lateral acceleration. The same goes the other way around, if all friction is used for lateral acceleration, any additional longitudinal acceleration would result in violation of the friction limit, causing under or oversteer. Also shown in Figure 12 is an example, if the yellow dot represents the current lateral force, then the green span would represent the limitation used on the longitudinal force.

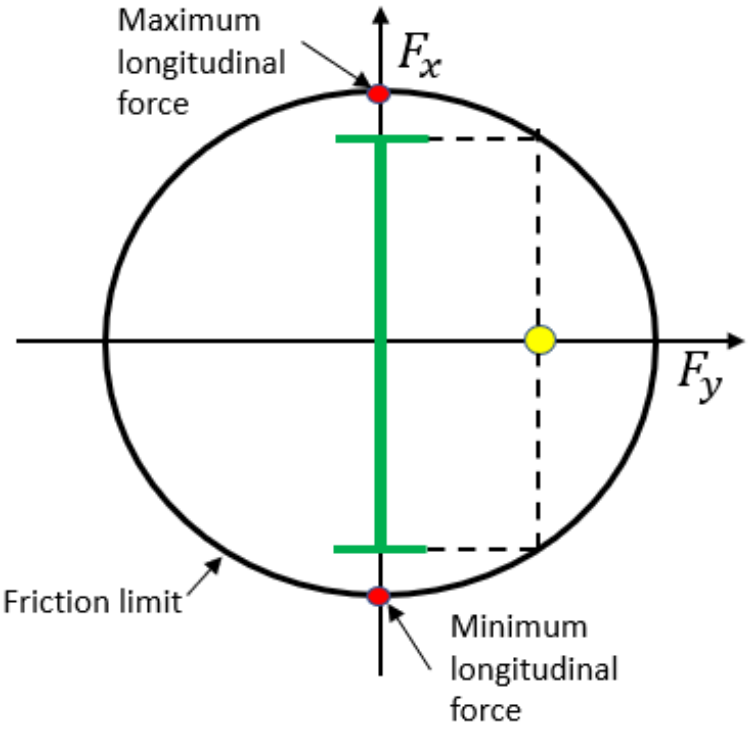

Figure 12. A visual representation of the friction circle.

The limit of the friction circle F_{tot}, is based on the vertical tire force $F_{z,r}$ and the friction coefficient μ as shown by Equation (49). Only the rear tire forces are considered in the equation although the same approach can be applied for the front tires. The vertical force can be obtained from Equation (50), where l_f is the length between the vehicle COG to the front wheels, h is the hight of the vehicle COG and L is the length between the front and rear wheels.

$$F_{tot} = \mu F_{z,r} \tag{49}$$

$$F_{z,r} = \frac{mgl_f + a_{long}h}{L} \tag{50}$$

After the friction circle is obtained, the current lateral force needs to be estimated. This is done by using the Magic formula tire model shown in Equation (51), where α_r is the rear tire slip angle and B, C, D, E are tire coefficients [31]. Furthermore, the rear tire slip angle can be obtained using Equation (52), where φ' is the vehicle yaw rate [15].

$$F_{y,r} = Dsin\left(Ctan^{-1}\left(B\alpha_r - E\left(B\alpha_r - tan^{-1}(B\alpha_r)\right)\right)\right) \tag{51}$$

$$\alpha_r = -tan^{-1}\left(\frac{V_y - l_r\varphi'}{V_x}\right) \tag{52}$$

By inserting Equation (49) and (51) into (37), the maximum longitudinal tire force $F_{x,r}$ can be obtained. This is as mentioned used to limit the output of the longitudinal controller if its output exceeds the maximum longitudinal force.

3.1.5 Feedforward-feedback lateral control

The feedforward-feedback lateral controller utilizes both a feedback term minimizing the "lookahead error" and a feedforward term based on the kinematics and dynamics of the vehicle at hand. The lookahead error can be described as the projected deviation from the planned path a distance in front of the vehicle, see Figure 13. The parameter e_{LA} represents the projected lookahead error, d is the current lateral error and s is the planned path, which should be followed [1].

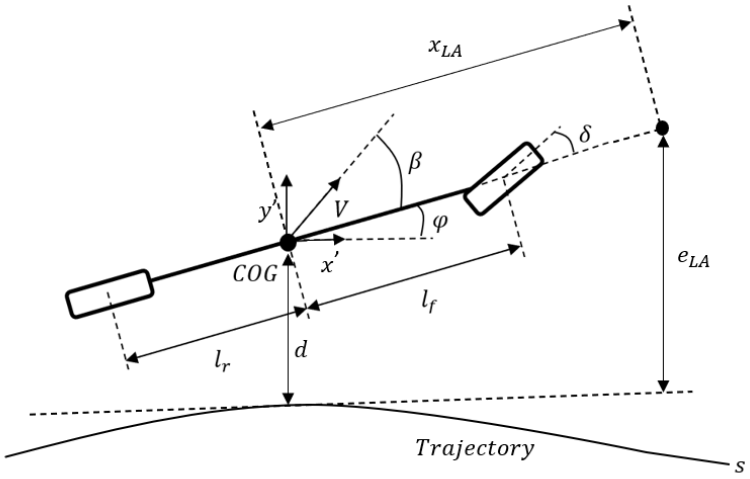

Figure 13. A schematic describing the lookahead error.

The objective of the feedforward steering controller is to calculate an estimated steering angle required to follow the path. Although, errors will enter the system and get magnified. The feedback term's objective is thus to compensate for these uncertainties and errors. The advantage of the feedforward-feedback control structure compared to simple a feedback controller, is the feedforward term's ability to immediately act before an error has occurred resulting in a faster controller response time. The feedforward term furthermore minimizes the compensation required by the feedback term which results in a possibility to make the feedback term less aggressive. The complete steering output is simply the sum of the feedback steering output and the feedforward steering output as shown in Equation (53).

$$\delta = \delta_{FF} + \delta_{FB} \tag{53}$$

The objective of the lateral controller is to minimize both the lateral error e as well as the heading error $\Delta\varphi$. The vehicle only supplies steering of the front wheels and can thus only manipulate and minimize one variable at the time. Therefore, a common approach is to minimize a weighted combination of the two by minimizing the lookahead error e_{LA} a distance x_{LA} in front of the COG as shown in Figure 13. A short lookahead distance results in higher consideration towards lateral error while a long lookahead distance results in higher consideration of heading error. This approach has been used employing several different vehicle and tire models with varying complexity [14].

The feedforward lateral controller furthermore needs to calculate the steering angle desired to follow the path depending on velocity and curvature as shown in Equation (54). The variable $V_{x,des}$ is the planned velocity, not the actual velocity, to avoid coupling errors between the feedforward and feedback controller [1].

$$\begin{aligned} F_{y,f,FF} &= \frac{ml_r}{L} V_{x,des}^2 \kappa \\ F_{y,r,FF} &= \frac{ml_f}{L} V_{x,des}^2 \kappa \end{aligned} \tag{54}$$

The vehicle and tire model are required to calculate the relationship between steering angle and actual lateral force. Altering the steering angle will in turn generate front and rear tire slip angles α_f and α_r as shown in Equation (55), which in turn generate front and rear lateral tire forces $F_{y,f,FF}$ and $F_{y,r,FF}$. The relationship between the steering and slip angles can be estimated using a bicycle model while the relationship between slip angles and lateral tire forces can be obtained using a tire model. The approach of the feedforward design presented below is a replication of the approach presented by Kapania in his dissertation. The equations used are generated from a kinematic bicycle model combined with a single friction coefficient Fiala brush model. The Fiala brush model for the front wheels can be seen in Equation (56), although the exact same approach can be applied to the rear wheels [1]. The constant C is the same tire cornering stiffness constant used in the Magic formula tire model explained in Equation (51). Note that the rear and front tire coefficients, C_r and C_f, are set equal resulting in the parameter C_{rf}.

$$\delta_{FF} = L\kappa - \alpha_{f,FF} + \alpha_{r,FF} \tag{55}$$

$$F_{y,f} = \begin{cases} -C_{rf}tan(\alpha_f) + \frac{{C_{rf}}^2}{3\mu F_{z,f}}|tan(\alpha_f)|tan(\alpha_f) \\ \quad -\frac{{C_{rf}}^3}{27\mu^2 F_{x,f}^2}tan^3(\alpha_f), & |\alpha_f| < tan^{-1}\left(\frac{3\mu F_{z,f}}{C_{rf}}\right) \\ -\mu F_{z,f} sgn(\alpha_f), & otherwise \end{cases} \quad (56)$$

As shown in Equation (54), the desired tire forces required to follow the curvature depending on the planned velocity can be obtained for the front and rear wheels. Thereafter, the front and rear slip angles need to be obtained from the tire model. This is done by generating an interpolated lookup table using Equation (56), with the slip angle on one axis and the resulting lateral force on the other axis. The lookup table is further used to estimate the slip angle required to generate the desired lateral force. After the slip angles are computed they can be inserted into Equation (55) to obtain the feedforward steering angle.

The lateral feedback controller's objective is to minimize a combination of the lateral deviation as well as the angular error. This is done by using the lookahead error as shown in Figure 13. The lookahead error serve as a combination of the lateral deviation and angular error, which can be seen in Equation (57). The lookahead error could be reformulated to include the path curvature as well, although, this would cause coupling between the feedforward and feedback controllers. Also shown in Equation (57) is the steering feedback term, which is dependent on the lookahead error multiplied by a scaling factor $k_{p,lat}$. After obtaining the feedforward and feedback terms, they are combined as previously shown in Equation (53).

$$\begin{aligned} e_{LA} &= d + x_{LA}\Delta\varphi \\ \delta_{FB} &= -k_{p,lat}e_{LA} \end{aligned} \quad (57)$$

3.2 Two-layer model predictive controller

Model Predictive Control (MPC) is a type of control structure allowing for more complex actions and behaviours unlike conventional control structures such as PID-control. MPC utilizes an iterative process for optimizing chosen object states while manoeuvring the object for a certain number of steps into the future called the prediction horizon. The object states are essentially the desired variables to be optimized, which are chosen dependent on the task at hand. For instance, the Frenet coordinates (s, d) are typical object states if a task of maximizing the vehicle progress along the centreline is desired, where *s* should be maximized while *d* should be minimized. Inclusion of additional object states might furthermore be necessary to be able to formulate equality constraints. The MPC predictions are made using a process model, or vehicle model in the field of autonomous racing, describing the object kinematics or dynamics. A general formulation describing the generation of predictions one step into the future is given in Equation (58).

$$\vec{x}_{i+1} = \vec{x}_i + \vec{x}_i' \cdot T_s \quad (58)$$

The vector $\vec{x}_i$ contains the object states at time step *i*, $\vec{x}_i'$ is a vector of state derivatives at time step i, T_s is the sample time and $\vec{x}_{i+1}$ is the vector of states at the next time step $(i + 1)$. The process model is used to obtain the state derivatives [32].

The MPC contains three parts: a cost function, a process model, and an optimization algorithm. The cost function represents the expected behaviour of the MPC, which is built up by an

optimization problem. The optimization problem is essentially a comparison of the object variables, subject to weight parameters, inequality as well as equality constraints, such that the optimization problem is either minimized or maximized. An example formulation of an MPC optimization problem is given by Equation (59).

$$Minimize: \sum_{i=0}^{N} W_a(\vec{x}_i - \vec{r}_i) \tag{59}$$

The vector $\vec{x}_i$ contains object states at time step i, $\vec{r}_i$ contains the reference states at time step i, N is the prediction horizon and W_a are the weights applied. The above optimization problem minimizes the distance to a reference trajectory described by $\vec{r}_i$ without taking any constraints into consideration [32].

The second part, the process model, is employed to obtain predicted future states or the so-called equality constraints, utilizing Equation (58). The third part, the optimization algorithm, is utilized to solve the optimization problem resulting in optimized object states [32]. There are several optimization libraries available for different programming languages, including several optimization algorithms selectable. The Operator Splitting Quadratic Program (OSQP) and SciPy are examples of optimization libraries for python. OSQP utilizes the Alternating Direction Method of Multipliers (ADMMs) optimization algorithm. SciPy allow for different optimization algorithms, such as Sequential Least-Squares Quadratic Programming (SLSQP) [33], [34].

A 2-layer MPC utilizes all the parts described above but differs slightly from a regular or so called "one-layer MPC". A regular MPC outputs the control inputs directly from the optimization problem, unlike the 2-layer MPC where it is necessary to convert the optimization outputs to obtain the control inputs. A low-level controller is thus essential in the 2-layer MPC which transforms the optimization output to control inputs [4].

3.2.1 Cost Function

As mentioned in Section 1.1, the trajectory planner's task is to achieve fast completion times while avoiding sharp steering and acceleration inputs. These tasks must be considered when choosing a cost function. Alcalá et al. suggests a cost function maximizing the vehicle velocity along a centreline at each time step while minimizing oversteer as well as understeer, see Equation (60) [3].

$$Minimize: \sum_{i=0}^{N} \left\| \alpha_{f,k+i} - \alpha_{r,k+i} \right\|_R^2 - \sum_{i=0}^{N} \left\| V_{t,k+i} \right\|_Q^2 \tag{60}$$

The parameter N is the prediction horizon, $\alpha_{f,k+i}$ is the front tire slip angle at time $(k+i)$, $\alpha_{r,k+i}$ is the rear tire slip angle at time $(k+i)$, $V_{t,k+i}$ is the velocity of the vehicle along the centreline at time $(k+i)$, and Q as well as R are weight scalars. The variable k describes the starting point of the current optimization, which depends on the last optimization. Another method for avoiding sharp steering and acceleration inputs is to simply minimize the change in longitudinal acceleration as well as steering, resulting in Equation (61).

$$Minimize: \sum_{i=0}^{N} \left\| \Delta a_{long,k+i} \right\|_R^2 + \sum_{i=0}^{N} \left\| \Delta \delta_{k+i} \right\|_R^2 - \sum_{i=0}^{N} \left\| V_{t,k+i} \right\|_Q^2 \tag{61}$$

The variable $\Delta a_{long,k+i}$ is the difference in longitudinal acceleration at time $k+i$ compared to time $k+i+1$ and $\Delta\delta_{k+i}$ is the difference in steering angle at time $k+i$ compared to time $k+i+1$. The velocity of the vehicle along the centreline can further be formulated using Equation (62).

$$V_{t,k+i} = \frac{V_{x,k+i}\cos(\varphi_{e,k+i}) - V_{y,k+i}\sin(\varphi_{e,k+i})}{1 - d_{k+i}\kappa_{k+i}} \tag{62}$$

The variable $V_{x,k+i}$ is the velocity of the vehicle in the local x-direction, $V_{y,k+i}$ is the velocity of the vehicle in the local y-direction, $\varphi_{e,k+i}$ is the angular error of the vehicle with respect to the centreline of the track, d_{k+i} is the lateral distance between the vehicle and the centreline and κ_{k+i} is the curvature of the centreline [3].

The cost function in Equation (61) is thus dependent on the following variables: $V_x, V_y, d, \varphi_e, a_{long}$ and δ, which is why V_x, V_y, d and φ_e are chosen to be the MPC states while a_{long} and δ are chosen as the MPC inputs. An additional state, φ', is, however, necessary to be able to formulate the equality constraints later, this will be discussed in the next chapter. The resulting state and input vectors are given in Equation (63).

$$\vec{x}_k = \begin{bmatrix} V_{x,k} \\ V_{y,k} \\ \varphi'_k \\ d_k \\ \varphi_{e,k} \end{bmatrix}, \vec{u}_i = \begin{bmatrix} \delta_k \\ a_{long,k} \end{bmatrix} \tag{63}$$

Substituting in Equation (62) into Equation (61) results in a cost function consisting of two quadratic terms and one non-linear term, $V_{t,k+i}$. It is preferable to find a linear-quadric formulation of the non-linear term, to reduce computational time as well as convexifying the problem. A convex problem further implies that any local minimum found is the sole minimum of the problem, which is favourable [35]. Alcalá et al. suggests a Quadratic programming (QP) formulation for convexifying the term [3]. Equation (64) shows a general formulation of a QP [33].

$$\left\|V_{t,k}\right\|_Q^2 \approx \frac{1}{2}\vec{x}_k^T Q \vec{x}_k + \vec{q}\vec{x}_k \tag{64}$$

The vector $\vec{x}_k$ contains the chosen states, Q is the optimization matrix acting on the states squared and q is the optimization vector acting on the states to the power of one [33].

To obtain the optimization matrix, Q, and optimization vector, $\vec{q}$, a Taylor approximation of Equation (62) is suggested. The general formulation of a quadratic Taylor approximation is given by Equation (65).

$$f(\vec{x}) \approx f(\vec{a}) + \left((\vec{x} - \vec{a}) \cdot \nabla f(\vec{a})\right) + \frac{1}{2}((\vec{x} - \vec{a})^T \cdot H(\vec{a}) \cdot (\vec{x} - \vec{a})) \tag{65}$$

$f(\vec{x})$ is the function to be approximated at point $\vec{x}$, $\vec{a}$ is the point in which the function is approximated around, ∇f is the gradient of the function and H is the Hessian matrix of the function. Assuming all constants equal to zero results in Equation (66) [36].

$$f(\vec{x}) \approx \vec{x} \cdot \nabla f(\vec{a}) + \frac{1}{2}(\vec{x}^T \cdot H(\vec{a}) \cdot \vec{x}) \tag{66}$$

When comparing Equation (66) and (64), it is obvious that the gradient as well as the Hessian at an approximation point, $\vec{a}$, of the function can be used to estimate the optimization vector, $\vec{q}$, respective the optimization matrix, Q, employing the assumptions mentioned above. The approximation results in the following optimization matrix and optimization vector when expressing the derivates with respect to the chosen states, see Equation (67-75). Observe that the gradient and hessian matrix are computed for $-V_{t,k+i}$.

$$\nabla f(\vec{a}) \approx \vec{q} = \left\{ \vec{a} = \begin{bmatrix} V_{x,0} \\ V_{y,0} \\ \varphi_0{}' \\ d_0 \\ \varphi_{e,0} \\ \kappa_0 \end{bmatrix} \right\} = \begin{bmatrix} \frac{\cos(\varphi_{e,0})}{\kappa_0 d_0 - 1} \\ \frac{\sin(\varphi_{e,0})}{1 - \kappa_0 d_0} \\ 0 \\ \frac{\kappa_0 V_{y,0} \sin(\varphi_{e,0}) - \kappa_0 V_{x,0} \cos(\varphi_{e,0})}{(\kappa_0 d_0 - 1)^2} \\ \frac{V_{x,0} \sin(\varphi_{e,0}) + V_{y,0} \cos(\varphi_{e,0})}{1 - \kappa_0 d_0} \end{bmatrix} \tag{67}$$

$$H(\vec{a}) \approx Q = \begin{bmatrix} 0 & 0 & 0 & H_{14} & H_{15} \\ 0 & 0 & 0 & H_{24} & H_{25} \\ 0 & 0 & 0 & 0 & 0 \\ H_{41} & H_{42} & 0 & H_{44} & H_{45} \\ H_{51} & H_{52} & 0 & H_{54} & H_{55} \end{bmatrix} \tag{68}$$

$$H_{14} = H_{41} = -\frac{\kappa_0 \cos(\varphi_{e,0})}{(1 - \kappa_0 d_0)^2} \tag{69}$$

$$H_{15} = H_{51} = \frac{\sin(\varphi_{e,0})}{1 - \kappa_0 d_0} \tag{70}$$

$$H_{24} = H_{42} = \frac{\kappa_0 \sin(\varphi_{e,0})}{(1 - \kappa_0 d_0)^2} \tag{71}$$

$$H_{25} = H_{52} = \frac{\cos(\varphi_{e,0})}{1 - \kappa_0 d_0} \tag{72}$$

$$H_{44} = -\frac{2\kappa_0{}^2 (V_{x,0} \cos(\varphi_{e,0}) - V_{y,0} \sin(\varphi_{e,0}))}{(1 - \kappa_0 d_0)^3} \tag{73}$$

$$H_{45} = H_{54} = -\frac{\kappa_0 (-V_{x,0} \sin(\varphi_{e,0}) - V_{y,0} \cos(\varphi_{e,0}))}{(1 - \kappa_0 d_0)^2} \tag{74}$$

$$H_{55} = -\frac{V_{y,0} \sin(\varphi_{e,0}) - V_{x,0} \cos(\varphi_{e,0})}{1 - \kappa_0 d_0} \tag{75}$$

The first row of the hessian matrix as well as the gradient represents the derivatives with respect to V_x, the second row the derivatives with respect to V_y, the third row the derivatives with respect to φ', the fourth row the derivatives with respect to d and the last row the derivatives with respect

to φ_e. The new cost function, resulting from the approximation above is given by Equation (76) [3].

$$Minimize: \sum_{i=0}^{N} \left\| \Delta a_{long,k+i} \right\|_R^2 + \sum_{i=0}^{N} \left\| \Delta \delta_{k+i} \right\|_R^2 - \frac{1}{2} \sum_{i=0}^{N} \vec{x}_{k+i}^T Q \vec{x}_{k+i} + q\vec{x}_{k+i} \tag{76}$$

3.2.2 Process Model

The weakness of many studies in the field of autonomous racing is the vehicle representation used. Alcalá et al. therefore, proposes a more complex non-linear dynamic bicycle model utilizing a reformulated version of the simplified Pacejka tire model to describe the vehicle physics and thus act as the process model of the MPC. These equations are given by Equations (6-15) in Section 2.2.2 [3]. The process model constitutes the equality and inequality constraints of the MPC. The equality constraints use the current state to calculate the next predicted state according to Equation (58), a matrix formulation of this equation is given below, see Equation (77).

$$\vec{x}_{k+1} = A_k \vec{x}_k + B_k \vec{u}_k \tag{77}$$

The vector $\vec{x}_k$ contains the current state, $\vec{u}_k$ contains the current input, A_k and B_k are matrices composed of constants defining the dependencies between the current state and the next state, $\vec{x}_{k+1}$ [3].

It is furthermore necessary to linearize Equations (6-15) to allow for the matrix representation given by Equation (77). The arc-tangent term in the slip angle equations can be neglected since V_y and $\dot{\varphi}$ are assumed to be close to zero, while V_x take on much higher values. The resulting slip angle equations are given by Equation (78) and (79).

$$\alpha_f = \delta - \left(\frac{V_y + l_f \varphi'}{V_x} \right) \tag{78}$$

$$\alpha_{long} = - \left(\frac{V_y - l_r \varphi'}{V_x} \right) \tag{79}$$

The equality constraint matrices, A_k and B_k, can now be constructed using Equation (77) and the dynamic bicycle model substituting in the derivate of each respective state. An example using the first state is given in Equation (80) and (81).

$$V_{x,i+1} = V_{x,i} + V_{x,i}{}' T_s \tag{80}$$

$$V_{x,i+1} = V_{x,i} + (a_{long} + \frac{-F_{y,f} \sin(\delta) - F_{d,f}}{m} + \varphi' V_y) \cdot T_s \tag{81}$$

The equality matrices are given by Equations (82-93).

$$A_k = \begin{bmatrix} A_{11} & A_{12} & A_{13} & 0 & 0 \\ 0 & A_{22} & A_{23} & 0 & 0 \\ 0 & A_{32} & A_{33} & 0 & 0 \\ 0 & 1 & 0 & 0 & A_{45} \\ A_{51} & A_{52} & 1 & 0 & 0 \end{bmatrix} \tag{82}$$

$$B_k = \begin{bmatrix} B_{11} & 1 \\ B_{21} & 0 \\ B_{31} & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{83}$$

$$A_{11} = 1 + \left(-\frac{\mu g}{V_x} - \frac{d_{wind} V_x}{m}\right) T_s \tag{84}$$

$$A_{12} = \frac{C_f \sin(\delta)}{m V_x} T_s \tag{85}$$

$$A_{13} = \left(\frac{C_f l_f \sin(\delta)}{m V_x} + V_y\right) T_s \tag{86}$$

$$A_{22} = 1 + \left(-\frac{C_f \cos(\delta)}{m V_x} - \frac{C_r}{m V_x}\right) T_s \tag{87}$$

$$A_{23} = \left(-\frac{C_f l_f \cos(\delta)}{m V_x} + \frac{C_r l_r}{m V_x} - V_x\right) T_s \tag{88}$$

$$A_{32} = \left(-\frac{C_f l_f \cos(\delta)}{I_z V_x} + \frac{C_r l_r}{I_z V_x}\right) T_s \tag{89}$$

$$A_{33} = 1 + \left(-\frac{C_f l_f^2 \cos(\delta)}{I_z V_x} - \frac{C_r l_r^2}{I_z V_x}\right) T_s \tag{90}$$

$$B_{11} = -\frac{C_f \sin(\delta)}{m} \tag{91}$$

$$B_{12} = \frac{C_f \cos(\delta)}{m} \tag{92}$$

$$B_{13} = \frac{C_f l_f \cos(\delta)}{I_z} \tag{93}$$

Observe that the initial states as well as the initial inputs are necessary to construct the equality matrices and equality constraints [3], see Equation (94).

$$\vec{x}_o = \begin{bmatrix} V_{x,init} \\ V_{y,init} \\ \varphi'_{init} \\ d_{init} \\ \varphi_{e,init} \end{bmatrix}, \vec{u}_0 = \begin{bmatrix} \delta_{init} \\ a_{long,init} \end{bmatrix} \tag{94}$$

The inequality constraints of the MPC are essentially constraints describing the value limits of each of the states and inputs. The states and inputs as well as the state predictions are only allowed to take on values included in the range specified. A general formulation of the inequality constraints regarding the states and inputs at hand are given in Equation (95).

$$
\begin{gathered}
V_{x,min} \leq V_x \leq V_{x,max} \\
V_{y,min} \leq V_y \leq V_{y,max} \\
\varphi_{min}{}' \leq \varphi' \leq \varphi_{max}{}' \\
d_{min} \leq d \leq d_{max} \\
\varphi_{e,min} \leq \varphi_e \leq \varphi_{e,max} \\
\delta_{min} \leq \delta \leq \delta_{max} \\
a_{long,min} \leq a_{long} \leq a_{long,max}
\end{gathered}
\tag{95}
$$

Combining the cost function, equality and inequality constraints, results in the complete optimization problem for the MPC, see Equation (96) [3].

$$
Minimize: \sum_{i=0}^{N} \left\| \Delta a_{long,k+i} \right\|_R^2 + \sum_{i=0}^{N} \left\| \Delta \delta_{k+i} \right\|_R^2 - \frac{1}{2} \sum_{i=0}^{N} \vec{x}_{k+i}^T Q \vec{x}_{k+i} + q \vec{x}_{k+i}
$$

$$
\begin{gathered}
Subject\ to: \\
\vec{x}_{k+1} = A_k \vec{x}_k + B_k \vec{u}_k \\
V_{x,min} \leq V_x \leq V_{x,max} \\
V_{y,min} \leq V_y \leq V_{y,max} \\
\varphi_{min}{}' \leq \varphi' \leq \varphi_{max}{}' \\
d_{min} \leq d \leq d_{max} \\
\varphi_{e,min} \leq \varphi_e \leq \varphi_{e,max} \\
\delta_{min} \leq \delta \leq \delta_{max} \\
a_{long,min} \leq a_{long} \leq a_{long,max} \\
\vec{x}_o = \begin{bmatrix} V_{x,init} \\ V_{y,init} \\ \varphi'_{init} \\ d_{init} \\ \varphi_{e,init} \end{bmatrix}, \vec{u}_0 = \begin{bmatrix} \delta_{init} \\ a_{long,init} \end{bmatrix}
\end{gathered}
\tag{96}
$$

3.2.3 Low level control

Solving the optimization problem renders new optimized states and inputs. In order to control towards these new states a low-level controller is necessary. An almost identical longitudinal feedforward-feedback controller as described in Section 3.1.4 can be applied here as well. The MPC structure includes lateral force consideration when planning, which is why the acceleration limits of the longitudinal feedforward-feedback controller has been omitted. This is the only difference between the longitudinal controller of the 2-layer MPC and the FF-FB trajectory planner. However, for lateral control a simple feedback controller presented by Svensson is suggested [37]. The feedback lateral controller uses a similar feedback approach described as in 3.1.5, but uses the actual deviation from the planned path a lookahead distance in front of the vehicle instead of the projected deviation. This approach takes more information about the track properties into consideration when controlling the vehicle than the feedback part of the FF-FB trajectory planner, which is why it is suggested. A figure describing the actual deviation from a planned path a distance in front of the vehicle is shown in Figure 14 [37].

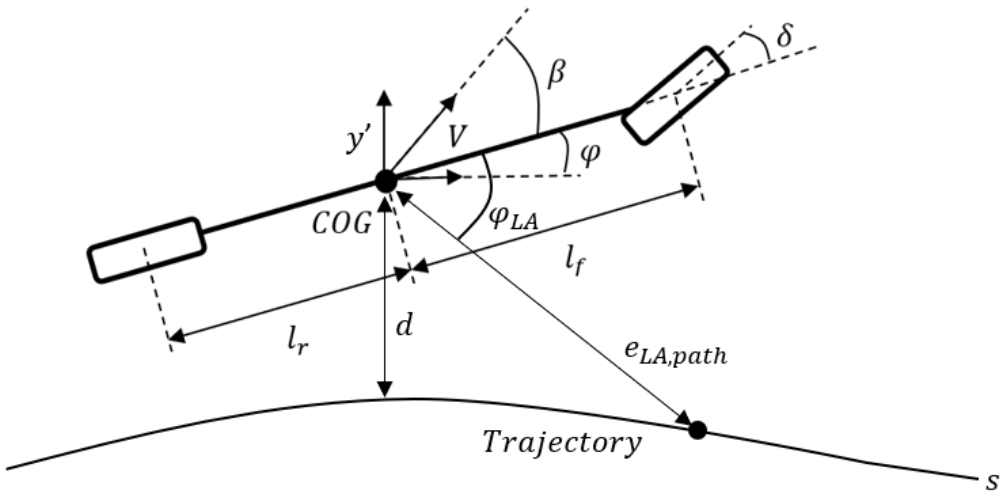

Figure 14. A schematic describing the lookahead error projected onto the track.

To obtain the lookahead error, $e_{LA,path}$, and lookahead angle, φ_{LA}, the local vehicle x- and y-coordinates, a point a set of indices ahead of the vehicle as well as current heading angle needs be extracted. Thereafter, from simple trigonometry the lookahead distance $e_{LA,path}$ and lookahead angle φ_{LA} can be calculated. The feedback steering angle is furthermore calculated using Equation (97) [37].

$$\delta_{FB} = \frac{2sin(\varphi_{LA})}{e_{LA,path}}\left(l_f + l_r\right) \tag{97}$$

The longitudinal controller suggested requires a reference velocity to be followed, this velocity is given by the first state of the optimization problem, V_x. The lateral controller suggested, however, requires the x- and y-coordinates of the optimized trajectory generated. It is therefore necessary to convert the optimized states and inputs into x- and y-coordinates. An approach suitable for this conversion is to firstly obtain the Frenet coordinates of the path generated and consequently collect the x- and y-coordinates from the Frenet to Cartesian coordinate transform described in Section 2.3. The d coordinate is already given by the optimized d-state, however, obtaining the s coordinate requires some calculations. Equation (58) is used to acquire the s coordinate, where the derivative of s is the velocity along the trajectory described in Equation (62). The complete equation is given by Equation (98)

$$s_{i+1} = s_i + \frac{V_{x,opt,i}\cos\left(\varphi_{e,opt,i}\right) - V_{y,opt,i}\sin\left(\varphi_{e,opt,i}\right)}{1 - d_{opt,i}\kappa_i} T_s \tag{98}$$

The variable i iterates from 0 to N, s_i is the s coordinate at iteration i, κ_i is the curvature of the centreline at iteration i and s_0 is set to zero. The x- and y-coordinates, which are used in the low-level control, can now be obtained using the Frenet to Cartesian transform [3], [37].

3.3 Offline and online trajectory planning

The task of a trajectory planner is to find a suitable trajectory to reach a goal while avoiding certain static or dynamic obstacles. The trajectory can be developed in two different ways, as an offline or online trajectory planner. The offline trajectory planner computes the entire trajectory to the

goal before motion begins. In contrast, an online trajectory planner generates trajectories incrementally while driving towards the goal in a real time manner. The advantage of an online planner is that it can adjust the trajectory to compensate for uncertainties such as unknown obstacles. Online trajectory planning often is a computationally heavy procedure, limitations within computational resources may hinder a trajectory of being computed fast enough [38].

4 IMPLEMENTATION

In this chapter describes the implementation of the trajectory planners to be simulated. The vehicle setup and the test cases are also presented here.

4.1 Experimental set up

To properly compare the performance of the trajectory planners, a set of test cases were designed. They were designed to be challenging for the trajectory planners as well as allow evaluation of a few key functionalities. A custom vehicle model and a set of customized tracks were furthermore developed to be able to perform the tests.

4.1.1 Selection of test cases

A set of test cases were required to investigate and compare the performance of the two different trajectory planners. The tracks created to compare completion times are constructed to represent a couple of specific curves typical in the field of autonomous racing. Two simple 90-degree turns were constructed to test the trajectory planners' ability to plan for a plain curve. Furthermore, two double hairpin curves were constructed to test the trajectories' ability to adjust the plan of the second arc depending on the first. Moreover, two scenarios with altered vehicle mass were used to test the sensitivity of trajectory planners with respect to modelling errors. The parameter of mass effects both low-level and high-level functions, such as velocity generation, control, and path generation, which is why it is assumed to form an adequate study of sensitivity. Lastly, two obstacle popup scenarios were constructed to test minimum detection distance of each trajectory planner. The set of test cases are presented in Table 2.

Table 2. Selection of test cases

Selection of test cases	
Test case	**Observed parameter**
Double hairpin track, Radius 6m	Completion time
Double hairpin track, Radius 12m	Completion time
90-degree turn track, Radius 6m	Completion time
90-degree turn track, Radius 12m	Completion time
Double hairpin track, Radius 6m, 20% extra mass	Completion time
Double hairpin track, Radius 6m, 20% less mass	Completion time
Obstacle avoidance track, $V_{des} = 8\ m/s$	Minimum detection distance
Obstacle avoidance track, $V_{des} = 12\ m/s$	Minimum detection distance

Each test case investigating completion time was conducted three times in a row where each of the completion times were saved.

For the test cases regarding obstacle avoidance, a set of experiments were conducted using different obstacle popup distance. The test starts by the vehicle accelerating until reaching the desired velocity, V_{des}, and thereafter keeping a constant speed. When an obstacle is detected, the vehicle can choose to steer and decelerate at will to avoid the obstacle. The vehicle completes the test if it avoids the obstacle without driving off the track or spinning out of control. The test cases are initially conducted using a long obstacle detection distance, which is reduced by one meter if the trajectory planner completes two successive successful runs out of two. The lowest completed detection distance achieved by each of the trajectory planners is noted.

4.1.2 Vehicle model

The vehicle model was created with the intention of, in the future, implementing the trajectory planners on a 1:10 scale model race car. The visualization and the vehicle parameters were therefore determined using the size and functionality the model race car implies. The steering was assumed to only act on the front wheels since it is the most common setup for model race cars as well as conventional race cars. The acceleration and braking are, however, usually different in a model race car compared to a regular race car. For instance, a model race car usually does not have a separate braking system, which in turn lead to the forces only being applied by the same wheels and motors used for acceleration. The vehicle model used within this project was assumed to be a rear wheel drive, therefore, both the acceleration and braking were assumed only to act on the rear wheels. The vehicle parameters used were estimated based on a normal 1:10 scale model race car. The inertia was furthermore calculated using an approximation assuming a rectangular weight distribution of the car and a circular weight distribution of the wheels with radius r_{wheel}. The vehicle parameters used are presented in Table 3.

Table 3. Vehicle parameters

Vehicle parameters			
Name	**Parameter**	**Value**	**Unit**
Mass vehicle	m	2	kg
Mass wheel	m_{wheel}	0.1	kg
Length COG to front wheels	l_f	0.15	m
Length COG to rear wheels	l_r	0.15	m
Wheel radius	r_{wheel}	0.03	m
Hight of COG	h	0.06	m
Friction coefficient	μ	1	-
Inertia around z-axis	I_z	0.022	$(\frac{kg}{m^2})$
Inertia of wheel	I_{wheel}	0.023	$(\frac{g}{m^2})$
Tire stiffness coefficient	B	10	-
Tire shape coefficient	C	1.9	-
Tire peak coefficient	D	1	-
Tire curvature coefficient	E	0.97	-
Rear tire coefficient	C_r	80	$(\frac{N}{rad})$
Front tire coefficient	C_f	80	$(\frac{N}{rad})$

4.1.3 Software implementation

The Robot Operating System, or ROS, was chosen as a base for the project due to its many functionalities and modularity. The API used was ROS Kinetic since most of the accessible software, including FSSIM, perform well using this stable old version of ROS.

Both F1tenth and FSSIM were evaluated as potential environments for implementation. Although, after investigating the environments, a decision to use FSSIM was made. The main reasons were the advanced physics engine, user friendliness and documentation. Perception, localization, and mapping are outside the scope of this thesis, resulting in these parts being adopted from FSSIM and [37]. The FSSIM simulation environment has the internal capabilities of outputting global coordinates of the track and current vehicle position, although, it lacks a clear local coordinate system. Therefore, a conversion function from global to local coordinates was imported and used from [37]. Most of the code was written using python 2.7.12 excluding the A* search algorithm

which was written using C++. The program was furthermore executed on a PC running Ubuntu 16.04 LTS with an Intel Core i7-6600U CPU and an Intel HD Graphics 520 (Skylake GT2) graphics card.

Each test was conducted in the customized FSSIM environment. The vehicle parameters were changed within FSSIMs vehicle generation method to be able to use their physics engine. A new model of the vehicle was furthermore added to the environment to match the size of the model race car, see Figure 15. The design of the vehicle is simplified and does not look like an actual 1:10 scale model race car. Although, the simplification is justified because the model is only necessary to get a feeling of size within the simulation environment and does not affect the performance.

Figure 15. A simple visualization of the vehicle model used within FSSIM.

To be able to conduct the test cases, five custom tracks were made. Two double hairpin tracks, two 90-degree turns tracks, and one straight track were made. All the tracks are presented in Figure 16-a-e. On the straight track the obstacle is shown, although as mentioned, the obstacle is not detectable for the trajectory planners before they are within the set detection distance. All tracks have a starting line, a finishing line as well as a vehicle starting position, which was set half a meter before the starting line. Both trajectory planners thus started on the same position resulting in comparable time readings. The timer started when the vehicle passed the starting line. The timer stopped when the vehicle reached the finish line and the time passed was calculated with a resolution of 0.005 seconds.

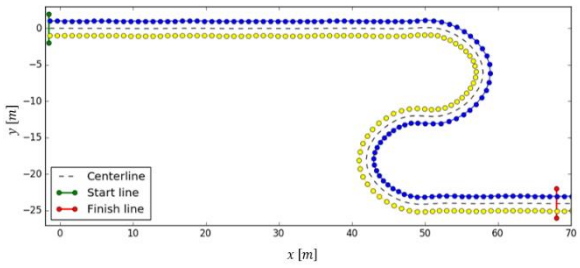

Figure 16-a. The double hairpin track with a curve radius of 6 m.

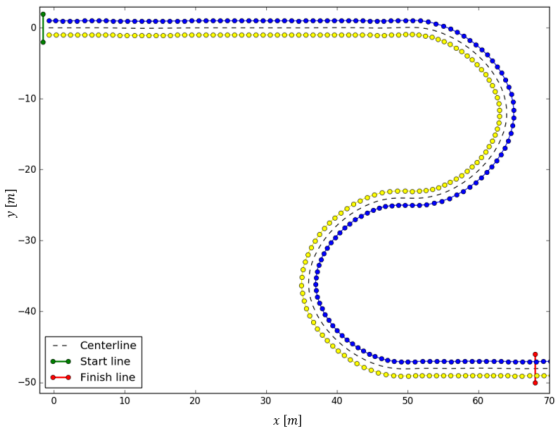

Figure 16-b. The double hairpin track with a curve radius of 12 m.

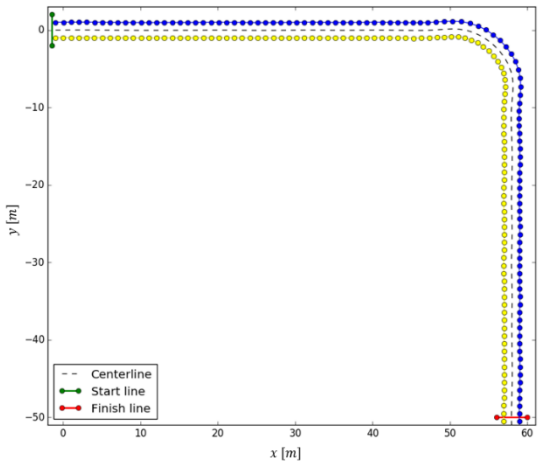

Figure 16-c. The 90-degree track with a curve radius of 6 m.

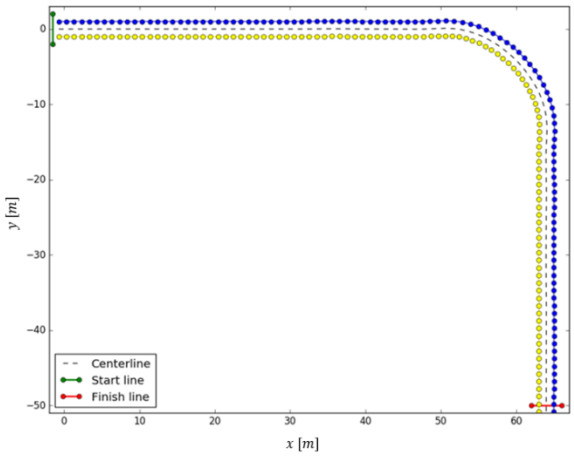

Figure 16-d. The 90-degree track with a curve radius of 12 m.

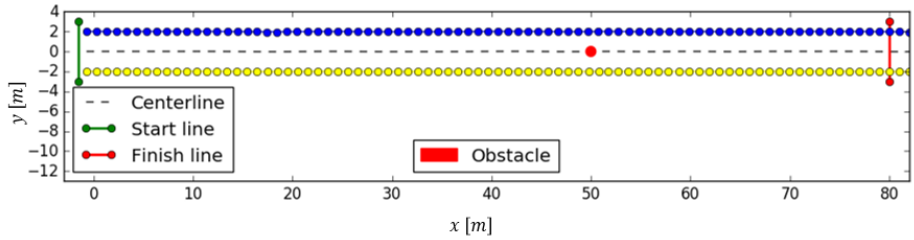

Figure 16-e. The straight track with obstacle appearing placed on the centreline.

4.2 Implementation of trajectory planners

The implementation of both trajectory planners was inspired by research papers already made. Customization and optimization of the trajectory planners to fit the simulation environment and task at hand, however, had to be made. The implementation process of the two trajectory planners is presented in this section.

4.2.1 Feedforward-feedback trajectory planner

A* search algorithm

The A* search algorithm is, as described in Section 3.1.1, a search algorithm that generates the shortest path from one point to another. The search algorithm, however, had to be remade to suit the needs of the trajectory planner and simulation environment. The algorithm was at first developed using hard boundaries for the track limits, which resulted in the program braking each time the vehicle drove slightly outside the track. This behaviour was not acceptable, which is why soft boundaries henceforth were used, utilizing an added weighting parameter if the planner suggested a path outside the track boundaries. When driving along a straight section the planner alternated significantly between paths. The planner was made to plan the shortest path to a point on the centreline a set distance in front of the vehicle. The alternation was present when the vehicle position deviated from the centreline. Consequently, the planner sometimes planned straight from the vehicle position and turned to the centreline at the end, other times it turned directly to the centreline at the start and thereafter followed it. The issue was that both ways of planning resulted in the same cost, causing this constant alternation. Therefore, the weight was, additionally, reduced slightly when planning along the centreline. The slightly reduced weight when driving along the

centreline caused the planner to favour driving to the centreline at the start since following it yielded a reduced cost, resulting in a significant reduction of path alternation. A visual representation of a path suggested by the A* path planner applied to the double hairpin track with radius of 6 meters is presented in Figure 17.

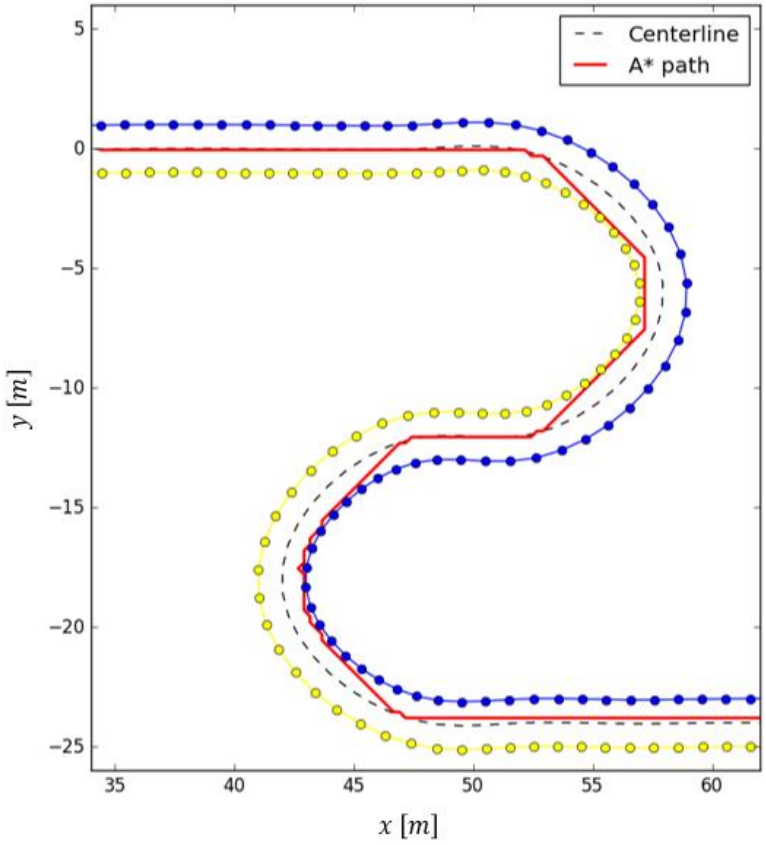

Figure 17. The offline A* path. Shown on the double hairpin track with curve radius 6m.

Minimum curvature path planner

The minimum curvature path planner is implemented using the approach presented in Section 3.1.2 to generate a path with the lowest curvature, thus yielding the path similar to the geometric racing line presented in Section 2.1. To implement a minimum curvature path planner Heilmeier et al. proposes a pre-processing of the original path before minimizing the curvature. The pre-processing is done to smoothen out the reference line since the optimization problem is quite sensitive to a noisy reference line [2]. A pre-processing session consisting of an interpolation of the reference line using cubic splines was carried out. Although, the pre-processing of the reference line did not result in any observable difference regarding performance while increasing the computation time. The reference line was quite smooth to begin with which might be the reason why the pre-processing did not cause a significant difference. Another attempt was carried out by increasing the number of indices along the track while using the same cubic spline interpolation, however this attempt did not render the desired results either. Therefore, the pre-processing was ignored and not used in the final version of the trajectory planner.

Offline minimum curvature path planner

To find the absolute minimum curvature racing line of the entire track, the optimization problem needed to consider the whole track. This in turn resulted in a quite large optimization problem with a heavy computational load. Therefore, this approach was only utilized within the offline trajectory planner. The advantage of the offline trajectory planner is as mentioned that it finds the absolute

minimum curvature racing line of the track and thus obtains an optimized path as can be seen in Figure 18. The optimization problem was solved using the SLSQP solver within SciPy.

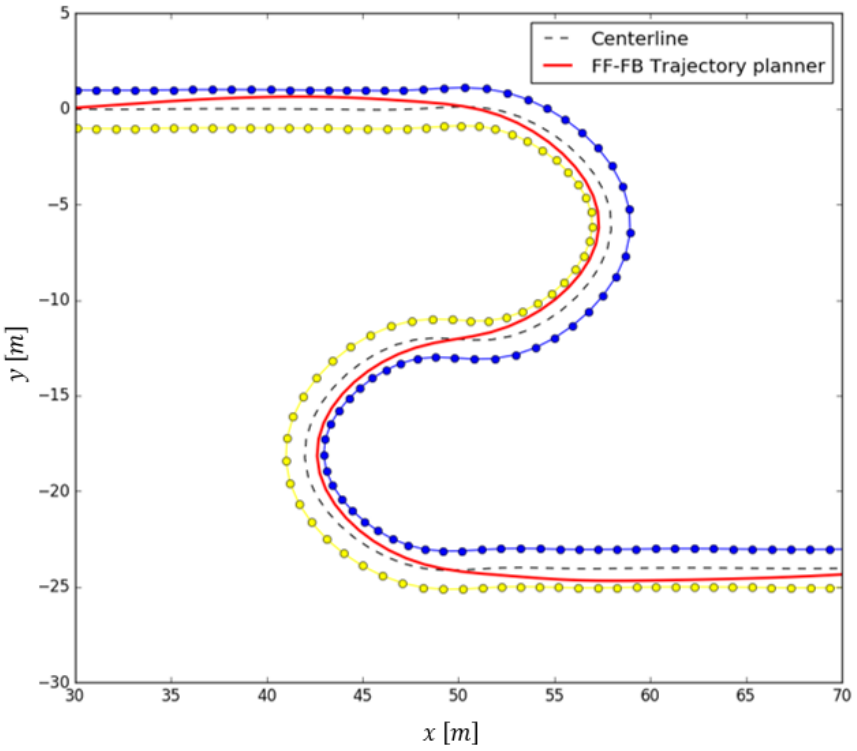

Figure 18. The offline minimum curve radius path. Shown on the track double hairpin track with radius 6m.

Online minimum curvature path planner

For better obstacle avoidance and comparability with the 2-layer MPC an online minimum curvature path planner was developed. Initially, the online minimum curvature path planner used the vehicle starting position as starting index and planned fixed distance in front of the vehicle. In order to obtain a somewhat minimum curvature path the path planner required at least 30 meters of planning distance in front of the vehicle. This in turn combined with the computational load of the velocity planner required about one second of computation time, considerably delaying the new trajectory. The trajectory planner thus sent an invalid starting position to the path planner because a new vehicle position had been reached during that second of delay. The result was a rough transition between the old path to the new which sometimes resulted in a crash.

The solution was to initially plan a normal trajectory from the vehicle starting position some planning distance ahead and thereafter lock the 10 meters closest to the vehicle. In each iteration of planning the first 10 meters in front of the vehicle were locked to remain the same as the previous trajectory and the last 35 meters were thus used to obtain a low curvature. The distance locked is further on referred to as the locking distance S_{locked}. The vehicle furthermore never had time to drive further than 10 meters before the trajectory was updated and therefore the transition between the trajectories were smooth. An increased planning distance would reduce the comparability between the trajectory planners as well as increase computational time. The increased computational time would in turn require a longer locked distance which would result in worse obstacle avoidance. A trial-and-error session where the planning and locking distance were altered to obtain a sufficient performance regarding both completion times on all tracks and obstacle avoidance resulted in $S = 35$ and $S_{locked} = 10$. The path of the online minimum curvature path planner applied to the double hairpin track with a curve radius of 6 meters is presented in Figure 19. The green lines represent the previously planned paths with the end of each

path marked by a green dot. The last point of each trajectory furthermore must be positioned on the centreline for best performance, which also can be seen in Figure 19. The figure captures only the middle part of the track, which is the part of interest regarding trajectory planner performance. One can see that the first path planned (green line) was straight since the curve was not visible for the planner, but as the vehicle progresses it spots the curve and consequently plans out to the left to obtain a somewhat minimized curvature.

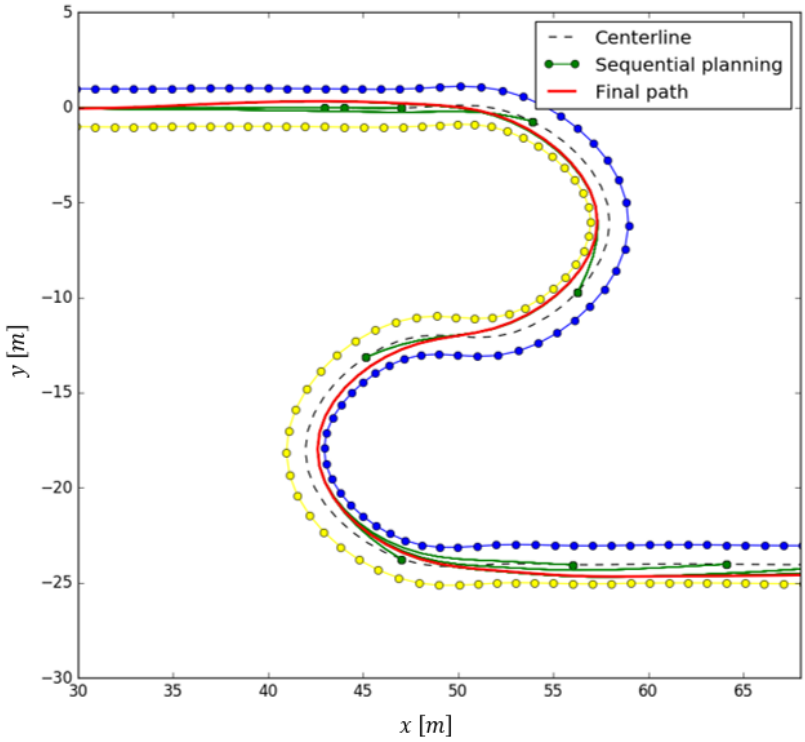

Figure 19. The online minimum curvature radius path planner, with sequential planning steps. Shown on the track double hairpin track with curve radius 6m.

Due to the locking distance and lengthy planning distance of the FF-FB trajectory planner, problems regarding obstacle avoidance were expected. A new parameter configuration of the planning distance and locking distance constructed for optimal obstacle avoidance was therefore partly used for the obstacle avoidance. The parameter configuration used was $S = 15$ and $S_{locked} = 7$ for the 8 m/s test case and $S = 24$ and $S_{locked} = 9$ for 12 m/s test case.

Decision between Online and Offline

The offline trajectory planner performed slightly better when considering completion times if all circumstances were known. The offline trajectory planner was, however, not able to re-plan depending on any uncertainties along the track and needed to know the entire track beforehand. The online FF-FB trajectory planner was therefore chosen for further development while the offline planner was neglected for further development. The online FF-FB trajectory planner furthermore compares better to the 2-layer MPC which was developed online, see Section 4.2.2.

Combining the A* search algorithm and the minimum curvature path planner

The optimal path proposed by Kapania is a weighted combination of the minimum curvature path and the shortest path, which can be generated using the A* search algorithm. Utilizing a static weight, however, proved to perform best with a weight highly favouring the minimum curvature path planner and thus almost neglecting the shortest path planner. Kapania therefore proposes a dynamic weight changing over time, to mimic data from a professional race driver [1]. Combining the two path planners proved some issues. The paths were combined using Equation (20). The A* search algorithm has sharp transitions between points as can be seen in Figure 17, due to the limited

set of directions available in the algorithm definition. Increasing the set of available directions furthermore resulted in an increased computation time and was therefore not a feasible solution. Furthermore, when combining the A* algorithm and minimum curvature path planners, the angle and curvature of the newly generated path had to be calculated using cubic splines which also increased computation times vastly. Attempts were made but none of them proved to perform as good as the minimum curvature path planner by itself. Therefore, the A* path planner was not used in final FF-FB trajectory planner. The resulting path planner used was thus the online minimum curvature path planner.

Velocity profile generation

With the path calculated, the next step was to generate a velocity profile that maximizes the velocity along the path. The velocity profile was generated according to the three-step process explained in section 3.1.3. When implementing the velocity planner three properties were considered: the path curvature, the vehicle acceleration, and the vehicle deceleration. The most important part was to have a good approximation of the curvature along the track. If the curvature in a point was badly approximated the velocity planner would propose a too high or too low velocity in that point, leading to a bad performance. The calculation of the curvature was done by interpolating the planned path and approximating the curvature using Equation (25). The approximation of the curvature proved to work decent but not perfect, therefore, it was processed afterwards. The processing was accomplished by taking the mean of the current and adjacent curvatures, which improved performance.

Another aspect to consider was the planning distance. If the planning distance were to short it could not correctly plan for the deceleration. The planning distance required for the velocity planner is, however, shorter than for the path planner and was thus not a limiting factor.

The acceleration and deceleration proved to be unsynchronized with respect to the capabilities of the vehicle. Testing resulted in a too fast desired acceleration by the velocity profile, even when accelerating the vehicle fully it could not follow the planned velocity profile. The term $F_{x,max,acc}$ proposed by Kapania, used in Equation (42) only depends on the vehicle engine force limits and the lateral demands on the tires [1]. Including the effect of the drag forces on the vehicle in the equation improved the performance greatly.

Feedforward-feedback longitudinal controller

The longitudinal controller was used to control the vehicle acceleration and braking to correctly follow the desired velocity profile. The longitudinal controller within the FF-FB was initially made as a feedback PID-controller. Although, implementing the feedforward-feedback longitudinal controller presented in Section 3.1.4, increased performance significantly. The feedforward part of the longitudinal controller was developed from the vehicle dynamics and was converted into the required forces to compensate for acceleration and drag. The feedback controller was thereafter converted into a P-controller which was sufficient to compensate for the errors. After the desired longitudinal force was obtained it was further limited by the current lateral force demand on the wheels. This approach worked well in the curves, although it did limit the acceleration somewhat on the straight sections due to small spikes in lateral tire demand. Therefore, if the lateral forces were below a certain limit, they were set to zero.

Feedforward-feedback lateral controller

To alter the steering angle of the vehicle to follow the desired path, a feedforward-feedback lateral controller was implemented as explained in Section 3.1.5. The feedforward lateral controller first calculates desired tire forces required at the front and rear wheels. Thereafter, a lookup table is needed to convert the desired forces into desired slip angles. A plot representing lookup table for the lateral feedforward controller is presented in Figure 20. The lookup table was created by

looping through slip angles from the lower to the upper slip condition and calculating the lateral force obtained at each slip angle. The generated data point was thereafter linearly interpolated.

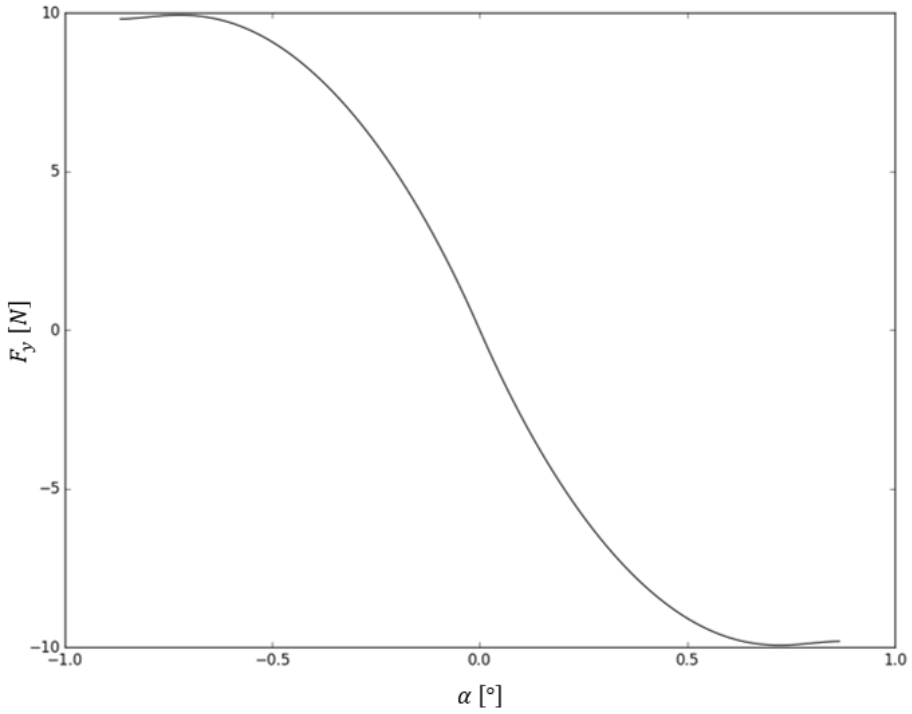

Figure 20. A visual representation of lookup table for the Fiala tire model.

FF-FB parameters

The complete set of parameters used within the FF-FB trajectory planner is presented in Table 4.

Table 4. The parameters used by the FF-FB trajectory planner.

FF-FB parameters			
Name	**Parameter**	**Value**	**Unit**
Longitudinal gain	$k_{p,long}$	2	-
Lateral gain	$k_{p,lat}$	0.05	-
Lookahead distance	x_{LA}	15	m
Planning distance	S	45	m
Locked distance	S_{locked}	10	m

4.2.2 Two-layer Model predictive controller

Cost Function

The 2-layer MPC was implemented utilizing a python package for solving optimization problems called OSQP [33]. The OSQP solver assumes a cost function expressed using a QP formulation on the following form, see Equation (99).

$$\begin{aligned} &Minimize\text{: } \frac{1}{2}\vec{w}^T P\vec{w} + \vec{w}^T L \\ &subject\ to\ l_{bound} \leq A\vec{w} \leq u_{bound} \end{aligned} \tag{99}$$

The vector $\vec{w}$ contains the states and inputs of the MPC, P is a weight matrix acting on the states and inputs squared, L is a weight vector acting on the states and inputs to the power of one, l_{bound} is a vector containing lower bounds, u_{bound} is a vector containing upper bounds and A is the

constraint matrix describing MPC restrictions. The cost function described in Section 3.2.1 was used by the 2-layer MPC. To utilize the OSQP package, a reformulation of the cost function and constraints was necessary. A reformulation of the cost function was first executed. Since the third term of the cost function, the velocity term, is already on QP form its conversion was very simple, see Equation (100).

$$\frac{1}{2}\sum_{i=0}^{N}\vec{x}_{k+i}^{T}Q\vec{x}_{k+i}+q\vec{x}_{k+i}=\vec{z}_k^{T}\underbrace{\begin{bmatrix}Q & 0 & \cdots & 0\\ 0 & Q & \ddots & 0\\ \vdots & \ddots & \ddots & 0\\ 0 & 0 & 0 & Q\end{bmatrix}}_{P_1}\vec{z}_k+\vec{z}_k^{T}\underbrace{\begin{bmatrix}q\\ q\\ \vdots\\ q\end{bmatrix}}_{L_1}$$
$$\vec{z}_k=\begin{bmatrix}\vec{x}_{k+1}\\ \vec{x}_{k+2}\\ \vdots\\ \vec{x}_{k+N}\end{bmatrix} \tag{100}$$

The reformulation essentially converts the summation into a QP consisting of one matrix P_1 containing matrices and one vector L_1 containing vectors. Reformulation of the first and second term of the cost function, however, slightly increased the complexity. Expanding one of the terms resulted in the following equation, see Equation (101). Note that the same expansion can be done regarding both terms.

$$\begin{gathered}\sum_{i=0}^{N}\left\|\Delta a_{long,k+i}\right\|_{R_2}^{2}=\{a_{long}=a\}=\sum_{i=0}^{N}R_2(a_{k+i}-a_{k+i+1})^2=\\ R_2(a_{k+1}-a_{k+2})^2+R_2(a_{k+2}-a_{k+3})^2+\cdots+R_2(a_{k+N}-a_{k+N+1})^2=\\ R_2({a_{k+1}}^2-2a_{k+1}a_{k+2}+2{a_{k+2}}^2-2a_{k+2}a_{k+3}+2{a_{k+3}}^2+\cdots+2{a_{k+N}}^2\\ -2a_{k+N}a_{k+N+1}+{a_{k+N+1}}^2\end{gathered} \tag{101}$$

The parameter R_2 is a weighting scalar and a is the longitudinal acceleration of the vehicle. The resulting expression in Equation (101) was reformulated on for both terms using QP notation, which rendered the following equation, see Equation (102).

$$\sum_{i=0}^{N}\left\|\Delta a_{long,k+i}\right\|_{R_2}^{2}+\sum_{i=0}^{N}\left\|\Delta\delta_{k+i}\right\|_{R_2}^{2}=$$
$$\vec{c}_k^{T}\begin{bmatrix}R_1 & 0 & -R_1 & 0 & \cdots & 0 & 0 & 0 & 0\\ 0 & R_2 & 0 & -R_2 & \ddots & 0 & 0 & 0 & 0\\ -R_1 & 0 & 2R_1 & 0 & \ddots & 0 & 0 & 0 & 0\\ 0 & -R_2 & 0 & 2R_2 & \ddots & -R_1 & 0 & 0 & 0\\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 & -R_2 & 0 & 0\\ 0 & 0 & 0 & -R_1 & 0 & 2R_1 & 0 & -R_1 & 0\\ 0 & 0 & 0 & 0 & -R_2 & 0 & 2R_2 & 0 & -R_2\\ 0 & 0 & 0 & 0 & 0 & -R_1 & 0 & R_1 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & -R_2 & 0 & R_2\end{bmatrix}\vec{c}_k+\vec{c}_k^{T}\begin{bmatrix}0\\ 0\\ 0\\ 0\\ \vdots\\ 0\\ 0\\ 0\\ 0\end{bmatrix}$$
$$\vec{c}_k=\begin{bmatrix}\vec{u}_{k+1}\\ \vec{u}_{k+2}\\ \vdots\\ \vec{u}_{k+N}\end{bmatrix} \tag{102}$$

The parameter R_1 is the weighting scalar which applies to the steering angle and R_2 is the weighting scalar which applies to the acceleration. The weighing scalars were chosen dependent on the importance of the objective. For instance, a high weighting scalar R_1 results in extra importance of minimizing the second term of the cost function, the difference in steering angle squared. The weighting scalars of the 2-layer MPC were set to $R_1 = 6$ and $R_2 = 6$.

All terms of the cost function on QP form were combined, which resulted in the complete representation of the cost function, see Equation (103).

$$Minimize: \vec{w}_k{}^T \underbrace{\begin{bmatrix} Q & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & Q & \ddots & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & Q & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & Q & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & R_1 & 0 & -R_1 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & R_2 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & -R_1 & 0 & 2R_1 & \ddots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -R_2 & 0 & R_2 \end{bmatrix}}_{P} \vec{w}_k + \vec{w}_k{}^T \underbrace{\begin{bmatrix} \vec{q} \\ \vec{q} \\ \vdots \\ \vec{q} \\ \vec{q} \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{L} \tag{103}$$

$$\vec{w}_k = \begin{bmatrix} \vec{x}_k \\ \vec{x}_{k+1} \\ \vdots \\ \vec{x}_{k+N-1} \\ \vec{x}_{k+N} \\ \vec{u}_k \\ \vec{u}_{k+1} \\ \vec{u}_{k+2} \\ \vdots \\ \vec{u}_{k+N} \end{bmatrix}$$

Equality and inequality constraints

Next, the equality and inequality constraints required reformulation. The $\vec{w}$ vector was given by the cost function formulation; the task was now to formulate the equality and inequality constraints using this vector. An easy reformulation using the identity matrix was achievable for the equality constraints resulting in Equation (104).

$$\vec{x}_{i+1} = A_{k,i}\vec{x}_i + B_{k,i}\vec{u}_i \rightarrow 0 = I_{5x5}\vec{x}_{i+1} - A_{k,i}\vec{x}_i - B_{k,i}\vec{u}_i \tag{104}$$

The formulation I_{5x5} denotes the identity matrix of size 5x5. The equality constraints were expanded *N* points into the future resulting in the matrix formulation, see Equation (105).

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} = \underbrace{\begin{bmatrix} I_{5x5} & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ -A_{k,1} & I_{5x5} & 0 & \ddots & 0 & 0 & -B_{k,1} & 0 & \ddots & 0 & 0 \\ 0 & -A_{k,2} & I_{5x5} & \ddots & 0 & 0 & 0 & -B_{k,2} & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \ddots & I_{5x5} & 0 & 0 & 0 & \ddots & -B_{k,N-1} & 0 \\ 0 & 0 & 0 & \cdots & A_{k,N-1} & I_{5x5} & 0 & 0 & \cdots & 0 & -B_{k,N} \end{bmatrix}}_{A_{eq,0}} \underbrace{\begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vec{x}_3 \\ \vdots \\ \vec{x}_{N-1} \\ \vec{x}_N \\ \vec{u}_1 \\ \vec{u}_2 \\ \vdots \\ \vec{u}_{N-1} \\ \vec{u}_N \end{bmatrix}}_{\vec{w}_0} \tag{105}$$

The variable *N* is the prediction horizon of the MPC. Equation (106) shows the general formulation of the equality constraints using the OSQP structure. The OSQP structure is presented in Equation (99).

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \leq A_{eq,k}\vec{w}_k \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{106}$$

A reformulation of the inequality constraints was achieved using an identity matrix applied to the vector $\vec{w}_k$ as well as constructing two vectors containing the lower and the upper bounds of the states and inputs. The identity matrix was constructed to specify which state or input to restrict while the bounding vectors defines the restriction on that specific state or input. The inequality constraints on matrix form are presented in Equation (107).

$$\underbrace{\begin{bmatrix} \vec{l}_x \\ \vec{l}_x \\ \vdots \\ \vec{l}_x \\ \vec{l}_u \\ \vec{l}_u \\ \vdots \\ \vec{l}_u \end{bmatrix}}_{\vec{l}_{ineq}} \leq I_{7Nx7N} \begin{bmatrix} \vec{x}_k \\ \vec{x}_{k+1} \\ \vdots \\ \vec{x}_{k+N} \\ \vec{u}_k \\ \vec{u}_{k+1} \\ \vdots \\ \vec{u}_{k+N} \end{bmatrix} \leq \underbrace{\begin{bmatrix} \vec{g}_x \\ \vec{g}_x \\ \vdots \\ \vec{g}_x \\ \vec{g}_u \\ \vec{g}_u \\ \vdots \\ \vec{g}_u \end{bmatrix}}_{\vec{g}_{ineq}} \tag{107}$$

The vector $\vec{l}_x$ contains the lower limits regarding the state vector, $\vec{x}_k$, $\vec{l}_u$ is a vector of lower limits regarding the input vector, $\vec{u}_k$, $\vec{g}_x$ is a vector of upper limits regarding the state vector and $\vec{g}_u$ is a vector for upper limits regarding the input vector. The size of the identity matrix, *I*, is dependent on the prediction horizon and the number of states respective inputs. Since there are five states and two inputs in the current MPC, a total size of 7Nx7N was set for the identity matrix.

The matrix formulations of the equality and inequality constraints were combined, resulting in the complete constraint formulation on the form desired, see Equation (108).

$$\underbrace{\begin{bmatrix} 0_{7Nx1} \\ \vec{l}_{ineq} \end{bmatrix}}_{\vec{l}_{bound}} \leq \underbrace{\begin{bmatrix} A_{eq,k} \\ I_{7Nx7N} \end{bmatrix}}_{A} \vec{w}_k \leq \underbrace{\begin{bmatrix} 0_{7Nx1} \\ \vec{g}_{ineq} \end{bmatrix}}_{\vec{g}_{bound}} \tag{108}$$

The formulation 0_{7Nx1} represents a matrix containing zeros of size 7Nx1. The complete optimization problem was now formulated using the desired structure, see Equation (109).

$$\begin{aligned} Minimize&: \vec{w}_k{}^T P \vec{w}_k + \vec{w}_k{}^T L \\ Subject\ to&: \vec{l}_{bound} \leq A\vec{w}_k \leq \vec{g}_{bound} \end{aligned} \tag{109}$$

Approximating Q and $\vec{q}$

The matrix Q and the vector $\vec{q}$, was furthermore approximated using the Taylor approximation method described in Section 3.2.1. The point linearized around, $\vec{a}$, used by the 2-layer MPC is presented below, see Equation (110).

$$\vec{a} = \begin{bmatrix} V_{x,0} \\ V_{y,0} \\ \varphi_0{}' \\ d_0 \\ \varphi_{e,0} \\ \kappa_0 \end{bmatrix} = \begin{bmatrix} 7.0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{110}$$

A point situated somewhat in the middle of the vehicle state range, according to the maximum respective minimum values set for all states, see Equation (114), was linearized around. The resulting Q matrix and $\vec{q}$ vector is presented in Equation (111).

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 7.0 \end{bmatrix}, q = \begin{bmatrix} -1.0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{111}$$

A gain acting on the lateral velocity, V_y, was furthermore applied to avoid unnecessary turning during straight sections of the track. The gain was set to 3.0, resulting in a new Q matrix, see Equation (112).

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 3.0 & 0 & 0 & 1.0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 7.0 \end{bmatrix} \tag{112}$$

Parameter values of the inequality constraints

The inequality constraints acting on the states, $\vec{l}_x$ and $\vec{g}_x$, respective the inputs, $\vec{l}_u$ and $\vec{g}_u$, were moreover set. The limit applied to the acceleration was set by $F_{x,max,acc}$ and $F_{x,max,dec}$ dependent on the engine force, lateral demand on the tires as well as the wind resistance, described thoroughly in Section 4.2.1 and 3.1.3. The limit applied to the lateral deviation from the centreline was set using the width of the racetrack as well as the width of the vehicle, according to Equation (113).

$$\begin{aligned} d_{max} &= \frac{w_{track} - w_{vehicle}}{2} \\ d_{min} &= -d_{max} \end{aligned} \tag{113}$$

The parameter w_{track} is the width of the track and $w_{vehicle}$ is the width of the vehicle. An experimental session was conducted regarding the rest of the states and inputs with the goal of rendering the best results. The concluding inequality constraints acting on the states and inputs are presented in Equation (114).

$$u_x = \begin{bmatrix} 16 & [m/s] \\ 1 & [m/s] \\ 10 & [rad/s] \\ d_{max} & [m] \\ \frac{\pi}{4} & [rad] \end{bmatrix}, l_x = \begin{bmatrix} 0 & [m/s] \\ -1 & [m/s] \\ -10 & [rad/s] \\ d_{min} & [m] \\ -\frac{\pi}{4} & [rad] \end{bmatrix} \quad (114)$$

$$u_u = \begin{bmatrix} 1.6 & [rad] \\ F_{x,max,acc} & [m/s^2] \end{bmatrix}, l_u = \begin{bmatrix} -1.6 & [rad] \\ F_{x,max,dec} & [m/s^2] \end{bmatrix}$$

Online or Offline planning

As mentioned above, in Section 4.2.1 and Section 3.3 , the choice of making the trajectory planners online or offline was investigated. An offline trajectory planner essentially generates a trajectory for the entire track before following it, resulting in a trajectory taking the entire structure of the track into consideration. An online trajectory planner on the other hand generates trajectories as it goes, resulting trajectories taking only the structure of the track a certain distance into future into consideration. The racing line rendered by the offline planner therefore generally has a more optimal appearance, while the online planner, due to its iterative behaviour, generally has more robustness against uncertainties on the track. An offline approach regarding the MPC unfortunately rendered quite unpredictable paths, making it difficult to obtain consistent and comparable results. The complexity of the problem exponentially increases when enlarging the prediction horizon, *N*, resulting in these unpredictable paths. The resulting trajectory from the offline MPC was thus not trustable. A figure showing the difference in paths generated from the offline MPC is presented in Figure 21. The offline MPC was therefore neglected for further development and an online MPC was developed.

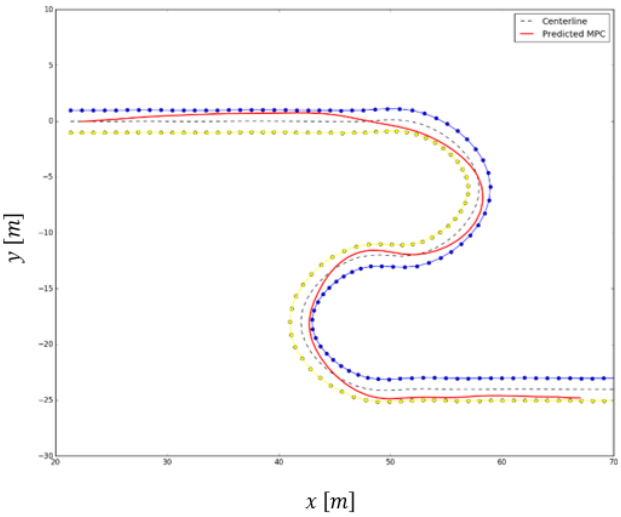

Figure 21-a. A bad path generated from the offline MPC.

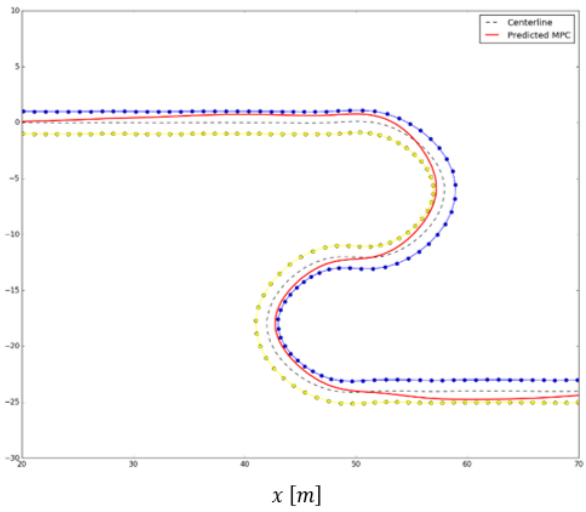

Figure 21-b. A good path generated from the offline MPC.

Online planning

The online MPC was implemented so that the optimized states computed were acted on a certain number of steps into the future, using the low-level controller described in Section 3.2.3, before new optimized states were calculated. The low-level controller utilizes the same longitudinal feedforward-feedback controller as the FF-FB trajectory planner, but another lateral controller. The feedforward part of the lateral controller used by the FF-FB trajectory planner caused instability issues when transitioning from one trajectory to another when applied to the 2-layer

MPC. A different lateral controller only utilizing feedback was therefore used by the MPC. The lateral controller of the MPC was furthermore implemented using a variable lookahead distance, x_{LA}, dependent on the vehicle longitudinal velocity. Equation (115) describes this variable lookahead distance. A too short lookahead distance resulted in an oscillative controller; a lower boundary was therefore added.

$$\begin{aligned} x_{LA} &= g_{LA} \cdot V_x \\ x_{LA} &\geq 0.8 \end{aligned} \tag{115}$$

The parameter g_{LA} is a constant applied to the current vehicle longitudinal velocity, V_x. The equation resulted in a shorter lookahead distance during low velocities and longer lookahead distance during high velocities. This approach was suitable to obtain a controller not cutting corners when approaching a curve. The longitudinal controller of the MPC was, on the other hand, implemented without any lookahead. Algorithm 1 describes the control loop of the MPC.

Algorithm 1: Control loop

1. Generate N optimized states and inputs, $\vec{x}_{k,opt}, \vec{u}_{k,opt}$.
2. Convert states and inputs into Frenet coordinates, s *and* d.
3. Calculate lookahead: $x_{LA} = g_{LA} \cdot V_x$
4. If $x_{LA} < 0.8$ then
   $x_{LA} = 0.8$
5. Find which index, m_n, in the s vector is closest to x_{LA}.
6. Convert Frenet coordinates into cartesian coordinates, x and y.
7. Initialize iterator: $j = 0$
8. For $i = m_n : m_n + m_{it}$
   If $i \geq N$
   break
   $longetudenal\_control(V_x(j))$
   $lateral\_control(x(i), y(i))$
   $j \mathrel{+}= 1$
9. end

The parameter m_{it} describes the number of steps acted on. The algorithm started by computing a starting index, m_n, using the lookahead distance calculated in Equation (115) and the Frenet coordinate, s, of the path generated. The algorithm thereafter used an interval of Cartesian coordinates of the path generated decided by the lookahead distance and m_{it} for lateral control. The algorithm furthermore used an interval from zero to m_{it} of the velocity profile for longitudinal control.

2-layer MPC parameters

The complete set of parameters used within the 2-layer MPC is presented in Table 5.

Table 5. 2-layer MPC parameters used.

2-layer MPC parameters			
Name	**Parameter**	**Value**	**Unit**
Longitudinal gain	$k_{p,long}$	2	-
Prediction horizon	N	30	Steps
Lookahead distance gain	g_{LA}	0.4	$\frac{1}{s}$
Number of iterations controlled towards	m_{it}	10	-

The low value of prediction horizon, N, was chosen due to the heavy computational load a high value of N renders. As mentioned above, the complexity of the problem increases exponentially when enlarging the prediction horizon. A value of 30 was therefore assumed to supply some sense of the environment while keeping the predictions steady and not driving a too high computational load.

5 Results

In this chapter the generated results from the test cases are presented. The results are divided into three sections presenting the results for each of the three parts of the research question.

5.1 Completion time

In this section the completion time will be observed and compared amongst the two trajectory planners. Completion time is defined as the time passed from when the vehicle drives over the starting line until it passes the finish line. Each subsection contains completion times and a minimum of two figures representing the path and velocity during the first test run when the trajectory planners are applied to each of the test tracks investigating completion time. The figures presenting the paths driven focuses on the non-straight section of the track since this is where the trajectory planners differ the most. The path and velocity profile does vary slightly between each run within each test case, although the general behaviour is shown by the first run shown in the figures. The test cases are conducted using the real time applied trajectory planner in the simulation environment FSSIM.

Figures 22, 26, 28, and 30 contains two paths of the planned trajectory (dashed lines), two paths of the actual trajectory driven (solid lines) by the vehicle during the test and one path describing the centreline of the track. Figures 24, 27, 29, and 31 contains two velocities describing the planned velocity of the trajectory planners and two additional velocities describing the actual velocity of the trajectory planner.

The test case investigating the double hairpin track with a curve radius of 6 meters contains some additional information. These tests are conducted an additional two times to gain better approximation of the mean and standard deviation. The control outputs are furthermore investigated for this test case.

5.1.1 The double hairpin track, curve radius 6 meters

The resulting planned and driven paths can be seen in Figure 22. The 2-layer MPC achieves an early apex in both the first and second curve as can be seen by the two green dots. In comparison, the FF-FB trajectory planner achieves a normal apex as can be seen by the two red dots. The difference in paths also explain the differences in the velocity profiles. The FF-FB brake earlier to be able to turn earlier while the 2-layer MPC enters with a higher velocity, which in turn forces the vehicle to drive further forward before turning.

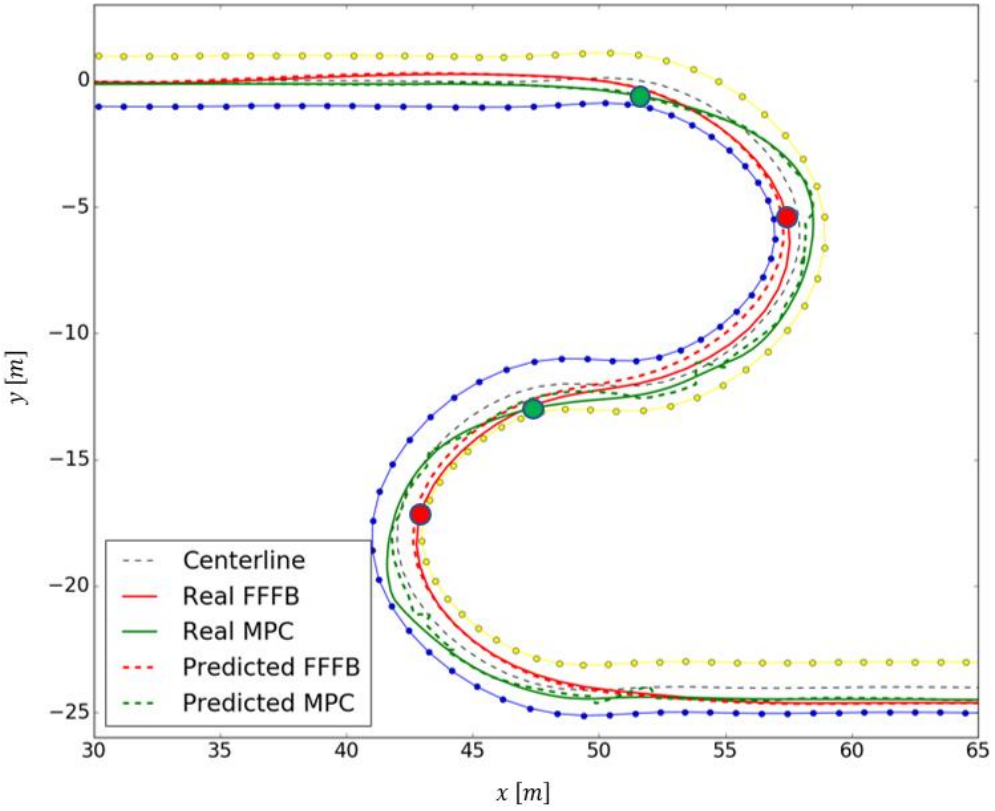

Figure 22. The test case investigating completion time on the double hairpin track, curve radius 6 m. A comparison between paths. Including dots to mark out the apex for the driven paths.

The output steering angle from the lateral controllers can be seen in Figure 23. The FF-FB lateral controller initiates an earlier decrease in steering angle compared to the 2-layer MPC, which also can be seen in Figure 23. The FF-FB trajectory planner furthermore increases the steering angle slightly before the first turn, which indicates a behaviour of trying to minimize the curvature, while the 2-layer MPC shows no behaviour of the kind. Also shown in the figure is the slightly higher peak steering angle of the FF-FB trajectory planner compared to the 2-layer MPC.

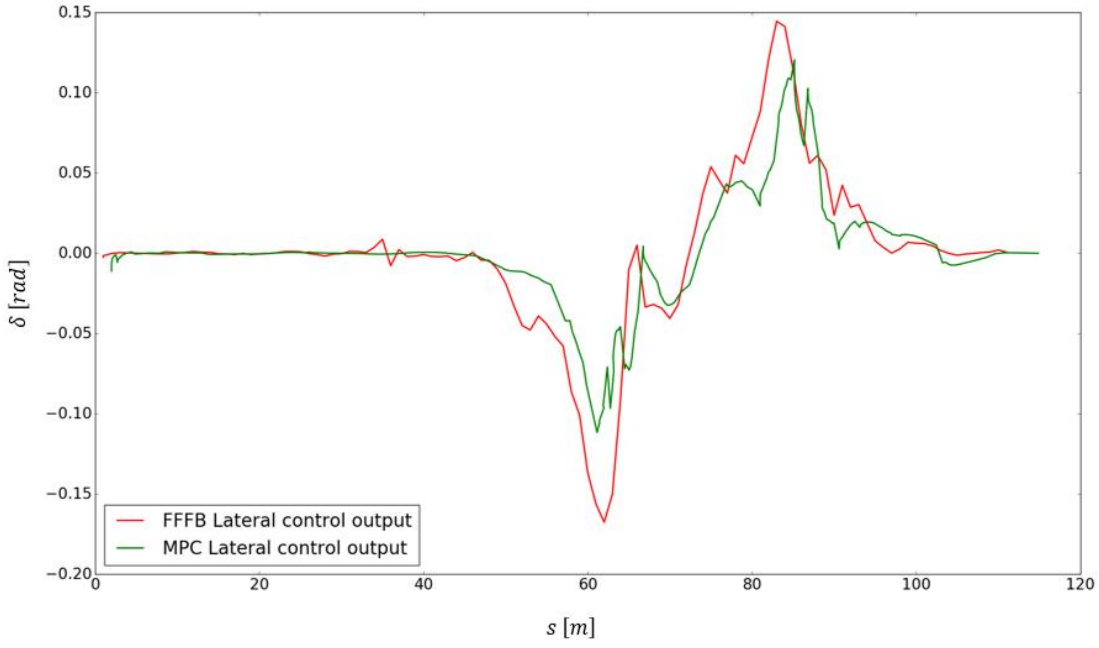

Figure 23. The test case investigating completion time on the double hairpin track, curve radius 6 m. Control output from lateral controller.

The longitudinal controllers' ability to follow the generated velocity profile can be observed in Figure 24 by comparing the planned velocities with the real velocities. As can be seen, both the 2-layer MPC and FF-FB follow the planned velocity with only minor deviations. Both planners have about the same average velocity in the middle of the curve and accelerate similarly out of it. The FF-FB has a slightly higher error within the curvature, this can partially be due to the upper limit saturation shown in Figure 25. Worth noting is that the 2-layer MPC include feedback when planning the velocity profile which the FF-FB does not. In other words, the 2-layer MPC plan the next velocity profile depending on the current vehicle velocity. The FF-FB on the other hand, plan the next velocity profile depending on the current velocity profile and not the vehicle state.

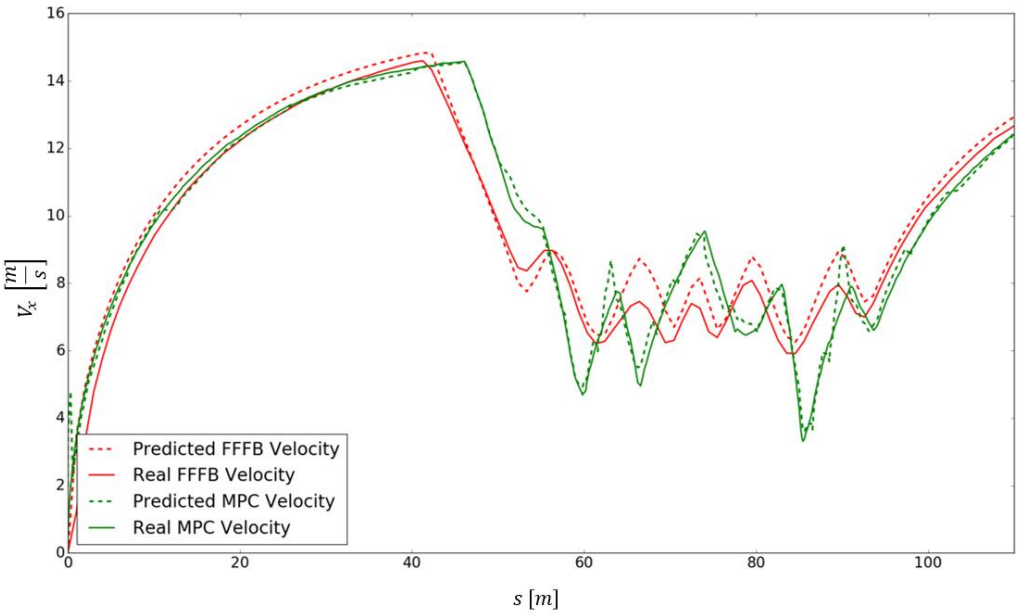

Figure 24. The test case investigating completion time on the double hairpin track, curve radius 6 m. A comparison between velocities.

The output from the longitudinal controllers can be seen in Figure 25-a and Figure 25-b. The upper and lower limits applied to the longitudinal force are also presented. As mentioned, the setup was made to mimic a 1:10 scale race vehicle, which in turn assumes acceleration as well as deceleration applied to only the rear wheels. Therefore, when braking the weight distribution results in less traction for the rear wheels, in turn resulting in a lower deceleration limit, $F_{x,max,dec}$, compared to acceleration limit, $F_{x,max,acc}$. The output saturation applied to the 2-layer MPC is static and set to $F_{x,max,acc}$ and $F_{x,max,dec}$. The saturation applied to the FF-FB controller is on the other hand dynamic and set according to Section 3.1.4. Comparing the velocity profile of the FF-FB trajectory planner and its longitudinal controller output one can observe the same amount of acceleration and deceleration phases. However, when comparing the velocity profile of the 2-layer MPC and its longitudinal controller output the same behaviour cannot be observed. The 2-layer MPC replans the whole path and velocity profile in each iteration which may cause a less smooth transition and a higher number of acceleration and deceleration phases.

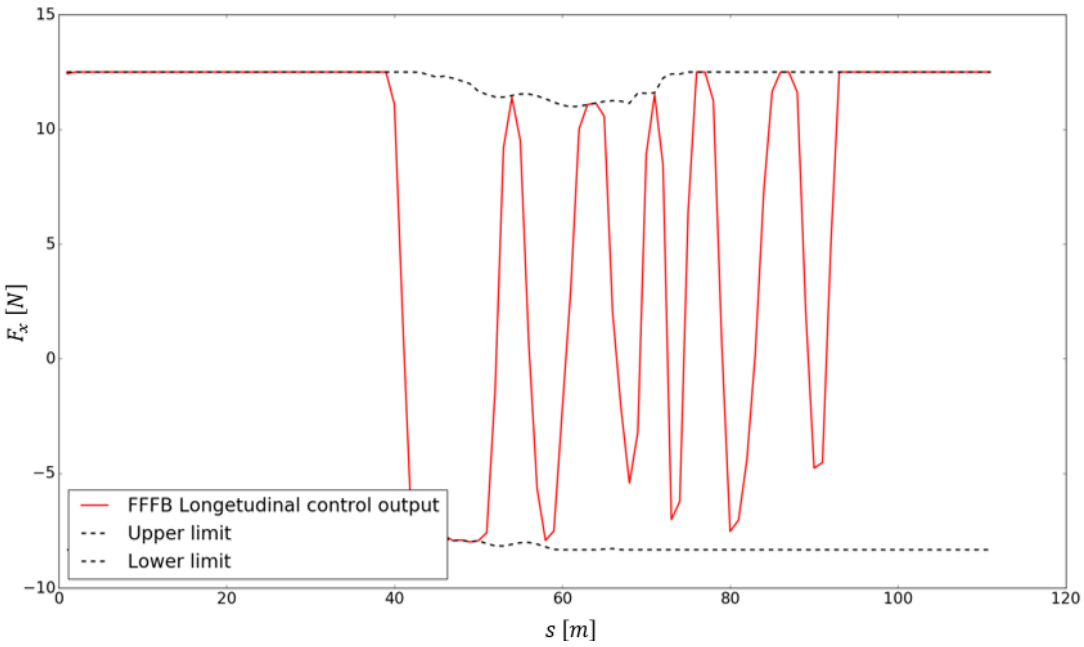

Figure 25-a. The test case investigating completion time on the double hairpin track, curve radius 6 m. Output from longitudinal controller for the FF-FB.

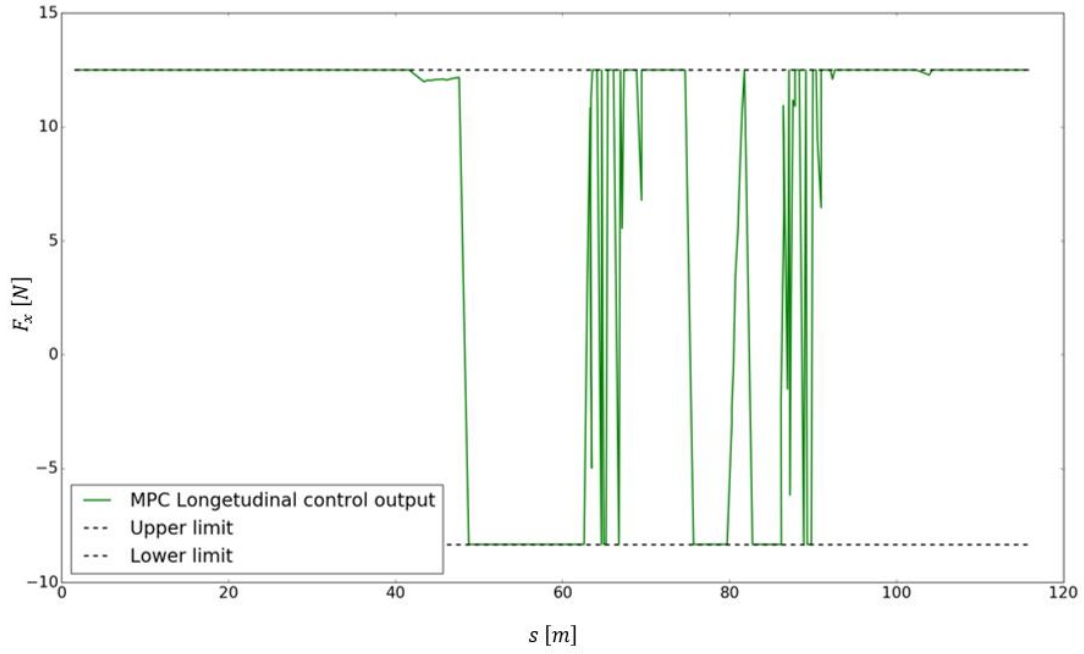

Figure 25-b. The test case investigating completion time on the double hairpin track, curve radius 6 m. Output from longitudinal controller for the MPC.

The resulting completion times are presented in Table 6. The test case investigating the double hairpin track with a curve radius of 6 meters was furthermore executed two times extra, in order to get a better approximation of mean and standard deviation when performing the significance test. The resulting average, or mean, completion times of the FF-FB trajectory planner as well as the 2-layer MPC are also presented in Table 6. Subtracting the average completion time of the FF-FB with the MPC furthermore results in a difference of 0.37 seconds, implying better performance of the FF-FB trajectory planner.

Table 6. Test results consisting of completion times for double hairpin track with a curve radius of 6 meters.

Trajectory planner	1st run	2nd run	3rd run	4th run	5th run	Mean	Standard deviation
FF-FB	11.825	11.775	11.730	11.760	11.730	11.764	0.035
2-layer MPC	12.010	12.390	12.025	12.105	12.100	12.126	0.137

The significance test assumes the measured values to be normally distributed with the standard deviations shown in Table 6. The FF-FB trajectory planner got a mean of 11.76 and a standard deviation of 0.04, resulting in, with 95% certainty using a coverage factor of k=1.96, a completion time of 11.76 ± 0.07. The 2-layer MPC on the other hand has a mean of 12.13 and a standard deviation of 0.14, resulting in, with 95% certainty using the same coverage factor, a completion time of 12.13 ± 0.27. The upper limit regarding completion time of the FF-FB, according to the significance test, is thus 11.83 seconds, which is faster than the lower limit of the MPC which is 11.86 seconds, implying that the FF-FB does achieve faster completion times with a significant difference.

5.1.2 The double hairpin track, curve radius 12 meters

The test case investigating the double hairpin track with a curve radius of 12 meters shows similar behaviour to the previous test case. The 2-layer MPC enters the curve with a slightly higher velocity and start steering into the curve later. Also observable, is a small steady state error acting on the FF-FB in the middle of the curves, resulting in a slightly wider path driven compared to the path planned. The paths can be seen in Figure 26.

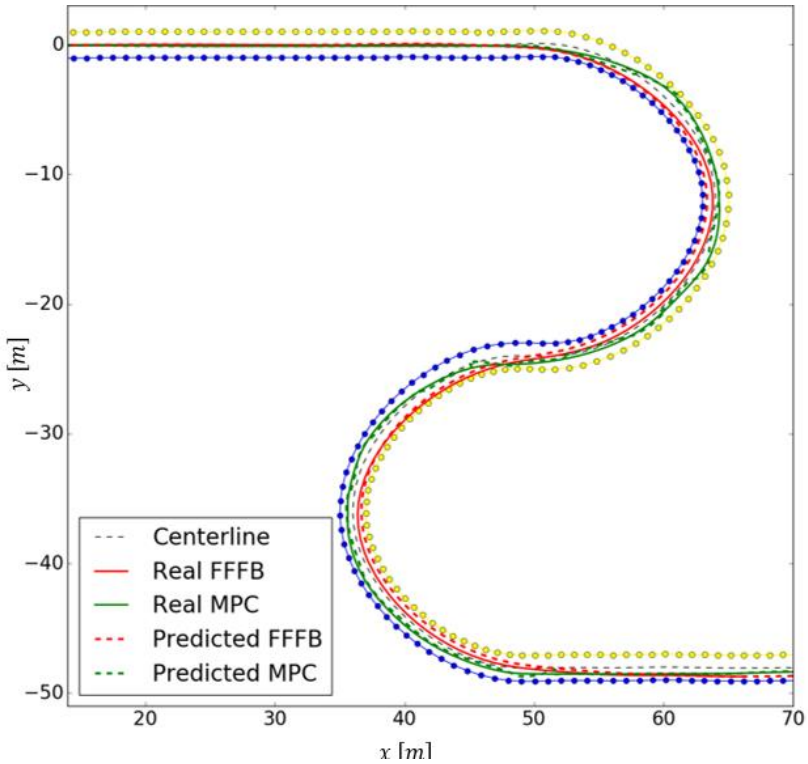

Figure 26. The test case investigating completion time on the double hairpin track, curve radius 12 m. A comparison between paths.

Initially both the MPC and FF-FB accelerate in a similar manner. Although, in the middle section of the curves there is an observable difference in average velocity, shown in Figure 27. The FF-

FB keeps a slightly higher average velocity in the curve and accelerates earlier out of the second turn.

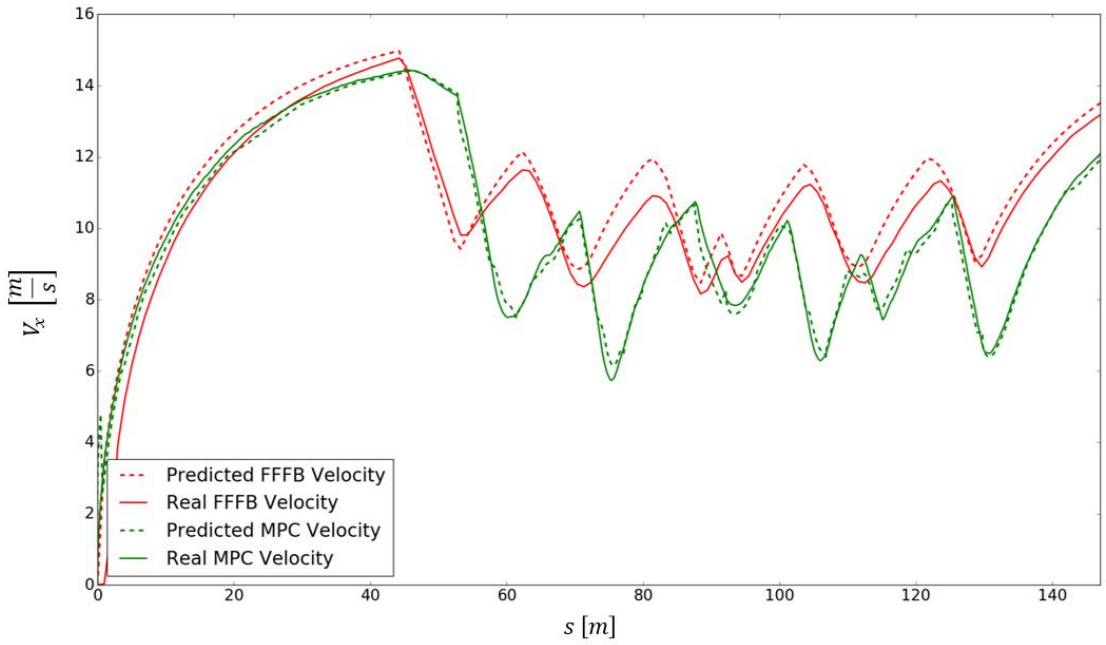

Figure 27. The test case investigating completion time on the double hairpin track, radius 12 m. A comparison between velocities.

The completion times of the test case investigating the double hairpin track with a curve radius of 12 m are presented in Table 7. The average completion time of the FF-FB trajectory planner is 13.88 seconds and 15.47 seconds for the 2-layer MPC, implying that the FF-FB trajectory planner is on average 1.59 seconds faster.

Table 7. Test results consisting of completion times for double hairpin track with curve radius 12 meters.

Trajectory planner	**1st run**	**2nd run**	**3rd run**	**Mean**
FF-FB	13.855	13.845	13.945	13.882
2-layer MPC	15.715	15.420	15.270	15.468

5.1.3 The 90-degree turn track, curve radius 6 meters

The paths driven and planned regarding each of the two trajectory planners when applied to the 90-degree turn track with a curve radius of 6 meter are presented in Figure 28. The MPC drives slightly further into the curve and follows the outer edge of the track. The FF-FB drives close to a path with the absolute minimum curvature and hits a normal apex similarly to the geometric racing line shown in Figure 1. Note that the FF-FB trajectory planner turns slightly left before entering the curve.

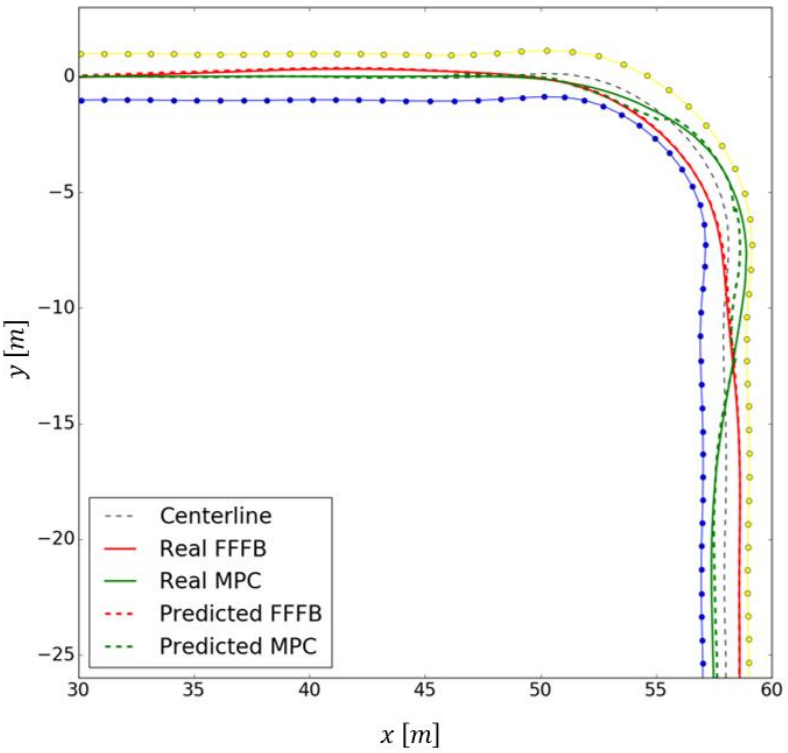

Figure 28. The test case investigating completion time on the 90-degre turn track, curve radius 6 m. A comparison between paths.

The velocity profiles during the test can be observed in Figure 29. The trajectory planners show a similar behaviour as in the test cases investigating the double hairpin track, the 2-layer MPC for instance brake later into the curve than the FF-FB trajectory planner does.

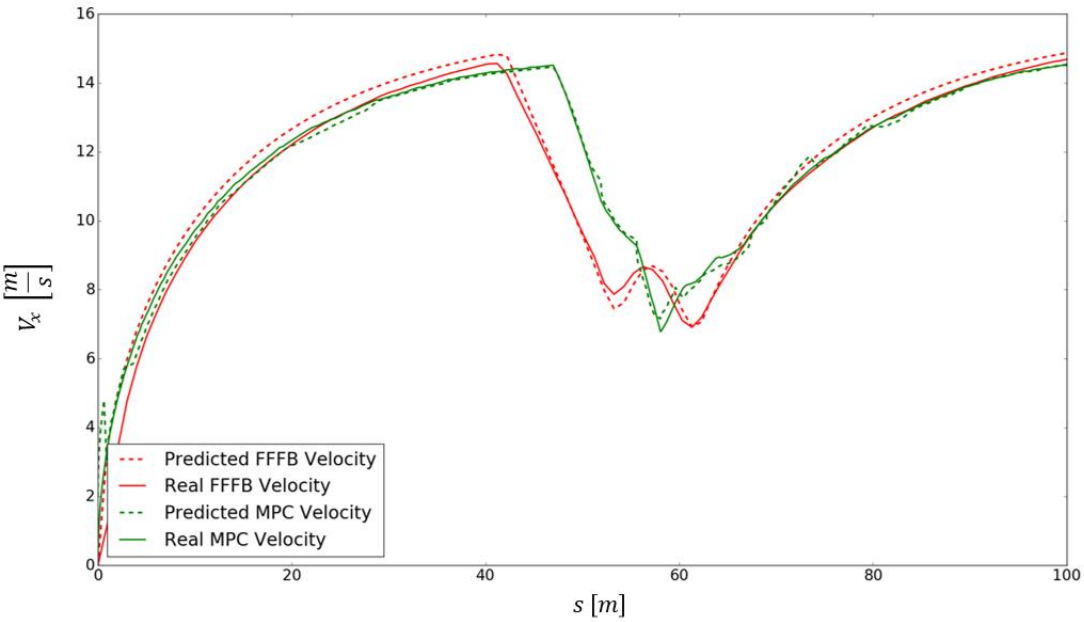

Figure 29. The test case investigating completion time on the 90-degree turn track, curve radius 6 m. A comparison between velocities.

The completion times regarding the test case investigating the 90-degree turn track with 6 meters curve radius are presented in Table 8. The average completion time of the FF-FB trajectory planner is 9.28 seconds and 9.30 seconds for the 2-layer MPC, implying that the FF-FB trajectory planner is on average 0.02 second faster.

Table 8. Test results consisting of completion times for 90-degee turn track with curve radius 6 meters.

Trajectory planner	1st run	2nd run	3rd run	Mean
FF-FB	9.305	9.285	9.255	9.282
2-layer MPC	9.210	9.375	9.305	9.297

5.1.4 The 90-degree turn track, curve radius 12 meters

A similar path as presented in Figure 28 can also be seen in Figure 30. The only observable difference is that the FF-FB does not turn to the left before entering the curve to reduce the curvature of the race line.

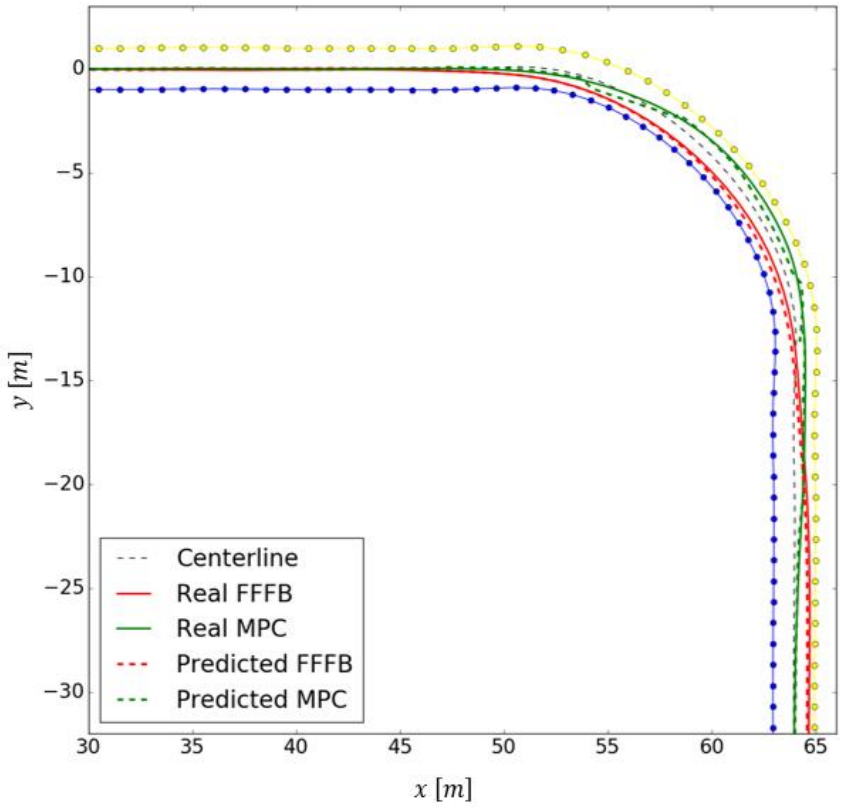

Figure 30. The test case investigating completion time on the 90-degree turn track, curve radius 12 m. A comparison between paths.

A comparison of the velocity profiles regarding the test cases investigating the 90-degree turn track with a curve radius of 12 meters is presented in Figure 31. There is an observable slight shift sideways in the velocity profile of the MPC and the velocity profile of the FF-FB trajectory planner. The FF-FB also has a higher average velocity in the curve in comparison to the MPC.

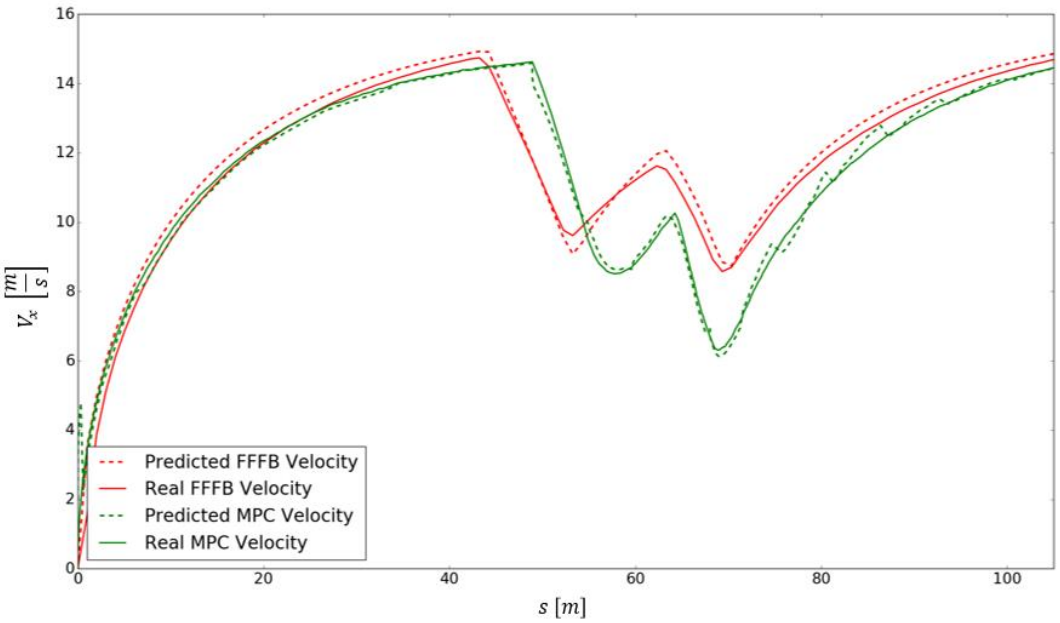

Figure 31. The test case investigating completion time on the 90-degree turn track, curve radius 12 m. A comparison between velocities.

The completion times of the test case are presented in Table 9. The average completion time of the FF-FB trajectory planner is 9.23 seconds and 9.51 seconds for the 2-layer MPC, implying that the FF-FB trajectory planner is on average 0.28 second faster.

Table 9. Test results consisting of completion times for 90-degree turn track with a curve radius of 12 meters.

Trajectory planner	1st run	2nd run	3rd run	Mean
FF-FB	9.235	9.210	9.245	9.230
2-layer MPC	9.830	9.360	9.330	9.507

5.2 Sensitivity against mass alteration

An observation and comparison of the trajectory planners regarding their sensitivity against mass alteration will be carried out in this section. Each subsection contains completion times and a minimum of two figures representing the path and velocity during the first test run. The trajectory planners are applied to the double hairpin track with a curve radius of 6 m with either increased or decreased mass. The test cases are conducted using the real time applied trajectory planner in the simulation environment FSSIM.

Figure 32 and Figure 35 contains two paths of the planned trajectory (dashed lines), two paths of the actual trajectory driven (solid lines) by the vehicle during the test and one path describing the centreline of the track. Figure 34 and Figure 36 contains two velocities describing the planned velocity of the trajectory planners and two additional velocities describing the actual velocity of the trajectory planner.

5.2.1 The double hairpin track, curve radius 6 m and 20% increased mass

The resulting paths of each of the trajectory planners regarding the test case investigating the double hairpin track with a curve radius of 6 meters and 20% increased mass are presented in

Figure 32. The trajectory planners show a similar behaviour as shown in Figure 22 but with a decrease in performance. The 2-layer MPC drives and plans slightly longer into the first curve before turning compared to the test with normal mass, the same goes for the second turn. As mentioned, the 2-layer MPC possess feedback while planning which is why there is almost no difference between the planned and driven path. The FF-FB trajectory planner, however, plans almost the same path as without altered mass, although it has difficulties following the generated path due to the extra mass. The controller cannot steer to keep the planned path with the planned velocity and thus slides away from the desired path. The FF-FB trajectory planner furthermore drives on top of some cones in the second curve as can be seen in Figure 32.

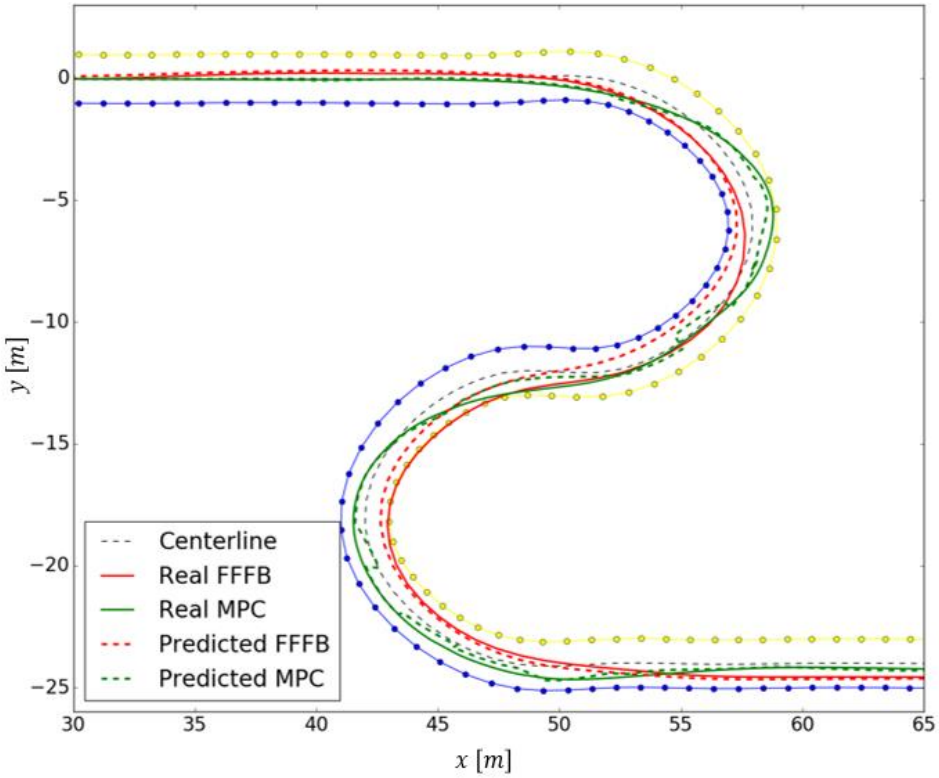

Figure 32. The test case investigating sensitivity to mass alteration on the double hairpin track, curve radius 6 m and increased mass. A comparison between paths.

The lateral controller output of the FF-FB trajectory planner is presented in Figure 33-a and in Figure 33-b for the 2-layer MPC. In the figures the lateral output from the test case with normal mass can also be observed. As shown, both controllers output a higher steering angle in the first turn with an increased mass. In the second turn the difference is steering angle is less obvious. The 2-layer MPC furthermore has a jerkier behaviour with an increased mass, while the behaviour of the FF-FB trajectory planner it is quite similar to the behaviour observed with original mass.

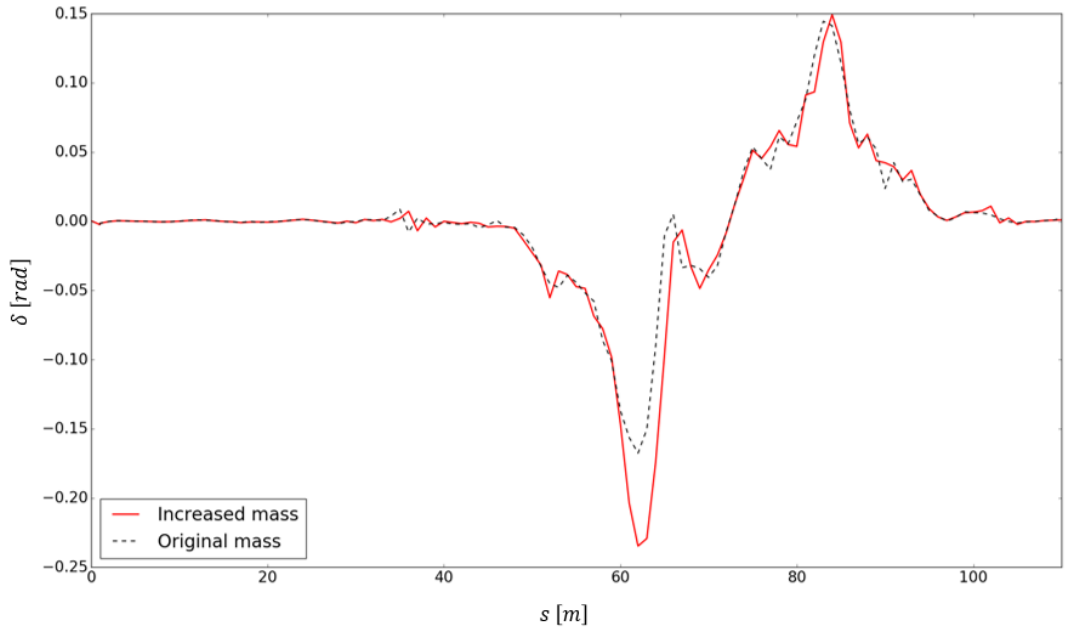

Figure 33-a. The test case investigating sensitivity to mass alteration on the double hairpin track, curve radius 6 m and increased mass. Lateral control output for the FF-FB trajectory planner.

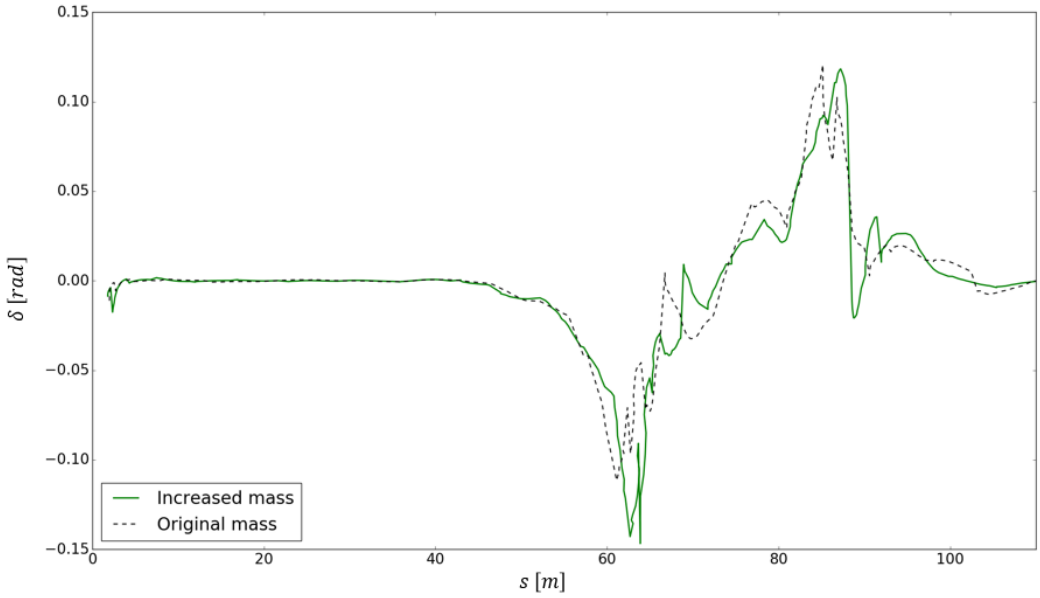

Figure 33-b. The test case investigating sensitivity to mass alteration on the double hairpin track, curve radius 6 m and increased mass. Lateral control output for the 2-layer MPC.

A comparison of the velocity profiles regarding the same test case is presented in Figure 34. The FF-FB trajectory planner does not manage to accelerate or decelerate to follow the planned velocity with the increased mass. It can, for instance, be seen during the initial acceleration part; the vehicle does not reach the planned velocity. Moreover, even though the actual velocity is lower than planned velocity it does not manage to decelerate to the desired velocity before entering the first curve. The 2-layer MPC also plans to accelerate quicker, although because the MPC uses feedback while planning the predicted velocity deviates less from the actual velocity. This behaviour can be observed in the initial acceleration phase. Small spikes can be observed in the planned velocity of the MPC which implies that it tries to accelerate faster than possible as a result of the increased mass.

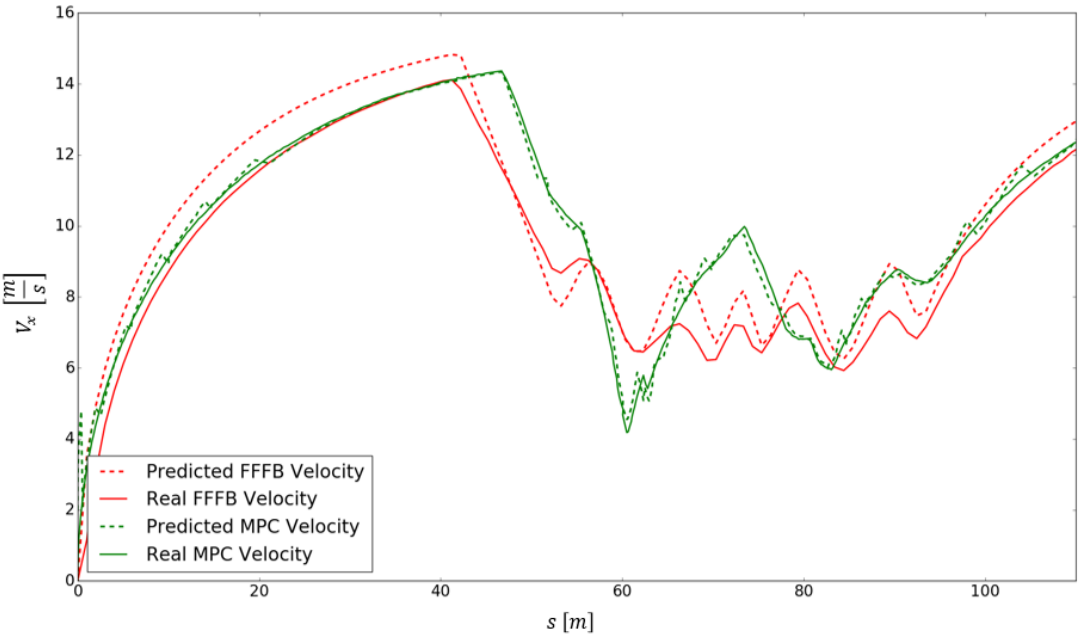

Figure 34. The test case investigating sensitivity to mass alteration on the double hairpin track, curve radius 6 m and increased mass. A comparison between velocities.

The completion times regarding the test case investigating the double hairpin track with a 6 meters curve radius and 20% increased mass are presented in Table 10. The average completion time of the FF-FB trajectory planner is 12.09 seconds and 12.30 seconds for the 2-layer MPC, implying that the FF-FB trajectory planner is on average 0.21 seconds faster.

Table 10. Test results consisting of completion times for double hairpin track with a curve of radius 6 meter and with 20% increased mass.

Trajectory planner	**1st run**	**2nd run**	**3rd run**	**Mean**
FF-FB	12.065	12.120	12.075	12.087
2-layer MPC	12.305	12.375	12.225	12.302

5.2.2 The double hairpin track, curve radius 6 meters and 20% reduced mass

The resulting paths of each of the trajectory planners regarding the test case investigating the double hairpin track with a curve radius of 6 meters and 20% decreased mass are presented in Figure 35. There is no obvious difference between the paths with the reduced mass compared to the case with the normal mass, shown in Figure 22.

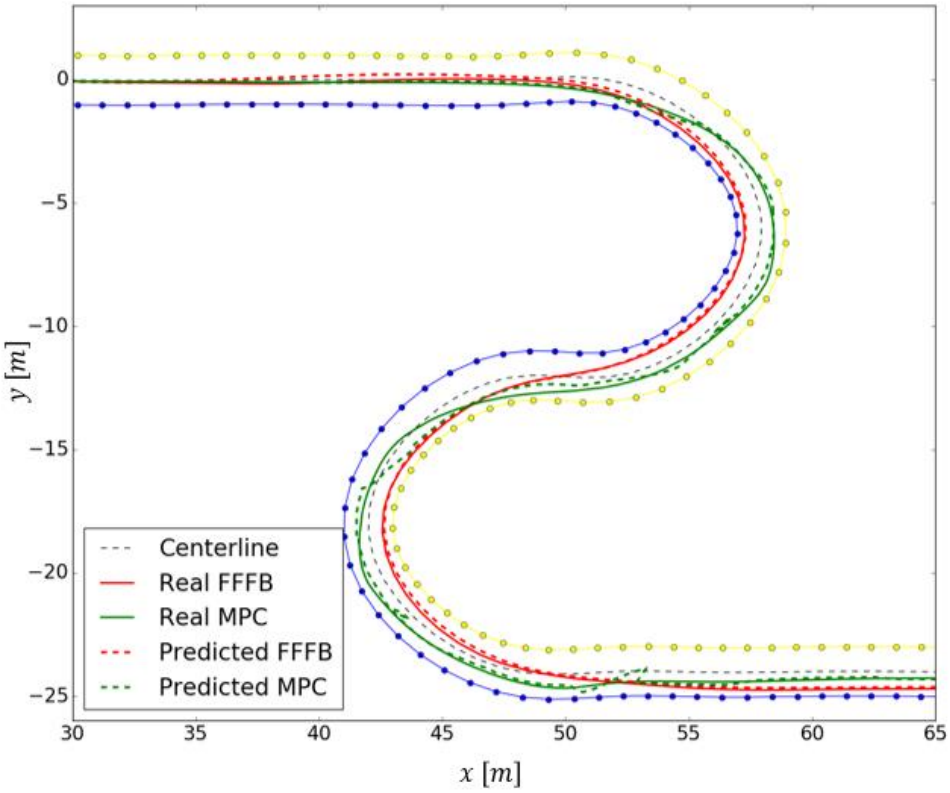

Figure 35. The test case investigating sensitivity to mass alteration on the double hairpin track, curve radius 6 m and decreased mass. A comparison between paths.

A comparison of the velocity profiles regarding the same test case is presented in Figure 36. Both trajectory planners follow the velocity profile closely. The only observable difference when comparing the normal mass, see Figure 24, to the reduced one is that the FF-FB initially accelerates and decelerates slightly faster than planned until the feedback longitudinal control compensates for the error.

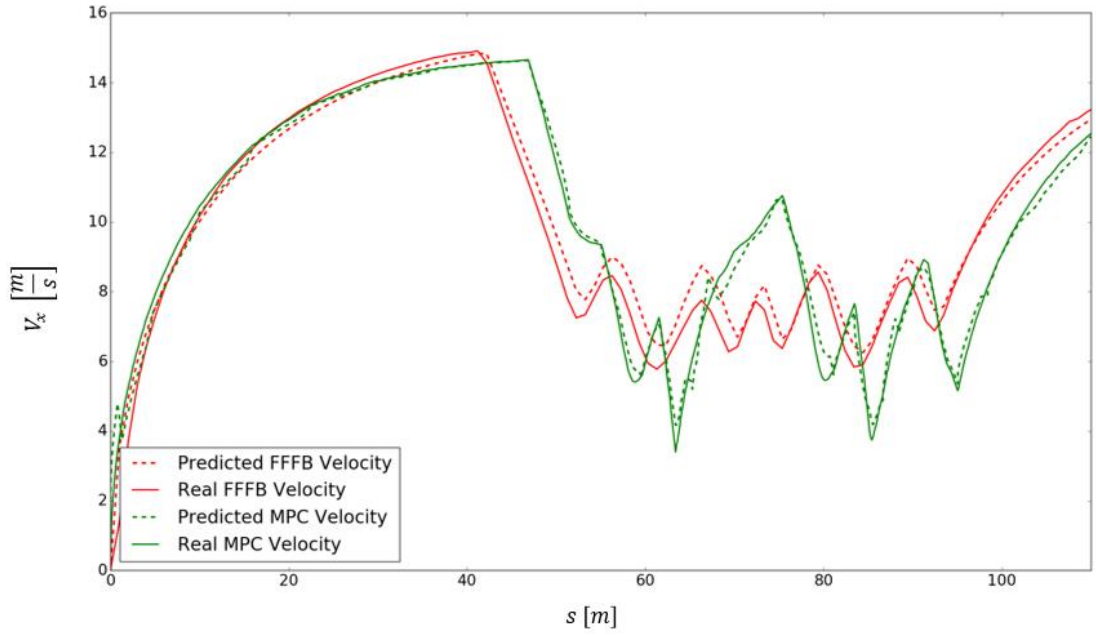

Figure 36. The test case investigating sensitivity to mass alteration on the double hairpin track, curve radius 6 m and reduced mass. A comparison between velocities.

The completion times regarding the test case investigating the double hairpin track with a 6 meters curve radius and 20% decreased mass are presented in Table 11. The average completion time of the FF-FB trajectory planner is 11.50 seconds and 12.44 seconds for the 2-layer MPC, implying that the FF-FB trajectory planner is on average 0.94 seconds faster.

Table 11. Test results consisting of completion times for double hairpin track with a curve radius of 6 meter and with 20% reduced mass.

Trajectory planner	**1st run**	**2nd run**	**3rd run**	**Mean**
FF-FB	11.445	11.515	11.540	11.500
2-layer MPC	12.190	12.590	12.530	12.437

5.3 Obstacle avoidance

To test the trajectory planners' ability to avoid obstacles a set of two tests were conducted. The tests were conducted using the approach presented in Section 4.1.1. The first failed run disqualifies the trajectory planner and the shortest successive successful detection distances of each of the trajectory planners are compared.

Figure 37 and Figure 38 contains two paths of the planned trajectory (dashed lines), two paths of the actual trajectory driven (solid lines) by the vehicle during the test and one path describing the centreline of the track. The figures furthermore include a line representing when the trajectory planner is able to detect the obstacle.

5.3.1 Obstacle avoidance, 8 m/s

The detection distances where the trajectory planners successfully or unsuccessfully avoided the obstacle without driving outside the track are presented in Table 12. Each run marked with *Completed* means that the trajectory planners managed to complete the run. Distances marked with a blue colour indicates the shortest detection distance for one trajectory planner. The green boxes represent the original parameter configuration used in the previous test cases while the yellow boxes represent customized parameters specifically made to shorten the detection distance of the FF-FB trajectory planner at 8 m/s. In the current case, the yellow boxes represent a parameter configuration of $S = 15$ and $S_{locked} = 7$ unlike the original parameter configuration of $S = 45$ and $S_{locked} = 10$. The reduction in the parameters values of S and S_{locked} results in a shortened planning and locking distance which in turn reduce computational time and thus increase update frequency. As can be seen in Table 12, it does reduce the minimum obstacle detection distance from 18 meter to 11 meters. The 2-layer MPC on the other hand manages to avoid the obstacle with a 7 meters detection distance without alternating the parameter configuration.

Table 12. Test results for obstacle avoidance at 8 m/s.

Distance (m)	FF-FB 1st	FF-FB 2nd	2-layer MPC 1st	2-layer MPC 2nd
20	Completed	Completed	Completed	Completed
19	Completed	Completed	Completed	Completed
18	Completed	Completed	Completed	Completed
17	Failed	Failed	Completed	Completed
16	Completed	Completed	Completed	Completed
15	Completed	Completed	Completed	Completed
14	Completed	Completed	Completed	Completed
13	Completed	Completed	Completed	Completed
12	Completed	Completed	Completed	Completed
11	Completed	Completed	Completed	Completed
10	Failed	Completed	Completed	Completed
9	-	-	Completed	Completed
8	-	-	Completed	Completed
7	-	-	Completed	Completed
6	-	-	Failed	Failed

5.3.2 Obstacle avoidance, 12m/s

The detection distances where the trajectory planners successfully or unsuccessfully avoided the obstacle without driving outside the track are presented in Table 13. The data is presented in the same manner as explained in Section 5.3.1. The customized parameter configuration for the FF-FB trajectory planner was altered differently to match the velocity of 12 m/s. In the current case the yellow boxes represent parameter configuration of $S = 24$ and $S_{locked} = 9$, unlike the original parameter configuration of $S = 45$ and $S_{locked} = 10$. The original FF-FB setup managed an obstacle detection distance of 24 meters while the customized FF-FB setup managed an obstacle detection distance of 18 meters. The 2-layer MPC managed an obstacle detection distance of 12 meters without altering the parameter configuration.

Table 13. Test results for obstacle avoidance at 12 m/s.

Distance (m)	FF-FB 1st	FF-FB 2nd	2-layer MPC 1st	2-layer MPC 2nd
25	Completed	Completed	Completed	Completed
24	Completed	Completed	Completed	Completed
23	Failed	Completed	Completed	Completed
22	Completed	Completed	Completed	Completed
21	Completed	Completed	Completed	Completed
20	Completed	Completed	Completed	Completed
19	Completed	Completed	Completed	Completed
18	Completed	Completed	Completed	Completed
17	Completed	Failed	Completed	Completed
16	-	-	Completed	Completed
15	-	-	Completed	Completed
14	-	-	Completed	Completed
13	-	-	Completed	Completed
12	-	-	Completed	Completed
11	-	-	Failed	Completed

The minimum detection distance of the FF-FB trajectory planner when driving 12 m/s is thus 24 meters with the original parameter configuration. A comparison of 2-layer MPC and FF-FB trajectory planner utilizing a detection distance of 24 meters is presented in Figure 37. The reaction distance is significantly shorter for the 2-layer MPC, it manages to drive to the side of the road before the FF-FB has started to turn.

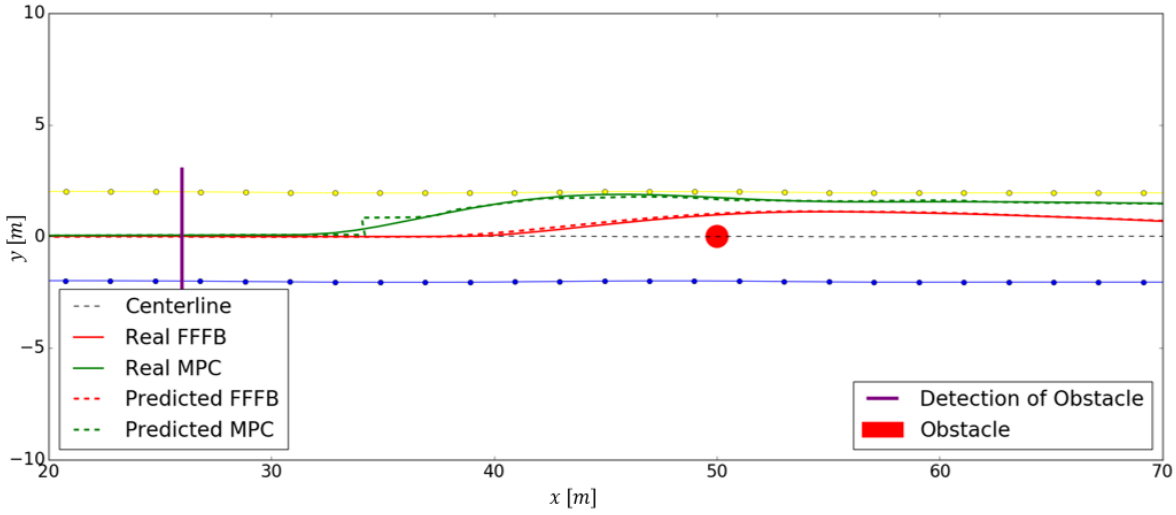

Figure 37. The planned and driven path of obstacle scenario at 12 m/s with a detection distance of 24 meters.

A comparison of the FF-FB trajectory planner and the 2-layer MPC regarding a detection distance of 12 meters, which is the minimum detection distance of the MPC at 12 m/s, is furthermore presented in Figure 38. The FF-FB trajectory planner fail by driving straight through the obstacle, as shown in the figure. The FF-FB path planner, however, manages to plan around the object, although the velocity is too high for the vehicle to follow the planned path. Worth noting is that the performance of the FF-FB varied between each run with the current detection distance. It did, for instance, during some tests not even react on the obstacle before it had passed it. The performance depended highly on where it was in the planning algorithm when reaching the detection distance. The FF-FB trajectory planner furthermore does not get back to the planned path immediately after the obstacle which can be explained by a conflict between the feedforward and feedback part of the lateral controller. The 2-layer MPC, on the other hand, avoids the obstacle successfully.

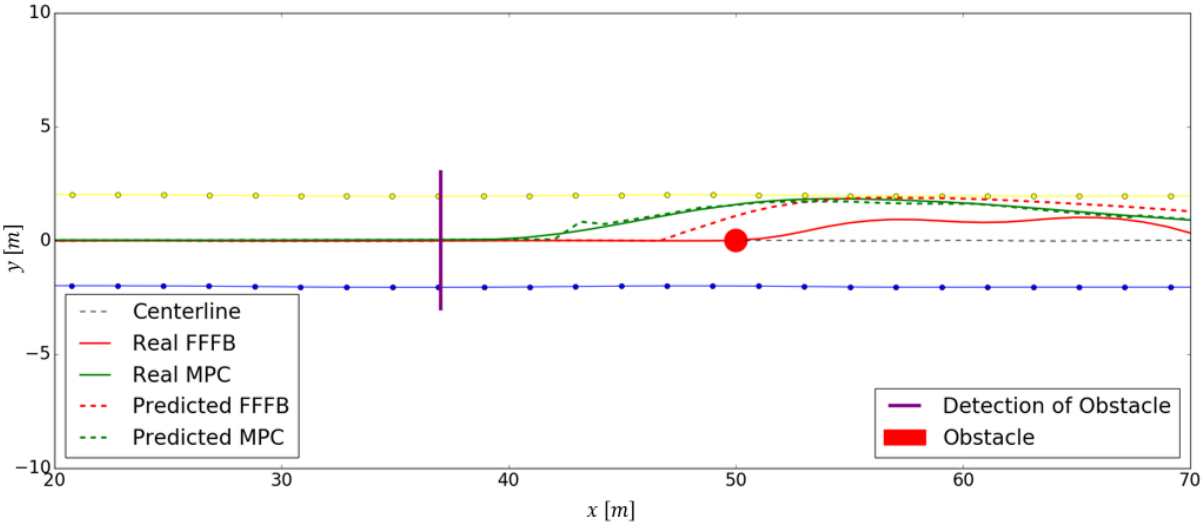

Figure 38. The planned and driven path of obstacle scenario at 12 m/s with a detection distance of 12 meters.

6 Discussion

In this chapter the trajectory planners are evaluated and discussed. The results are investigated and used to properly compare the trajectory planners in terms of the stated research question.

6.1 Discussion

To properly understand the results presented they will be further discussed in this section. The section is divided into three subsections, one subsection for each of the three parts forming the complete research question. As a reminder, the research question is:

- Considering the control strategies of a feedforward-feedback trajectory planner and a 2-layer MPC, which trajectory planner has a better performance regarding the following:
    - Minimizing completion time.
    - Sensitivity against modelling errors.
    - Minimizing detection distance when an obstacle is introduced to the track.

6.1.1 Comparison of trajectory planners concerning completion time

To investigate and compare the performance considering minimizing completion time, four test cases were conducted for each of the two trajectory planners. In each of the four test cases, the FF-FB trajectory planner rendered a shorter average completion time than the 2-layer MPC. Although, the difference in average completion time varied significantly from 0.02 seconds on the 90-degree turn track with radius 6 meters to 1.59 seconds on the double hairpin track with radius 12 meters. The double hairpin track with a radius of 6 meters rendered a difference of 0.36 seconds and the 90-degree turn track with a radius of 12 meters rendered a difference of 0.28 seconds, both in favour of the FF-FB trajectory planner. The FF-FB trajectory planner was in average 0.56 seconds faster, comparing the completion times on all tracks. Section 5.1.1 furthermore shows that the difference in completion time is statistically significant for the double hairpin track with a curve radius of 6 m test case. The difference in average completion time is furthermore greater when the trajectory planners are running on a track with a 12 m curve radius compared to a track with 6 m curve radius as well as when running on a double hairpin track compared to a 90-degree turn track. Subtracting all completion times of the FF-FB trajectory planner with the completion times of the 2-layer MPC results in the following table, see Table 14.

Table 14. Representation of the difference in completion time between FF-FB and MPC in the different test cases.

	6 meter radius	12 meter radius	Average
90-degree turn track	0.015	0.277	0.1460
Double hairpin track	0.362	1.586	**0.9740**
Average	0.1885	**0.9315**	

The difference with respect to completion time regarding the trajectory planners can be explained by their driven paths shown in Section 5.1.1-5.1.4. The 2-layer MPC acts in accord to what is given by the cost function explained in Section 4.2.2, which is aims to maximize velocity while minimizing deviations in steering and acceleration dependent on certain weighting scalars.

In other words, the MPC will keep the highest possible velocity and avoid steering until it can detect a curve and thereafter act by reducing the velocity and initiate steering in order to avoid violating the limits of the track. The 2-layer MPC consequently drives along the outer edges of the

track when applied to the double hairpin tracks, see Figure 22 and Figure 26. The behaviour can be explained by the short prediction horizon of the 2-layer MPC in comparison to the FF-FB trajectory planner. A planner with longer prediction horizon (or planning distance) can plan further ahead and thus spot a curve or obstacle earlier. The FF-FB trajectory planner has longer planning distance and can thus, in the case of the double hairpin track, spot the outer edge of the track before entering it and plan to drive slightly outwards before entering the curve, resulting in better racing line. In Figure 22, the FF-FB trajectory planner drives slightly to the left before entering the curve to the right which is a direct effect of having a longer detection horizon.

One reason why the difference in average completion time is reduced on the 90-degree turn track could be because the track has less curves that can affect the completion time. The exit of the 90-degree turn track does not affect the entry of any following turn which is favourable for the 2-layer MPC due to the short prediction horizon. Another reason could be that the total length of the track is shorter in comparison to the double hairpin tracks.

If the MPC, however, would have a longer prediction horizon, it might have been able to plan for the next turn and thus obtain a better racing line. This racing line can be expected to look something like the paths of the offline MPC, see Figure 21-a and Figure 21-b. The offline MPC sometimes have a racing line with similarities to the optimal racing line including a late apex on the double hairpin track as shown in Figure 2.

The difference in average completion time with larger curve radius could, in this case as well, be explained by the short prediction horizon of the 2-layer MPC, in addition to the increased velocities. When driving at high velocities it is crucial to still be able to brake and steer, therefore the trajectory planner must plan to be able to brake before the end of the prediction horizon which could limit the top speed within the curve. There is an observable difference where the FF-FB has a higher average velocity within the curves during the test cases with 12 meter curve radius compared to 6 meter curve radius, as can be observed in Figure 27 and Figure 31.

When considering a pure racing scenario both trajectory planners seem to generate a better racing line with increased prediction horizon. The FF-FB would, for instance, perform better as an offline planner since it does not require any feedback in the generation of path or velocity profile which the 2-layer MPC does. The parameters used while conducting the tests were chosen to allow for sufficient performance regarding both completion time and obstacle avoidance. When considering completion time, the FF-FB would presumably show a slight improvement in performance with a longer planning distance. The 2-layer MPC would also presumably show increased performance with longer prediction horizon if the computation time and instability issues while planning did not limit it like it did.

6.1.2 Sensitivity against mass alteration

To observe and compare the trajectory planners with respect to sensitivity against modelling errors a mass alteration test was conducted. When observing the results from the tests with 20% increased and decreased mass the FF-FB trajectory planner showed better average completion times in both cases, as can be seen in Table 15.

Table 15. Average completion time on the double hairpin track with radius of 6 meters with mass alterations.

	20% reduced mass	Normal mass	20% increased mass
FF-FB	**11.500**	**11.764**	**12.087**
2-layer MPC	12.437	12.126	12.302

The FF-FB trajectory planner did have a shorter completion time when considering normal mass which results in an inaccurate comparison when comparing only completion times instead of the differences. The difference in time between the tests with normal mass and the tests with altered mass was therefore calculated, see Table 16. As can be seen, the FF-FB reduced its average completion time while the MPC increased its average completion time, when considering reduced mass. When considering the increased mass test case, the MPC, however, increased its completion time less than the FF-FB. One can thus assume that the FF-FB is less sensitive against reduced mass alterations while the MPC is less sensitive against increased mass alterations.

Table 16. The average difference in completion time depending on mass alteration.

	20% reduced mass	Normal mass	20% increased mass
FF-FB	**-0.264**	0	0.323
2-layer MPC	0.311	0	**0.176**

When considering sensitivity against mass alteration, it is important for the vehicle to keep both a short completion time but also to be able to complete the lap. For instance, during one of the test runs, the FF-FB touched some cones which suggests that it would have issues remaining inside the track limits with increased mass alterations. The 2-layer MPC on the other hand drove quite similar to the case with normal mass, which suggest that it would remain inside the track limits even with higher mass alterations. One reason why the MPC might be less sensitive towards increased mass could be the fact that it contains feedback when planning. The MPC would thus, if it starts to slide or the velocity remains higher than planned, construct the next trajectory according to these new circumstances. The FF-FB trajectory planner, on the other hand, assumes that the vehicle is driving on the path generated at a speed corresponding to the velocity profile. Therefore, if an error enters the system, it can only compensate for it using the controller utilized. The FF-FB trajectory planner furthermore, unlike the 2-layer MPC, incorporates a lateral feedforward-feedback controller depending on the now inaccurate mass of the vehicle, resulting in inaccurate steering outputs acted on. Moreover, a feedforward-feedback controller, unlike the simple feedback controller used by the MPC, results in less impact by the feedback part of the controller also affecting the trajectory planner's ability to compensate for errors.

When considering reduced mass, the FF-FB trajectory planner clearly has greater performance. The FF-FB reduced the average completion time with reduced mass, which is understandable, lowered mass in turn allows for faster acceleration and deceleration. The MPC on the other hand increased the average completion time more with reduced mass compared to increased mass which intuitively seems strange. The completion times for the 2-layer MPC, however, varied significantly between each run and were only conducted three times. The result is a high standard deviation of 0.18. Comparing the results of normal mass to the reduced mass did not result in a statistically significant difference and the results can thus be partially due to a coincidence. However, the results still suggest an extra sensitivity to reduced mass regarding the MPC as well as an extra sensitivity to increased mass regarding the FF-FB trajectory planner.

6.1.3 Obstacle avoidance

To observe and compare the trajectory planners' ability to avoid an obstacle introduced to the track, an obstacle avoidance test was conducted, testing the shortest detection distance achievable for each of the trajectory planners. When considering obstacle avoidance, the 2-layer MPC proved to work a lot better than the FF-FB. At both 8 m/s and 12 m/s, the MPC managed to avoid the obstacle with significantly lower detection distance as can be observed in Table 17. Even when the parameters S and S_{locked} of the FF-FB trajectory planner were customized to increase obstacle avoidance capabilities, it did not get close to the shortest distance managed by the MPC. The MPC even successfully manages to avoid the obstacle at a shorter detection distance while driving at 12 m/s than what the FF-FB manages while driving at 8 m/s with the original parameters.

Table 17. A comparison of lowest detection distance between the trajectory planners.

	Original FF-FB	Customized FF-FB	2-layer MPC
8m/s	18 meters	11 meters	**7 meters**
12m/s	24 meters	18 meters	**12 meters**

As mentioned in Section 4.2.1, the choice of making the trajectory planners offline was investigated. An offline FF-FB trajectory planner would result in a close to optimal minimum curvature race line and therefore probably lower completion times. An offline planner would, however, not be able to detect an obstacle at all, due to the path and velocity planning done prior to driving. To encapsule both these behaviours without troubles with computational load, the FF-FB trajectory planner was made online using a locking distance. A locking distance S_{locked} of 10 meters was used in the original FF-FB trajectory planner. The definition of the locking distance is the distance ahead, starting from the current vehicle position, which cannot be altered by the path planner. A locking distance of 10 m therefore results in a fixed path 10 m ahead of the vehicle position, consequently limiting the obstacle avoidance capabilities. The FF-FB trajectory planner, additionally, has a computation time of almost one second for the planning algorithm, further limiting the obstacle avoidance. An obstacle appearing at the start of the planning algorithm thus delays the trajectory's planner ability to act by approximately one second. An alternative version of the FF-FB trajectory planner was therefore developed to increase obstacle avoidance utilizing a parameter configuration with altered planning and locking distance. The new parameter configuration, however, deteriorate the racing performance with respect to completion time. The choice of parameter configuration, for the FF-FB trajectory planner, must therefore be set taking the obstacle avoidance capabilities and racing performance into consideration, which is what the original parameter configuration was optimized for.

Unlike the FF-FB, the MPC does not require any locking distance at all and updates the trajectory more frequently which is the reason it manages such low detection distances. Figure 37 clearly shows how much faster the MPC initiates steering after detection of an obstacle. Even though both trajectory planners manage to avoid the obstacle, the 2-layer MPC starts steering toward the road limits earlier compared to the FF-FB trajectory planner. With the shorter detection distance shown in Figure 38, a similar behaviour is observable where the MPC starts steering to the side of the road earlier after detection. The FF-FB trajectory planner, however, has time to plan a path around the obstacle, although the path generated is unfeasible, resulting in the vehicle driving through the obstacle. The path of the FF-FB is generated before the velocity profile and thus does not take velocity into account, which may result in a path that the vehicle cannot follow.

Generally, the MPC both performed more stable and achieved a lot lower detection distance. The FF-FB performance varied significantly between runs depending on where it was in the planning algorithm when reaching the detection distance. The MPC also did have variations depending on where in the planning algorithm detection occurred, although the shorter computation time for each updated step reduced the effect.

7 Conclusion and Future Work

In this chapter the conclusion of the project as well as recommendations for future work are presented. Additional steps for further development and testing by implementation on a real test vehicle are also presented.

7.1 Conclusion

Autonomous racing is an interesting field which could lead to innovations applicable both to the racing industry as well as to autonomous cars. For increased margin of safety, the vehicle at hand must be able to plan and manoeuvre at the limits of traction. Both trajectory planners have proven that it is possible to handle a vehicle close to the limit of traction in a controlled and satisfactory manner even when slight modelling errors are present. At the same time the trajectory planners have highlighted the differences between them, including the advantages respectively disadvantages they contain.

When considering traditional autonomous racing with an empty track which the vehicle should complete as fast as possible, the FF-FB trajectory planner does achieve the fastest completion times. The difference varies depending on the test case, although generally this is the case. If the vehicle model would have been incorrectly modelled for example by having slightly incorrect mass the FF-FB would still achieve the fastest completion times. The results have, although, shown that the 2-layer MPC is less sensitive considering an increased mass alteration and is expected to keep itself within track limits in a more satisfactory manner in this case. The FF-FB trajectory planner is, however, less sensitive considering an alteration reducing the mass of the vehicle.

The 2-layer MPC required a significantly shorter detection distance to avoid an appearing obstacle in a satisfactory manner. The MPC required about half the distance to be able to replan and control to avoid the obstacle. Even when using a customized parameter configuration of the FF-FB trajectory planner made to reduce the detection distance, the FF-FB still required a significantly longer distance.

In a classic autonomous race competition, the FF-FB trajectory is preferred if the vehicle model is somewhat properly modelled. The FF-FB trajectory planner furthermore requires an empty track when driving and a map of the track beforehand, which commonly is the case for autonomous racing. Although, if a map of the track is not given in advance, the result is a higher required map update frequency resulting in a shorter planning distance. Decreasing the planning distance of the FF-FB trajectory planner furthermore decreases its performance and thus also its advantage compared to the 2-layer MPC. Furthermore, if other vehicles or unknown obstacles were introduced along the track, the 2-layer MPC is preferred, due to superior obstacle avoidance. Moreover, employing the trajectory planners in safety systems, in for instance an autonomous car, a further development of the 2-layer MPC is favourable due to its ability to act on unknown impediments in a faster manner compared to the FF-FB trajectory planner.

7.2 Future work

In the scope of the project at hand two trajectory planners have been developed and implemented in a simulation environment. The following chapter discusses additional tasks to improve the outcome of the project as well as proposes ways to continue the project.

7.2.1 Implementation on a test vehicle

The process of implementing the trajectory planners on a 1:10 scale race vehicle requires a few functions. Since trajectory planning and control were the main focuses of this thesis, perception, localization, and mapping were ignored which obviously would be required on a test vehicle. The trajectory planners in their current configuration furthermore require the track centreline to start planning.

The trajectory planners are constructed using ROS, indicating an untroublesome employment of the trajectory planners on the 1:10 scale race. The ROS environment provides an easy transition from the simulated signals given by the simulation environment to real signals given by sensors and accelerometers etc on the test vehicle. Some issues are, however, of course to be expected. For instance, the microcontroller must be quite powerful to generate trajectories in real time as well as act on the generated trajectories.

7.2.2 Reduction of computation time

The coding language used when developing both the 2-layer MPC and the FF-FB trajectory planner was python. Python supplies an easy syntax and lots of packages usable in the field of trajectory planning but does not deliver when it comes to computation times. A more suitable coding language when it comes to programs where the computational time is of essence, is for instance C++. Tamim shows in his article “How fast is C++ compared to Python?” an example showing that C++ is approximately 25 times faster than python in the case of generating DNA k-mers, articulating the massive difference in efficiency of C++ versus Python [39].

The algorithms forming the trajectory planners were furthermore not optimized in terms of lowering the computation time. Some functions could be remade to allow for faster computation within different steps of the planning process. The optimization solvers could, for instance, be replaced with a different one which may reduce computational times.

Reduced computation time could further be used to improve the trajectory planners’ performance, consequently allowing a reduced locking distance of the FF-FB trajectory planner increasing its performance in regard to obstacle avoidance. A reduced computation time would furthermore allow an increased prediction horizon of the 2-layer MPC, in turn presumably leading to a more optimal planner trajectory and thus reduced completion times.

7.2.3 Optimizing the prediction horizon of the Two-layer MPC

The prediction horizon of the 2-layer MPC was set to 30, proving the overall best performance after a trial-and-error session. The MPC furthermore control towards 10 of these calculated states before recalculation. A longer prediction horizon would increase the performance of the 2-layer MPC, because it would allow the planner to detect and plan for emerging curvatures and obstacles earlier. This prediction horizon is highly dependent on the computation time and sample time of the MPC, a longer prediction horizon results in longer computation times. A reduction of the computation time would hence allow a longer prediction horizon and thus a better performance. A future task could therefore be to optimize the prediction horizon and sample time of the 2-layer MPC with respect to its computation time. Qazani et al., for instance, suggests a optimizing the prediction horizon as well as the states controlled towards using a butterfly optimization problem [40].

8 References

[1] N. R. Kapania, “Trajectory planning and control for an autonomous race vehicle,” Ph.D. dissertation, Dept. Mech. Eng., Stanford University, 2016.

[2] A. Heilmeier, A. Wischnewski, L. Hermansdorfer, J. Bets, M. Lienkamp and B. Lohmann, “Minimum curvature trajectory planning and control for an autonomous race car,” *International Journal of Vehicle Mechanics and Mobility*, 2019.

[3] E. Alcalá, V. Puig and J. Quevedo, “LPV-MP planning for autonomous racing vehicle considering obstacles,” Robotics and Autonomous Systems, vol. 124, 2019.

[4] M. Ahlberg, “Optimization based trajectory planning for autonomous racing,” M.S. thesis, Dept. Veh. Eng., KTH, Stockholm, 2019.

[5] S. McCombes, “How to do a case study,” 2019. [Online]. Available: https://www.scribbr.com/methodology/case-study/. [Accessed Jan. 21, 2021].

[6] S. Crowe, K. Cresswell, A. Robertson, G. Huby, A. Avery and A. Sheikh, “The case study approach,” 2011. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3141799/. [Accessed Jan. 21, 2021].

[7] R. Bevans, “A guide to experimental design,” 2019. [Online]. Available: https://www.scribbr.com/methodology/experimental-design/. [Accessed Jan. 21, 2021].

[8] J. Seawright and J. Gerring, “Case Selection Techniques in Case Study Research,” *Political Research Quarterly*, vol. 61, no. 2, pp. 294-308, 2008. [Online]. Available: Sage Journals, https://journals.sageoub.com. [Accessed Jan, 21, 2021].

[9] L. D'Olimpio, “The trolley dilemma: would you kill one person to save five?,” 2016. [Online]. Available: https://theconversation.com/the-trolley-dilemma-would-you-kill-one-person-to-save-five-57111. [Accessed May. 10, 2021].

[10] A. Maxmen, “Self-driving car dilemmas reveal that moral choices are not universal,” Nature, 2018. [Online]. Available: https://www.nature.com/articles/d41586-018-07135-0. [Accessed May. 10, 2021].

[11] Institute of Mechanical Engineers, “Formula Student Rules 2020,” 2020. [Online]. Available: https://www.imeche.org/events/formula-student/team-information/rules. [Accessed Apr. 13, 2021].

[12] S. Mansell, “Driver61, How to Drive the Perfect Racing Line,” [Online]. Available: https://driver61.com/uni/racing-line/. [Accessed Apr. 13, 2021].

[13] Brittish automobile racing club, “DRIVER GUIDE,” [Online]. Available: https://www.bskc.co.uk/driverguide. [Accessed Apr. 13, 2021].

[14] K. M. Kritayakirana, “Autonomous vehicle control at the limits of handling,” Ph.D. dissertation, Dept. Mech. Eng., Stanford University, 2012.

[15] P. Polack, F. Altché, B. D'Andrea Novel and A. de La Fortelle, “The Kinematic Bicycle Model: a Consistent Model for Planning Feasible Trajectories for Autonomous Vehicle?,” in *2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA*. IEEE, 2017. pp. 812-818.

[16] J.-Y. Yu, “Lecture 0, Geophysics Fluid Dynamics,” presented to ESS227, University of California., CA, USA, Jan. 10, 2017. [PowerPoint slides]. Available: https://www.ess.uci.edu/~yu/class/ess228/lecture.0.introduction.2017.all.pdf. [Accessed Apr. 5, 2021].

[17] Stanford, "The tire model," [Online]. Available: http://www-cdr.stanford.edu/dynamic/bywire/tires.pdf. [Accessed Apr. 19, 2021].

[18] H. B. Pacejka, Tire and vehicle dynamics, Amsterdam: Boston : Elsevier/BH, 2012.

[19] E. Forsen, “Teaching Cars To Drive - Highway Path Planning,” 2018. [Online]. Available: https://towardsdatascience.com/teaching-cars-to-drive-highway-path-planning-109c49f9f86c. [Accessed Mar. 12, 2021].

[20] J. Fickenscher, S. Schmidt, F. Hannig, M. Essayed Bouzoura and J. Teich, “Path Planning for highly Automated Driving on Embedded GPUs,” *Low Power Electronics and Applications,* vol. 8, no. 35, 2018.

[21] ROS, “ROS,” [Online]. Available: https://www.ros.org/. [Accessed Mar. 26, 2021].

[22] J. Kabzan, M. d. l. I. Valls, V. Reijgwart, H. F. C. Hendrikx, C. Ehmke, M. Prajapat, A. Bühler, N. Gosala, M. Gupta, R. Sivanesan, A. Dhall, E. Chisari, N. Karnchanachari and S. Brits, “AMZ Driverless: The Full Autonomous Racing System,” Cornell University, 2019.

[23] K. Loung and H. Zheng, “F1TENTH Racecar Simulator,” 2020. [Online]. Available: https://github.com/f1tenth/f1tenth_simulator. [Accessed Mar. 12, 2021].

[24] R. Baijayanta, “A-Star (A*) Search Algorithm,” 2019. [Online]. Available: https://towardsdatascience.com/a-star-a-search-algorithm-eb495fb156bb. [Accessed Mar. 12, 2021].

[25] Wikimedia Commons, “File:Pathfinding A Star.svg,” 2006. [Online]. Available: https://commons.wikimedia.org/wiki/File:Pathfinding_A_Star.svg. [Accessed Mar. 12, 2021].

[26] E. Velenis and P. Tsiotras, “Minimum-Time Travel for a Vehicle with Acceleration Limits: Theoretical Analysis and Receding-Horizon Implementation,” *Journal of Optimization Theory and Applications,* vol. 138, no. 2, pp. 275-296, 2008.

[27] R. Griffiths, “Minimum time simulation of a racing car,” M.S. thesis, Dept. Mech. Eng., Cranfield University, 1992.

[28] J. C. Gerdes and J. Subosits, “Autonomous vehicle control for emergency maneuvers: The effect of topography,” in *2015 American Control Conference, Chicago, IL, USA, July 1-3,* 2015.

[29] Engineers Edge, “Rolling Resistance Equation and Calculation,” 2021. [Online]. Available: https://www.engineersedge.com/mechanics_machines/rolling_resistance_13633.htm. [Accessed May. 10, 2021].

[30] E. M. López, “Air Resistance: The Invisible Enemy in Vehicle Design,” 2019. [Online]. Available: https://www.simscale.com/blog/2017/06/air-resistance-vehicle-design/. [Accessed May. 10, 2021].

[31] H. B. Pacejka, “ THE MAGIC FORMULA TIRE MODEL,” in *Tyre and Vehicle Dynamics*, 2006, pp. 165-202.

[32] M. Agrawal, “What is Model Predictive Control (MPC)?,” *Control Automation,* 2020.

[33] B. Stellato, G. Banjac, P. Goulart, A. Bemporad and S. Boyd, “QSQP: an operator splitting solver for quadratic programs,” *Mathematical Programming Computation,* vol. 12, no. 4, pp. 637-672, 2020.

[34] “Scipy.org,” 2008. [Online serial]. Available: https://docs.scipy.org. [Accessed Apr. 19, 2021].

[35] J. Heider, “Quadratic programming,” 2015. [Online serial]. Avaible: https://optimization.mccormick.northwestern.edu/index.php/Quadratic_programming. [Accessed Apr. 19, 2021].

[36] D. Nykamp, “Introduction to Taylor's theorem for multivariable functions.,” Math Insight. [Online serial]. Available: https://mathinsight.org/taylors_theorem_multivariable_introduction. [Accessed Apr. 14, 2021].

[37] L. Svensson, “Github(larsvens/tamp_ws),” 2020. [Online]. Available: https://github.com/larsvens/tamp_ws. [Accessed Mar. 15, 2020].

[38] Z. Shiller, “Off-Line and On-Line Trajectory Planning,” in Motion and Operation Planning of Robotic Systems. G. Carbone and F.Gomez-Barvo, Ed. Cham: Springer International Publishing Switzerland, 2015, pp. 29-62.

[39] N. Tamimi, “How fast is C++ compared to Python?,” Towards Data Science, 2020.

[40] M. Qazani, S. Jalali, H. Asadi and S. Nahavandi, “Optimising Control and Prediction Horizons of a Model Predictive Control-Based Motion Cueing Algorithm Using Butterfly Optimization Algorithm,” in *IEEE CEC 2020*, Glasgow, 2020.