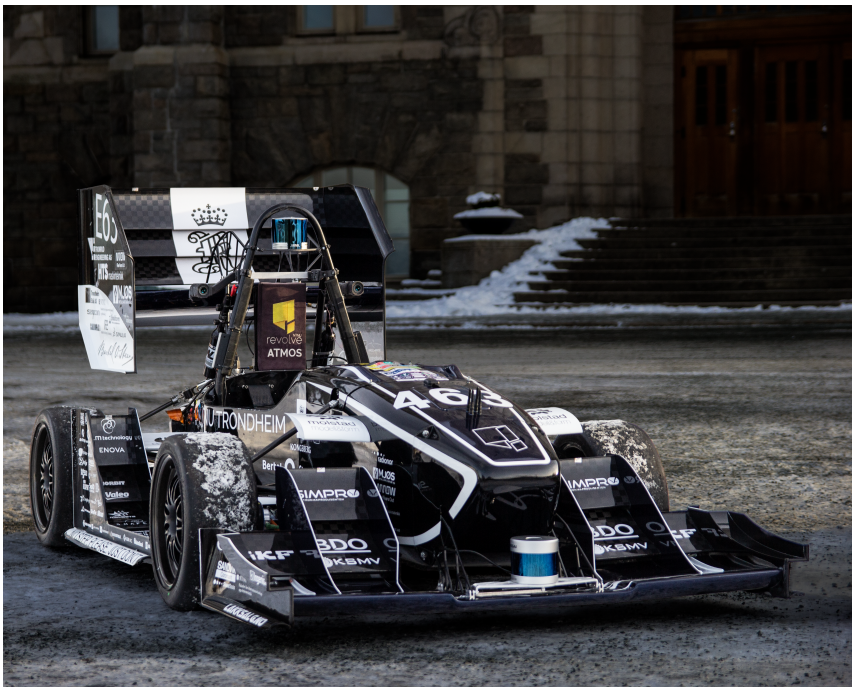
Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Christine Sääv Borg

Model Predictive Control for Lateral Path Tracking of an Autonomous Formula Student Race Car

Master's thesis in Cybernetics and Robotics
Supervisor: Sebastien Gros
June 2020



NTNU
Norwegian University of Science and Technology

Christine Sääv Borg

Model Predictive Control for Lateral Path Tracking of an Autonomous Formula Student Race Car

Master's thesis in Cybernetics and Robotics
Supervisor: Sebastien Gros
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Abstract

For decades, Formula Student has been a well-established engineering competition for students all over the world. The engineering competition was first introduced for students within the mechanical and electrical fields. However, in 2018 Formula Student held their first competition for driverless race cars. The autonomous competition opens for several new fields, from advanced perception systems to complex control systems. Within the field control systems, an important task is to track the desired path in the lateral direction. Deciding the optimal wheel angle, while avoiding unstable behavior may not be trivial, especially during the velocities and accuracy a race car aims for.

In this thesis, a lateral control system has been developed and implemented for the autonomous race car developed by the Formula Student team, Revolve NTNU. In the specialization topic associated with this thesis, an analysis was conducted which concluded that Model Predictive Control (MPC) will give the best results based on the desired behavior and the current stage of the development of the driverless vehicle. The MPC method is an optimal controller that will give the desired steering angle based on the predicted behavior of the vehicle. Two models are proposed for predicting vehicle behavior, a kinematic model, and a dynamic model. The kinematic model only takes into account the vehicle positions and velocities, whilst the dynamic model includes the forces and moments generated by the vehicle actions.

The resulting MPC implementations have shown results of varying levels through the three experiments, straight-line driving, constant radius circle, and the Formula Student inspired track. The MPC implementation using the kinematic vehicle model shows stable results for all given experiments, while the implementation using the dynamic vehicle model has difficulties when it comes to tracking the given curvatures. Additionally, the experiments have shown shortcomings when it comes to the implemented simulator. Using the MPC implementation that uses the kinematic vehicle model, the autonomous race car manages to do small radius cornering at unrealistic high velocities.

Sammendrag

I flere tiår har Formula Student vært en velletablert ingeniørkonkurranse for studenter over hele verden. Konkurransen har i hovedsak vært arrangert for å teste studentenes kunnskap og evner innenfor det mekaniske og elektriske fagfeltet. Som et resultat av den teknologiske utviklingen i verden, valgte Formula Student å holde sin første konkurranse for autonome racer biler sommeren 2018, noe som åpnet opp mulighetene for et helt nytt ingeniørfelt. Alt fra avanserte oppfatningssytemer til komplekse reguleringssystemer blir nå tested i ekstreme forhold. En av oppgavene til reguleringssystemet er å bestemme optimal styrevinkel basert på racerbanen, og samtidig unngå ustabil oppførsel, noe som ikke er trivielt for de gitte omstendighetene.

I denne oppgaven har et slikt lateralt reguleringssystem blitt utviklet og implementert for den førerløse bilen til Formula Student laget Revolve NTNU. I fordypningsprosjektet som er relatert til denne oppgaven, ble det utført en analyse som konkluderte med at modellprediktiv regulering (MPC) vil gi best resulat basert på ønsket oppførsel, i tillegg til utviklingsstadiet til den førerløse bilen i dag. MPC er en optimal regulator som predikerer bilens oppførsel, og basert på dette kan gi den optimale styrevinkelen til racer bilen. To modeller er forseslått for å predikere bilens oppførsel, en kinematisk og en dynamisk. Den kinematiske modellen tar kun hensyn til bilens posisjon og hastighet, mens den dynamiske også inkluderer krefter og momenter.

De resulterende MPC implementasjonene har vist varierende resultater gjennom de tre eksperimentene, rett bane, sirkel med konstant radius og en Formula Student inspirert bane. MPC implementasjonen som bruker den kinematiske bilmodellen viser stabile resultater for alle testene, mens MPC implementasjonen som bruker den dynamiske bilmodellen viser vanskeligheter når det kommer til å følge de gitte kurvaturene. Eksperimentene har i tillegg vist svakheter ved den implementerte simulatoren, ved at den autonome racer bilen kan gjennomføre svinger med urealistiske høye hastigheter.

Preface

This thesis is a representation of my work as a member of the Formula Student team, Revolve NTNU. This year has not been as I hoped, due to abnormal situation the whole world has faced in the last months. Unfortunately, my implementation will not be tested at the planned competitions in Spain and Germany. However, I would like to thank Revolve NTNU for giving me the opportunity to be a part of the team and letting me contribute to their driverless vehicle. Hopefully, my work will be revisited and the development will continue through a new member of the team. Revolve NTNU has taught me that hard work, dedication, and good teammates are the key to something great. I am proud to be an alumnus of Revolve NTNU.

I would like to thank Sebastien Gros for being my supervisor. Sebastien Gros helped me find the path that resulted in this project, by asking the correct questions. I would also like to thank my family and friends for always supporting me, even though they may not understand what I am doing.

Christine Sääv Borg

Trondheim, June 21, 2020

Contents

List of Tables

List of Figures

Acronyms

CG Center of Gravity. 9, 10, 12, 13, 37, 46, 49

GNSS Global Navigation Satellite System. xvii, 90, 96

HPIPM High-Performance Interior-Point. 30, 31

INS Inertial Navigation System. 90

LIDAR Light Imaging, Detection and Ranging. 90

LOS Line of Sight. 16, 18, 19

LPV Linear Parameter-Variant. 23, 83

MIMO Multiple-Input-Multiple-Output. 23

MPC Model Predictive Control. i, iii, xiii, xvi, 4–6, 21–24, 29, 31, 32, 35, 36, 39, 41, 43–45, 51, 71–73, 75–79, 81–83, 97, 101

OCP Optimal Control Problem. 30

PID Proportional–Integral–Derivative. 16, 18, 19

QP Quadratic Programming. 23, 26, 27, 29–31, 45, 77, 79, 81

ROS Robot Operating System. 83

sf Serret-Frenet. xiii, 8, 9, 16–18

SLAM Simultaneous Localization and Mapping. 90–92

Chapter 1

Introduction

This chapter is based on Chapter 1 from the specialization project associated with this thesis. Modifications have been made according to changes during the latter part of the project.

Autonomous vehicles are on their way to everyday roads, whether humanity likes it or not. In some countries, the first autonomous cars are already being tested in the wild. Autonomous cars will revolutionize transportation, how cities are built and probably change the lives of many people in unknown ways. The EU has estimated that autonomous vehicles will give 84.4 Billion Euro trade surplus for the European market[1]. Likewise, the industry is in a huge research and development phase around the world. Tesla is introducing self-driving equipment for all of their new cars, including traffic-aware cruise control and autosteer [2]. Even more impressive may be the autonomous race cars competing in Roborace, a global championship for autonomous race cars [3]. In Roborace the autonomous vehicles are tested at their limits, where both control and perception systems must perform at an excellent level during high-speed racing.

The engineering competition Formula Student has recognized the technology that comes with driverless vehicles as important for the future. Formula Student has therefore established a competition to encourage students around the world to develop skills regarding driverless race cars. One of the teams competing in the Formula Student Driverless class is the Norwegian based Formula Student team, Revolve NTNU, and this project is a contribution to the driverless vehicle made by Revolve NTNU.

The Formula Student Driverless competition is described in Section 1.1, while the Revolve NTNU is described in Section 1.2. In Section 1.3, the contributions of the product from

[1] For further information about the automobile industry in Europe, see `https://www.acea.be/automobile-industry/facts-about-the-industry`

[2] For further information about Tesla AutoPilot, see `https://www.tesla.com/autopilot`

[3] For further information about Roborace, see `https://roborace.com/`

this thesis are presented, and the structure of the report is described in Section 1.4.

1.1 Formula Student Driverless

Formula Student is a student competition organized to conceive, design, fabricate, develop and compete with small, formula style, race cars [1]. The competitions are arranged all over the world, where the most prestigious competition is at Hockenheimring, Germany. The driverless class of the Formula Student competition was introduced in 2017 and the first competitions were arranged during the summer of 2018.

The competition consists of two event categories, static and dynamic. The static events are divided into

- *Business Plan Presentation*,
- *Cost & Manufacturing*, and
- *Engineering Design*.

These static events are meant for testing the team's understanding of the vehicle and how decisions have been made throughout its development phase.

The dynamic events are divided into five events,

- *Skid Pad*,
- *Acceleration*,
- *Autocross*,
- *Efficiency* and
- *Trackdrive*,

where the goal is to push the vehicle to its limits. The points awarded to each event, both static and dynamic, are presented in Figure 1.1.

Before a vehicle is allowed to enter the dynamic events it has to go through scrutineering. During scrutineering, judges inspect the vehicle and make sure it follows all the rules the competition requires, given in the Formula Student Rules [1]. The rules may be specific for each competition, and therefore, it is important to have a good understanding of the rules and make the vehicle rule compliant.

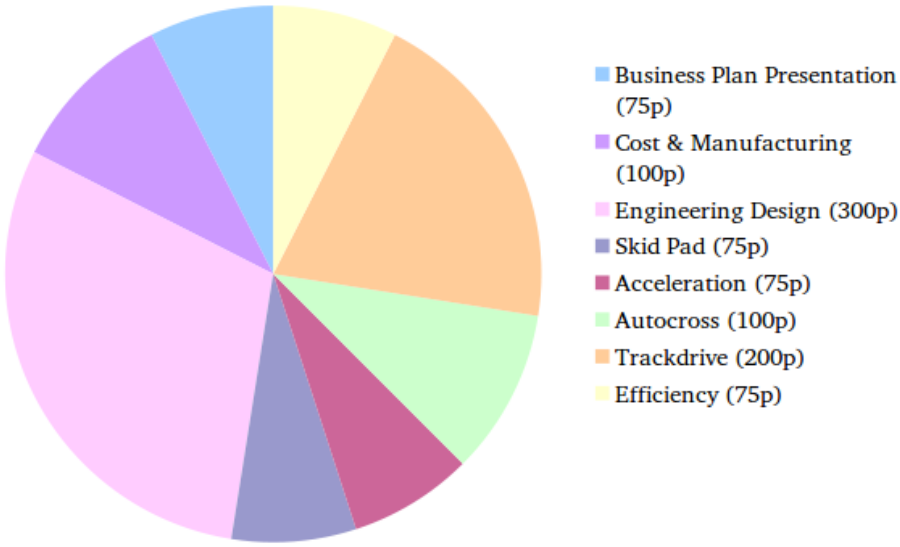


Figure 1.1: Points rewarded for each event

Formula Student also has an electrical and a combustion class, which is similar to Formula Student Driverless. However, in these classes, the race car is manned. This report only regards the driverless class, therefore, the electrical and combustion classes will not be described in detail.

1.2 Revolve NTNU

Revolve NTNU is an independent student organization founded in 2010 with the goal of competing in the Formula Student competition. Over the last decade, the team has expanded and is now competing in both the driverless class and the electrical class [2]. Every year, the electrical team of Revolve NTNU develops and builds a new electrical race car, while the driverless team improves the autonomous systems of the driverless vehicle. The team aims to compete in the respective classes all over Europe.

During last season, the performance of the driverless race cars in the Formula Student competitions had a major leap. The autonomous systems were more accurate and the vehicles drove faster. As a team, Revolve NTNU has always had high ambitions and aims at being one of the best Formula Student teams in the world. As a result, high goals are set for pushing each subsystem of the vehicle to perform at the desired level. The goals for the driverless vehicle are listed in Table 1.1, where the main goal is to be one of the top three teams in every competition Revolve NTNU attends.

Sub-goals	Dynamic Goals	Static Goals
Test ready car by the 13th of April	3.75 sec on acceleration	70/100 Cost and Manufacturing
Top 3 in all dynamic events	5.5 sec on skidpad	70/75 Business Presentation
Final in Engineering Design	10 m/s avg. on autocross	230/300 Engineering Design
	17 m/s avg. on trackdrive	

Table 1.1: Goals for Revolve NTNU Driverless

The ambitious dynamic goals, represented in Table 1.1, demands controllers with high accuracy and robustness, accounting for nonlinear behavior during high-speed events, like trackdrive. See Appendix B for a brief analysis of the nonlinear region of Atmos. Therefore it is of high interest to research the field of lateral controller that allows the vehicle to reach the high velocities needed to accomplish the given goals. In the specialization topic associated with this thesis [3], an analysis was conducted which concluded that an Model Predictive Control (MPC) will be the best approach for reaching the presented goals, given the current stage of the development of Revolve NTNU's autonomous race car. The control system designed in this thesis will be embedded into Atmos Driverless, shown in Figure 1.2. See Appendix A for an introduction to Atmos Driverless' sensor configuration and pipeline.

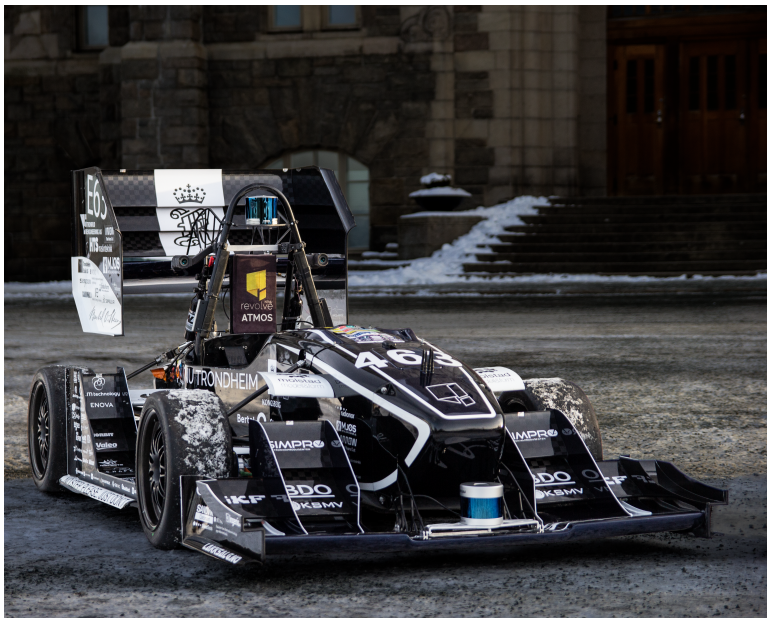


Figure 1.2: Revolve NTNU's driverless vehicle, named Atmos Driverless.

1.3 Contributions

This project is a contribution to the driverless vehicle of Revolve NTNU, and the following contributions have been made by the author:

- implementation of the simulation environment in MATLAB,
- implementation of the feedback linearization method in MATLAB,
- development and implementation of the MPC method using both a kinematic and a dynamic vehicle model in MATLAB,
- a discussion of the performance of the two MPC methods, including a comparison with the feedback linearization controller, and
- a timing comparison between the two MPC methods and the feedback linearization controller

1.4 Report Structure

Background

The background chapter aims to give a good insight into the lateral controller approached used by the previous Revolve NTNU team. To do so, a walkthrough of the notation and frames used throughout the report is conducted, also a description of the lateral modeling of the vehicle, including both the kinematic and the dynamic of the vehicle, is given.

Theory

The theory chapter gives an insight into the theory behind the MPC method, which is needed for understanding the two implementations of the method. The linearization method, in addition to the discretization method, are also presented for understanding the entirety of the MPC method.

Implementation

The two MPC implementations using the kinematic vehicle model and the dynamic vehicle model are presented in the implementation chapter. An overview of the implementation interface and the simulation environment is also given in this chapter.

Results

The results from the three experiments are presented in the results chapter. The experiments conducted are

- straight-line driving,
- constant radius cornering, and

- a track containing all elements of Formula Student Driverless Track described by the Formula Student rules [1].

Also, the computational effort of the MPC method, both for the kinematic and the dynamic vehicle modeling, in addition to the feedback linearization controller, are represented.

Discussion

The results of both the performance and the computational effort of the two MPC implementations are discussed in this chapter. Also, a comparison with the feedback linearization is conducted.

Epilogue

The conclusion of the overall performance of the MPC implementations is given in this chapter. Additionally, a discussion about further work is conducted.

Chapter 2

Background

This chapter is based on the Chapter 2 and the Chapter 3 from the specialization project associated with this thesis. Modifications have been made according to changes during the latter part of the project.

Some background knowledge is highly important for understanding and designing a control system for an autonomous race car. The notations and frames used throughout this report are described in Section 2.1. Further, the lateral vehicle modeling is described in Section 2.2, and the earlier approach, is described in Section 2.3 for later comparison with the new control system.

2.1 Nomenclature

To have a common understanding of the notation and frames used throughout the thesis, a summary is given. A more thorough description is given wherever needed. Some standard notations are used and described in Section 2.1.1. Further, the frames used throughout the report are represented in Section 2.1.2.

2.1.1 Mathematical Notation

Throughout the report, variables are set in italic, vectors and matrices are in bold, and constants in roman.

Some standard matrices are used throughout the report. $\mathbf{0}$ represents a zero matrix or a zero vector, and $\mathbf{I}$ denotes the identity matrix.

The standard notation that is given in SNAME (1950) [4] is used throughout the report and the notation can be seen in table 2.1. The SNAME (1950) notation, is the notation used in the maritime sector. However, controlling the motion of a vehicle is similar to controlling

the motion of a vessel. Therefore, the notation can be used in vehicle modulation and control.

DoF	Description	Forces and moments	Linear and angular velocities	Position and Euler angles
1	Motion in the x-direction	X	u	x
2	Motion in the y-direction	Y	v	y
3	Motion in the z-direction	Z	w	z
4	Rotation about the x-direction (roll)	K	p	ϕ
5	Rotation about the y-direction (pitch)	M	q	θ
6	Rotation about the z-direction (yaw)	N	r	ψ

Table 2.1: Summary of SNAME (1950) notation.

Some exceptions from the SNAME (1950) notation are done, where the forces are defined as F_X, F_Y, and F_Z for the x-, y- and z-direction respectively. The same yields for the moments where M_X, M_Y, and M_Z describe the moments in the x-, y- and z-direction respectively.

2.1.2 Frames

There are three main frames used in this report. These are the inertial frame, the base link frame, and the Serret-Frenet (sf) frame, all shown in Figure 2.1.

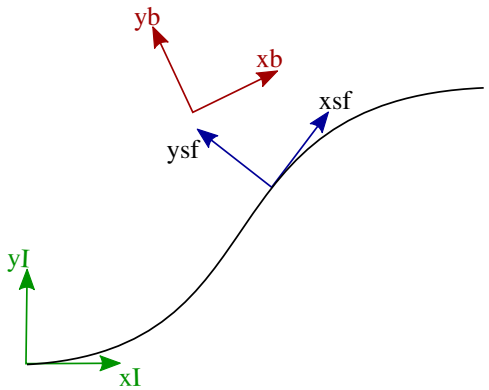


Figure 2.1: The inertial frame in green, the base-link frame in red and the sf frame in blue. The black line represents the centerline of the path.

The inertial frame, also called map frame, origins where the vehicle stands when the autonomous system boots. The x-component points towards the front of the vehicle, the y-component points towards the left of the vehicle, and the z-component points upwards. In Figure 2.1 the x- and y component of the inertial frame are visualized by the green arrows. This is a static frame, meaning it will keep the same position throughout the whole race and works as a reference frame.

The base-link frame, or body frame, is defined in the same way as the inertial frame. However, where the inertial frame is static, the base-link frame is centered in the Center of Gravity (CG) of the vehicle at all times. The base link frame is represented by the red x- and y-axis in Figure 2.1.

To connect the body frame to the inertial frame the odometry frame is applied, which is based on calculations from the state estimation module. The relationship between the three frames, inertial frame, body frame, and odometry frame, is

$$\mathbf{p}_b^I = \mathbf{R}_o^I \mathbf{p}_b^o + \mathbf{p}_o^I, \tag{2.1}$$

where $\mathbf{p}$ is a position vector and $\mathbf{R}$ is a rotation matrix. The subscriptions I, o and b represent the inertial, odometry and body frame, respectively.

The sf frame is used as a reference frame following the desired path [4]. The sf frame is used due to its property, where the frame components are parameterized using the given curve the frame is following. This property is preferred when following a curved path, and the curvature of the path can easily be extracted from the parameterized components. The x-component of the frame is always tangential to the curvature of the path and the y-component is pointing to the left, perpendicular to the x-component of the frame. The blue x- and y components in Figure 2.1 represents the sf frame.

2.2 Lateral Vehicle Modeling

To describe the lateral vehicle behavior, the bicycle model is applied for both the kinematic and the dynamic modeling [5]. The bicycle model is a simplification of the vehicle model where the two axles are represented as one wheel, as seen in Figure 2.2. This model is widely used and often in the automotive industry.

The conference paper, [6], represents a comparison between the bicycle model and the 9 degrees of freedom vehicle model in a model predictive control approach for motion planning. It concludes that the bicycle model is consistent for this purpose, as long as the lateral acceleration is constrained.

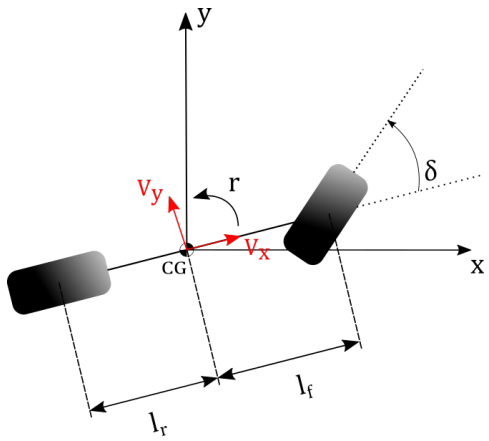


Figure 2.2: Bicycle Model with steering angle δ and the track width $L = l_r + l_f$.

The resulting model simplifies the kinematics and dynamics of the vehicle, compared to a higher degree of freedom model. It is assumed only front-wheel steering and that the CG is placed at the origin of the coordinate frame in Figure 2.2, i.e. base-link frame.

2.2.1 Lateral Kinematics of Bicycle Model

The lateral kinematic of the vehicle describes the vehicle motion from a geometric perspective, thus the influence of forces and torques are neglected. This is considered a valid assumption for low velocities, where low velocity is defined as $0m/s \leq u \leq 5m/s$. When assuming low speed, the no-slip assumption is also valid and simplifies the mathematical representation. As the vehicle is running on a flat track, the linear motions in the z-direction are neglected. The stated assumptions result in only x, y, and ψ as the defining states for the vehicle motions. x and y define the position of the vehicle, while ψ defines the rotation of the vehicle around the z-axis, located in CG, all with respect to the inertial frame.

For modeling the vehicle motions it is assumed that the vehicle is in a circular motion. In this case, driving a straight line is equivalent to cornering with an infinite radius, $R = \infty$.

To model the kinematics of the vehicle, a triangle can be drawn between the center of the circle that the vehicle is following, the CG, and the front wheel. Additionally, a triangle between the center of the circle, CG, and the rear wheel can be drawn, as visualized in Figure 2.3. By deploying the sine rule, the relationship between the circular path and the vehicle can be deducted.

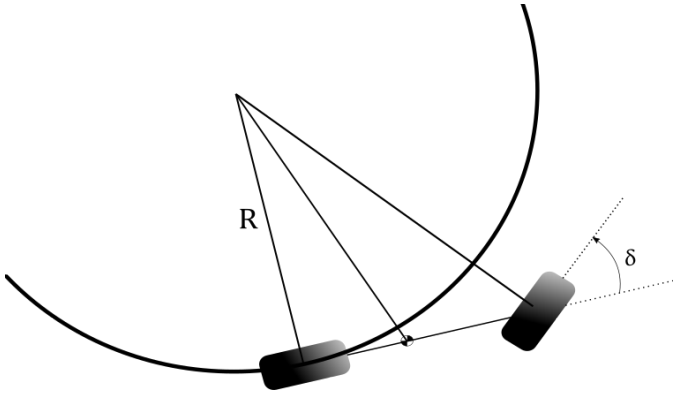


Figure 2.3: Bicycle Model driving a constant radius circle.

For the front triangle, the sine rule results in

$$\frac{sin(\delta - \beta)}{l_f} = \frac{sin(\frac{\pi}{2} - \delta)}{R}, \tag{2.2}$$

where β is the body slip angle of the vehicle. The body slip angle is the difference in the direction of the velocity vector and the direction the vehicle is heading.

The sine term in Equation 2.2 can be extended using the algebraic method and the resulting expression is

$$tan(\delta)cos(\beta) - sin(\delta) = \frac{l_f}{R}. \tag{2.3}$$

Similarly, for the rear triangle, the sine rule results in

$$\frac{sin(\beta)}{l_r} = \frac{sin(\frac{\pi}{2})}{R} = \frac{1}{R} \Rightarrow sin(\beta) = \frac{l_r}{R}. \tag{2.4}$$

Thus, by inserting Equation 2.4 into Equation 2.3 the relationship between the vehicle and the circular path can be described by

$$tan(\delta)cos(\beta) = \frac{l_f + l_r}{R}. \tag{2.5}$$

When using the assumption of low velocity, the assumption $\dot{\psi} = r = \frac{V}{R}$, is also considered valid. By combining Equation 2.5 with this assumption the change in ψ can be described as

$$\dot{\psi} = r = \frac{V cos(\beta)}{l_f + l_r} tan(\delta). \tag{2.6}$$

The changes in x- and y-direction are then defined by

$$\begin{aligned} \dot{x} &= V cos(\psi + \beta), \\ \dot{y} &= V sin(\psi + \beta). \end{aligned} \tag{2.7}$$

2.2.2 Lateral Dynamics of Bicycle Modeling

When a vehicle drives at higher velocities the dynamics of the vehicle must be included in the modeling of the behavior, due to that the assumption about no-slip stated in the previous section does not hold anymore. Thus, the forces and torques must be included in the description of the change in the vehicle states.

To define the lateral forces acting on the vehicle, Newton's second law of motion is applied.

$$m a_y = F_{Y,f} + F_{Y,r}, \tag{2.8}$$

where a_y is the acceleration of the vehicle at CG with respect to inertial frame, and $F_{Y,f}$ and $F_{Y,r}$ are the forces acting on the front and rear tires, respectively.

The acceleration a_y is a combination of the acceleration along the y-axis and the centripetal acceleration, thus the acceleration can be stated as

$$a_y = \ddot{y} + ur. \tag{2.9}$$

When combining Equation 2.8 and Equation 2.9, Newton's second law of motion is described by

$$m(\ddot{y} + ur) = F_{Y,f} + F_{Y,r}. \tag{2.10}$$

Similar to Newton's second law of motion, the moment balance about the z-axis placed in CG is defined as

$$I_Z \dot{r} = l_f F_{Y,f} - l_r F_{Y,r} \tag{2.11}$$

where the tire forces, $F_{Y,f}$ and $F_{Y,r}$, must be estimated, and I_Z is the yawing moment of inertia.

In summary, the lateral dynamics of the bicycle model can be described by

$$\begin{aligned} \ddot{y} &= \frac{1}{m}(F_{Y,f} + F_{Y,r}) - ur \\ \dot{r} &= \frac{l_f F_{Y,f} - l_r F_{Y,r}}{I_Z}. \end{aligned} \tag{2.12}$$

2.2.3 Tire Modeling

Tires are the only elements of the vehicle in contact with the ground, and consequently are the only source of generating acceleration. The lateral force acts on the center of the contact patch in the horizontal plane. This force acts, at all times, perpendicular to the heading of the tire, assuming zero camber and inclination angle [7], following the coordinate system in Figure 2.4.

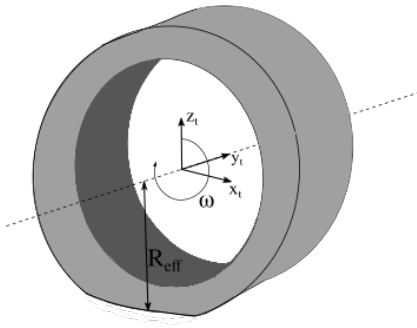


Figure 2.4: Illustration of a tire, including tire frame, angular velocity, ω, and the effective radius, R_{eff}.

As a result of the tire movement relative to the ground, called slip, the tire experiences a deformation due to the friction forces at the contact patch. The natural behavior of a tire is to act against this deformation and it tries to regain its original shape, which creates the lateral force. The lateral slip angle can be described as the angle between the velocity vector and the geometric direction of the respective tire. The slip angle is mathematically defined as

$$\begin{aligned} \alpha_f &= \delta - arctan(\frac{\omega_f l_f + v}{u}) \\ \alpha_r &= arctan(\frac{\omega_r l_r - v}{u}) \end{aligned} \tag{2.13}$$

where $\omega_{f,r}$ is the angular velocity, and $l_{f,r}$ is the distance from CG to the front and rear tire, respectively. The slip angle is illustrated in Figure 2.5.

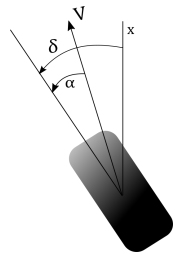


Figure 2.5: Illustration of the slip angle, α, where δ is steering angle, V is the velocity vector and x is the longitudinal axis.

To model the lateral forces, the empirical model *Pacejka magic formula* is used [8]. The Pacejka magic formula is a widely used model for estimating tire forces in the automotive industry. The simplification of constant coefficients [9] is applied, where it is assumed zero camber angle and constant load as pitching motions are neglected. The resulting lateral force for a tire is then

$$F_{Y,i} = -F_{Z,i} \cdot D_{Y,i} sin(C_{Y,i} \cdot arctan[Bv\alpha_i - E_{Y,i}(B_{Y,i}\alpha_i - arctan(B_{Y,i}))]), \quad (2.14)$$

where i indicates the front or rear wheels. The parameters D, C, B and E represents the peak, shape, stiffness and curvature, respectively, and are estimated by fitting the curve defined by Equation 2.14 to empirical data giving the characteristic graph given in Figure 2.6.

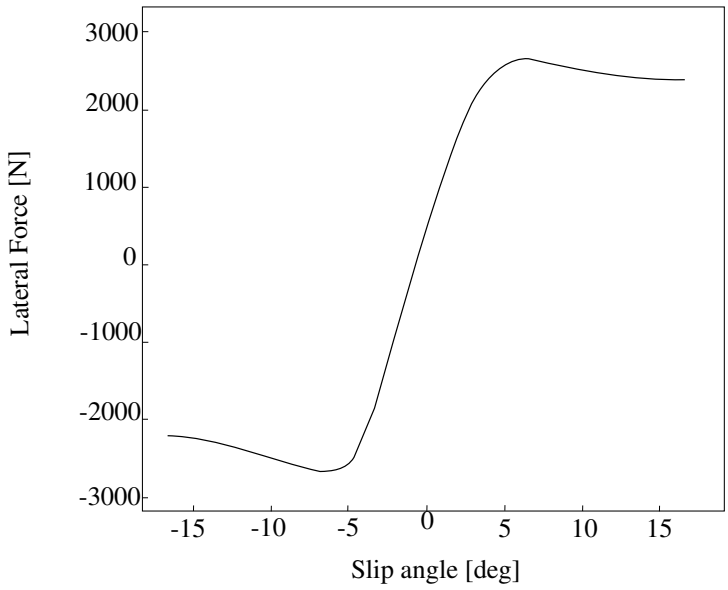


Figure 2.6: Characteristic graph of the lateral force VS slip angle.

The final lateral dynamics of the vehicle, when substituting Equation 2.14 into Equation 2.12, is

$$\begin{aligned}\ddot{y} = -\frac{1}{m}\Bigg(& F_Z^f \cdot D_Y^f sin\Big(C_Y^f \cdot arctan[B_Y^f \alpha_f - E_Y^f (B_Y^f \alpha_f - arctan(B_Y^f))]\Big) \\ & + F_Z^r \cdot D_Y^r sin\Big(C_Y^r \cdot arctan[B_Y^r \alpha_r - E_Y^r (B_Y^r \alpha_r - arctan(B_Y^r))]\Big)\Bigg) \\ & - ur\end{aligned} \tag{2.15}$$

$$\begin{aligned}\dot{r} = & -\frac{l_f F_Z^f \cdot D_Y^f sin\Big(C_Y^f \cdot arctan[B_Y^f \alpha_f - E_Y^f (B_Y^f \alpha_f - arctan(B_Y^f))]\Big)}{I_Z} \\ & + \frac{l_r F_Z^r \cdot D_Y^r sin\Big(C_Y^r \cdot arctan[B_Y^r \alpha_r - E_Y^r (B_Y^r \alpha_r - arctan(B_Y^r))]\Big)\Big)}{I_Z}.\end{aligned} \tag{2.16}$$

2.2.4 Linear Tire Modeling

The aforementioned tire forces are highly nonlinear, which may be too complex for a lateral controller. Therefore, a linear approximation of the lateral forces can be deducted [10]. For small slip angles, the lateral force can be linearized, which results in

$$\begin{aligned}F_{Y,f} &= B_{cs,f}\alpha_f, \\ F_{Y,r} &= B_{cs,r}\alpha_r.\end{aligned} \tag{2.17}$$

The variables $B_{cs,f}$ and $B_{cs,r}$ are the cornering stiffness of the front and rear tire, respectively. The cornering stiffness is a description of the tires ability to resist deformation while cornering.

Due to the small slip angle assumption in Equation 2.17, the slip angle can also be linearized around small angles. Consequently, the slip angles can be approximated as

$$\begin{aligned}\alpha_f &= \delta - arctan(\frac{\omega_f l_f + v}{u}) \approx \delta - \frac{\omega_f l_f + v}{u}, \\ \alpha_r &= arctan(\frac{\omega_r l_r + v}{u}) \approx \frac{\omega_r l_r + v}{u}.\end{aligned} \tag{2.18}$$

Substituting the linear formula for slip angle, Equation 2.18, into the linear relation of lateral force, Equation 2.17, the linear relation results in

$$\begin{aligned}F_{Y,f} &= B_{cs,f}(\delta - \frac{\omega_f l_f + v}{u}), \\ F_{Y,r} &= B_{cs,r}(\frac{\omega_r l_r + v}{u}).\end{aligned} \tag{2.19}$$

The lateral dynamics of a vehicle with a linear tire model can then be described by

$$\begin{aligned} \ddot{y} &= \frac{1}{m}((B_{cs,f}(\delta - \frac{\omega_f l_f + v}{u})) + (B_{cs,r}(\frac{\omega_r l_r + v}{u}))) - ur, \\ \dot{r} &= \frac{1}{I_Z}(l_f(B_{cs,f}(\delta - \frac{\omega_f l_f + v}{u})) - l_r(B_{cs,r}(\frac{\omega_r l_r + v}{u}))). \end{aligned} \tag{2.20}$$

2.3 Earlier Approach

The approach used before this thesis started was a cascade of three controllers, a Line of Sight (LOS) controller, a feedback linearization controller, and finally a Proportional–Integral–Derivative (PID) controller, including an advanced path and timing law. For this report, the earlier approach is used for comparison purposes and a basis for further development.

2.3.1 Path Representation

The path $\pi(\cdot)$, defined by the centerline of the track, is represented as a parameterized path in the planar vector space, where it is assumed approximately arc length parameterization, meaning

$$\pi(\varpi_i) - \pi(\varpi_{i-1}) \approx \varpi_i - \varpi_{i-1}, \tag{2.21}$$

where ϖ is a continuous path variable. Additionally, the path follows the geometric boundary conditions stated in the Formula Student rules, [1]. This implies a maximum track width of 3 meters, in addition to a maximum curvature of $0.22m^{-1}$. Further, the path is assumed second derivative continuous and regular, which allows calculating velocities and accelerations of the sf frame, described in Section 2.1. A path being regular implies that

$$|\pi'(\varpi)| = \frac{d\pi(\varpi)}{d\varpi} \neq 0. \tag{2.22}$$

Projection

At all times, it is important to keep track of where the vehicle is with respect to the path. Both the odometry and the centerline are given in the world frame, but the position of where they are in the world with respect to each other is not. To resolve this, a minimization problem can be defined.

$$\theta_{\mathcal{P}}(\pi) \text{ min } \{||\pi^d(\theta) - p||_2 : \theta \in [0, L]\}. \tag{2.23}$$

In Equation 2.23, $\theta_{\mathcal{P}}$ is the projection operator, where $\mathcal{P}$ is the set of all legal points, p, along the track with a length L. Further, the 2-norm, $||\cdot||_2$, is defined as

$$||x||_2 = \sqrt{\sum_{k=1}^{n} |x_k|^2}. \tag{2.24}$$

This finds the shortest distance from the vehicle to the path, perpendicular to the curvature of the path.

Timing Law

Both to keep track of the progression along the path and be able to use a lookahead distance, a timing law is crucial. Two different timing laws are used, one for fully detected and known tracks and one for unknown tracks [11].

A simple projective guidance law is implemented for the unknown tracks, where the pose of the vehicle is projected onto the track with a given lookahead distance, see Figure 2.7.

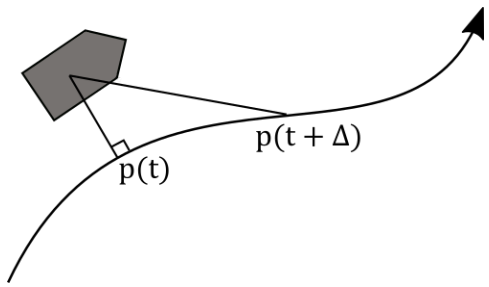


Figure 2.7: Projective guidance law.

For the fully detected and known track, a kinematic path following timing law for tracking the reference frame, the sf frame, attached to the path is implemented. Assigning a virtual velocity to the sf frame leads to a more predictive timing law, which results in better performance, as the lookahead distance is dynamic. The change in distance traveled is defined as

$$\dot{s} = \frac{v_f}{\sqrt{\Delta^2 + y_e^2 + x_e^2}}(\sqrt{\Delta^2 + x_e^2} - x_e) \tag{2.25}$$

The kinematic path following, and the variables in Equation 2.25, are visualized in Figure 2.8.

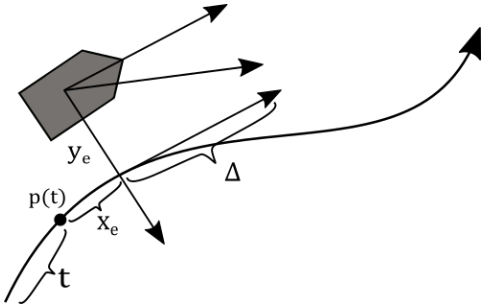


Figure 2.8: Kinematic path following.

2.3.2 Lateral Controller

The existing control system for the lateral movement of the race car is based on the path-following kinematic controller in [4]. This controller utilizes the bicycle model, presented and illustrated in Section 2.2. The complete control strategy was developed by a previous member of Revolve NTNU and is described in [12], excluding the aforementioned kinematic timing law. Even though the control design is presented in [12], the controller will be described in this Section for the reader's holistic understanding.

The controller for lateral motion is a cascade of three controllers,

- a LOS guidance controller,
- a feedback linearization controller, and
- a PID controller.

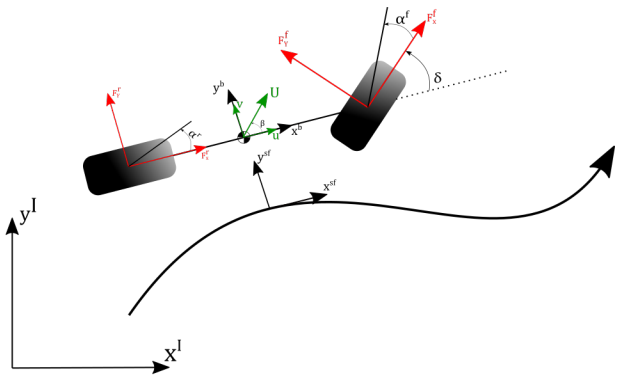


Figure 2.9: Bicycle Model with frames

The LOS guidance controller finds the desired heading for approaching the path with a given lookahead distance. The path is represented as a spline, with the respective target velocity as a function of the curvature. The sf frame, at the path, seen in Figure 2.9, is

used as a virtual target for the vehicle. When the desired course is known, a feedback linearization controller controls the vehicle for keeping the desired course. By utilizing the yaw dynamic of the bicycle model, the desired steering angle can be calculated for maintaining the desired yaw rate from the feedback linearization controller. Then a PID controller draws the control error to zero.

The control law of the LOS guidance controller is defined as

$$\chi_r^{sf} = atan(\frac{-y^{sf}}{\Delta}) \tag{2.26}$$

where χ is the heading of the vehicle and Δ is the lookahead angle. The lookahead angle refers to how far in front of the vehicle it is aiming, to reach the centerline when assuming a straight path.

The feedback linearization controller aims to keep the desired course angle from Equation 2.26. Therefore, it is necessary to find the yaw dynamics described in the sf frame. This is given by

$$\dot{\chi}^{sf} = r + \dot{\beta} - \kappa u^{sf} \tag{2.27}$$

By rearranging Equation 2.27, the yaw rate is described by

$$r = \dot{\chi}^{sf} - \dot{\beta} + \kappa u^{sf}, \tag{2.28}$$

where

$$\dot{\beta} = \frac{d}{dt}\left(arctan\left(\frac{v}{u}\right)\right) \tag{2.29}$$

and κ is the curvature of the path.

The feedback linearization controller results in the yaw rate reference, defined as

$$r_r = \dot{\chi}_r^{sf} - \dot{\beta} + \kappa u^{sf} + K_{P,r}\tilde{\chi}^{sf}. \tag{2.30}$$

where

$$\tilde{\chi}^{sf} = \chi^{sf} - \chi_r^{sf}, \tag{2.31}$$

The yaw dynamic for the bicycle model is given by

$$\dot{r} = \frac{1}{I_Z}(-B_{cs,f}\alpha_f l_f + B_{cs,r}\alpha_r l_r) \tag{2.32}$$

where $B_{cs,f}$ and $B_{cs,r}$ are the cornering stiffness for the respective tires. The tire model is assumed to be linear as described in Section 2.2. Thus, by substituting the equation for the slip angle, Equation 2.13, into Equation 2.32 and solving for the steering angle, δ, it results in

$$\delta = \frac{I_Z \dot{r}}{B_{cs,f} l_f} - arctan(\frac{\omega l_f + v}{u}) + \frac{l_r}{l_f} arctan(\frac{\omega l_r - v}{u}) \tag{2.33}$$

Hence, the steering angle reference, including a proportional controller, is

$$\delta_{ref} = \frac{I_Z \dot{r}}{B_{cs,f} l_f} - arctan(\frac{\omega l_f + v}{u}) + \frac{l_r}{l_f} arctan(\frac{\omega l_r - v}{u}) + K_{P,\delta} \tilde{r} \tag{2.34}$$

where $\dot{r}_d$ refers to the derivative of Equation 2.30 and $\tilde{r} = r - r_r$.

Chapter 3

Theory

The various spectra of control regimes have previously been used for controlling the lateral motion of autonomous vehicles. One of the advanced methods is the optimal control regime Model Predictive Control (MPC), a method for finding the optimal control input based on the prediction of the vehicle motions. A thorough introduction of the MPC method and the intended use is described in Section 3.1. For this thesis, the linear version of the problem formulation of the MPC is employed. Therefore, the linearization scheme is described in Section 3.2. The MPC is planned to run on a digital computer, which requires a discrete formulation of the MPC. Discretization methods are therefore described in Section 3.3.

3.1 Model Predictive Control

MPC, also called receding horizon predictive control, is a widely used control method for several purposes and in different fields of studies [13]. MPC was introduced in the 1960s and was quickly implemented in the industry. The MPC method is highly computational, and therefore, the method did not get the desired attention. Due to the increase in the processing power of computers, the interest in MPC has grown during the last decade.

3.1.1 Motivation for using Model Predictive Control

For a race car driver, it is important to act as a result of the path seen in front of the driver. It is equally important to know the behavior of the vehicle in every situation. If a race car driver only utilizes the current state of the vehicle and neglects future parts of the track, it is impossible to compete against world-class drivers. This theory also yields for autonomous race cars. Taking actions based on predicted behavior is crucial when the goal is to drive at high velocities. This is known as an active control regime, which is the opportunity the MPC method provides.

Using the MPC method for tracking a reference path is not a new technology. In the literature several approaches using the MPC method are already presented, e.g. [14], [15], [16], and [17] all utilizing the method differently. Both the approach in [15] and the approach in [16] are designing an MPC to use the shortest amount of time finishing the given track. Even though they are aiming for the same thing, [15] includes machine learning in the formulation, which [16] is not using. The approach in [14] proposes a method using the kinematic vehicle model for path tracking, while the approach in [17] is using a dynamic vehicle model for the MPC implementation. The MPC approach can be implemented in many ways, and utilize a wide specter of technology, which makes the approach interesting.

One of the benefits of the MPC method is that it is a modular control regime, allowing one to change the object of the controller, without changing the entire implementation. This makes it easy to increase the complexity of the controller stepwise, depending on both the accuracy and complexity needed. The different levels of complexity can also be utilized for the different dynamic events of the Formula Student competition, described in Section 1.1, where different events require a different level of accuracy and robustness.

3.1.2 General Formulation of Model Predictive Control

The method, MPC, seeks to find the optimal control input based on predictions of future behavior of the controlled system, and consists of three main elements:

- Prediction Model,
- Objective Function,
- Constraints.

These elements are designed as an optimization problem for fulfilling the three steps of the control method.

1. The prediction model describes the behavior of the controlled system and is used for predicting future outputs, based on the current system state. The prediction is calculated for every time step in a given and finite prediction horizon denoted N.
2. The objective function is designed for optimizing the system control input while keeping the given reference for the given prediction horizon. Often systems are limited, due to physical limits or desired limits, then both the states and the control inputs may be limited. The limits are formulated as constraints in the optimization problem.
3. The resulting optimization problem will optimize the control input over the given horizon, where only the first control input is fed into the controlled system.

Mathematically, the MPC problem can be summarized as the optimization problem represented in Equation 3.1 [18]. The objective function, $f(x)$, is minimized subject to the constraints, where $\mathcal{E}$ represents the set of equality constraints and $\mathcal{I}$ represents the set of inequality constraints.

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_x}} \quad & f(x) \\ \text{s. t.} \quad & c_i(x) = 0 \quad i \in \mathcal{E}, \\ & c_i(x) \geq 0 \quad i \in \mathcal{I}. \end{aligned} \tag{3.1}$$

Objective Function

The objective function in an MPC is often formulated as a quadratic cost function, resulting in a Quadratic Programming (QP) problem. The general formulation is

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{G}\mathbf{x} + \mathbf{x}^T\mathbf{c}, \tag{3.2}$$

where $G \in \mathbb{R}^{n_x \times n_x}$ is the Hessian matrix and $c \in \mathbb{R}^{n_x}$ is the gradient [18]. A QP problem can always be solved or shown to be infeasible within a finite number of iterations. If the Hessian matrix of the quadratic cost function is a positive semidefinite Hessian matrix, the function is convex. A convex function implies that a local minimum is a global optimum.

The QP formulation allows for Multiple-Input-Multiple-Output (MIMO) system, meaning the systems have several inputs and outputs [19]. When formulating the MPC problem as a QP problem, the resulting control inputs over the horizon are treated as individual outputs. Additionally, several industrial optimization solvers utilize the structure of a QP problem. Therefore, the QP structure is preferable for an MPC formulation.

Prediction Model

To predict the future system outputs, a model of the system is necessary. The prediction model is a crucial part of the MPC problem and a significant contributor to the computational load of the MPC. Therefore, it is important to choose a model that is both complex enough to catch the important behavior and dynamics of the system, and at the same time simple enough for solving the resulting optimization problem in real-time. The model can be both linear and nonlinear. In this thesis, the linear model is considered as the prediction model, due to the reduction of complexity and the decrease in computational load. The linear state-space formulation of the prediction model used in the MPC formulation is

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k + \mathbf{E}_k\mathbf{d}_k, \\ \mathbf{y}_k &= \mathbf{C}_k\mathbf{x}_k + \mathbf{D}_k\mathbf{u}_k. \end{aligned} \tag{3.3}$$

In the states-space formulation, Equation 3.3, the subscript indicates the discrete time step index. Allowing the system to change state matrices for every time step in the MPC, is often a useful tool. State-independent and known model parameters may change over the prediction horizon, N, and by letting these parameters change for every time step over the horizon, it will lead to a more accurate solution. Changing the state-independent and known model parameters, results in a Linear Parameter-Variant (LPV) system. The system states in the state-space formulation are $\mathbf{x}_k \in \mathbb{R}^{n_x}$, the system control inputs are

$\mathbf{u}_k \in \mathbb{R}^{n_u}$, the known system disturbances are $\mathbf{d}_k \in \mathbb{R}^{n_d}$ and $\mathbf{y}_k \in \mathbb{R}^{n_y}$ are the output variables of the system. The associating system matrices are the discrete system matrices of the following sizes

- $\mathbf{A}_k \in \mathbb{R}^{n_x \times n_x}$,
- $\mathbf{B}_k \in \mathbb{R}^{n_x \times n_u}$,
- $\mathbf{E}_k \in \mathbb{R}^{n_x \times n_d}$,
- $\mathbf{C}_k \in \mathbb{R}^{n_y \times n_x}$,
- $\mathbf{D}_k \in \mathbb{R}^{n_y \times n_u}$.

Constraints

In real-world systems, there are often limitations based on physical laws or desirable behavior. These limitations can be embedded into the MPC formulation as constraints. Both the system states and the control input are often bounded by a defined interval which makes them a perfect candidate for inequality constraints.

$$\begin{aligned} \mathbf{x}^{low} \leq \mathbf{x} \leq \mathbf{x}^{high} \\ \mathbf{u}^{low} \leq \mathbf{u} \leq \mathbf{u}^{high} \end{aligned} \tag{3.4}$$

In Equation 3.4 the system states, $\mathbf{x}$, are limited by a lower value, $\mathbf{x}^{low}$, and an upper value, $\mathbf{x}^{high}$, restricting the states from entering an illegal area. The same applies to the control input $\mathbf{u}$.

Often it is desirable to limit the change in control input to avoid rapid movements that can damage the controlled system. The inequalities are then

$$\boldsymbol{\Delta}\mathbf{u}^{low} \leq \boldsymbol{\Delta}\mathbf{u} \leq \boldsymbol{\Delta}\mathbf{u}^{high}, \tag{3.5}$$

where $\boldsymbol{\Delta}\mathbf{u}$ is the change in control input, defined as $\boldsymbol{\Delta}\mathbf{u} = \mathbf{u}_k - \mathbf{u}_{k-1}$.

3.1.3 Model Predictive Control for Trajectory Tracking

MPC is well suited for trajectory tracking as the method can find the optimal control input for minimizing the error between the trajectory of the system and the desired trajectory. The aforementioned formulation of the MPC problem is also applicable for the trajectory tracking problem. The objective function and the constraints for the trajectory tracking problem will be described more thoroughly in the following sections.

Objective Function

The MPC method attempts to find the optimal control inputs,

$$\bar{\mathbf{u}} = [\mathbf{u}_k^T, \mathbf{u}_{k+1}^T, \ldots, \mathbf{u}_{k+N-1}^T] \in \mathbb{R}^{n_u \times N}, \tag{3.6}$$

for the given horizon N, minimizing the error evolution

$$\bar{\mathbf{e}} = [\mathbf{e}_{k+1}^T, \mathbf{e}_{k+2}^T, \ldots, \mathbf{e}_{k+N}^T] \in \mathbb{R}^{n_x \times N}. \tag{3.7}$$

The error, $\mathbf{e}$, is defined as the difference between the system states and the trajectory references, denoted $\mathbf{e}_{k+1}^T = \mathbf{x}_{k+1}^T - \mathbf{r}_{k+1}^T$. The resulting objective function can be formulated as

$$f(\mathbf{e}, \mathbf{u}) = \frac{1}{2} \sum_{k=0}^{N} \mathbf{e}(\mathbf{x}, \mathbf{r})_{k+1}^T \mathbf{Q}_{k+1} \mathbf{e}(\mathbf{x}, \mathbf{r})_{k+1} + \mathbf{u}_k^T \mathbf{R}_k \mathbf{u}_k. \tag{3.8}$$

The matrices $\mathbf{Q}_{k+1} \in \mathbb{R}^{n_y \times n_y}$ and $\mathbf{R}_k \in \mathbb{R}^{n_u \times n_u}$ are weighting matrices. $\mathbf{Q}_{k+1} \succeq 0$, is a positive semi-definite matrix weighting the importance of a small state error, $\mathbf{e}_{k+1}^T \to 0$, and $\mathbf{R}_k \succ 0$ is a positive definite matrix weighting the usage of the control input [20].

Equation 3.8 can be reformulated to match the formulation in Equation 3.2 when utilizing the vectors in Equation 3.6 and Equation 3.7,

$$f(\mathbf{z}) = \frac{1}{2} \bar{\mathbf{e}}^T \bar{\mathbf{Q}} \bar{\mathbf{e}} + \bar{\mathbf{u}}^T \bar{\mathbf{R}} \bar{\mathbf{u}} = \frac{1}{2} \mathbf{z}^T \begin{bmatrix} \bar{\mathbf{Q}} & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{R}} \end{bmatrix} \mathbf{z} = \frac{1}{2} \mathbf{z}^T \bar{\mathbf{G}} \mathbf{z}. \tag{3.9}$$

The optimization variables in Equation 3.9 are now both the control input and the state error, denoted $\mathbf{z} = [\mathbf{e}_{k+1}^T, \mathbf{e}_{k+2}^T, \ldots, \mathbf{e}_{k+N}^T, \mathbf{u}_k^T, \mathbf{u}_{k+1}^T, \ldots, \mathbf{u}_{k+N-1}^T] = [\bar{\mathbf{e}}^T \quad \bar{\mathbf{u}}^T]$. The new weighting matrices, $\bar{\mathbf{Q}}$ and $\bar{\mathbf{R}}$, are the state and control input weighting over the whole horizon, defined as

$$\bar{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q}_{k+1} & \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{k+2} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{Q}_{k+3} & & \mathbf{0} & \mathbf{0} \\ & \vdots & & \ddots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{Q}_{k+N-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{Q}_{k+N} \end{bmatrix}, \tag{3.10}$$

and

$$\bar{\mathbf{R}} = \begin{bmatrix} \mathbf{R}_k & \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{k+1} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_{k+2} & & \mathbf{0} & \mathbf{0} \\ & \vdots & & \ddots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{R}_{k+N-2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{R}_{k+N-1} \end{bmatrix}. \tag{3.11}$$

This formulation of the weighting matrices, $\bar{\mathbf{Q}}$ and $\bar{\mathbf{R}}$, allows different weighting at each time step over the horizon.

Defining Constraints

The prediction model and the state and control input limitations have to be embedded into the QP formulation as constraints. There are mainly two approaches for embedding the constraints and prediction model into the formulation when aiming for a numeric solution,

- a sequential approach, and
- a simultaneous approach [21].

For the sequential approach, both the system states and the control inputs are defined as the optimization variables, whereas in the simultaneous approach, only the control input is treated as the optimization variables.

Simultaneous Formulation

The constraints of the simultaneous formulation are derived from the prediction model, defined by Equation 3.3. The evolution of the states can be stated as

$$\begin{aligned}
\mathbf{x}_{k+1} &= \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k + \mathbf{E}_k\mathbf{d}_k \\
\mathbf{x}_{k+2} &= \mathbf{A}_{k+1}\mathbf{x}_{k+1} + \mathbf{B}_{k+1}\mathbf{u}_{k+1} + \mathbf{E}_{k+1}\mathbf{d}_{k+1} \\
&= \mathbf{A}_{k+1}(\mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k + \mathbf{E}_k\mathbf{d}_k) + \mathbf{B}_{k+1}\mathbf{u}_{k+1} + \mathbf{E}_{k+1}\mathbf{d}_{k+1} \\
&= \mathbf{A}_{k+1}\mathbf{A}_k\mathbf{x}_k + \mathbf{A}_{k+1}\mathbf{B}_k\mathbf{u}_k + \mathbf{A}_{k+1}\mathbf{E}_k\mathbf{d}_k + \mathbf{B}_{k+1}\mathbf{u}_{k+1} + \mathbf{E}_{k+1}\mathbf{d}_{k+1} \\
\mathbf{x}_{k+3} &= \mathbf{A}_{k+2}\mathbf{x}_{k+2} + \mathbf{B}_{k+2}\mathbf{u}_{k+2} + \mathbf{E}_{k+2}\mathbf{d}_{k+2} \\
&= \mathbf{A}_{k+2}(\mathbf{A}_{k+1}\mathbf{A}_k\mathbf{x}_k + \mathbf{A}_{k+1}\mathbf{B}_k\mathbf{u}_k + \mathbf{A}_{k+1}\mathbf{E}_k\mathbf{d}_k + \mathbf{B}_{k+1}\mathbf{u}_{k+1} \\
&\quad + \mathbf{E}_{k+1}\mathbf{d}_{k+1}) + \mathbf{B}_{k+2}\mathbf{u}_{k+2} + \mathbf{E}_{k+2}\mathbf{d}_{k+2} \\
&\vdots
\end{aligned} \tag{3.12}$$

By continuing this evolution for the entire horizon N, it can be condensed, using the states and control inputs for the horizon, formulation by

$$\bar{\mathbf{A}}\bar{\mathbf{x}} + \bar{\mathbf{B}}\bar{\mathbf{u}} = \bar{\mathbf{E}}, \tag{3.13}$$

where the state matrices are defined as

$$\bar{\mathbf{A}} = \begin{bmatrix} -\mathbf{I} & \mathbf{0} & \mathbf{0} & & \mathbf{0} \\ \mathbf{A}_{k+1} & -\mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{k+2} & -\mathbf{I} & & \vdots \\ & \vdots & & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{A}_{k+N-1} & -\mathbf{I} \end{bmatrix}, \tag{3.14}$$

$$\bar{\mathbf{B}} = \begin{bmatrix} \mathbf{B}_k & \mathbf{0} & & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{k+1} & \cdots & \vdots \\ & \vdots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{B}_{k+N-1} \end{bmatrix} \text{ and } \bar{\mathbf{E}} = \begin{bmatrix} -\mathbf{A}_k\mathbf{x}_k - \mathbf{E}_k\mathbf{d}_k \\ -\mathbf{E}_{k+1}\mathbf{d}_{k+1} \\ -\mathbf{E}_{k+2}\mathbf{d}_{k+2} \\ \vdots \\ -\mathbf{E}_{k+N-1}\mathbf{d}_{k+N-1} \end{bmatrix}. \tag{3.15}$$

Equation 3.13, can be written even more compact by including the optimization variable, $\mathbf{z}$, in the formulation can then be stated as

$$\begin{bmatrix} \bar{\mathbf{A}} & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{B}} \end{bmatrix} \mathbf{z} = \begin{bmatrix} \bar{\mathbf{E}} \\ \mathbf{0} \end{bmatrix}. \tag{3.16}$$

Including the objective function in Equation 3.9, the resulting QP formulation is

$$\begin{aligned} \min_{\mathbf{z}\in\mathbb{R}^{(m+n)\times N}} \quad & \frac{1}{2}\mathbf{z}^T\bar{\mathbf{G}}\mathbf{z} \\ \text{s. t.} \quad & \begin{bmatrix} \bar{\mathbf{A}} & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{B}} \end{bmatrix} \mathbf{z} = \begin{bmatrix} \bar{\mathbf{E}} \\ \mathbf{0} \end{bmatrix}, \\ & \mathbf{z}_{lb} \leq \mathbf{z} \leq \mathbf{z}_{ub}. \end{aligned} \tag{3.17}$$

This formulation requires good estimates of the state evolution $\bar{\mathbf{x}}$, which may not be trivial.

Sequential Formulation

For the sequential formulation, both the objective function and the constraints containing the prediction model and limitations must be formulated using the control input. In this case, the evolution is described similarly as for the simultaneous formulation, however, the control input is the main variable, and it can be described as

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k + \mathbf{E}_k\mathbf{d}_k \\ \mathbf{x}_{k+2} &= \mathbf{A}_{k+1}\mathbf{x}_{k+1} + \mathbf{B}_{k+1}\mathbf{u}_{k+1} + \mathbf{E}_{k+1}\mathbf{d}_{k+1} \\ &= \mathbf{A}_{k+1}(\mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k + \mathbf{E}_k\mathbf{d}_k) + \mathbf{B}_{k+1}\mathbf{u}_{k+1} + \mathbf{E}_{k+1}\mathbf{d}_{k+1} \\ &= \mathbf{A}_{k+1}\mathbf{B}_k\mathbf{u}_k + \mathbf{B}_{k+1}\mathbf{u}_{k+1} + \mathbf{A}_{k+1}\mathbf{E}_k\mathbf{d}_k + \mathbf{E}_{k+1}\mathbf{d}_{k+1} + \mathbf{A}_{k+1}\mathbf{A}_k\mathbf{x}_k \\ \mathbf{x}_{k+3} &= \mathbf{A}_{k+2}\mathbf{x}_{k+2} + \mathbf{B}_{k+2}\mathbf{u}_{k+2} + \mathbf{E}_{k+2}\mathbf{d}_{k+2} \\ &= \mathbf{A}_{k+2}(\mathbf{A}_{k+1}\mathbf{B}_k\mathbf{u}_k + \mathbf{B}_{k+1}\mathbf{u}_{k+1} + \mathbf{A}_{k+1}\mathbf{E}_k\mathbf{d}_k + \mathbf{E}_{k+1}\mathbf{d}_{k+1} \\ &\quad + \mathbf{A}_{k+1}\mathbf{A}_k\mathbf{x}_k) + \mathbf{B}_{k+2}\mathbf{u}_{k+2} + \mathbf{E}_{k+2}\mathbf{d}_{k+2} \\ &= \mathbf{A}_{k+2}\mathbf{A}_{k+1}\mathbf{B}_k\mathbf{u}_k + \mathbf{B}_{k+2}\mathbf{u}_{k+2} + \mathbf{B}_{k+2}\mathbf{u}_{k+2} + \mathbf{A}_{k+2}\mathbf{A}_{k+1}\mathbf{E}_k\mathbf{d}_k \\ &\quad + \mathbf{A}_{k+2}\mathbf{E}_{k+1}\mathbf{d}_{k+1} + \mathbf{E}_{k+2}\mathbf{d}_{k+2} + \mathbf{A}_{k+2}\mathbf{A}_{k+1}\mathbf{A}_k\mathbf{x}_k. \\ &\vdots \end{aligned} \tag{3.18}$$

By continuing the evolution over the horizon N, the state prediction can be written using matrices as

$$\begin{aligned}\begin{bmatrix}\mathbf{x}_{k+1}\\ \mathbf{x}_{k+2}\\ \mathbf{x}_{k+3}\\ \vdots\end{bmatrix} = &\begin{bmatrix}\mathbf{B}_k & \mathbf{0} & \mathbf{0} & \cdots\\ \mathbf{A}_{k+1}\mathbf{B}_k & \mathbf{B}_{k+1} & \mathbf{0} & \\ \mathbf{A}_{k+2}\mathbf{A}_{k+1}\mathbf{B}_k & \mathbf{A}_{k+2}\mathbf{B}_{k+1} & \mathbf{B}_{k+2} & \\ & \vdots & & \ddots\end{bmatrix}\begin{bmatrix}\mathbf{u}_k\\ \mathbf{u}_{k+1}\\ \mathbf{u}_{k+2}\\ \vdots\end{bmatrix}\\ &+\begin{bmatrix}\mathbf{E}_k & \mathbf{0} & \mathbf{0} & \cdots\\ \mathbf{A}_{k+1}\mathbf{E}_k & \mathbf{E}_{k+1} & \mathbf{0} & \\ \mathbf{A}_{k+2}\mathbf{A}_{k+1}\mathbf{E}_k & \mathbf{A}_{k+2}\mathbf{E}_{k+1} & \mathbf{E}_{k+2} & \\ & \vdots & & \ddots\end{bmatrix}\begin{bmatrix}\mathbf{d}_k\\ \mathbf{d}_{k+1}\\ \mathbf{d}_{k+2}\\ \vdots\end{bmatrix}\\ &+\begin{bmatrix}\mathbf{A}_k\\ \mathbf{A}_{k+1}\mathbf{A}_k\\ \mathbf{A}_{k+2}\mathbf{A}_{k+1}\mathbf{A}_k\\ \vdots\end{bmatrix}\mathbf{x}_k,\end{aligned} \tag{3.19}$$

The condensed formulation of Equation 3.19 is

$$\hat{\mathbf{x}} = \hat{\mathbf{B}}\bar{\mathbf{u}} + \hat{\mathbf{E}}\bar{\mathbf{d}} + \hat{\mathbf{A}}\mathbf{x}_k. \tag{3.20}$$

If not all system states require minimization, the system output matrix $\mathbf{C}$ should be embedded into the objective function for reducing the problem size. The new weighting matrix, $\bar{\bar{\mathbf{Q}}}$, is defined as

$$\bar{\bar{\mathbf{Q}}} = \begin{bmatrix}\mathbf{C}_{k+1}^T\mathbf{Q}_{k+1}\mathbf{C}_{k+1} & \cdots & \mathbf{0}\\ \vdots & \ddots & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{C}_{k+N}^T\mathbf{Q}_{k+N}\mathbf{C}_{k+N}\end{bmatrix}. \tag{3.21}$$

Equation 3.20 can be substituted into the objective function, Equation 3.9, resulting in

$$\begin{aligned}f(\mathbf{u}) =&\frac{1}{2}((\hat{\mathbf{B}}\bar{\mathbf{u}} + \hat{\mathbf{E}}\bar{\mathbf{d}} + \hat{\mathbf{A}}\mathbf{x}_k)^T - \bar{\mathbf{r}}^T)\bar{\bar{\mathbf{Q}}}((\hat{\mathbf{B}}\bar{\mathbf{u}} + \hat{\mathbf{E}}\bar{\mathbf{d}} + \hat{\mathbf{A}}\mathbf{x}_k) - \bar{\mathbf{r}})\\ &+ \bar{\mathbf{u}}^T\bar{\mathbf{R}}\bar{\mathbf{u}},\end{aligned} \tag{3.22}$$

where only the control inputs for the horizon are the optimization variables. Rearranging Equation 3.22, the objective function can be written on the standard form as

$$\begin{aligned}f(\mathbf{u}) =&\frac{1}{2}\bar{\mathbf{u}}^T(\hat{\mathbf{B}}^T\bar{\bar{\mathbf{Q}}}\hat{\mathbf{B}} + \bar{\mathbf{R}})\bar{\mathbf{u}} + (\mathbf{x}_k^T\hat{\mathbf{A}}^T + \bar{\mathbf{d}}^T\hat{\mathbf{E}}^T - \bar{r}^T)\bar{\bar{\mathbf{Q}}}\hat{\mathbf{B}}\bar{\mathbf{u}}\\ &+ \frac{1}{2}(2\hat{\mathbf{A}}^T\mathbf{x}_k^T\bar{\bar{\mathbf{Q}}}\hat{\mathbf{E}}\bar{\mathbf{d}} - 2\hat{\mathbf{A}}^T\mathbf{x}_k^T\bar{\bar{\mathbf{Q}}}\bar{r} - 2\hat{\mathbf{E}}^T\mathbf{d}^T\bar{\bar{\mathbf{Q}}}\bar{r}\\ &+ \hat{\mathbf{A}}^T\mathbf{x}_k^T\bar{\bar{\mathbf{Q}}}\hat{\mathbf{A}}\mathbf{x}_k + \hat{\mathbf{E}}^T\mathbf{d}^T\bar{\bar{\mathbf{Q}}}\hat{\mathbf{E}}\mathbf{d} + \bar{r}^T\bar{\bar{\mathbf{Q}}}\bar{r})\end{aligned} \tag{3.23}$$

The constant terms of Equation 3.23 does not have any effect on the minimization value, it will only introduce an offset. Hence, the objective function can be written as

$$\begin{aligned} f(\mathbf{u}) &= \frac{1}{2}\bar{\mathbf{u}}^T(\hat{\mathbf{B}}^T\bar{\bar{\mathbf{Q}}}\hat{\mathbf{B}} + \bar{\mathbf{R}})\bar{\mathbf{u}} + (\mathbf{x}_k^T\hat{\mathbf{A}}^T + \bar{\mathbf{d}}^T\hat{\mathbf{E}}^T)\bar{\bar{\mathbf{Q}}}\hat{\mathbf{B}}\bar{\mathbf{u}} \\ &= \frac{1}{2}\bar{\mathbf{u}}^T\mathbf{G}\bar{\mathbf{u}} + \mathbf{c}^T\bar{\mathbf{u}}. \end{aligned} \tag{3.24}$$

The system prediction is embedded into the objective function. Therefore, the limitations of the states and the control inputs have to be included as constraints. However, the limitations need to be formulated using the prediction matrices, due to only the control inputs being the optimization variables. This results in constraints of a higher degree of complexity than for the simultaneous formulation. The inequality constraints for the states are

$$\bar{\mathbf{x}}_{lb} \leq \hat{\mathbf{x}} \leq \bar{\mathbf{x}}_{ub}. \tag{3.25}$$

When the prediction matrices are included, the inequality matrices are written as

$$\bar{\mathbf{x}}_{lb} - \hat{\mathbf{E}}\bar{\mathbf{d}} - \hat{\mathbf{A}}\mathbf{x}_k \leq \hat{\mathbf{B}}\bar{\mathbf{u}} \leq \bar{\mathbf{x}}_{ub} - \hat{\mathbf{E}}\bar{\mathbf{d}} - \hat{\mathbf{A}}\mathbf{x}_k. \tag{3.26}$$

The inequality constraints in Equation 3.26 can be written more generally as

$$\bar{\mathbf{x}}_{lb} \leq \hat{\mathbf{B}}\bar{\mathbf{u}} \leq \bar{\mathbf{x}}_{lb}. \tag{3.27}$$

The resulting minimization problem, using the sequential formulation, is then

$$\begin{aligned} \min_{\mathbf{u}\in\mathbb{R}^{n\times N}} \quad & \frac{1}{2}\bar{\mathbf{u}}^T\mathbf{G}\bar{\mathbf{u}} + \mathbf{c}^T\bar{\mathbf{u}} \\ \text{s. t.} \quad & \bar{\mathbf{b}}_{lb} \leq \hat{\mathbf{B}}\bar{\mathbf{u}} \leq \bar{\mathbf{b}}_{lb}, \\ & \bar{\mathbf{u}}_{lb} \leq \bar{\mathbf{u}} \leq \bar{\mathbf{u}}_{ub}. \end{aligned} \tag{3.28}$$

3.1.4 Solvers for Model Predictive Control

The optimization problem is a widely researched field and many state-of-the-art solvers are available. For MPC- and QP problems some of the available solvers are

- qpOASES [22],
- FORCES PRO [23],
- HPIPM [24], and
- osQP [25].

An important aspect of the industrial solvers is the terms and conditions the solvers are restricted by. Some solvers require a valid license for getting access to the solver, while others are open-source and restricted by free software licenses like GNU General License[1], Apache License, Version 2.0[2], or the 2-Clause BSD License[3].

Often, solvers utilize different aspects of the problem formulation for decreasing the run time, as this often is the main problem during real-time optimization.

qpOASES

qpOASES is an open-source online active set strategy [22] that solves QP problems. The solver is based on a study revealing that the active set of a QP problem does not change significantly from one problem to the next and therefore, the active set is a useful tool when solving optimization problems. The solver is written in C and wrappers for several Third-Party Software are available. Additionally, a detailed manual and a well commented code are available to ease the implementation. qpOASES is restricted by the GNU General License.

FORCES PRO

FORCES PRO is an optimization solver made for embedded computers [23]. The main area of usage is problems performed in succession, with changing inputs and is executed in real-time. The solver is based on an interior point method using the Newton step, and the solver automatically generates C-code for fast execution. Although FORCES PRO is a license based solver, meaning a costly license is needed for using the solver, all academic projects can request free licenses.

HPIPM

The High-Performance Interior-Point (HPIPM) solver is an open-source solver and implemented in C. HPIPM is designed for small to medium-sized problems, and to efficiently and reliably solve QP problems. HPIPM supports three types of QP problems, dense QPs, Optimal Control Problem (OCP) QPs, and tree-structured OCP QPs. A limited guide, in addition to several test examples, is available for HPIPM. The solver is dependent on the high-performance linear algebra framework BLASFEO[4], and the solver is restricted by the 2-Clause BSD License.

osQP

osQP is an open-source solver using the alternating direction method of multipliers algorithm for solving convex QP problems [25]. The solvers utilize the spars pattern of the

[1] For further information on GNU General License, see `https://www.gnu.org/licenses/gpl-3.0.en.html`

[2] For further information on Apache License, Version 2.0, see `https://www.apache.org/licenses/LICENSE-2.0`

[3] For further information on 2-Clause BSD License, see `https://opensource.org/licenses/BSD-2-Clause`

[4] For further information aboout BLASFEO, see `https://blasfeo.syscop.de/overview/`

matrices in the QP problems, hence requires the same coefficient matrix at almost every iteration. One of the main benefits of this solvers is that it is division free as long as an initial matrix factorization has been done. This makes it a good solver for real-time applications. The solver includes beneficial features like warm start and to change matrices from iteration to iteration. The solver is restricted by the Apache License, Version 2.0.

Selected Solver

All the aforementioned solvers are suited, and highly relevant, for the MPC method. One of the main criteria for selecting a solver is that it is an open-source solver. The reason for this is that it makes it easier for all people involved in the project to further work and also test the MPC implementation. Additionally, being dependent on an expensive solver may affect further development if the licenses can not be renewed. Therefore, FORCES PRO is not an option.

In [24] an experiment is conducted including the three free license solvers, qpOASES, HPIPM, and osQP. The experiment is thoroughly conducted and results in HPIPM being the fastest and most robust solver. Also in [25], a benchmark test has been conducted where osQP shows the best result. Taking into account that [24] is a HPIPM, and [25] is a similar paper describing the algorithm for osQP, it seems that the solvers have different strengths, which determine the changing performance in the tests. Based on the test results from the mentioned papers it is difficult to select one of the solvers. However, the results for qpOASES are good for the tests in both [24] and [25], it is well documented, including good examples, and is well established in the industry. Therefore, qpOASES is selected as the solver used for solving the MPC problem in this project.

3.2 Linearization

In cases where the system model is nonlinear,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{3.29}$$

it is often desirable to approximate the model by linearizing it, to reduce the complexity of the system [19]. This is especially important in the MPC case, where linearizing will reduce the computational load of the optimization problem significantly.

The nonlinear system in Equation 3.29 can be linearized around an operating point, often equilibrium, denoted $\mathbf{x}_0$ and $\mathbf{u}_0$, using the Jacobean matrices defined as

$$\tilde{\mathbf{A}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \dots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \Bigg|_{\mathbf{x}=\mathbf{x}_0}, \qquad \tilde{\mathbf{B}} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \dots & \frac{\partial f_1}{\partial u_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial u_1} & \dots & \frac{\partial f_n}{\partial u_n} \end{bmatrix} \Bigg|_{\mathbf{u}=\mathbf{u}_0}. \tag{3.30}$$

The resulting equation for the linearized system is

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) \approx \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0) + \tilde{\mathbf{A}}\tilde{\mathbf{x}} + \tilde{\mathbf{B}}\tilde{\mathbf{u}}, \tag{3.31}$$

where $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x}_0$ and $\tilde{\mathbf{u}} = \mathbf{u} - \mathbf{u}_0$.

3.3 Discretization

In most cases, the linear system models are described in the continuous time form as

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}, \end{aligned} \tag{3.32}$$

whilst the MPC works in the discrete time form. Hence, the model must be discretized and formulated as in Equation 3.3. Several methods can be used to discretize a linear state-space model, e.g. exact discretization or Euler discretization, where the Euler discretization is somewhat less precise than the exact discretization.

3.3.1 Exact Discretization

Using exact discretization the desired discrete system matrices, $\mathbf{A}_d$, $\mathbf{B}_d$, $\mathbf{C}_d$, $\mathbf{D}_d$, are calculated using the continuous system matrices, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$. In cases where the control inputs are generated by a computer and then passed through a digital-to-analog converter using zero-order hold, results are in a staircase-like control input [19]. In this case, the general solution of a continuous-time state-space equation still holds,

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau. \tag{3.33}$$

Letting the control input be

$$\mathbf{u}(t) = \mathbf{u}(kT) =: \mathbf{u}[k], \quad \text{for } kT \leq t \leq (k+1)T, \tag{3.34}$$

and for $k = 0, 1, 2, \ldots$. Hence, when the control input only changes at discrete time instants and computing Equation 3.33 at $t = kT$, then the continuous state-space model in Equation 3.32 can be written as

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}_d\mathbf{x}_k + \mathbf{B}_d\mathbf{u}_k, \\ \mathbf{y}_k &= \mathbf{C}_d\mathbf{x}_k + \mathbf{D}_d\mathbf{u}_k. \end{aligned} \tag{3.35}$$

The system matrices are defined as

$$\begin{aligned}
\mathbf{A}_d &= e^{\mathbf{A}T} = \mathcal{L}^{-1}\{(s\mathbf{I} - \mathbf{A}\}_{t=T}, \\
\mathbf{B}_d &= \left(\int_{\tau=0}^{T} e^{\mathbf{A}T} d\tau \right) \mathbf{B} = \mathbf{A}^{-1}(\mathbf{A}_d - \mathbf{I})\mathbf{B}, \quad \text{if } \mathbf{A} \text{ is nonsingular}^5, \\
\mathbf{C}_d &= \mathbf{C}, \\
\mathbf{D}_d &= \mathbf{D}.
\end{aligned} \tag{3.36}$$

3.3.2 Euler Discretization

Sometimes the exact discretization may be inconvenient, due to the high complexity of the system model or computational complexity. In this case, the Euler approximation

$$\dot{\mathbf{x}} \approx \frac{\mathbf{x}(t+T) - \mathbf{x}(t)}{T}, \tag{3.37}$$

where T is a given step size, can be sufficiently accurate using a small step size [19]. The smaller step size, the more accurate solution. Substituting Equation 3.37 into the continuous linear model in Equation 3.32, and $t = kT$ for $k = 0, 1, \ldots$, the new discrete formulation of the model is

$$\begin{aligned}
\mathbf{x}((k+1)T) &= (\mathbf{I} + T\mathbf{A})\mathbf{x}(kT) + T\mathbf{B}\mathbf{u}(kT), \\
\mathbf{y}(kT) &= \mathbf{C}\mathbf{x}(kT) + \mathbf{D}\mathbf{u}(kT),
\end{aligned} \tag{3.38}$$

where $\mathbf{x} = \mathbf{x}\mathbf{I}$.

[5] If $\mathbf{A}$ is an $n \times n$ square matrix, and its determinant is nonzero, then $\mathbf{A}$ is a full rank matrix, called a nonsingular matrix [19].

Chapter 4

Implementation

The MPC method is implemented in cascade with a simulation environment for testing both the implementation, and also its performance. An overview of the interaction between the modules and the data flow is presented in Figure 4.1.

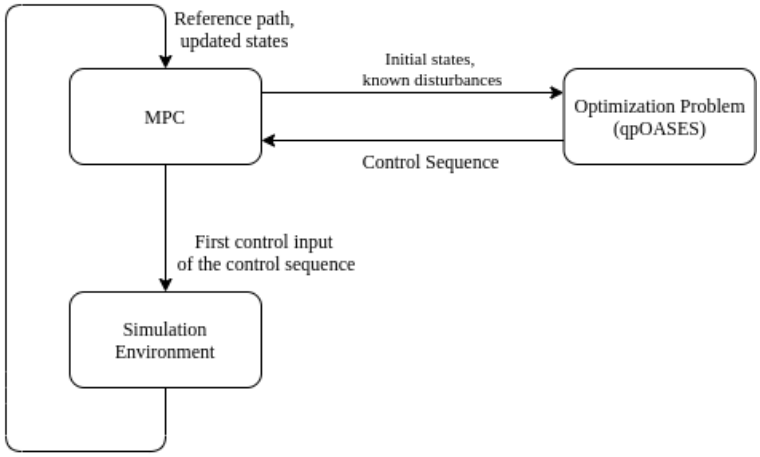


Figure 4.1: Overview of the interaction between the MPC, the solver of the optimization problem, and the simulation environment.

The problem formulation of the MPC, with the associated constraint and prediction model, is described in Section 4.1. The formulation described in this section is based on the findings of the associated specialization topic [3]. The implementation interface for the MPC, including the designated solver is represented in Section 4.2. Further, the entire simulation environment is described in Section 4.3.

4.1 Problem Formulation

For the path following problem, the main objective is to keep the distance error between the path and the vehicle, called *cross-track error*, at zero, in addition to keeping the angle error between them, called *heading error*, at zero. When the cross-track error and the heading error is equal to zero, the vehicle is following the given reference path. The cross-track error and the heading error are visualized in Figure 4.2 where the cross-track error is represented by e_d and the heading error is represented by e_ψ.

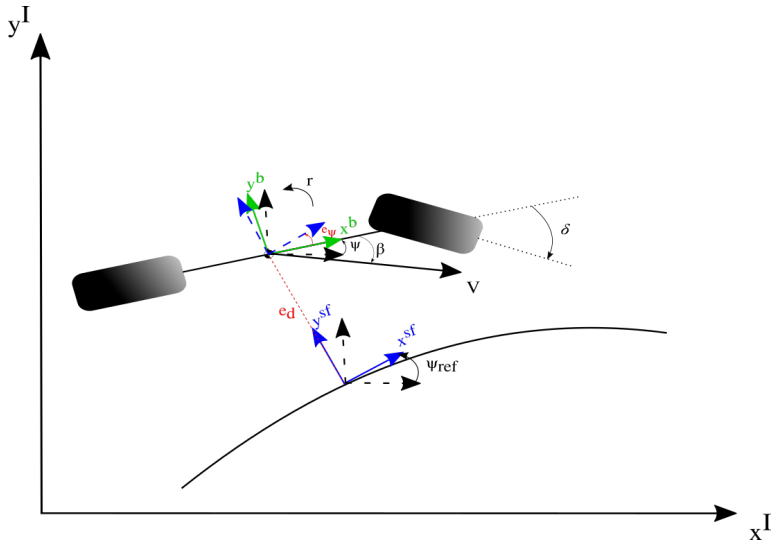


Figure 4.2: Visual representation of the MPC problem, with important variables.

The cross-track error is derived from the bicycle model, described in Section 2, and the curved reference path, described in the planar coordinate frame, the vehicle aims to follow. A representation of this is seen in Figure 4.2. The change in cross-track error is then defined in the body frame as

$$\dot{e}_d = \dot{x}sin(e_\psi) + \dot{y}cos(e_\psi). \tag{4.1}$$

The states, $\dot{x}$, and $\dot{y}$ are the velocities of the vehicle, as defined in Equation 2.7. It is considered a valid assumption that the rotational velocity of the vehicle is small for the given time step, in addition to the cornering radius being significantly larger than the cross-track error. Therefore, the small-angle approximation can be deployed to Equation 4.1. Also, it can be assumed that the velocity in the x-direction is significantly larger than the velocity in the y-direction, hence the velocity in the y-direction can be neglected. Including the given assumptions, the change in cross-track error results in

$$\dot{e}_d = \dot{x}\psi_e = ue_\psi. \tag{4.2}$$

The response of the path can be described by the change in heading and the curvature of the path, where the curvature is defined as one divided by the radius, denoted $\kappa_{ref} = \frac{1}{R}$. The representation of the path, introduced in Section 2.3.1, is a two times continues curve, therefore the evolution of the path can be described by the state-space model

$$\begin{bmatrix} \dot{\psi}_{ref} \\ \dot{\kappa}_{ref} \end{bmatrix} = \begin{bmatrix} 0 & u \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \psi_{ref} \\ \kappa_{ref} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \dot{\kappa}_{ref}. \tag{4.3}$$

For simplicity, it is assumed that the vehicle is following the osculating circle of the curvature.

The vehicle response may be described by either a kinematic or a dynamic model representation. The kinematic model of the vehicle behavior describes the low velocity response at a desirable level, where low velocity is, as aforementioned, defined as $0m/s \leq u \leq 5m/s$. However, when the velocity increases the nonlinear behavior will be present and a dynamic model is more suitable. Therefore, both the dynamic and the kinematic response are described.

4.1.1 Kinematic Formulation

The kinematic formulation uses the change of the vehicle position and heading to describe the response. The control input for changing the behavior to follow the given reference path is the change in steering angle. For the kinematic formulation, it is considered a valid assumption that the vehicle follows the heading of the steering angle, hence the change in vehicle heading can be described by Equation 2.6, and assuming zero body slip, due to driving at low velocities. When assuming small-angle approximation, the change in vehicle heading results in

$$\dot{\psi} = \frac{u}{l_f + l_r}\delta. \tag{4.4}$$

The change in cross-track error is described by Equation 4.2 where the velocity is assumed constant. The heading error, e_ψ, is influenced by both the vehicle heading and the path heading, $e_\psi = \psi - \psi_{ref}$. Since the heading error is described in the Center of Gravity (CG) of the vehicle, the cross-track error, described by both the vehicle heading and the path heading, is defined as

$$\dot{e}_d = u(\psi + \frac{l_r}{l_f + l_r}\delta) - u\psi_{ref}. \tag{4.5}$$

Combining the vehicle response and the path response the response is summarized as a kinematic state-space model

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & u & u\frac{l_r}{l_r+l_f} & -u & 0 \\ 0 & 0 & \frac{u}{l_r+l_f} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & u \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} d, \tag{4.6}$$

where the system states are $\mathbf{x} = [e_d \quad \psi \quad \delta \quad \psi_{ref} \quad \kappa_{ref}]^T$, the control input is $u = \dot{\delta}$ and $d = \dot{\kappa}_{ref}$, which is considered a known disturbance.

The required numerical values for the vehicle models, both the kinematic and the dynamic, are summarized in Table 4.1.

	Symbol	Unit	Numerical Value
Mass	m	[kg]	196.5
Wheel base front	l_f	[m]	0.813
Wheel base rear	l_r	[m]	0.717
Track front	t_f	[m]	1.180
Track rear	t_r	[m]	1.200
Cornering stiffness front	C_f	[N/rad]	80
Cornering stiffness rear	C_r	[N/rad]	80
Moment of inertia	I_z	[kg $\cdot$ m^2]	86.1

Table 4.1: Numerical values for vehicle parameters.

4.1.2 Dynamic Formulation

Inclusion of the dynamics of the vehicle in the problem formulation leads to a more complex model, yet more accurate during high velocities. The lateral dynamics defined in Section 2.2.2 is used to describe the behavior of the vehicle. Equation 2.20 describes the yaw rate and the lateral acceleration, with linear tire models. Assuming the longitudinal velocity, u is constant or regulated using a separate longitudinal controller, the yaw dynamic can be described by

$$\begin{aligned} \ddot{y} &= \frac{1}{m}\left(- B_{cs,r}\frac{\dot{y} - l_r r}{u} - B_{cs,f}\Big(\frac{\dot{y} + l_f r}{u} - \delta\Big)\right) - ur, \\ \dot{r} &= \frac{1}{I_Z}\left(- l_r B_{cs,r}\frac{\dot{y} - l_r r}{u} - l_f B_{cs,f}\Big(\frac{\dot{y} + l_f r}{u} - \delta\Big)\right), \end{aligned} \tag{4.7}$$

where only the lateral velocity and the yaw rate are the controllable variables.

For the dynamic case, the lateral velocity is included in the change in cross-track error, hence the cross-track error is now defined as

$$\dot{e}_d = u\psi + v - u\psi_{ref}, \tag{4.8}$$

where the small-angle approximation is still assumed valid.

The dynamic system can now be described as a states-space formulation as

$$\begin{aligned}\dot{\mathbf{x}} = &\begin{bmatrix} 0 & u & 1 & 0 & 0 & -u & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -\frac{B_{cs,r}+B_{cs,f}}{mu} & \frac{l_r B_{cs,r} - l_f B_{cs,f}}{mu} - u & \frac{B_{cs,f}}{m} & 0 & 0 \\ 0 & 0 & \frac{l_r B_{cs,r} + l_f B_{cs,f}}{u} & \frac{l_r^2 B_{cs,r} - l_f^2 B_{cs,f}}{mu} & \frac{l_f B_{cs,f}}{I_z} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & u \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x} \\ &+ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} d,\end{aligned} \tag{4.9}$$

where the new state matrix is $\mathbf{x} = [e_d \quad \psi \quad \dot{y} \quad \dot{\psi} \quad \delta \quad \psi_{ref} \quad \kappa_{ref}]^T$ and the control input and the known disturbance are still the change in steering angle and the curvature, respectively.

4.1.3 Minimization Variables

Several states can be minimized in the MPC problem. For the tracking problem the two main minimization states are, both for the kinematic and the dynamic formulation, the cross-track error, and the heading error, as described above. For the kinematic formulation, the output function results in

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} e_d \\ \psi \\ \delta \\ \psi_{ref} \\ \kappa_{ref} \end{bmatrix}. \tag{4.10}$$

For the dynamic formulation, the output function is defined as

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} e_d \\ \psi \\ \dot{y} \\ \dot{\psi} \\ \delta \\ \psi_{ref} \\ \kappa_{ref} \end{bmatrix}. \tag{4.11}$$

During high velocity cornering, the nonlinear behavior of the vehicle is highly present. Therefore, it may be preferable to minimize the lateral acceleration and yaw rate for a less varying, and more stable behavior.

4.1.4 System Constraints

Both the vehicle states and the path states have some desired limitations, in addition to physical limitations.

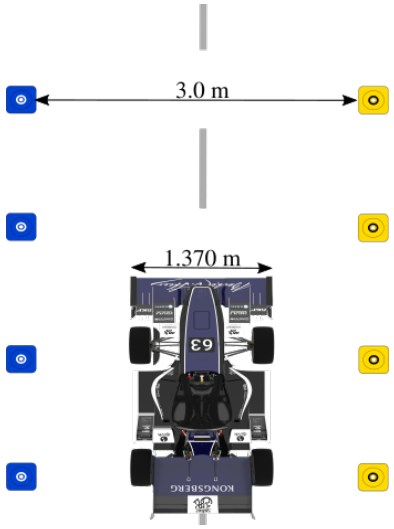


Figure 4.3: Vehicle following the centerline of a track, bounded by cones. 3.0 m is marking the track width, while 1.2 m is marking the widest part of the vehicle.

The Formula Student rules [1] states that the minimum track width is 3m. The main task of the vehicle is to follow the centerline, therefore, the vehicle can not deviate more than $\frac{3.0\text{m}-1.37\text{m}}{2} = 0.815\text{m}$ from the centerline without hitting cones. To have some safety margin, the allowed deviation should be somewhat less than 0.815 meters. The steering angle is physically limited by the steering actuator. The steering actuator does not allow an angle more than 110 degrees to both sides, which results in a steering wheel angle of approximately 25.6 degrees.

The minimum hairpin defined by the formula student rules [1] is a 9 meters outside diameter hairpin, resulting in a maximum curvature of

$$\kappa = \frac{1}{R} = \frac{1}{\frac{9m - 1.5m \cdot 2m}{2}} \approx 0.45 \frac{1}{m}, \tag{4.12}$$

with a track width of 3 meters. It is not necessary to include limitations of the curvature. However, it can be used as a safety, as a mathematical error can occur when calculating the curvature using the reference path. The reference path can have a curvature in both left and right directions, hence the limitation for the curvature is $\pm 0.8 \frac{1}{m}$, to have some margin.

To avoid entering the nonlinear region, the lateral velocity and the yaw rate of the vehicle may be limited. Based on data from Atmos as a manned vehicle, the lateral velocity is limited to $\pm \dot{y} = 4 \frac{m}{s^2}$ and the yaw rate to $\pm 2.7 \frac{rad}{s}$.

The desired and physical constraints can be summarized as

$$\begin{aligned} -0.8\text{m} \leq \quad & e_d & \leq 0.8\text{m}, \\ -26.5 \cdot \frac{\pi}{180} \text{ rad} \leq \quad & \delta & \leq 26.5 \cdot \frac{\pi}{180} \text{ rad}, \\ -0.8 \frac{1}{\text{m}} \leq \quad & \kappa_{ref} & \leq 0.8 \frac{1}{\text{m}}. \end{aligned} \tag{4.13}$$

4.1.5 Controller Tuning

To increase the accuracy and stability of the MPC method the weighting matrices, the prediction horizon must be tuned concerning its problem formulation. Therefore, the kinematic and dynamic formulations are tuned separately. As mentioned, the minimization variables, for both the kinematic and the dynamic formulation, are the heading error, e_ψ, and the cross-track error, e_d, hence the weighting matrices result in $\mathbf{Q} \in \mathbb{R}^{2 \times 2}$. Both formulations include only one control input, steering angle rate, $\dot{\delta}$, hence the associated weighting matrices are scalar, $R \in \mathbb{R}^{1 \times 1}$. When tuning the weighting matrices and the prediction horizon it will always be a trade-off between stability and fast response. Therefore, it may be difficult to find the balance between them. The procedure for finding the final values for the prediction horizon and the weighting matrices can be seen in Appendix C.

Kinematic MPC tuning

The final tuning of the weighting matrices results in

$$\mathbf{Q} = \begin{bmatrix} 5 & 0 \\ 0 & 35 \end{bmatrix}, \quad R = 0.001. \tag{4.14}$$

The resulting error responses with the associated weighting matrices in Equation 4.14, and prediction horizon, $H = 6$, are displayed in Figure 4.4. The state responses and the control

input response are displayed in Figure 4.5, where both the states and the control input is regulated to zero within a reasonable time, with a smooth response.

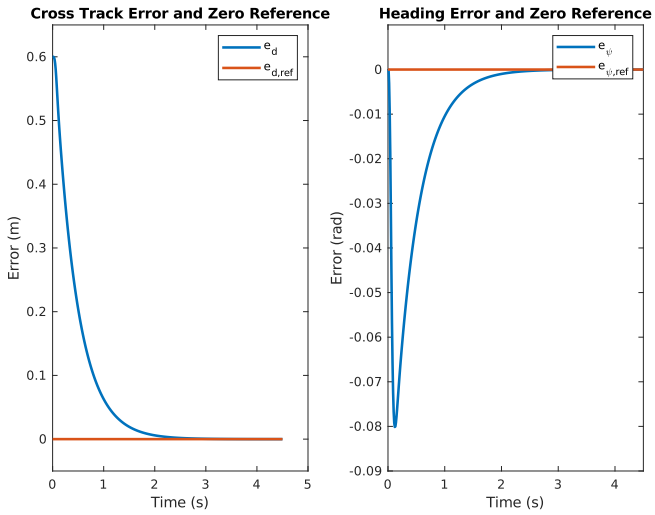


Figure 4.4: Error response, where the orange line represents the desired error value $e_d = e_\psi = 0$, for the kinematic formulation.

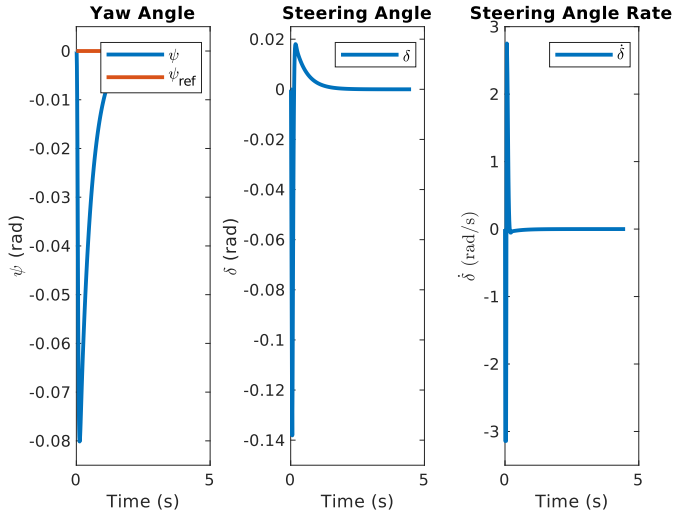


Figure 4.5: State response, where the orange line represents the desired error value $e_\psi = 0$, and control input response, for the kinematic formulation..

Dynamic MPC tuning

The final tuning of the weighting matrices results in

$$\mathbf{Q} = \begin{bmatrix} 5 & 0 \\ 0 & 35 \end{bmatrix}, \quad R = 0.001. \tag{4.15}$$

The resulting error responses with the associated weighting matrices in Equation 4.15, and prediction horizon, $H = 30$, are displayed in Figure 4.6. The state responses and the control input response are displayed in Figure 4.7, where both the states and the control input are regulated to zero within a reasonable time, with a smooth response, similar to the kinematic formulation response.

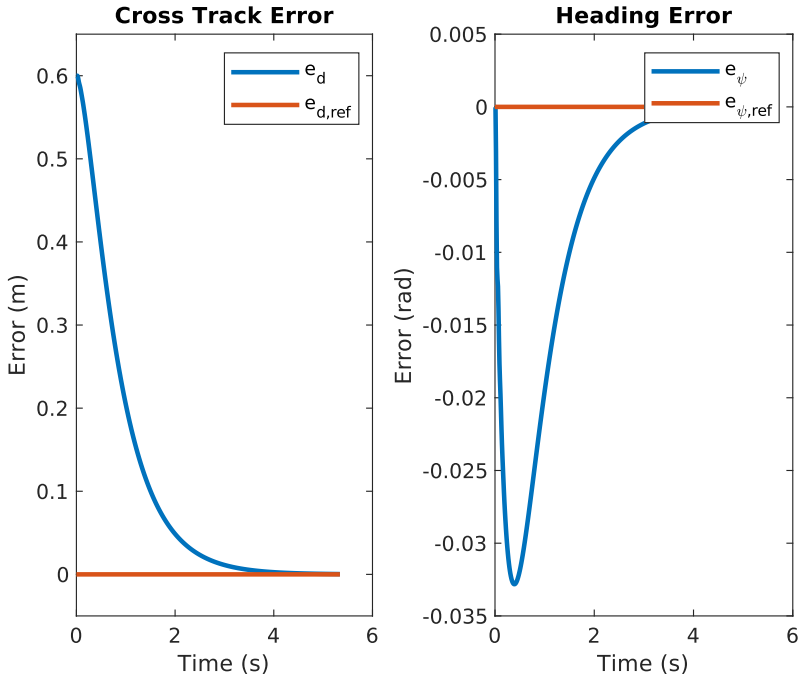


Figure 4.6: Error response, where the orange line represents the desired error value $e_d = e_\psi = 0$, for the dynamic formulation.

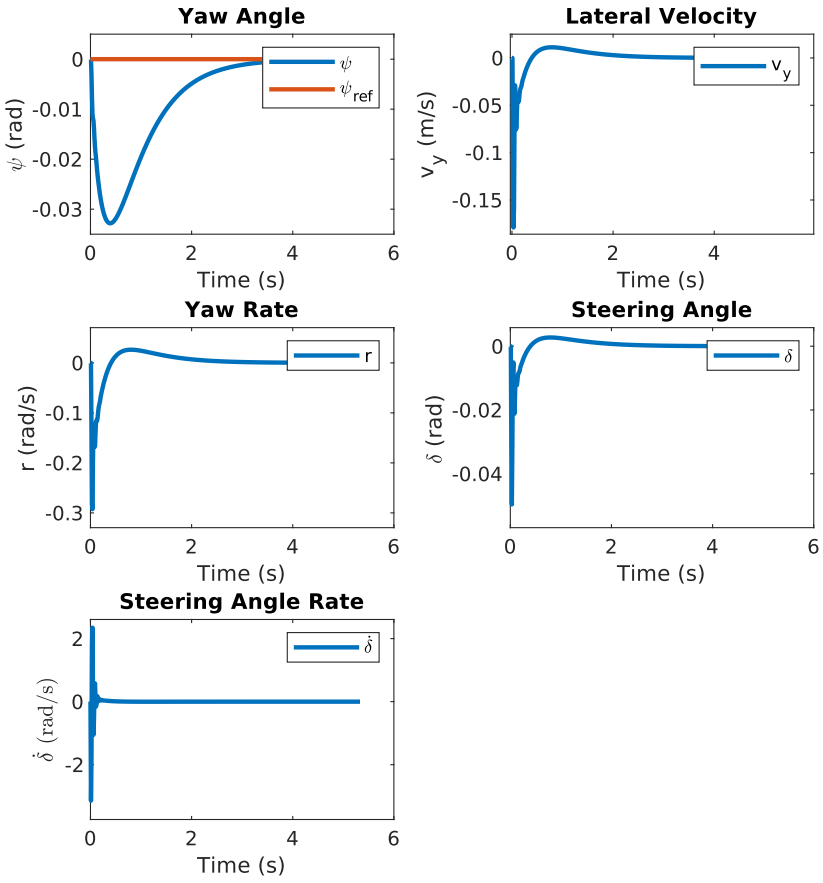


Figure 4.7: State response, where the orange line represents the desired error value $e_\psi = 0$, and control input response, for the dynamic formulation.

4.2 Implementation Interface

The two MPC methods are implemented in MATLAB [26] using the MATLAB version of the qpOASES solver. The system model is discretized using the MATLAB function `c2d()` from the MATLAB toolbox, control toolbox [27].

4.2.1 qpOASES Specific Implementation

The selected solver, qpOASES, requires the MPC problem stated as

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_x}} \quad & \frac{1}{2} x^T H x + x^T g(w_0) \\ \text{s. t.} \quad & lbA(w_0) \leq Ax \leq ubA(w_0), \\ & lb(w_0) \leq x \leq ub(w_0). \end{aligned} \tag{4.16}$$

To formulate the MPC method as desired, the sequential formulation described in Equation 3.28, is used. The matrices in Section 4.1 are substituted into the sequential formulation in Equation 3.28 and results in two MPC problems, one using the kinematic prediction model and one using the dynamic prediction model, resulting in the desired QP problem formulation. The code implementation using variables is presented in Appendix D, together with the interaction with the qpOASES solver.

The sequential formulation is used due to the smaller optimization problem which should reduce the computational load and that it is well suited for QP solver. Additionally, the sequential formulation should ease the implementation, as unlike for the simultaneous formulation, not all system states need calculating.

4.3 Simulation Environment

To simulate the implemented MPC both a model of the vehicle and a representation of the reference path must be implemented. To increase the value of the simulation a four-wheel model of the vehicle is used as the vehicle during the simulation. For the path representation, splines are used, due to their smooth and compact characteristics.

4.3.1 Vehicle Simulation

The simulation environment for validating the performance and robustness of the MPC consists of a dynamic model of a four-wheel-drive vehicle, including

- longitudinal load transfer,
- lateral load transfer,
- aerodynamics weight, and
- F_x and F_y tire force estimation using the Pacejka tire model.

All variables and parameters needed for calculating the different forces and moments are listed in Table 4.2.

	Symbol	Unit
Sprung mass	m_s	[kg]
Roll stiffness front	K_f	[N·m /deg]
Roll stiffness rear	K_r	[N·m/deg]
Height of CG of the unsprung mass front	$h_{cg,us,f}$	[m]
Height of CG of the unsprung mass rear	$h_{cg,us,r}$	[m]
Height of the roll center front	$h_{rc,f}$	[m]
Height of the roll center rear	$h_{rc,r}$	[m]
Height of vehicle CG	h_{cg}	[m]
Change in height	dh	[m]
Aerodynamic area	A	[m^2]
Air density	ρ_{air}	[kg/m^3]
Aerodynamic drag constant	C_d	[-]
Aerodynamic lift constant	C_l	[-]
Static weight distribution front	wd_f	[-]
Static weight distribution rear	wd_r	[-]

Table 4.2: Variables for calculating the different forces and torques for the simulation environment.

Vehicle Dynamics for a Four Wheel Model

For accuracy and realistic simulation results, the four-wheel vehicle model, as seen in Figure 4.8, is used to update the vehicle dynamics.

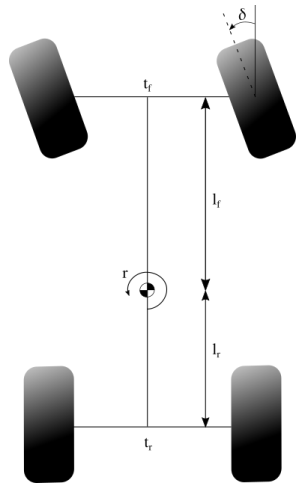


Figure 4.8: Four wheel model of the vehicle with designated constants and variables.

By using Newton's second law of motion and the moment balance about the z-axis of the vehicle, equations for linear and angular acceleration can be deducted. For the linear acceleration, each contribution of each tire is summarized for the longitudinal and lateral

acceleration. Likewise, for the angular acceleration, each tire's yaw moment contribution is summarized, which results in

$$\begin{aligned}
a_x =& \frac{1}{m}\Big((F_{x,fl}cos(\delta) + F_{x,fr}cos(\delta)) - (F_{y,fl}sin(\delta) + F_{y,fr}sin(\delta)) \\
&+ (F_{x,rl} + F_{x,rr}) - F_{x,aero}\Big) + v \cdot r, \\
a_y =& \frac{1}{m}\Big((F_{y,fl}cos(\delta) + F_{y,fr}cos(\delta)) + (F_{x,fl}sin(\delta) + F_{x,fr}sin(\delta)) \\
&+ (F_{y,rl} + F_{y,rr})\Big) - u \cdot r, \\
\dot{r} =& \frac{1}{I_z}(l_f(F_{x,fl}sin(\delta) + F_{x,fr}sin(\delta)) + l_f(F_{y,fl}cos(\delta) + F_{y,fr}cos(\delta)) \\
&+ \frac{t_f}{2}(F_{x,fr}cos(\delta) - F_{x,fl}cos(\delta)) + \frac{t_f}{2}(F_{y,fl}sin(\delta) - F_{y,fr}sin(\delta)) \\
&- l_r(F_{y,rl} + F_{y,rr}) + \frac{t_f}{2}(F_{x,rr} - F_{x,rl})).
\end{aligned} \tag{4.17}$$

The lateral and longitudinal tire forces are estimated using the Pacejka Tire Model and will be described more thoroughly in the next section.

The aerodynamic longitudinal force is also included in the calculation for the longitudinal acceleration. The vehicle experiences a high amount of drag, due to the complex aerodynamic package of the vehicle, and it can be described by

$$F_{x,aero} = \frac{\rho_{air} C_d A u^2}{2}, \tag{4.18}$$

where C_d is the aerodynamic drag constant, ρ_{air} is the air density, and A is the total aerodynamic device area.

Longitudinal and Lateral Tire Force Estimation

The longitudinal and lateral forces felt by the tires are estimated using the Pacejka Tire model described in Section 2.2.2. The same method that is used for lateral tire force estimation can be used for estimating the longitudinal tire forces. Similarly, as lateral forces are a result of the appearance of the slip angle, the longitudinal forces are a result of the appearance of the slip ratio. The slip ratio is defined as the relationship between the angular and the linear velocity for the tire in the longitudinal direction. Mathematically it is described as

$$sr = \frac{\omega R_{eff}}{V \cdot cos(sa)} \cdot \frac{1}{100}. \tag{4.19}$$

The slip ratio is a dimensionless variable and is in the interval between zero and one. The slip ratio is calculated independently for each tire. The effective radius, R_{eff}, is a

changing radius depending on the load on the tire and is calculated using Pacejka magic formula, where it is assumed that the tire acts like an undamped spring. The effective radius of a tire is visualized in Figure 2.4.

Tire Load Estimation

The tire load is the z-component of the force acting on each wheel. The total load acting on a tire is

$$F_z = F_{z,long} + F_{z,lat} + F_{z,static} + F_{z,aero}. \tag{4.20}$$

The static tire load is the load acting on the tires while standing still, a result of the vehicle mass and gravity is defined by

$$\begin{aligned} F_{z,static,f} &= \frac{m \cdot g \cdot wd_f}{2}, \\ F_{z,static,r} &= \frac{m \cdot g \cdot wd_r}{2} \end{aligned} \tag{4.21}$$

where $wd_{f,r}$ is the weight distribution, and the subscripts f and r refers to the front and rear tire, respectively.

Longitudinal and Lateral Load Transfer

When the vehicle moves, it will experience load transfer due to acceleration, both in the longitudinal and the lateral direction.

The longitudinal load transfer is

$$\begin{aligned} F_{z,long,f} &= \frac{m \cdot a_x \cdot h_{cg}}{L} \cdot wd_f, \\ F_{z,long,r} &= \frac{m \cdot a_x \cdot h_{cg}}{L} \cdot wd_r, \end{aligned} \tag{4.22}$$

for the front and rear tire, respectively.

The lateral load transfer depends on the roll stiffness of the vehicle, meaning how much the vehicle resists while rolling [7], and is defined as

$$\begin{aligned} F_{z,lat,f} &= a_y \left(\frac{m_s}{t_f} \left(dh \frac{Kf - l_r \cdot m_s \cdot \frac{dh}{L}}{Kf + Kr - m_s \cdot dh} + \frac{l_r}{L \cdot h_{rc,f}} \right) + \frac{m_{us,f}}{t_f \cdot h_{cg,us,f}} \right), \\ F_{z,lat,r} &= a_y \left(\frac{m_s}{t_r} \left(dh \frac{Kf - l_f \cdot m_s \cdot \frac{dh}{L}}{Kf + Kr - m_s \cdot dh} + \frac{l_f}{L \cdot h_{rc,f}} \right) + \frac{m_{us,r}}{t_r \cdot h_{cg,us,r}} \right), \end{aligned} \tag{4.23}$$

for the front and rear wheel respectively, and where

$$dh = h_{cg} - \frac{h_{rc,f} + h_{rc,r}}{2}. \tag{4.24}$$

The load transfer for the left side of the vehicle is initially negative, due to the coordinate frame placed in the CG.

Tire Load due to Vehicle Aerodynamics

The autonomous vehicle includes a complex aerodynamic package. The aerodynamic load is dependent on the velocity of the vehicle and is arising from drag. The aerodynamic load for each tire can be modeled as

$$F_{z,aero} = \frac{\rho_{air} C_l A u^2}{2}, \tag{4.25}$$

where C_l is the aerodynamic lift constant, ρ_{air} is the air density, and A is the total aerodynamic device area.

4.3.2 Path Representation

The path is defined by the centerline of the track, as aforementioned, and is represented as in Section 2.3 using splines. By using desired coordinates, also called waypoints, a polynomial function can be extracted, as seen in Figure 4.9. A cubic spline, constructed from piecewise third-order polynomials, holds the necessary assumptions of being second derivative continuous, and regular. When the vehicle position is known, a projection can be executed down to the reference spline to find the cross-track error at a given time instance, as described in Section 2.3.1.

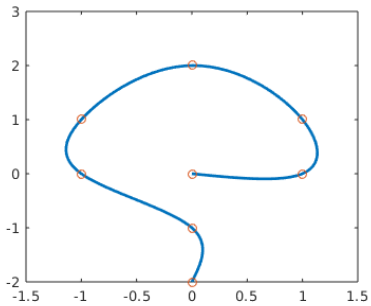


Figure 4.9: Example of a cubic spline representation, where the blue graph is the spline and the orange dots are the waypoints.

The spline representation is implemented in MATLAB using the MATLAB's Curve Fitting Toolbox [28].

Chapter 5

Results

In this chapter, the results of the MPC methods are presented. The results are divided into two categories

- performance results, and
- timing results.

The performance results intend to present the MPC's ability to track the reference path for a variety of tracks. As the MPC is suppose to be embedded into a real-time system, the run time of the controller may be as important as the ability to follow a given track. If the run time of the control system is too slow, the controller will not be able to give reasonable results, and by that not manage to follow the reference path. The performance results will be presented in Section 5.1, and the timing results will be presented in Section 5.2.

5.1 Performance Results

Using the aforementioned simulation environment, three experiments are conducted for testing and validating the performance of the implemented MPC,

- straight-line driving,
- constant radius cornering, and
- a track containing all elements of Formula Student Driverless Track described by the Formula Student rules [1].

These three experiments are selected as the straight-line driving and the constant radius corner validates the implementation and the functionality of the MPC. A full track is also included as an experiment, as this should validate some of the edge cases of the MPC

implementation. Some of the edge cases are a track with changing curvature and high-speed cornering. These experiments may also be linked up to the goals of Revolve NTNU, described in Section 1.2.

5.1.1 Straight-Line

The track setup for the straight-line driving is displayed in Figure 5.1. The experiment is conducted with a constant longitudinal velocity of $u = 15$m/s, as this may be the average velocity of the vehicle during the dynamic events.

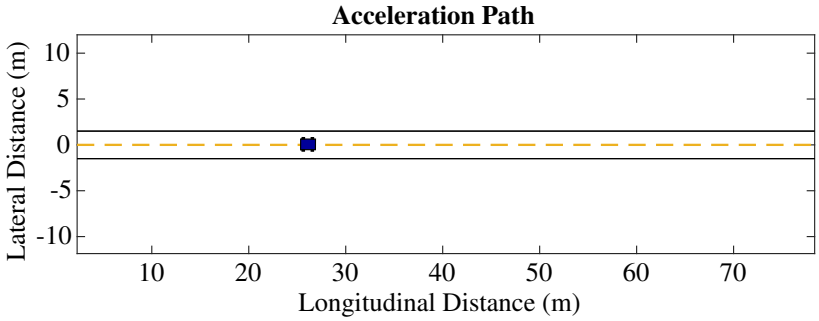


Figure 5.1: Track layout for straight-line driving. The yellow dotted line represents the centerline, and the black lines are the border of the track with a track width of 3m. The blue object represents the autonomous vehicle, Atmos Driverless.

Two experiments are conducted using the straight-line path, one experiment with an initial cross-track error, $e_d = 0.6$m, and one experiment with initial cross-track error, $e_d = 0.6$m, and initial heading error, $e_\psi = 0.03$rad. The heading error is chosen to be in the opposite direction of the natural direction for driving towards the centerline. The reason for choosing the heading error in the opposite direction is that it demands a higher control effort to change the heading of the vehicle and minimize the cross-track error. The experiments are conducted for both the kinematic vehicle model, and the dynamic vehicle model.

Kinematic Vehicle Model with Initial cross-track Error

The tracking of the reference path with an initial cross-track error, using the kinematic vehicle model is displayed in Figure 5.2. The respective error states are displayed in Figure 5.3, the problem states are presented in Figure 5.4, including the state constraints, and the resulting control input is displayed in Figure 5.5, also including the control input constraints.

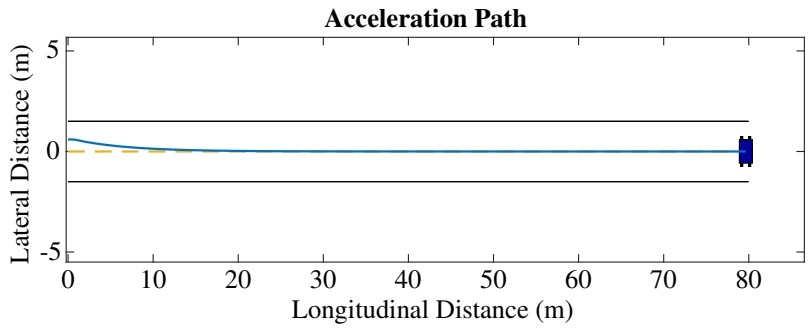


Figure 5.2: Lateral tracking of straight driving with the kinematic formulation with an initial cross-track error of 0.6m. The blue line is the path driven by the vehicle.

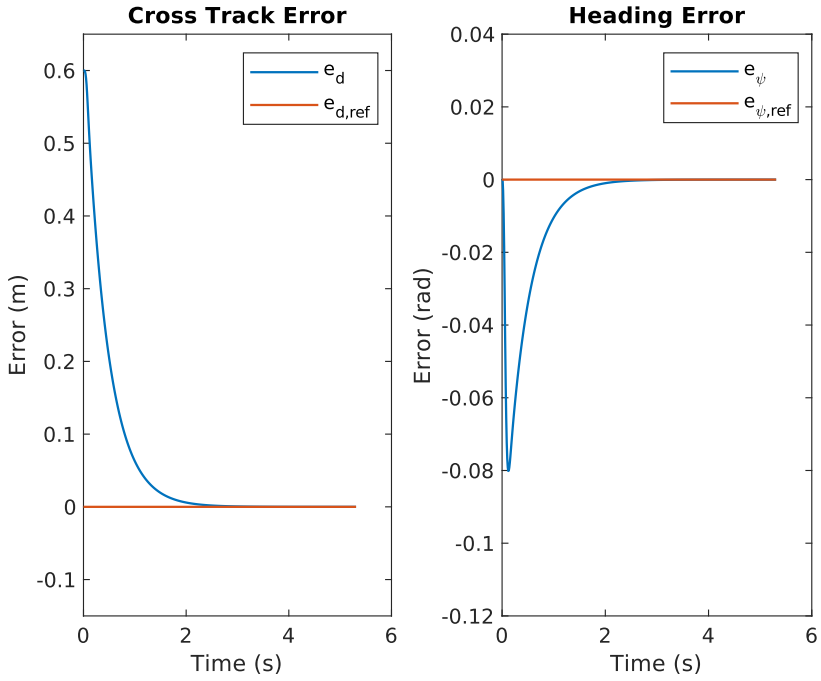


Figure 5.3: Error states while tracking the centerline with an initial cross-track error of 0.6m with the kinematic formulation. The orange lines represents the zero references.

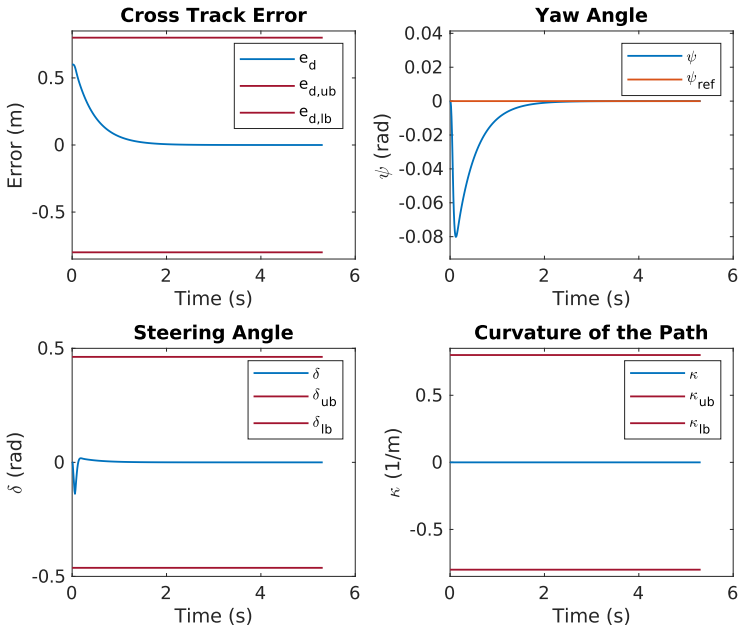


Figure 5.4: Vehicle states while tracking the centerline with an initial cross-track error of 0.6m with the kinematic formulation. The dark red lines represent the associated state constraints.

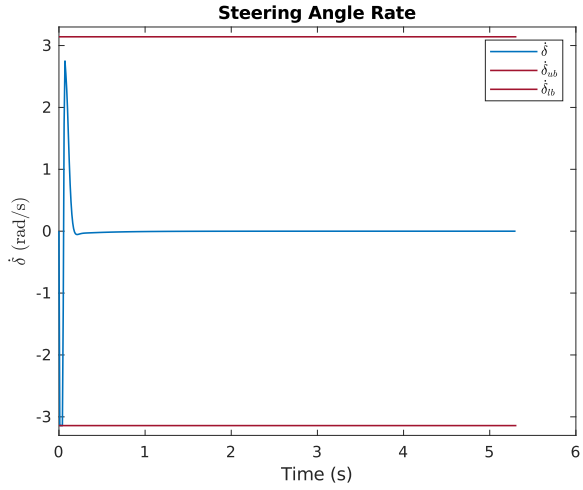


Figure 5.5: The control effort while tracking the centerline with an initial cross-track error of 0.6m with the kinematic formulation. The dark red lines represent the associated control input constraints.

Kinematic Vehicle Model with Initial cross-track Error & Heading Error

The error states for the kinematic vehicle model when tracking the reference path with an initial cross-track error of 0.6m, in addition to an initial heading error of 0.03rad are displayed in Figure 5.6. The problem states, with its respective constraints, are displayed in Figure 5.7, and the control input is displayed in Figure 5.8, including the control input constraints.

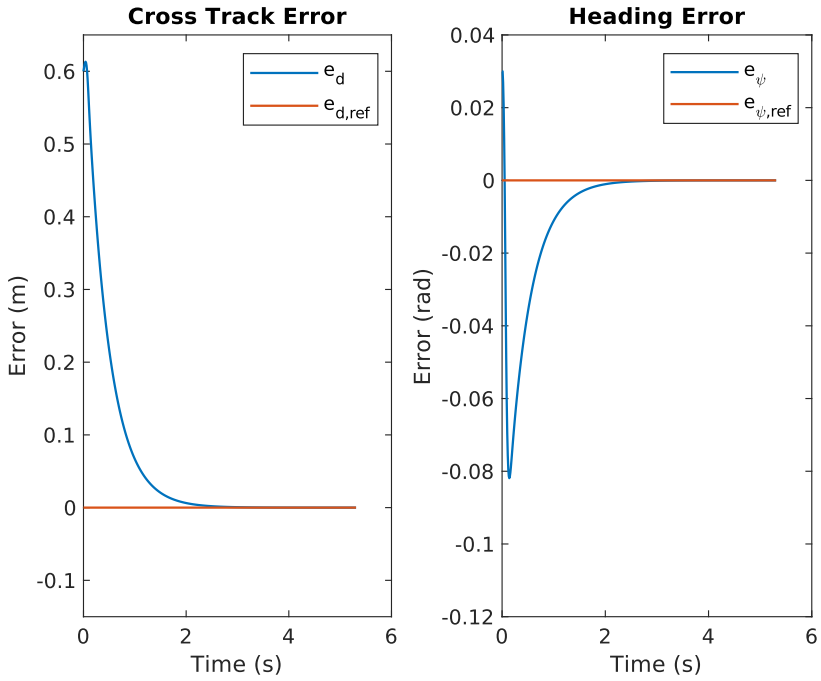


Figure 5.6: The error states while tracking the centerline with an initial cross-track error of 0.6m, in addition to an initial heading error of 0.03rad using the kinematic formulation. The orange lines represent the zero references.

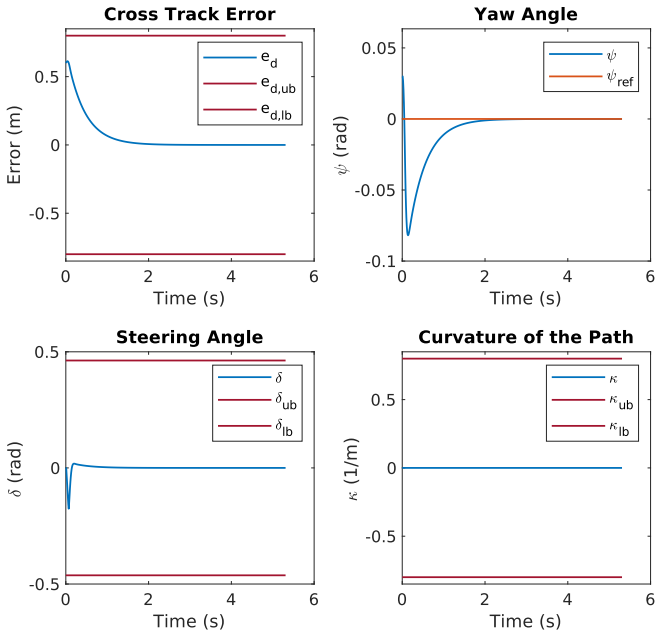


Figure 5.7: The vehicle states while tracking the centerline with an initial cross-track error of 0.6m, in addition to an initial heading error of 0.03rad using the kinematic formulation. The dark red lines represent the associated state constraints.

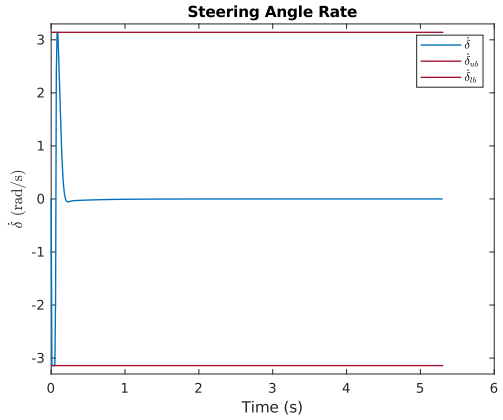


Figure 5.8: The control effort while tracking the centerline with an initial cross-track error of 0.6m, in addition to an initial heading error of 0.03rad using the kinematic formulation. The dark red lines represent the associated control input constraints.

Dynamic Vehicle Model with Initial Cross-Track Error

The tracking of the reference path with an initial cross-track error of 0.6m, using the dynamic vehicle model is displayed in Figure 5.9. The respective error states are displayed in Figure 5.10, the problem states are displayed in Figure 5.11, and the control input is displayed in Figure 5.12. The associated constraints are also displayed in the respective figures.

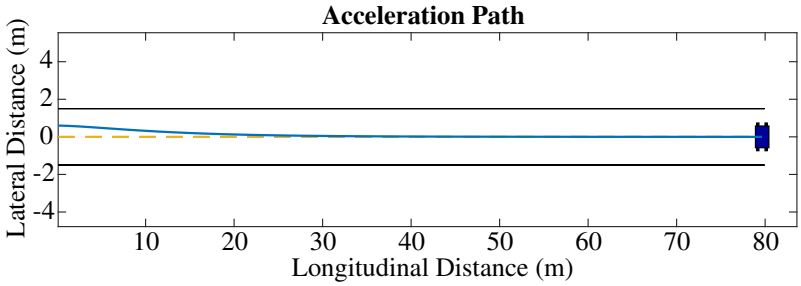


Figure 5.9: Lateral Tracking of straight driving with an initial cross-track error of 0.6m using the dynamic vehicle formulation. The blue line is the path driven by the vehicle.

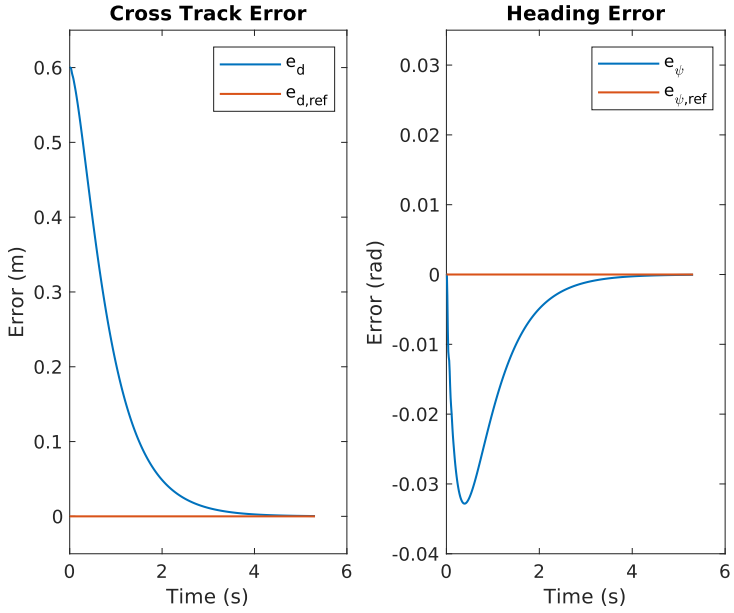


Figure 5.10: The error states while tracking the centerline with an initial cross-track error of 0.6m using the dynamic formulation. The orange lines represent the zero references.

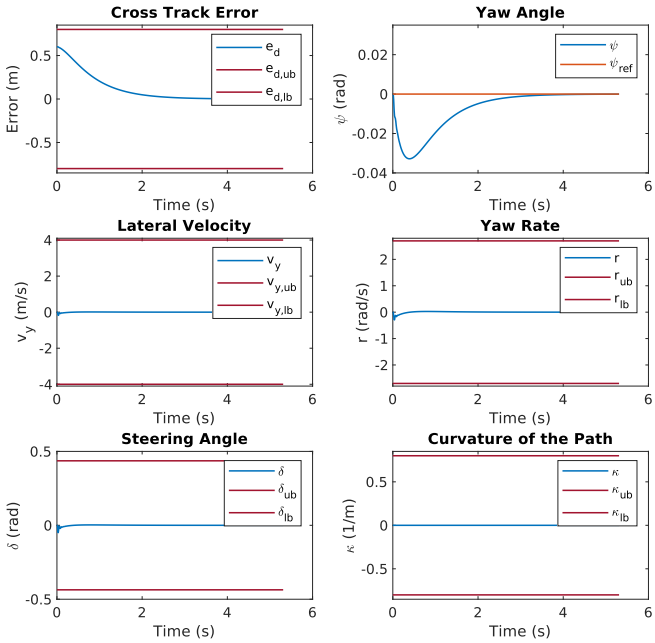


Figure 5.11: The vehicle states while tracking the centerline with an initial cross-track error of 0.6m using the dynamic formulation. The dark red lines represent the associated state constraints.

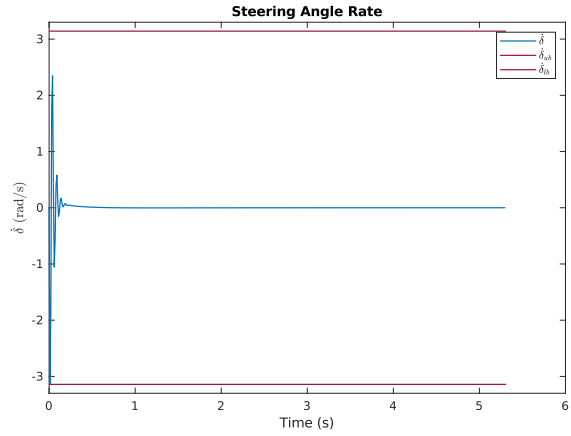


Figure 5.12: The control effort while tracking the centerline with an initial cross-track error of 0.6m using the dynamic formulation. The dark red lines represent the associated control input constraints.

Dynamic Vehicle Model with Initial Cross-Track Error & Heading Error

The error states for the dynamic vehicle model when tracking the reference path with an initial cross-track error of 0.6m, in addition to an initial heading error of 0.03rad are displayed in Figure 5.13. The problem states, with its respective constraints, are displayed in Figure 5.14, and the control input is displayed in Figure 5.15, including the control input constraints.

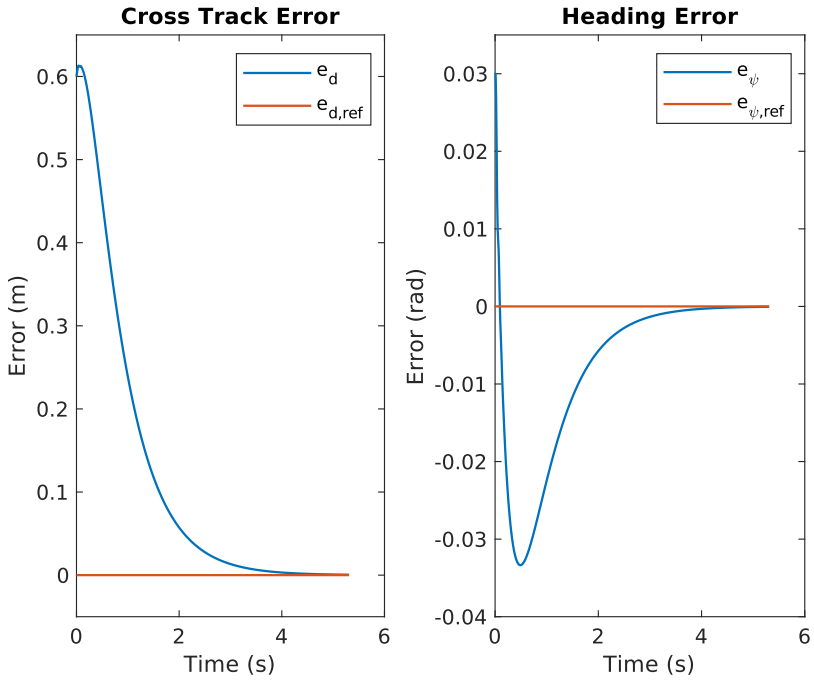


Figure 5.13: Error states while controlling the vehicle to the centerline with an initial cross-track error of 0.6m, in addition to an initial heading error of 0.3rad with the dynamic formulation. The orange lines represent the zero references.

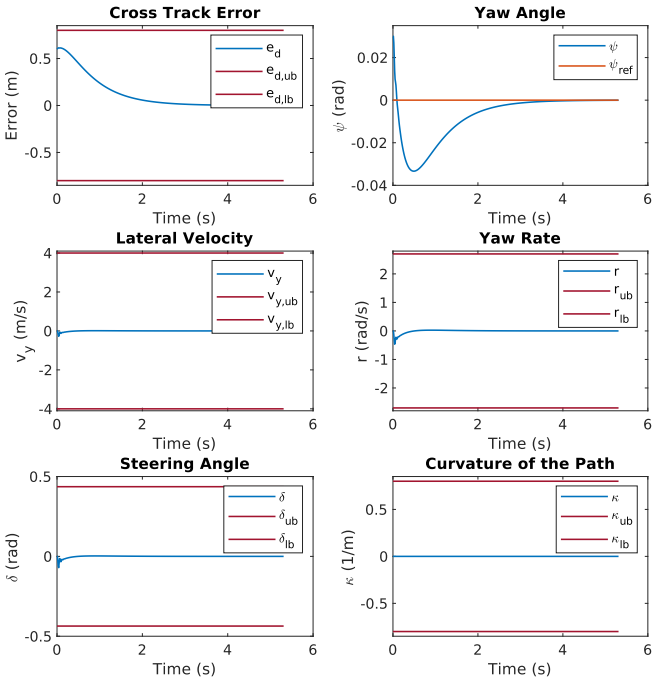


Figure 5.14: Vehicle states while controlling the vehicle to the centerline with an initial cross-track error of 0.6m, in addition to an initial heading error of 0.3rad with the dynamic formulation. The dark red lines represent the associated state constraints.

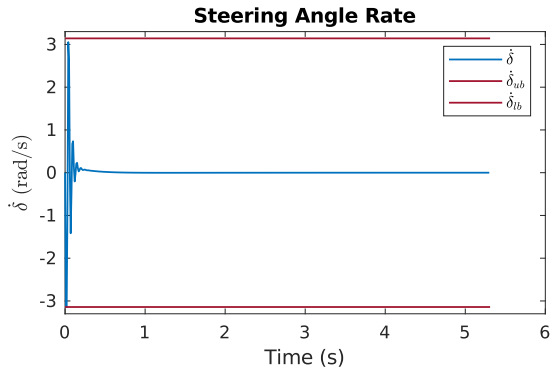


Figure 5.15: The control effort while controlling the vehicle to the centerline with an initial cross-track error of 0.6m, in addition to an initial heading error of 0.3rad with the dynamic formulation. The dark red lines represent the associated control input constraints.

5.1.2 Constant Radius Cornering

The track setup for the constant radius cornering is displayed in Figure 5.16. The experiment is conducted with a constant radius of $R = 9.125$m. This experiment is similar to the skidpad track described in the formula student rules [1].

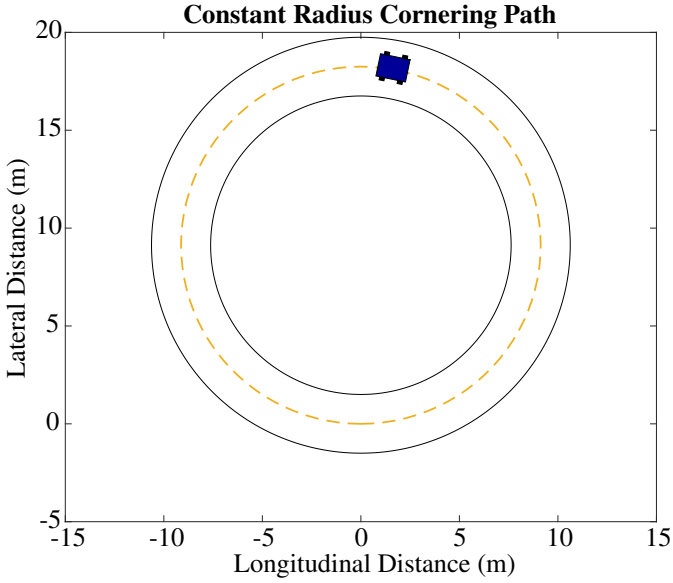


Figure 5.16: The track layout for constant radius cornering. The yellow dotted line represents the centerline, and the black lines are the border of the track with a track width of 3m. The blue object represents the autonomous vehicle, Atmos Driverless.

To validate that the control regime manages to track a path with a constant nonzero curvature, even within cross-track error, the experiment is conducted with an initial cross-track error, $e_d = 0.6$m, similar to the aforementioned straight-line experiment.

Kinematic Vehicle Model

The vehicle path when tracking the centerline of a constant radius circle is displayed in Figure 5.17. The respective error states is displayed in Figure 5.18, the vehicle states are displayed in Figure 5.19, and the control input is displayed in Figure 5.20, all with its associated constraints. The experiment is conducted with a constant longitudinal velocity of $u = 15$m/s.

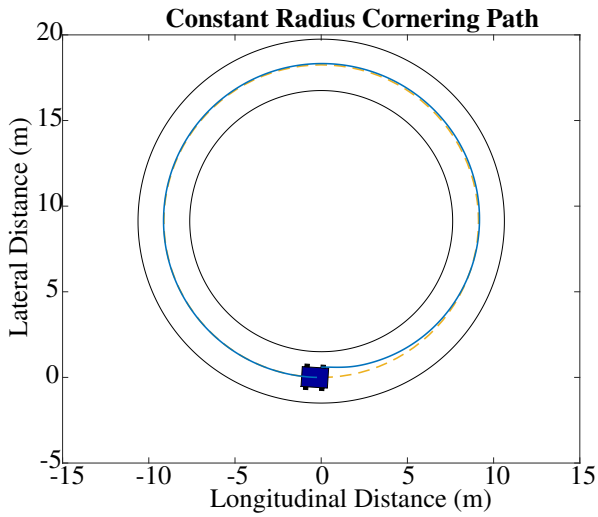


Figure 5.17: Lateral tracking of a constant radius circle with an initial cross-track error of 0.6m using the kinematic formulation. The blue line is the path driven by the vehicle.

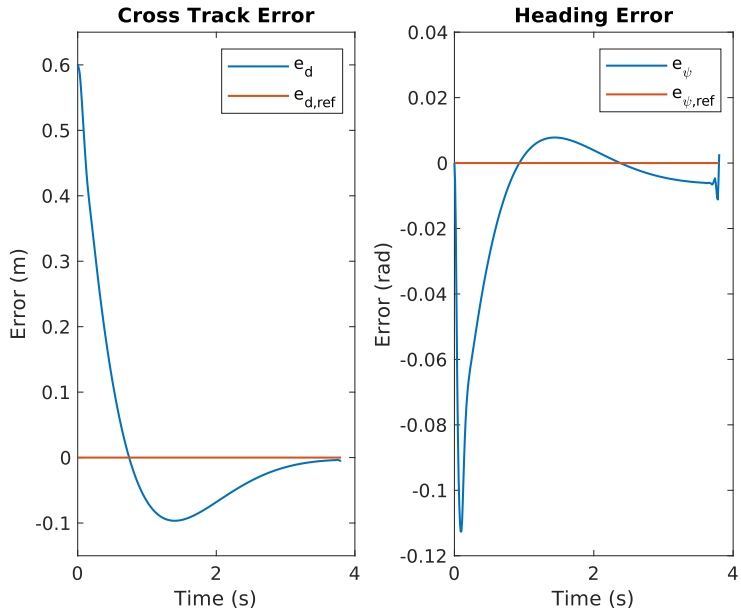


Figure 5.18: The error states while tracking the centerline with an initial cross-track error of 0.6m using the kinematic formulation. The orange lines represent the zero references.

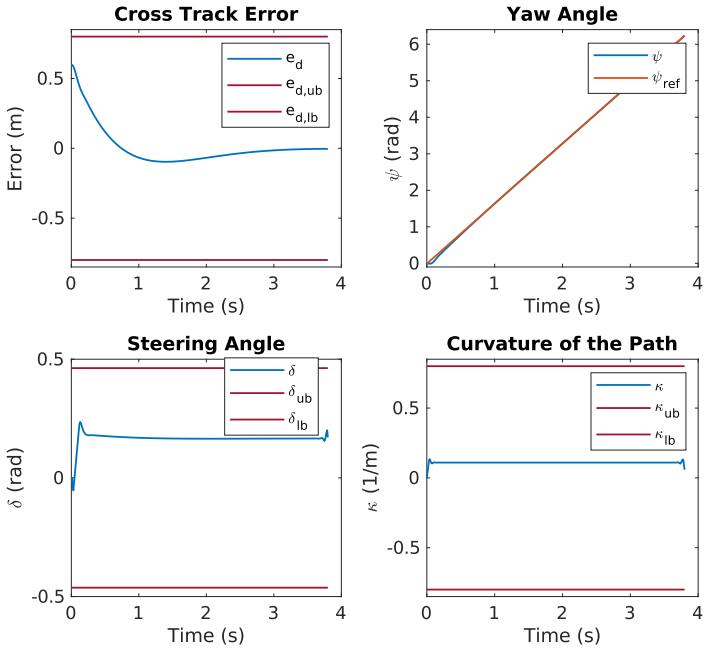


Figure 5.19: The vehicle states while tracking the centerline with an initial cross-track error of 0.6m using the kinematic formulation. The dark red lines represent the associated state constraints.

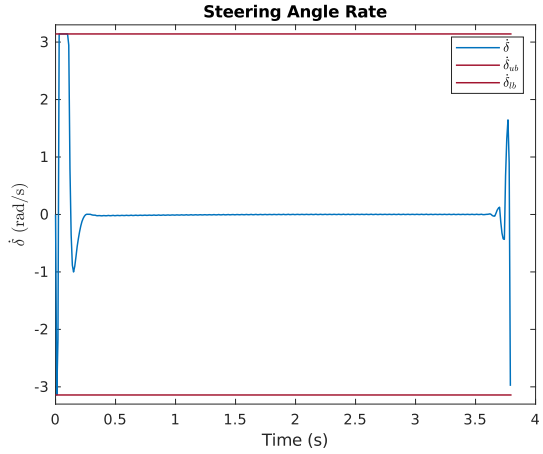


Figure 5.20: The control effort while tracking the centerline with an initial cross-track error of 0.6m using the kinematic formulation. The dark red lines represent the associated control input constraints.

Dynamic Vehicle Model

The vehicle path while tracking the centerline of a constant radius circle using the dynamic vehicle model is displayed in Figure 5.21. The respective error states are displayed in Figure 5.22, the control input is displayed in Figure 5.23, and the vehicle states are displayed in Figure 5.24, all with its associated constraints. The experiment is conducted with a constant longitudinal velocity of $u = 10$m/s.

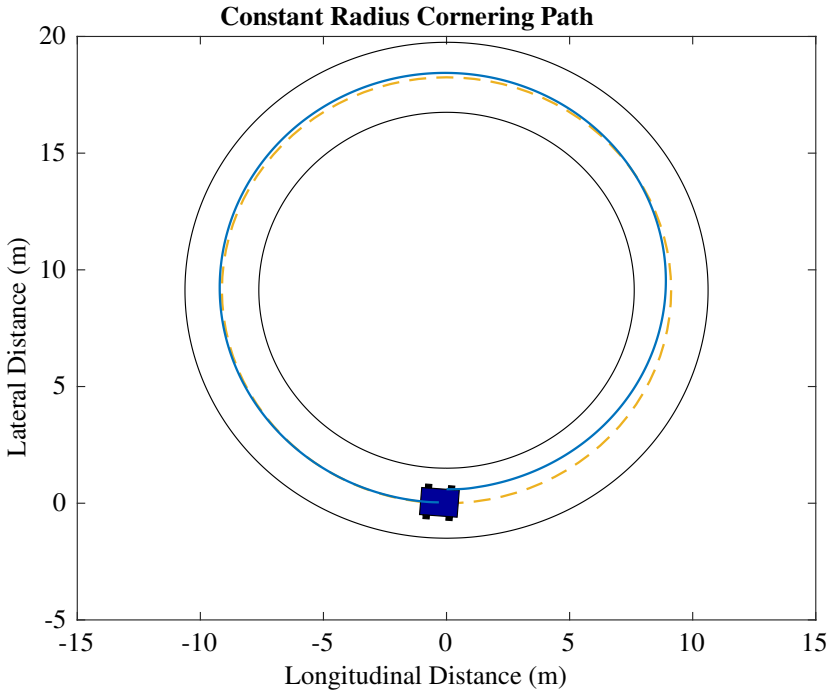


Figure 5.21: Lateral tracking of a constant radius circle with an initial cross-track error of 0.6m using the dynamic formulation. The blue line is the path driven by the vehicle.

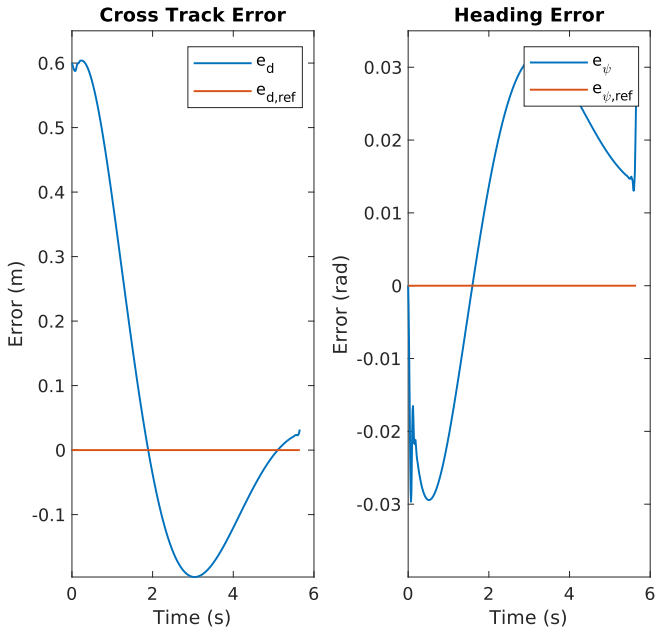


Figure 5.22: The error states while tracking the centerline with an initial cross-track error of 0.6m using the dynamic formulation. The orange lines represent the zero references.

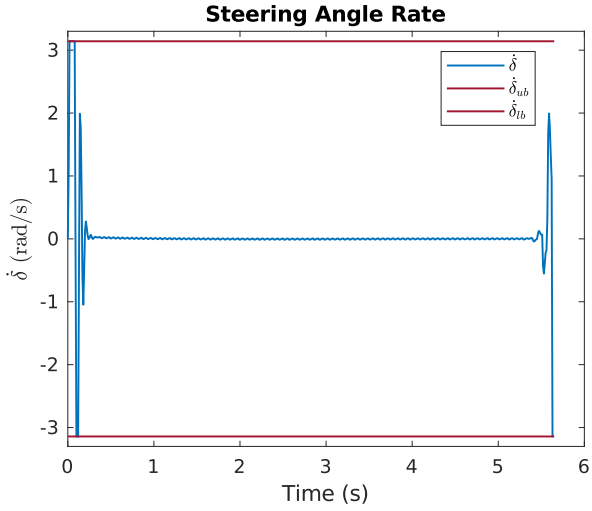


Figure 5.23: The control effort while tracking the centerline with an initial cross-track error of 0.6m using the dynamic formulation. The dark red lines represent the associated control input constraints.

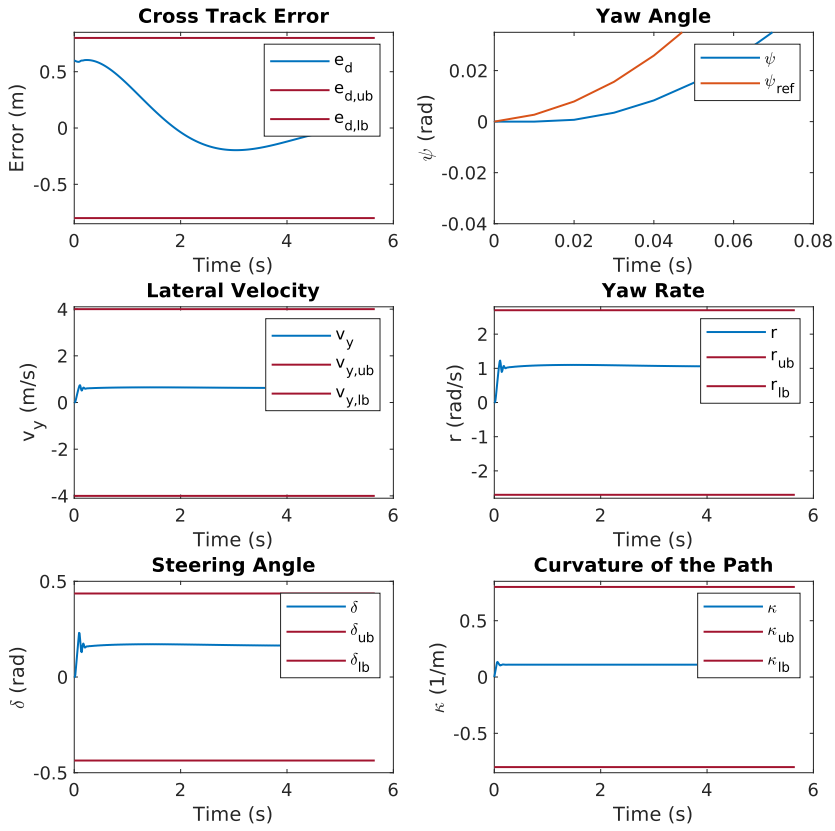


Figure 5.24: The vehicle states while tracking the centerline with an initial cross-track error of 0.6m using the dynamic formulation. The dark red lines represent the associated state constraints.

5.1.3 Formula Student Driverless Track

The final experiment is a track containing all elements in a Formula Student Track, including

- Straights: No longer than 80 m,
- Constant Turns: up to 50 m diameter,
- Hairpin Turns: Minimum of 9 m outside diameter (of the turn),
- Miscellaneous: Chicanes, multiple turns, decreasing radius turns, etc [1].

The generated track is displayed in Figure 5.25.

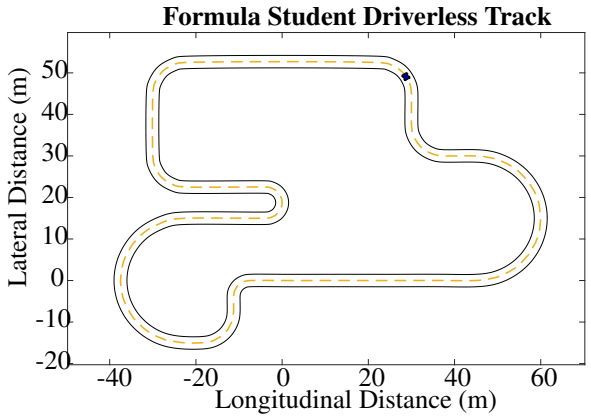


Figure 5.25: Track layout for the Formula Student Track. The yellow line represents the centerline, and the black lines are the boarder of the track with a track width of 3m. The blue objects represents the autonomous vehicle, Atmos Driverless.

Kinematic Vehicle Model

The vehicle path when tracking the centerline of the generated Formula Student Track is displayed in Figure 5.26. The respective error states are displayed in Figure 5.27, the control input is displayed in Figure 5.28, and the vehicle states are displayed in Figure 5.29, all with its associated constraints. The experiment was conducted with a constant longitudinal velocity of $u = 17$m/s.

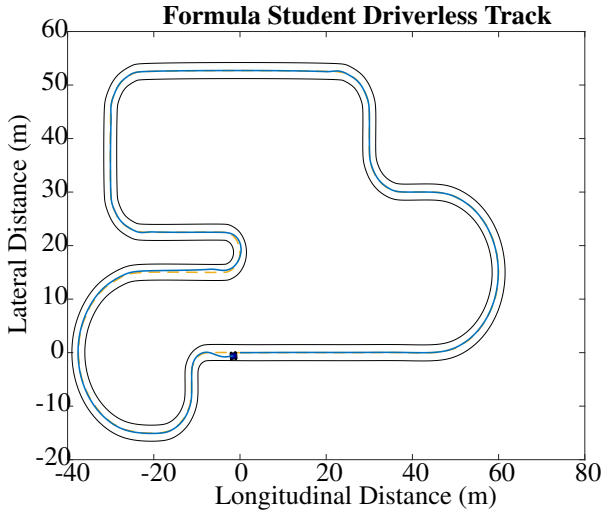


Figure 5.26: Lateral Tracking of the track using kinematic formulation. The blue line is the path driven by the vehicle.

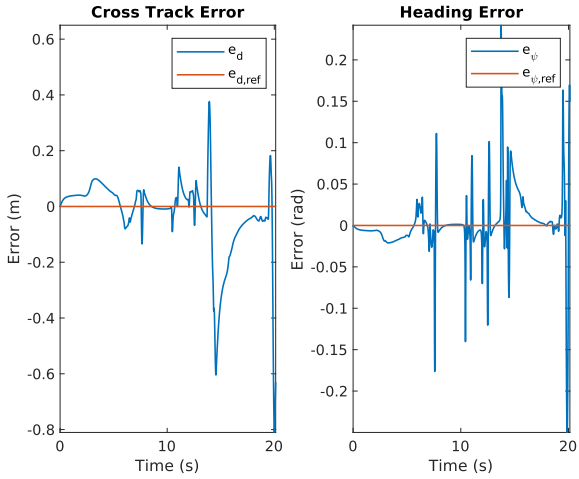


Figure 5.27: Error states while tracking the centerline of the Formula Student track using the kinematic formulation. The orange lines represent the zero references.

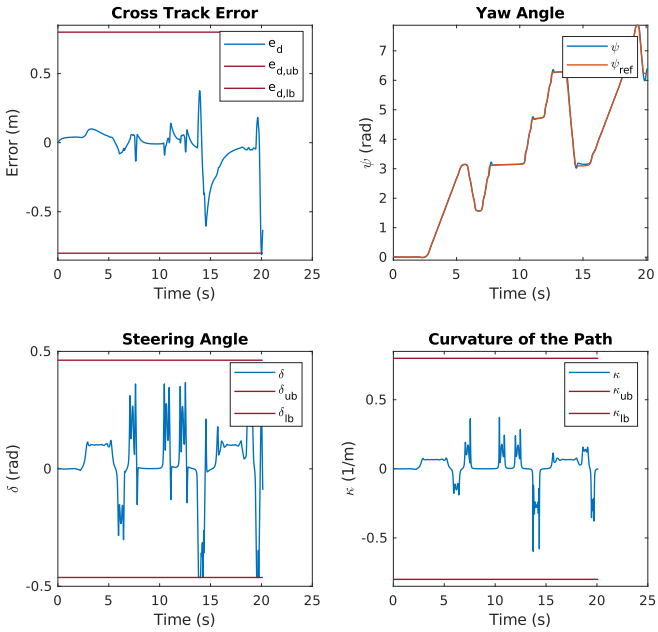


Figure 5.28: Vehicle states while tracking the centerline of the Formula Student track using the kinematic formulation. The dark red lines represent the associated state constraints.

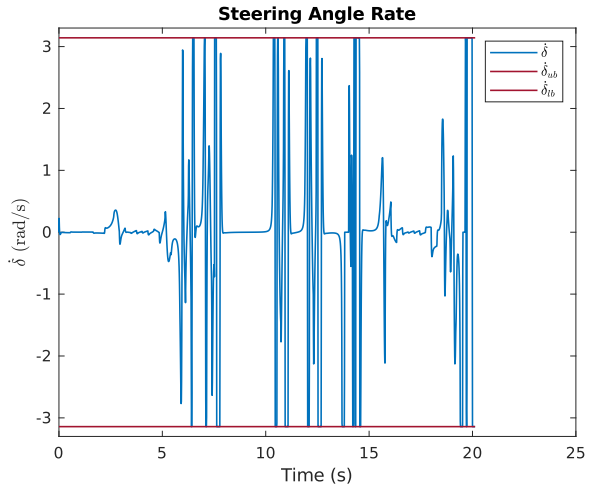


Figure 5.29: The control effort while tracking the centerline of the Formula Student track using the kinematic formulation. The dark red lines represent the associated control input constraints.

Dynamic Vehicle Model

The vehicle path when tracking the centerline of the generated Formula Student Track using the dynamic vehicle model is displayed in Figure 5.30. The respective error states are displayed in Figure 5.31, the control input is displayed in Figure 5.33, and the vehicle states are displayed in Figure 5.32, all with its associated constraints. The experiment was conducted with a constant longitudinal velocity of $u = 5$m/s.

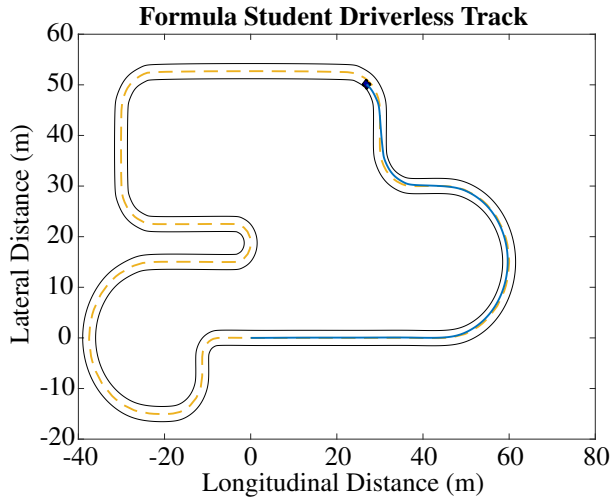


Figure 5.30: Lateral Tracking of the track using dynamic formulation. The blue line is the driven path of the vehicle.

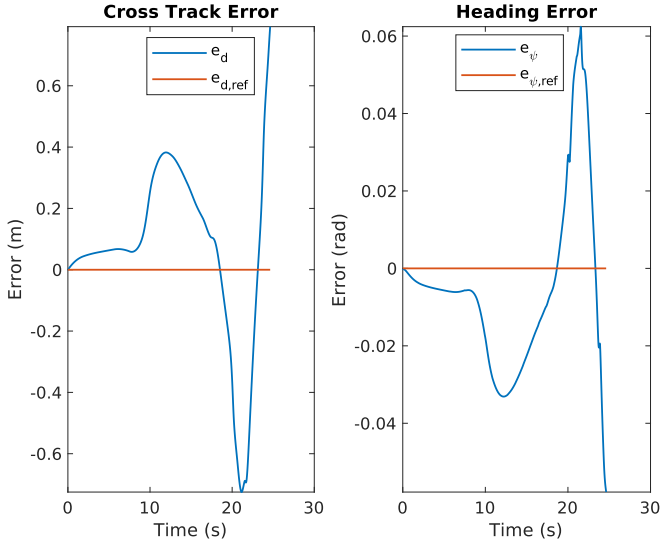


Figure 5.31: Error states while tracking the centerline of the Formula Student track using the dynamic formulation. The orange lines represent the zero references.

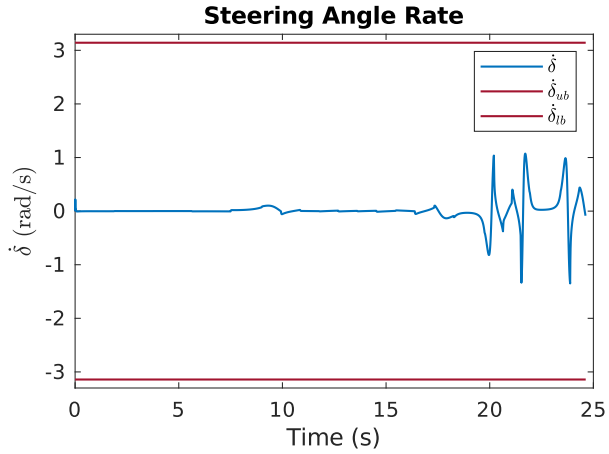


Figure 5.32: The control effort while tracking the centerline of the Formula Student track using the dynamic formulation. The dark red lines represent the associated control input constraints.

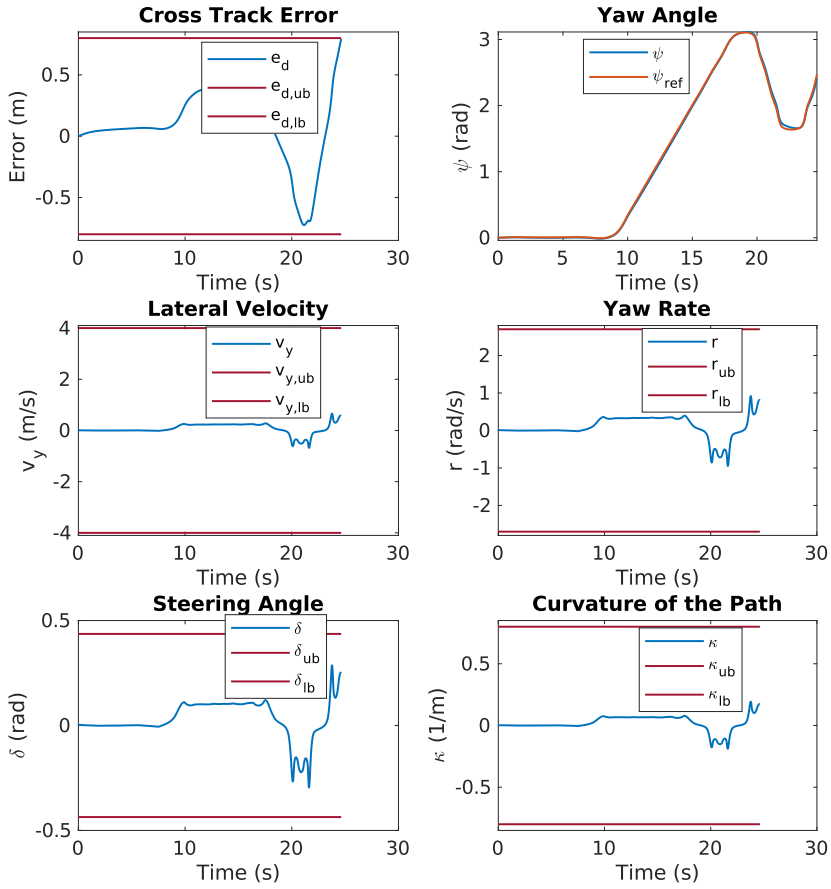


Figure 5.33: Vehicle states while tracking the centerline of the Formula Student track using the dynamic formulation. The dark red lines represent the associated state constraints.

5.2 Computational Effort Results

To calculate the computational effort of the MPC implementations, the MATLAB functions *tic* and *toc* have been used. Both the straight driving track and the constant radius track has been utilized in the experiments for better validation. The experiments are conducted with a constant velocity of $u = 5$m/s. When calculating the computational effort, only the formulation of the MPC matrices and the *qpOASES* solver are included.

Figure 5.34 displays the computational effort for the kinematic vehicle model while tracking a straight path, and Figure 5.35 shows the effort while tracking a constant radius circle of $R = 9.125$m. The same experiments are conducted for the dynamic vehicle model. Fig-

ure 5.36 represents the computational effort for the straight-line tracking, and Figure 5.37 displays the computational effort for tracking the constant radius, both using the dynamic vehicle model.

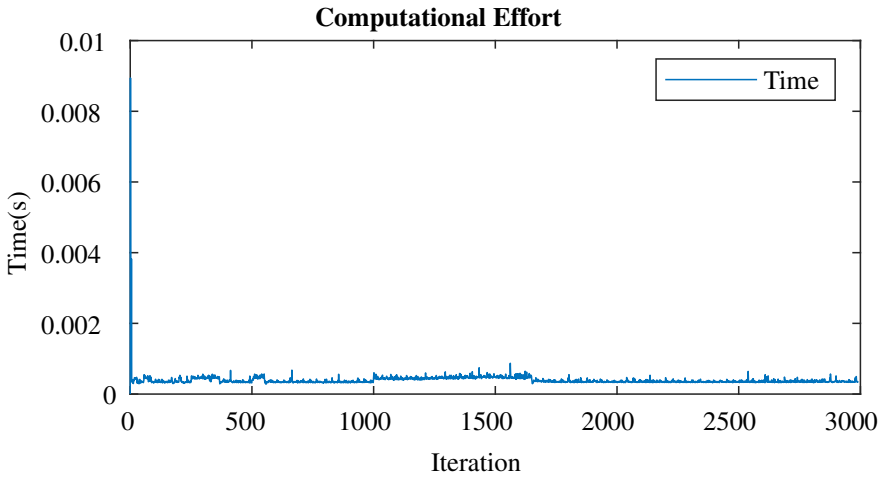


Figure 5.34: The computational effort of the MPC using the kinematic vehicle model, while driving a straight path.

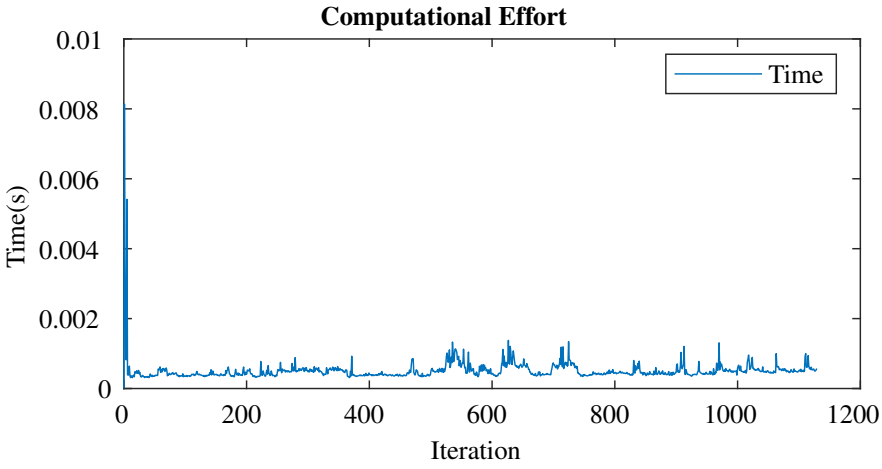


Figure 5.35: The computational effort of the MPC using the kinematic vehicle model, while driving a constant radius circle.

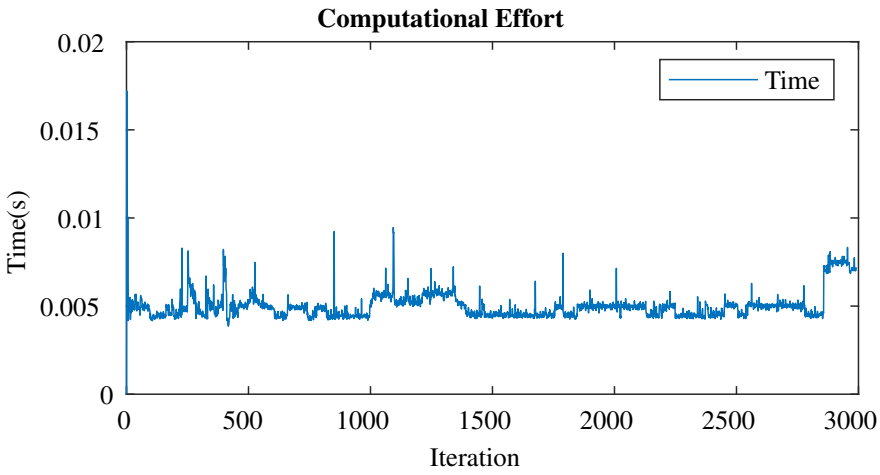


Figure 5.36: The computational effort of the MPC using the dynamic vehicle model, while driving a straight path.

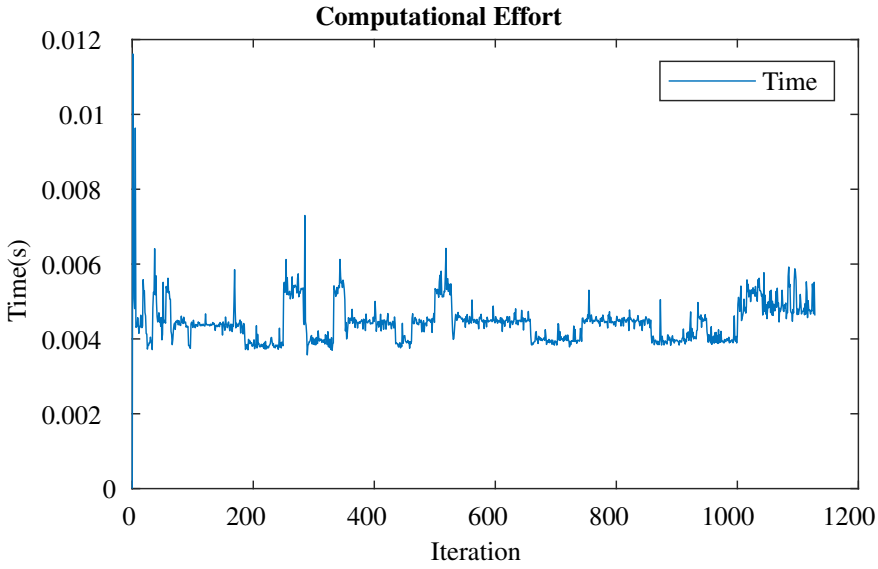


Figure 5.37: The computational effort of the MPC using the dynamic vehicle model, while driving a constant radius circle.

Chapter 6

Discussion

The results of the MPC implementations are presented in Chapter 5 and will be discussed in this chapter. The implementations are tested using the four-wheel model presented in Section 4.3, using the designated parameters for Atmos Driverless. The experiments conducted, and that are the basis of the discussion are listed in Table 6.1.

Track	Initial State	Prediction Model
Straight-Line	Initial cross-track error	Kinematic bicycle model
Straight-Line	Initial cross-track error and initial heading error	Kinematic bicycle model
Straight-Line	Initial cross-track error	Dynamic bicycle model
Straight-Line	Initial cross-track error and initial heading error	Dynamic bicycle model
Constant Radius Circle	Initial cross-track error	Kinematic bicycle model
Constant Radius Circle	Initial cross-track error	Dynamic bicycle model
Formula Student inspired track	-	Kinematic bicycle model
Formula Student inspired track	-	Dynamic bicycle model

Table 6.1: Experiments including track, initial state and the used prediction model.

For the constant radius cornering path and the Formula Student inspired track, the suddenly varying steering angle rate at the end of the reference spline is due to an unclosed spline.

A discussion about the performance result will be presented in Section 6.1, and a discussion about the computational effort results will be conducted in Section 6.2. The experiments conducted for testing the computational effort are listed in Table 6.2.

Track	Prediction Model
Straight-Line	Kinematic bicycle model
Straight-Line	Dynamic bicycle model
Constant Radius Circle	Kinematic bicycle model
Constant Radius Circle	Dynamic bicycle model

Table 6.2: Experiments for testing the computational effort including track and the used prediction model.

6.1 Performance

In Section 5.1 the performance results of the MPC with both a kinematic and a dynamic vehicle model are presented. Throughout the experiments conducted, both MPC methods have shown results of different levels.

Straight-Line Experiment

The straight-line experiments show that the MPC implementation with both the kinematic and the dynamic vehicle model, controls the vehicle to the given centerline. Figure 5.3 shows the error states during the experiments, and it is seen that the MPC implementation using the kinematic vehicle model controls the vehicle to both zero cross-track error and zero heading error within a short amount of time. The expected behavior is also seen where the heading error increases rapidly in the beginning and slowly increases as the vehicle closes up on the centerline. In Figure 5.4 and Figure 5.5, it is seen that the MPC states and the control input stay safely within the given constraints and act stable. When including an initial heading error, the response is still quite fast, as seen in Figure 5.6, where the error states are presented. In this case, the steering angle rate is limited by the given constraint, as seen in Figure 5.5. However, the state response, seen in Figure 5.7, does not have any undesired behavior as a result of the constrained control input.

For the straight-line experiment with an initial cross-track error, the error response and the state response of the MPC using the dynamic vehicle model is similar to the response of the MPC using the kinematic vehicle model. However, the response is somewhat slower than the response when using the kinematic vehicle model. Additionally, the control input, seen in Figure 5.12, changes quite rapidly, which may be a sign of bad tuning. The error response and the state response of the MPC implementation using the dynamic model are seen in Figure 5.10 and Figure 5.11, respectively. Similar response as for only initial cross-track error is seen when having both an initial cross-track error and an initial heading error. In Figure 5.13, it is seen that the response is somewhat slower when having both initial cross-track error and heading error, than only having an initial cross-track error. The same rapid change in steering angle is also seen for the experiment with both initial cross-track error and initial heading error, seen in Figure 5.15.

Comparing the straight-line results of the two implementations of the MPC methods and the feedback linearization result, presented in Appendix E, the MPC method shows greater results than the feedback linearization result. To tune the feedback linearization controller

to be critically damped is a non-trivial and highly time-consuming task. With the current tuning of the feedback linearization controller, the system response is under damped, in addition to having a constant deviation of approximately 0.3m. This response is seen in Figure E.1. The steering angle is also rapidly shifting from side to side, which may damage the control actuator.

Constant Radius Circle Experiment

The constant radius experiment is conducted with different velocities. For velocities higher than 10m/s the QP problem of the MPC implementation using the dynamic vehicle model was unfeasible. Even with lower velocities, the performance is somewhat poorer than for the MPC implementation using the kinematic vehicle model. The implementation using the kinematic vehicle model has the highest cross-track error at approximately 0.1m after correcting for the initial cross-track error, while the implementation with the dynamic vehicle model gets a cross-track error of approximately 0.2m after correcting for the initial error. This can be seen in Figure 5.18 and Figure 5.22 for the MPC method using the kinematic vehicle model and the dynamic vehicle model, respectively. Additionally, the cross-track error for the MPC with the kinematic vehicle model seems to stabilize at zero cross-track error, while the MPC with the dynamic vehicle model overshoots slightly.

The control input for the MPC using the dynamic vehicle model, seen in Figure 5.23, is a little too aggressive, which is visible in the heading error plot in Figure 5.22. The heading of the vehicle is rapidly changing and does not have the expected behavior, which can be seen in Figure 5.18 where the kinematic vehicle model is used. Additionally, small oscillations are visible in the control input figure, which is due to poor tuning. In this formulation, the control input is staying at its limit of steering angle rate until the desired steering wheel angle is met, as seen in Figure 5.20. This discussion reflects the state response displayed in Figure 5.19 and Figure 5.24, for the kinematic formulation and the dynamic formulation, respectively.

For the constant radius circle experiment, the feedback linearization does not show acceptable results. The cross-track error seen in Figure E.2, shows an under damped response, in addition to stabilizing at a cross-track error of 3m. The steering angle response, in Figure E.2, shows also an undesired behavior, which may not be safe for the vehicle, and an uncontrollable behavior may occur.

Formula Student Track Experiments

The implementation of the MPC using the kinematic formulation performs near to perfect. The MPC implementation stays within all constraints, as seen in Figure 5.27, Figure 5.28 and Figure 5.29, that present the error states, MPC states, and the control input, respectively. The MPC method manages to track curves with changing curvature, and maximum cross-track error while tracking is approximately 0.6m, which will be sufficient for a Formula Student dynamic event.

The experiment using the MPC implementation with the dynamic vehicle model is not performing as well as the MPC implementation using the kinematic vehicle model. Figure

5.31, Figure 5.32, and Figure 5.33 display the resulting error states, the control input, and the MPC states, respectively. As seen in Figure 5.31, the method does not manage to keep the vehicle within the given cross-track error constraint, even though the velocity is decreased to 5m/s. The poorer results may be because it is tested at higher velocities, which results in a higher acceleration than the method and the vehicle can operate at.

Even though, the method using the dynamic vehicle model does not show the best results in the simulations it has several features that may increase the performance in a real-world application. Features like limitations of the lateral velocity and yaw rate. In Section 2.2, it is mentioned that the bicycle model is only a valid vehicle model for an MPC method when the lateral acceleration is constrained.

The experiments conducted reveals some shortcomings when it comes to the used simulator. Figure 5.26 shows the simulation results when using the kinematic vehicle model in the MPC method. During the simulation the constant longitudinal velocity of the vehicle is $u = 17$m/s. The given track has a 3m radius corner, which results in a lateral acceleration of

$$a_y = \frac{(17m/s)^2}{3m} = 96.3333\frac{m}{s^2}. \tag{6.1}$$

In Appendix B, it is concluded that the nonlinear behavior of the vehicle is present already at approximately $a_y = 10\text{m/s}^2$. This behavior is not as clearly present in the simulations, as it would be in the real world.

6.2 Computational Effort

Section 5.2 presents the computational effort results for the MPC implementation using both the kinematic vehicle model and the dynamic vehicle model.

Table 6.3 lists all the key values,

- maximum computational effort,
- minimum computational effort, and
- average computational effort,

for all experiments, including the experiments for the feedback linearization controller. The key values are derived from the result figures in Section 5.2, thus Figure 5.34, Figure 5.35, Figure 5.36, and Figure 5.37. The computational effort results from the feedback linearization controller displayed in Appendix E, Figure E.3, and Figure E.4, are also included.

Controller/Event	Max Time [s]	Min Time [s]	Average Time [s]
MPC using kinematic model/ Straight-Line	0.0089	2.8000e-04	3.9090e-04
MPC using kinematic model/ Constant Radius Cornering	0.0081	3.0900e-04	5.0793e-04
MPC using dynamic model/ Straight-Line	0.0172	0.0039	0.0050
MPC using dynamic model/ Constant Radius Cornering	0.0116	0.0036	0.0045
Feedback Linearization/ Straight-Line	0.0134	0.0035	0.0046
Feedback Linearization/ Constant Radius Cornering	0.0158	0.0036	0.0047

Table 6.3: Computational Effort Results

The experiments reveal that the MPC method using the kinematic vehicle model has the lowest computational efforts. When comparing the two MPC methods, the method using the dynamic vehicle model requires a longer prediction horizon, than the method using the kinematic vehicle model. Using a longer prediction horizon, results in a bigger optimization problem, as the sequential formulation defines the size of the QP problem as $u \in \mathbb{R}^{n \times N}$, where u is the minimization variable, n is the number of minimization variables, and N is the prediction horizon.

The key values for the MPC method using the dynamic vehicle model and the feedback linearization method are quite similar, and they have a quite high computational effort. The feedback linearization controller includes several nonlinear elements which are computationally heavy and is one of the reasons of the high computational effort for this controller.

Chapter 7

Epilogue

The goal of this thesis is to design and implement a control regime for lateral tracking of a given path, performing better than the existing feedback linearization controller. The new regime should also reach the goals presented in Chapter 1. To improve the lateral tracking, two MPC methods, one using the kinematic vehicle mode, and one using the more complex dynamic vehicle model, are implemented. The methods are implemented in MATLAB using the industrial QP problem solver, *qpOASES*. The conclusion is presented in Section 7.1. Improvements can be made for both the MPC implementations and the simulation environments. These are presented in Section 7.2.

7.1 Conclusion

Several experiments are conducted, including static and dynamic curvature. The experiments are conducted for testing the functionality and performance of the implemented MPC methods. The MPC method using the kinematic vehicle model achieves great results of tracking the given path. For the straight-line experiment, the vehicle is controlled smoothly on the line, and follows it, even with initial cross-track and heading errors. For the constant radius experiment, the vehicle deviates somewhat from the line. However, it is within the given constraint of 0.8m. For the Formula Student inspired track, the method is following the given path, even during high-velocity cornering.

The MPC method using the dynamic vehicle model shows varieties of results. For the straight-line experiment, the method follows the given line with both initial cross-track error and heading error. For the constant radius circle, the method does not manage to follow the centerline at the same velocities as the MPC method using the kinematic vehicle model. However, at 10m/s, the method follows the circular path, and the vehicle is within its constraints. The MPC method has some difficulties staying within the given constraints for the varying curvature of the Formula Student inspired track, even at low velocities.

The computational effort of the implementations is in favor of the MPC method using the kinematic vehicle model. This method has lower computational effort than the method using the dynamic vehicle model due to a smaller model, and also shorter prediction horizon. The low computational effort is beneficial for real-time execution.

The MPC method using the kinematic vehicle model should be prioritized when testing the lateral behavior of the autonomous vehicle. This MPC method out-performs the feedback linearization controller in simulations, both in performance and computational effort results. Also, the results show signs of reaching the given goals, by competing the Formula Student inspired track at an average velocity of 17m/s.

7.2 Further Work

The work conducted throughout this thesis consists of two working implementations of the MPC method. However, as mentioned, the simulation environment is far from realistic. Therefore, it is important to analyze the environment and find solutions on how to improve it. An example of this is to include the Runge-Kutta method for integrating the states in the simulation environment [29], and not the simple Euler discretization, described in Chapter 3.3. With a more realistic simulation environment, aspects of the MPC method can be tested more thoroughly.

The MPC implementation using the dynamic vehicle model shows signs of being poorly tuned. The method should go through a new tuning process, and also tested more in simulations. There are beneficial features included in this implementation, as mentioned in Section 6.1. The possibility of limiting the lateral velocity or acceleration should be explored. Therefore, the method should be thoroughly tested using the autonomous race car to unveil its potential.

Several features can be implemented for improving the performance of the method, and are presented in the following sections. The method should be code generated before testing in a real-time environment. Some final tuning will also be necessary.

7.2.1 New Features

Slack Variables

There may be situations where the vehicle states are outside its given constraints. In these cases, it is important to have a safe way of returning to the legal area. In optimization, a useful tool for this problem is the introduction of slack variables [18].

Linearization

Throughout the report, the small-angle approximation has been used, which may not be a valid assumption during high-velocity cornering or small radius cornering. Therefore, it may be useful to use the method described in Section 3.2, and linearize functions included in the state-space models around the current operating point for every iteration, instead of at zero.

Changing Curvature over Horizon

When driving a track with varying curvature at high speed, the assumption about constant curvature is less valid. Therefore, including changing curvature over the entire Horizon may improve the accuracy of the MPC.

Include Velocity

The current MPC implementation facilitates LPV system, which is currently not utilized. It is a common understanding that the steering angle is highly dependent on the velocity, hence, including the change of longitudinal velocity over the entire horizon will most likely improve the performance.

7.2.2 Vehicle Implementation

The current implementation of the MPC does have some last requirements before it can run safely on an autonomous vehicle. The following must be done,

- code generation, and
- testing and tuning.

C-Code Generation

The MPC is currently implemented in MATLAB. However, this may not be the optimal solution when running the implementation in real-time. Code programmed in the C language generally executes faster than the same code programmed in the MATLAB language. This is one of the key factors for satisfactory results when driving in real-time. Additionally, the pipeline, described in Appendix A, is implemented in C++ using Robot Operating System (ROS)[1]. Therefore, it is beneficial to code generate the MPC implementation to C code to embed it into the existing pipeline for testing the systems altogether.

Testing & Tuning

When the code is embedded into the processing unit, that is the brain of the vehicle, the MPC implementation must be tested and tuned online. The tuning described in Section 4.1, is a preliminary tuning, and it is not a given that this will be sufficient when running the implementation online with a driving vehicle. The simulation environment is not complex enough for capturing all uncertainties in the real world.

[1] For further information about ROS, see https://www.ros.org/

Bibliography

[1] Formula student rules 2020. [rev. 1.0], 2019. URL https://www.formulastudent.de/fileadmin/user_upload/all/2020/rules/FS-Rules_2020_V1.0.pdf.

[2] Revolve NTNU. Revolve ntnu - about us. URL https://www.revolve.no/about-us/.

[3] C. S. Borg. Control system for a driverless formula student race car. 2020.

[4] Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley, 2011.

[5] Rajesh Rajamani. *Vehicle Dynamics and Control*. Springer, 2006.

[6] Philip Polack, Florent Altché, Brigitte Novel, and Arnaud de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? pages 812–818, 06 2017. doi: 10.1109/IVS.2017.7995816.

[7] W. F. Milliken and D. L. Milliken. *Race Car Vehicle Dynamics*, volume 400. Society of Automotive Enineers Warrendale, Inc, 1995.

[8] H. B. Pacejka. *Tire and Vehicle Dynamics*. Elsevier Butterworth Heinemann, 2005.

[9] Mathworks. Tire-road interaction (magic formula). URL https://se.mathworks.com/help/physmod/sdl/ref/tireroadinteractionmagicformula.html.

[10] M. Meywerk. *Vehicle Dynamics*. Automotive Series. Wiley, 2015. ISBN 9781118971369. URL https://books.google.no/books?id=qYG4CAAAQBAJ.

[11] Marcus Engebretsen. Revolve wiki, vehicle dynamics and control system/reactive control system. internal document. trondheim, no: Revolve ntnu. 2019.

[12] S. N. Midtskogen. Trajectory following for formula student driverless vehicle. august 2018.

[13] E. F. Camacho and C. Bordons. *Model Predictive Control*. Springer London, 2007.

[14] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099, 2015.

[15] Ugo Rosolia, Ashwin Carvalho, and Francesco Borrelli. Autonomous racing using learning model predictive control, 2016.

[16] Alexander Liniger. Path planning and control for autonomous racing. 2018.

[17] Jan Filip. *Trajectory Tracking for Autonomous Vehicles*. PhD thesis, 05 2018.

[18] J. Nocedal and S. Wright. *Numerical Optimization*. Springer New York, 2006.

[19] Chi-Tsong Chen. *Linear System Theory and Design*. Oxford University Press, 2013.

[20] Bjarne Foss and Tor Aksel N. Heirung. *Merging Optimization and Control*. Department of Engineering Cybernetics Norwegian University of Science and Technology — NTNU, 2016.

[21] Tor A. Johansen. Introduction to nonlinear model predictive control and moving horizon estimation, chapter 1.

[22] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

[23] Alexander Domahidi and Juan Jerez. Forces professional. Embotech AG, url=https://embotech.com/FORCES-Pro, 2014–2019.

[24] Gianluca Frison and Moritz Diehl. Hpipm: a high-performance quadratic programming framework for model predictive control. *ArXiv*, abs/2003.02547, 2020.

[25] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 2020. doi: 10.1007/s12532-020-00179-2. URL `https://doi.org/10.1007/s12532-020-00179-2`.

[26] MATLAB. *9.7.0.1190202 (R2019b)*. The MathWorks Inc., Natick, Massachusetts, 2019.

[27] Inc. The MathWorks. *Control System Toolbox*. Natick, Massachusetts, United State, 2019. URL `https://se.mathworks.com/help/control/`.

[28] Inc. The MathWorks. *Curve Fitting Toolbox*. Natick, Massachusetts, United State, 2019. URL `https://se.mathworks.com/help/control/`.

[29] Rolf Johansson. Modeling and simulation for automatic control, olav egeland, jan tommy gravdahl marine cybernetics, trondheim, norway, 2002, 1st edition, 656pp. isbn 82-92356-01-0. *International Journal of Robust and Nonlinear Control*, 14 (7):683–684, 2004. doi: 10.1002/rnc.915. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rnc.915.

[30] LLC VectorNav Technologies. Vn-300 user manual, 2017. URL https://www.vectornav.com/docs/default-source/documentation/vn-300-documentation/vn-300-user-manual-(um005).pdf?sfvrsn=c7f6e7b9_26.

[31] Ouster os1. URL https://ouster.com/products/os1-lidar-sensor.

[32] Basler. aca1300-200uc camera. URL https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca1300-200uc/.

[33] Adrian Skibelid. Odometry, mapping and localisation of an autonomous race car for revolve ntnu. august 2019.

[34] Christoph Sprunk. Planning motion trajectories for mobile robots using splines. 2008.

Appendix A

Introduction to Atmos Driverless

This chapter is based on Chapter 1 from the specialization project associated with this thesis. Modifications have been made according to changes during the latter part of the project.

Atmos Driverless is Revolve NTNU's driverless vehicle. The vehicle was designed and manufactured by Team 2018 of Revolve NTNU and competed in the electric vehicle class the respective summer before it was reconfigured to a driverless vehicle by Team 2019.

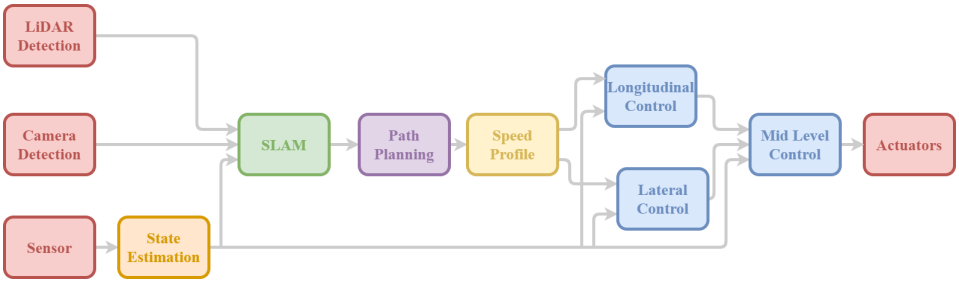


Figure A.1: Overview of the autonomous pipeline

To reconfigure a manned vehicle to an autonomous vehicle, significant changes are required for both the hardware and the software aspect of the vehicle. When removing the driver, one is forced to create a fully autonomous pipeline, including visual perception, in addition to an advanced control system. Figure A.1 shows a simplified overview of the complex pipeline of the autonomous vehicle, Atmos Driverless. The modules before the control system modules will be described briefly in the upcoming sections.

A.1 Sensors

Sensors mounted to the vehicle are the only method for capturing the different states of the vehicle, which is crucial for an autonomous race car. For observing the states of the vehicle there are four types of sensors mounted to the vehicle,

- a Global Navigation Satellite System (GNSS)-Aided Inertial Navigation System (INS),
- an optical sensor,
- a Light Imaging, Detection and Ranging (LIDAR) sensor, and
- a camera.

As the GNSS/INS sensor, a VN-300 [30] is mounted to the vehicle. The VN-300 outputs the linear acceleration and the angular velocities in three dimensions, in addition to the heading angle and the position data from the GNSS.

It is highly important to know the angular velocity of the four wheels, in addition to the steering wheel angle. To measure these states, an optical sensor is attached to the wheels.

To perceive the world, both LIDAR's and cameras are used in conjunction as a visual sensor. The objective of the visual sensors is to detect the cones that define the track boundaries, in addition to work as safety sensors for detecting hazardous situations where objects suddenly enter the track.

The LIDAR sends out a grid of light and uses the time it takes for the light to return to measure distances. Two LIDARs are placed on the vehicle for redundancy and a better field of view. The rotation of the LIDARs results in a 360° field of view. The onboard LIDARs are two Ouster O1 [31] with 16 and 64 channels, meaning it sends 16 or 64 arrays of light simultaneously as the LIDARs rotates.

For the same reasons as for the LIDARs, two cameras are mounted to the vehicle. The cameras are two Basler Ace cameras [32] and are placed at the front of the vehicle. The cameras are used both for detecting cones and estimating the distances between the vehicle and the cones.

The cone representation from the LIDARs and the cameras are passed to the Simultaneous Localization and Mapping (SLAM) module. Figure A.2 shows the LIDAR and camera detection.

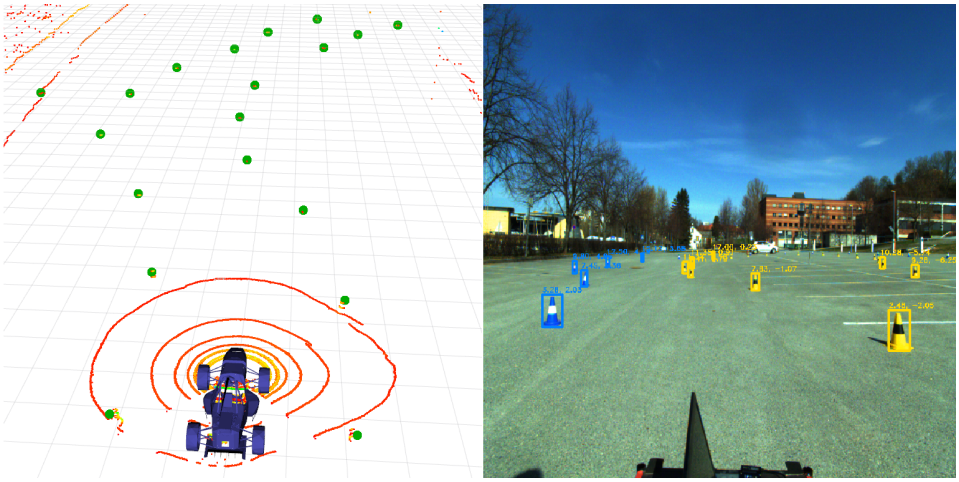


Figure A.2: Visualization of LiDAR detection to the left, where the red circles represent the light channels, all visualized using Rviz. The camera detection is visualized to the right, where the cones are marked using the camera detection algorithm.

A.2 State Estimation

The desired states not available for the sensors are estimated in the state estimation module. In addition to estimating new states of the vehicle, state estimation works as a safety check for validating the sensor signals and making sure the rest of the modules are using valid and correct sensor values for their tasks.

One of the main objectives of the state estimation module is odometry. The odometry utilizes all the sensor data for estimating the change in pose over time, which can further be exploited to keep track of where the vehicle is in the world frame [33].

A.3 Visual Perception

By using the aforementioned outputs from the state estimation module both the position of the cones in the world frame and the odometry can be estimated [33]. SLAM is deployed for creating a map of the world with the cones and places the vehicle in this map using the odometry. Figure A.3 visualize the odometry and the cone placement from the SLAM module.

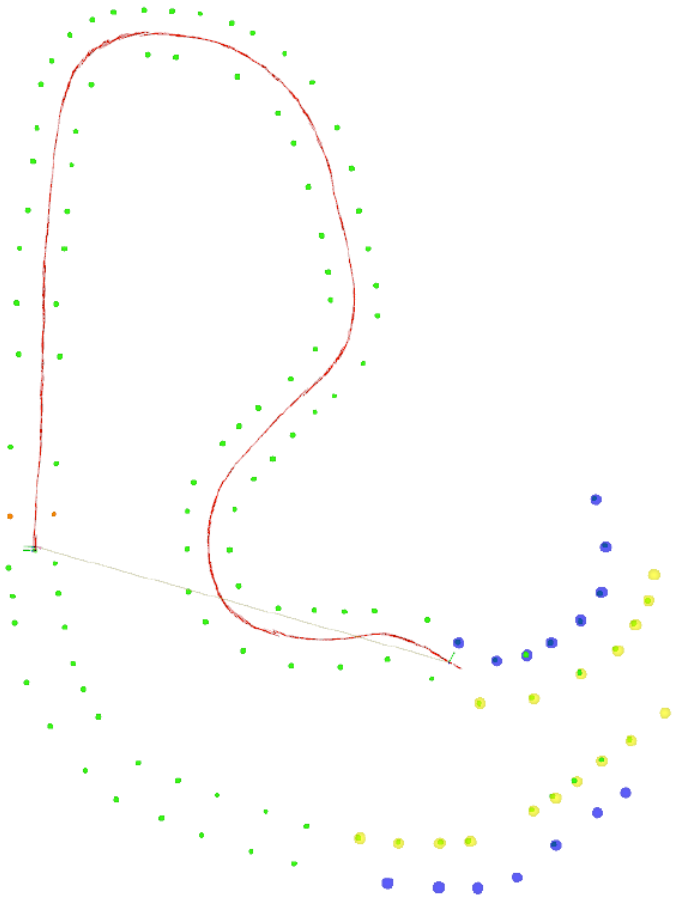


Figure A.3: Visualization of the estimated position indicated by the red line. The green circles are the predicted location of the cones and the yellow and blue circles are incoming cones. The visualization is from a simulation of the FSG 2018 race track using Rviz.

A.4 Path Planning

The path planning module is using the mapping from SLAM to find the most likely representation of the centerline of the track and passes this as a smooth input to both the speed profile and the control system.

A particle filter algorithm is used for estimating the centerline during the first round, as this is the only round with an unknown path. Hence when driving the *track drive* event, the particle filter will be planning during the first round and then optimizing the path for the rest of the race. Figure A.4 visualize the particle filter where the red lines have the highest probability of being the centerline and the green lines have a lower probability of being the centerline.

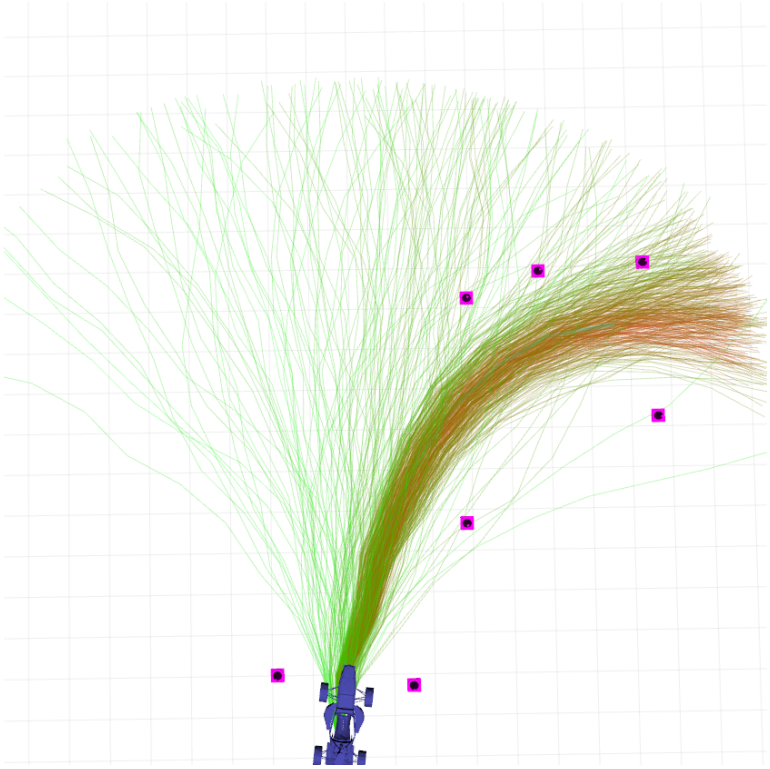


Figure A.4: Visualization of the particle filter using Rviz.

A.5 Speed Profile

The speed profile module aims to find the target velocity based on the curvature of the path and the kinematics of the vehicle. Using the forward-backward consistency algorithm described in [34] as the speed profile, a feasible velocity and acceleration can be computed along the given centerline. This algorithm allows one to constrain the acceleration to a reasonable value, regarding the demands for the current event.

Appendix B

Nonlinear Region Analysis

The lateral nonlinear behavior of the vehicle may be defined as when the slip angle of the rear wheels is significantly larger than the slip angle of the front wheels. The region where this behavior occurs is dependent on the vehicle design and is different from vehicle to vehicle.

Before Atmos driverless was reconstructed to a driverless vehicle, it was a highly functional vehicle driven by a driver. Hence, to estimate the nonlinear region of Atmos driverless data from when the vehicle was manned is utilized.

The lateral region is dependent on the velocity of the vehicle and the radius of the corner. It is known that the lateral acceleration is defined by these two variables as

$$a_y = \frac{u^2}{R}, \tag{B.1}$$

where a_y is the lateral acceleration, u is the longitudinal velocity and R is the radius of the current corner.

Figure B.1 displays the variables during a competition when Atmos driverless was a manned vehicle. For the current position, it is seen by the steering wheel angle graph that the driver is counter-steering for keeping the desired path. Simultaneously, the longitudinal velocity is low and the absolute value of the lateral acceleration is high. This may indicate that the vehicle has entered the nonlinear region. Hence, the nonlinear region is in the region where the lateral acceleration is at approximately $a_y = |-10|\text{m/s}^2$.

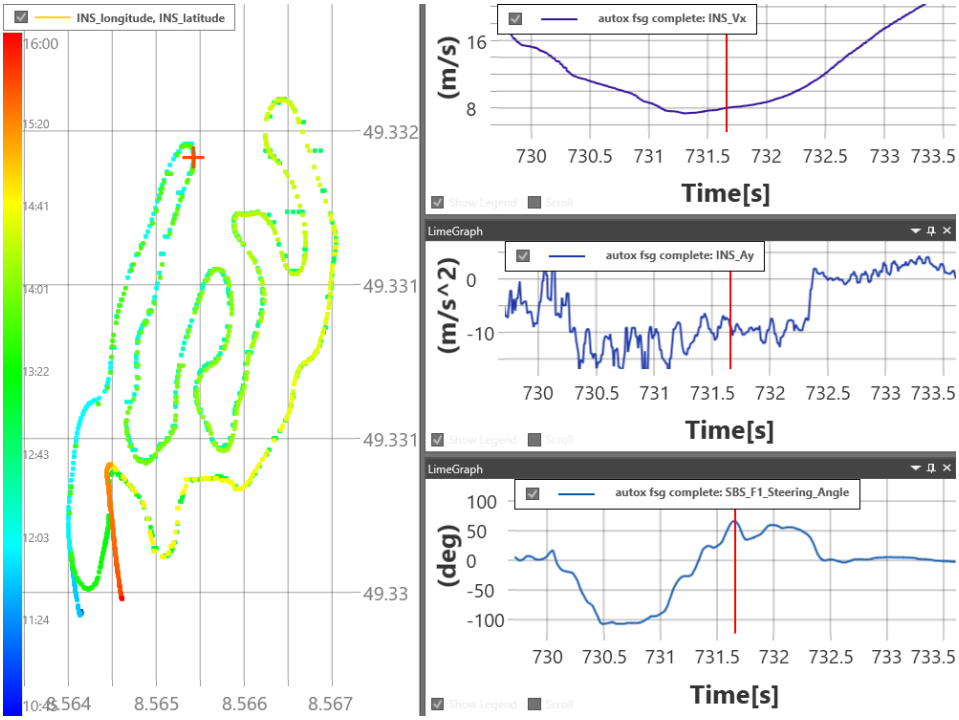


Figure B.1: Vehicle data from Autocross run during the Formula Student competition at Hockenheim. August 2018. Left plot displays the path of the track, using GNSS, where the red cross is the current position of the vehicle. The upper right plot shows the longitudinal velocity of the vehicle. The middle plot shows the lateral acceleration of the vehicle, and the bottom graph shows the steering wheel angle. The red line represents the current position in time.

Appendix C

Tuning Process

The tuning process of the MPC has been conducted using a straight-line reference path, driving at a velocity of 15m/s. The same process has been conducted for both the kinematic problem formulation and the dynamic problem formulation.

C.1 Prediction Horizon

The experiment conducted for deciding the prediction horizon is to let the vehicle follow a straight-line without initial errors. The initial weighting matrices for tuning the prediction horizon are chosen approximately similar to the magnitude of the error states, hence the weighting matrices result in

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}, \quad R = 1, \tag{C.1}$$

for both problem formulations.

When choosing a too short or too long prediction horizon the system results in unstable behavior. This behavior is displayed in Figure C.1 and Figure C.2 for the kinematic problem formulation.

The error response with an appropriate prediction horizon is deployed in Figure C.3 for the kinematic vehicle model and in Figure C.4 for the dynamic vehicle model.

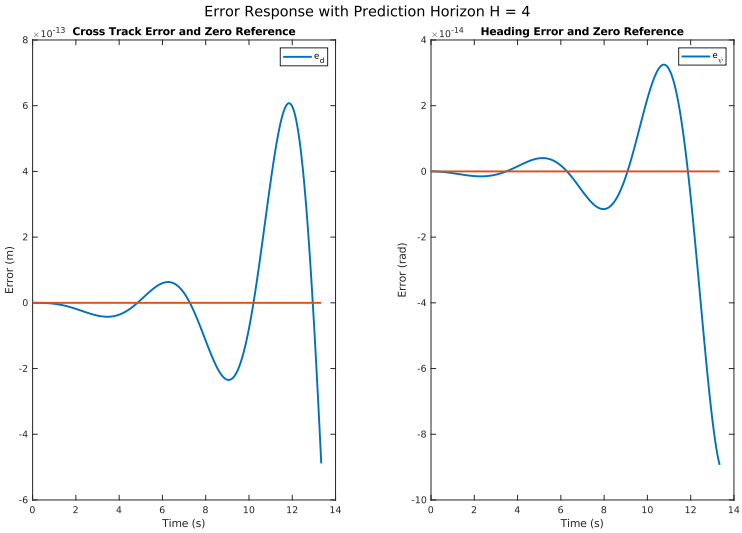


Figure C.1: Straight-line driving using the kinematic problem formulation with too short prediction horizon.

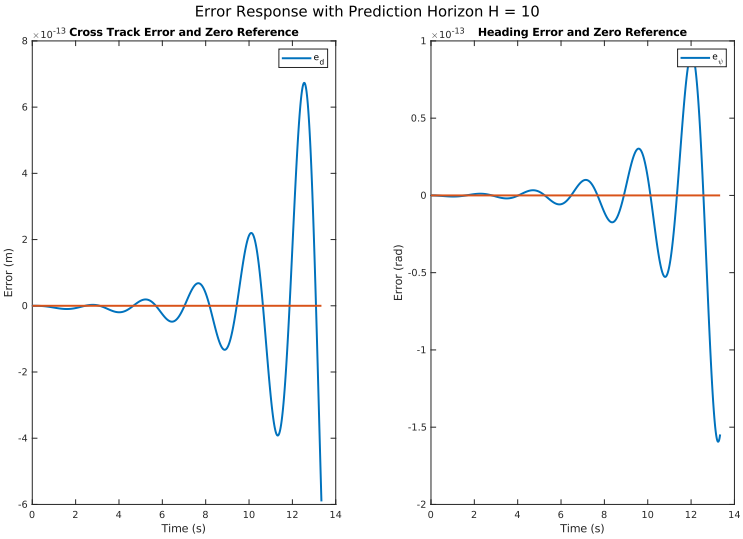


Figure C.2: Straight-line driving using the kinematic problem formulation with too large prediction horizon.

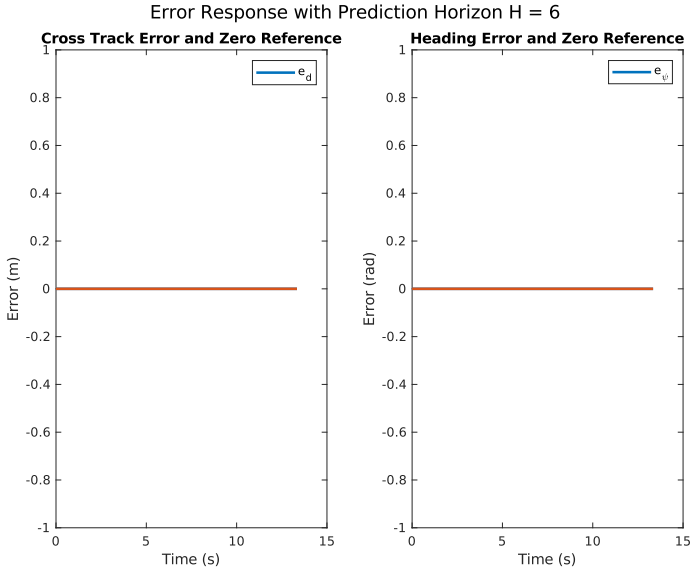


Figure C.3: Straight-line driving using the kinematic problem formulation with an appropriate prediction horizon.

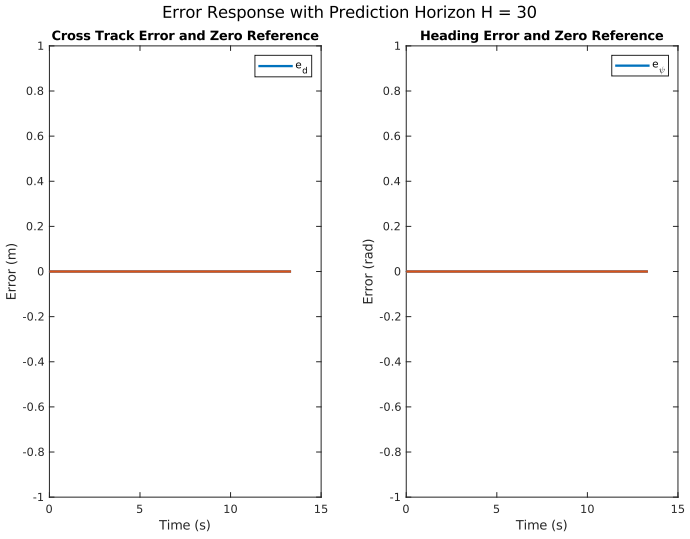


Figure C.4: Straight-line driving using the dynamic problem formulation with an appropriate prediction horizon.

C.2 Weighting Matrices

The experiment conducted for tuning the weighting matrices is to let the vehicle try to track a straight-line path with an initial cross-track error of 0.6m. This is to provoke a steering angle. With the weighting in Equation C.1, the vehicle has an undesirable unstable response leading to infeasible solutions that violate the constraints, both for the kinematic and the dynamic vehicle model, as seen in Figure C.5 and Figure C.6, respectively.

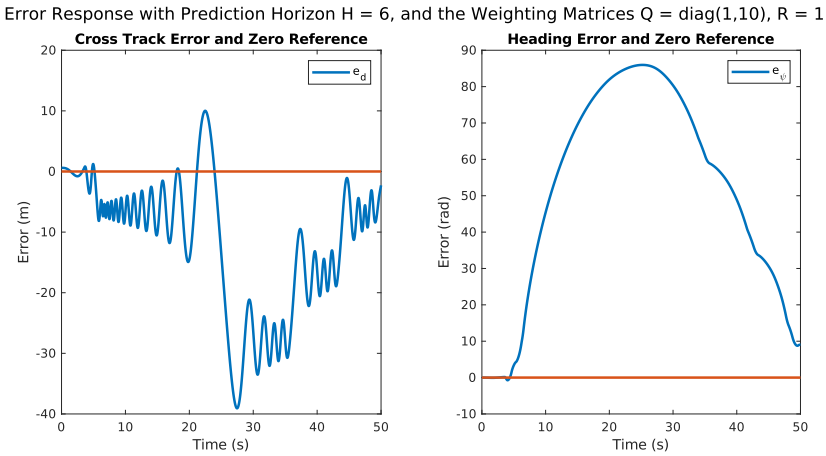


Figure C.5: Error state response for the kinematic vehicle model with the initial weighting matrices in Equation C.1.

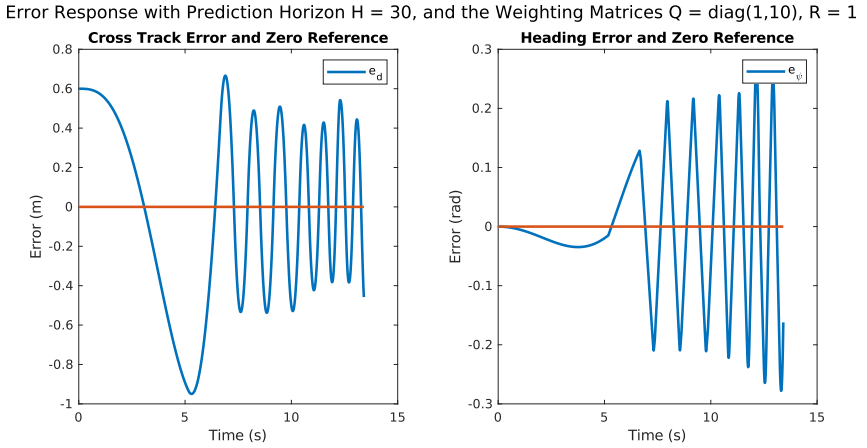


Figure C.6: Error state response for the dynamic vehicle model with the initial weighting matrices in Equation C.1.

When reducing the control input weight significantly, $R = 0.01$, for the MPC method using the kinematic vehicle model, the behavior stabilizes at the desired zero references with a quite smooth approach towards the reference. The response with a reduced control input weight is displayed in Figure C.7.

To decrease the response time, the cross-track error may be weighted higher, as seen in Figure C.8. However, the response is still not acceptable as the response is under damped. By also increasing the weight on the heading error, the system response gets critically damped, as seen in Figure C.9. The critically damped behavior is the desired behavior and the weighting matrices used for the critically damped system are

$$\mathbf{Q} = \begin{bmatrix} 5 & 0 \\ 0 & 30 \end{bmatrix}, \quad R = 0.01. \tag{C.2}$$

The tuning is also applicable to the dynamic model formulation as displayed in Figure C.10.

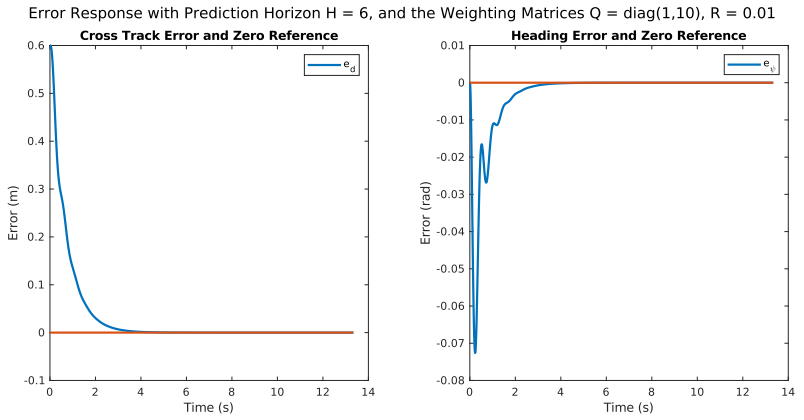


Figure C.7: Error state response for the kinematic vehicle model with decreased weight on the control input.

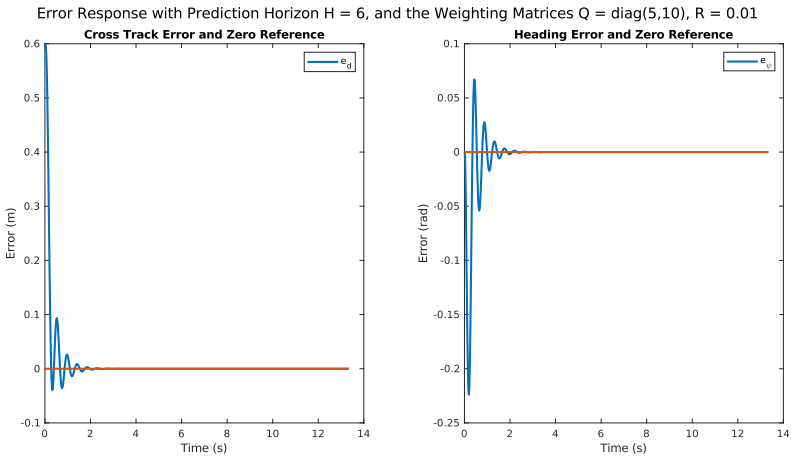


Figure C.8: Error state response for the kinematic vehicle model with increased decreased weight on the cross-track error, $Q(1,1) = 5$.

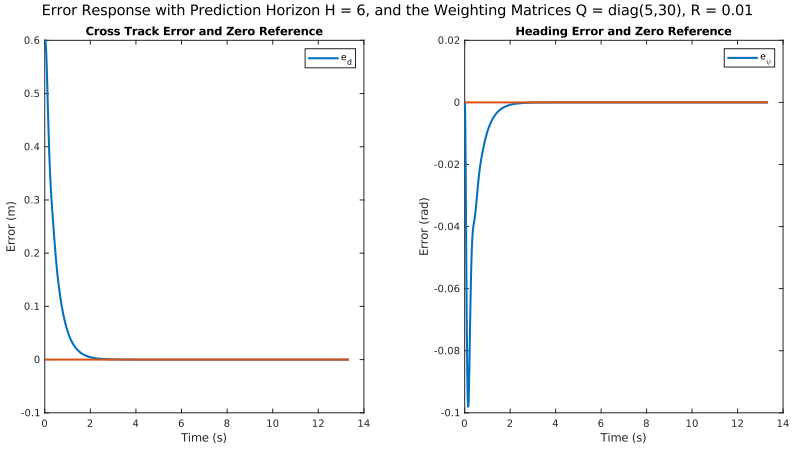


Figure C.9: Error state response for the kinematic vehicle model with increased decreased weight on the heading error, $Q(2,2) = 30$.

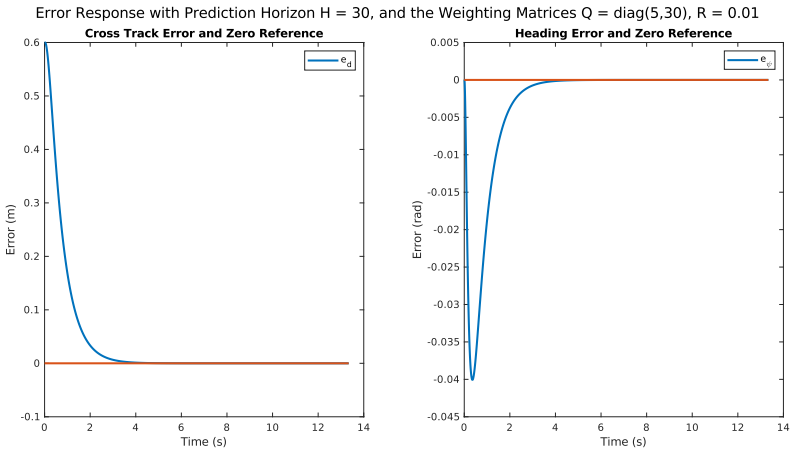


Figure C.10: Error state response for the dynamic vehicle model with the initial weighting matrices in Equation C.2.

Appendix D

Code

D.1 Sequential Reformulation

```
function [H_hat, g_hat, A_constraints, lbA, ubA] =
    sequential_reformulation(A, B, C, E, Q, R, m, n, H, x_n, d_bar, x_lb,
    x_ub)

    A_hat = A;
    A_hat((m + 1):2*m, 1:m) = A*A;
    AB = B;
    AB(m + 1:2 * m,1:n) = A*B;
    AE = E;
    AE(m + 1:2 * m,1:n) = A*E;

    Q_hat = C'*Q*C;

    A_exp = A;
    for i = m+1:m:H*m
        AB(i:(i + m - 1),1:n) = A_exp*B;
        AE(i:(i + m - 1),1:n) = A_exp*E;
        A_exp = A_exp*A;
        A_hat(i:(i + m - 1), 1:m) = A_exp;
    end

    counter = 1;
    for i = 1:n:n*H
        k = 1;
        for j = counter:m:H*m
            B_hat(j:(j + m - 1),i:(i+n-1)) = AB(k:(k + m - 1), 1:n);
            E_hat(j:(j + m - 1),i:(i+n-1)) = AE(k:(k + m - 1), 1:n);
            k = k + m;
        end
        counter = counter + m;
    end

```

```
31      for i = 1:m:(m*H)
32          Q_bar(i:(i+m-1),i:(i+m-1)) = Q_hat;
33          x_bar_lb(i:(i+m-1),1) = x_lb;
34          x_bar_ub(i:(i+m-1),1) = x_ub;
35      end
36
37      for i = 1:n:(n*H)
38          R_bar(i:(i+n-1),i:(i+n-1)) = R;
39      end
40
41      g_hat = B_hat'*Q_bar*[A_hat E_hat]*[x_n; d_bar];
42      H_hat = B_hat'*Q_bar*B_hat + R_bar;
43
44      A_constraints = B_hat;
45      lbA = x_bar_lb - A_hat*x_n - E_hat * d_bar;
46      ubA = x_bar_ub - A_hat*x_n - E_hat * d_bar;
47  end
```

D.2 Run qpOASES

```
1 [H_hat, g_hat, A_constraints, lbA, ubA] = sequential_reformulation(Ad, Bd,
       Cd, E, x_n, d_bar(:,i), config);
2 [DeltaU,fval,exitflag,iter] = qpOASES(H_hat,g_hat, A_constraints, -config.
      c_u, config.c_u, lbA, ubA);
```

Appendix E

Feedback Linearization Results

E.1 Performance Results

The performance results of the feedback linearization controller are presented in this section. Two experiments are conducted,

- straight-line driving with an initial cross-track error, and
- constant radius cornering with a radius of $R = 9.125$.

The simulation environment includes a four-wheel model of a vehicle, using the designated parameters for Atmos Driverless, as described in Section 4.3. The vehicle is driving at a velocity of $u = 15$m/s. The cross-track error and the resulting steering angle from the straight-line experiment is displayed in Figure E.1. For the constant radius cornering experiment, the cross-track error and the steering angle are seen in Figure E.2.

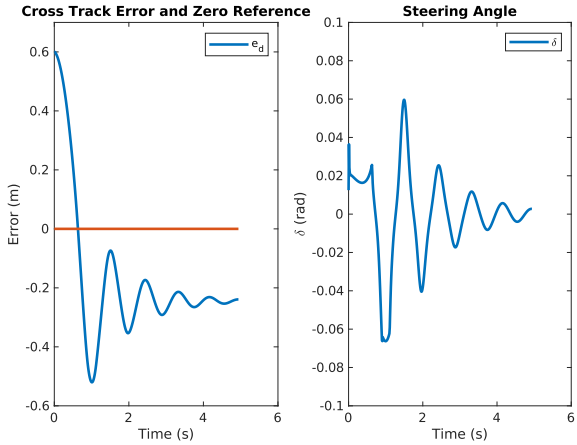


Figure E.1: Cross-track error and the resulting steering angle using the feedback linearization controller, driving a straight-line path. The initial cross-track error is 0.6m. The orange line represents the zero references.

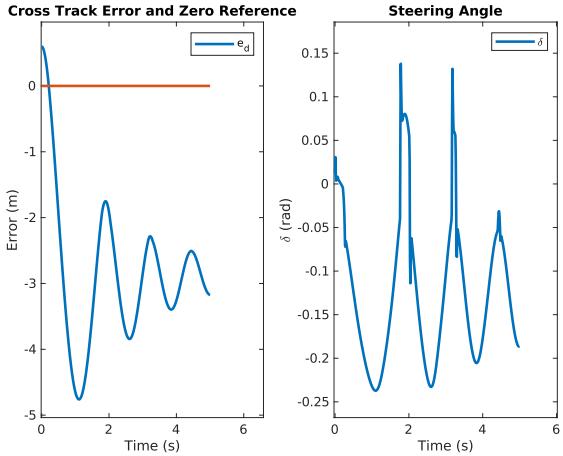


Figure E.2: Cross-track error and the resulting steering angle using the feedback linearization controller, driving a constant radius corner. The initial cross-track error is 0.6m. The orange line represents the zero references.

E.2 Computational Effort Results

To evaluate the computational effort the MATLAB function *tic* and *toc* are used. The tracking experiment of a straight-line path and the tracking experiment of a constant radius corner, with a radius of $R = 9.125$m, are conducted at a velocity of 5m/s. The three

controllers that are in cascade, are included in the timing experiment. The results of the timing of the controllers during straight-line driving are displayed in Figure E.3, while the timing results while driving a constant radius circle is shown in Figure E.4.

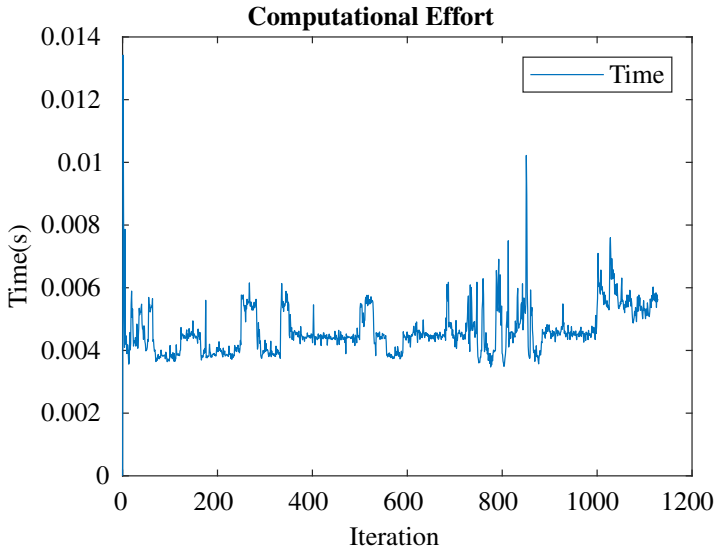


Figure E.3: The computational effort of the feedback linearization controller, while driving a straight path.

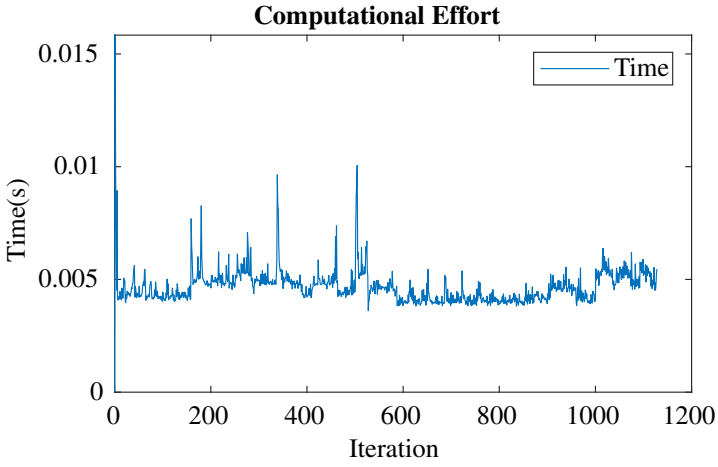


Figure E.4: The computational effort of the feedback linearization controller, while driving a constant radius circle path.

Christine Sääv Borg

Model Predictive Control for Lateral Path Tracking of an Autonomous Formula Student Race Car

NTNU
Norwegian University of
Science and Technology