Optimization Based Trajectory Planning for Autonomous Racing

Thesis · January 2018

DOI: 10.13140/RG.2.2.23680.38402

1 author:

Max Ahlberg
KTH Royal Institute of Technology
2 PUBLICATIONS **0** CITATIONS

SEE PROFILE

DEGREE PROJECT IN VEHICLE ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Optimization based trajectory planning for autonomous racing

MAX AHLBERG

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Abstract

Autonomous driving is one of the three new technologies that are disrupting the classical vehicle industry together with electrification and connectivity. All three are pieces in the puzzle to drastically reduce the number of fatalities and injuries from traffic accidents but also to reduce the total amount of cars, reduce the polluting greenhouse gases, reduce noise pollution and completely eliminate unwanted driving. For example would most people rather rest, read or do anything else instead of driving in congested traffic. It is not small steps to take and it will have to be done incrementally as many other things. Within the vehicle industry racing has always been the natural place to push the boundaries of what is possible. Here new technologies can be tested under controlled circumstances in order to faster find the best solution to a problem.

Autonomous driving is no exception, the international student competition "Formula Student" has introduced a driverless racing class and Formula E are slowly implementing Robo Race. The fact that race cars aim to drive at the limits of what is possible enable engineers to develop algorithms that can handle these conditions even in the every day life. Because even though the situations when normal passenger cars need to perform at the limits are rare, it is at these times it can save peoples lives. When an unforeseen event occurs and a fast manoeuvre has to be done in order to avoid the accident, that is when the normal car is driving at the limits. But the other thing to take into considerations when taking new technology into the consumer market is that the cars cannot cost as much as a race car. This means simpler computers has to be used and this in turn puts a constraint on the algorithms in the car. They can not be too computationally heavy.

In this thesis a controller is designed to drive as fast as possible around the track. But in contrast to existing research it is not about how much the limit of speed can be pushed but of how simple a controller can be. The controller was designed with a Model Predictive Controller (MPC) that is based on a point mass model, that resembles the Center of Gravity (CoG) of the car. A g-g diagram that describes the limits of the modeled car is used as the constraints and the cost function is to maximize the distance progressed along the track in a fix time step. Together with constraints on the track boundaries an optimization problem is giving the best possible trajectory with respect to the derived model. This trajectory is then sent to a low level controller, based on a Pure Pursuit and P controller, that is following the predicted race trajectory. Everything is done online such that implementation is possible.

This controller is then compared and evaluated to a similar successful controller from the literature but which has a more complicated model and MPC formulation. The comparison is made and some notable differences are that the point mass model is behaving similar to the more complex model from the literature. Though is the hypothesis not correct since the benefits of the simplification of the model, from bicycle to point mass model, is replaced when more complex constraints has to be set up, resulting in similar performance even in computational times.

A combination of the two models would probably yield the best result with acceptable computational times, this is left as future work to research.

Sammanfattning

Autonom körning är tillsammans med elektrifiering och uppkopplande av fordon en av tre teknologier som håller på att förändra den klassiska fordonsindustrin. Alla tre är delar i ett pussel för att drastiskt reducera antalet döda och skadade i trafiken men också för att reducera det totala antalet bilar, minska växthusgasutsläppen, mindre störande ljud och potentiellt helt kunna ta bort de timmar som man måste sitta framför en ratt utan att vilja. Det är inga små förändringar och detta kommer implementeras stegvis. Inom fordonsindustrin har racing alltid varit den naturliga arenan för att föra utvecklingen framåt. Här kan nya teknologier bli testade under kontrollerade former för att snabbare kunna hitta optimala lösningar. När det kommer till autonom körning ser det ut som att detta är fallet här också, den internationella studenttävlingen Formula Student har precis infört en klass för självkörande bilar och Formula E inför långsamt det som kallas Robo Race. De faktum att racebilar kör på gränsen av vad som är möjligt gör det möjligt för ingenjörer att utveckla algoritmer som kan hantera liknande situationer i vardagen. Även om de situationer där en vanlig bil behöver kunna köra på gränsen av vad som är möjligt är få, så är det vid dessa tillfällen som det finns möjlighet att rädda människoliv. När en oförutsädd händelse inträffar och en snabb manöver behöver utföras för att undvika en olycka, det är precis då som en vanlig bil behöver kunna köra på gränsen av vad som är möjligt. Men en annan sak att räkna in är att bilar som säljs till konsumenter inte kan kosta lika mycket som en racerbil. Detta betyder att enklare elektronik och hårdvara måste användas vilket i sin tur betyder att algoritmerna som används inte får vara för beräknings tunga.
I detta examensarbete designas en regulator som skall vara så snabb som möjligt runt en bana, men istället för att, som i mycket annan forskning, fokusera helt på att minimera varvtiden, har här valts att se hur enkel och därmed beräkningssnabb man kan göra en regulator som ändå ger acceptabla resultat. Regulatorn var designad som en Modell Prediktiv Regulator (MPC) som är baserad på en punkt-mass modell som motsvarar mass-centrum av bilen. Ett g-g diagram används för att beskriva bilens gränser på vad den kan klara och kostnadsfunktionen designas på ett sätt så att regulatorn vill ta sig så långt som möjligt på banan under en fix tid. Tillsammans med restriktioner på vart bilen får köra ger detta optimeringsproblem den optimala trajektorien. Den beräknade trajektorien är sedan skickad till lågnivåregulatorer, en Pure Pursuit för lateral styrning och en P regulator för longitudinel styrning. Allting är gjort så att det skall kunna implementeras på en riktig racerbil utan att beräkningar görs i förhand.
Den designade regulatorn är sedan jämförd med en mer komplicerad regulator som är publicerad och som har visat sig fungera. Vid jämförelsen visade det sig att båda beter sig likt i många situationer men vissa skillnader kunde observeras, som att den föreslagna hypotesen inte visade sig vara riktig. Även fast den designade regulatorn var enklare i sin modell, blev restriktionsbetingelserna svårare att hantera och därmed beräkningstiden ungefär den samma.
Att sammanfoga de två modellerna är av intresse för att hitta en bättre mer race-optimal lösning och kommer tas an i framtida arbete.

Contents

List of Tables

List of Figures

1 Introduction

1.1 Scope

As autonomous driving is evolving it is natural that an autonomous racing scene would start to grow. Racing has been, and still is, a great platform to push and develop new technologies within the vehicle industry. Racing more or less accelerate the innovation and development within a sector, since everything has to be at its physical limits to chase the victory. This is about to evolve when it comes to autonomous driving, Roborace is introduced on the professional scene and for students, Formula Student Germany has lead the way by introducing Formula Student Driverless. This is a completely new class where university teams have to compete in the exact same manner as before, in the combustion and electric class, but now without a driver. This is challenging and a great opportunity for students around the world to show their skills. KTH Formula Student is aiming at competing in the second driverless competition ever at Formula Student UK at Silverstone the 11th to 16th of July 2018. To reach this goal a fully autonomous system has to be developed from scratch. One part of that system is the trajectory planner that is going to choose where the car has to go and how it shall go there. The trajectory planner is the topic of this thesis and the goal is to compare the result of two different systems. The trajectory planner is a decision taking part of the algorithm which means that it is this part of the code that actively chooses where to go. This will have an impact on both things and people around the vehicle. That is why a model and cost function based on physical differential equations is decided to be used in favour of machine learning or AI algorithms that are a product of their learning cases. This makes it possible to track down every decision instead of having the answer coming from a black box.

Two different systems are developed to investigate how a simple model is compared to a more complex one. How does it deal with the fast phase environment it has to work within? How simple models and controllers can be used and still get a satisfying result?

To prove the logic a simulation environment will be developed and the results compared.

1.2 Problem definition

Formula Student introduced the Driverless class in the summer of 2017. In the competition format there are 6 events. The team shall present the project as a business, they shall defend their economy and cost choices, motivate their design from an engineering point of view, make the car drive 75 meters as fast as possible, drive in a circle of eight as fast as possible and finally to the only race-alike moment drive 10 laps around an unknown track. In the three dynamic events it is of great importance that the car can choose the most optimal race line in order to be as competitive as possible. A optimization problem like that is never easy to solve and requires a lot of computational power. The more computational power, the heavier the computer, which in turn results in a slower car. Developing a trajectory planning algorithm that delivers satisfactory results while still is simple to keep the computational burden down is the scope of this thesis. A multi layer controller is going to be compared with a single layer MPC controller to study the benefits of the different versions.

1.3 Previous work and contributions

To understand the history and development of the Advanced Driver Assistant Systems (ADAS) and Autonomous Driving (AD) this section is devoted to describe how a general Autonomous car is built up in both hardware and software. What the state of the art is within the Formula Student scene is presented thereafter and then the evolution of driving aids is presented trough a thorough literature review of everything from Cruise Control to Autonomous Racing.

1.3.1 General Structure of Autonomous Vehicles

The development of autonomous cars is taking place at multiple places around the world at the same time. Even though the development is spread and sometimes taking place behind closed doors is the overall layout of an autonomous system similar at all places. Systems like the cars in the DARPA Urban Challenge [1], the vehicle competing the Bertha-Benz-Memorial-Route [2] or cars from the company Waymo, whom is currently seen as the leader in the race towards fully autonomous cars, are all based on perception sensors like cameras, LiDARs and radars often placed as in Figure 1.

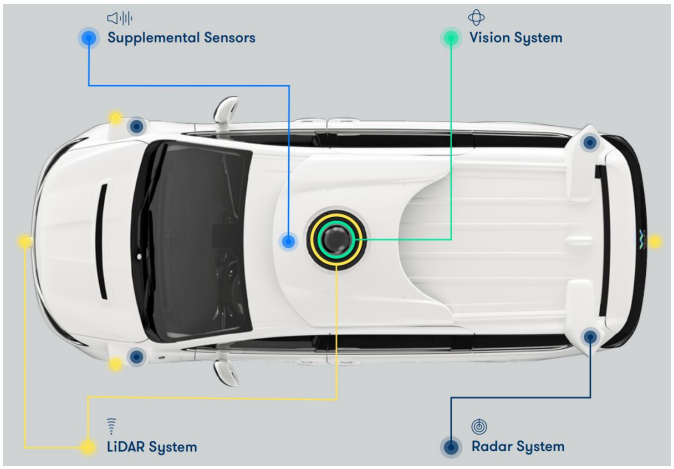

Figure 1: Autonomous vehicles perception sensors

Once the perception and recognition of objects around the car is done, the trajectory planning and actuation takes place. Deciding how to move in the surrounding space and executing the action. The new state and place the vehicle has entered after a certain time step is then recognized and updated by the sensors, closing the loop of the whole driver less system. In Figure 2 a coarse overview of a common self driving system is shown.

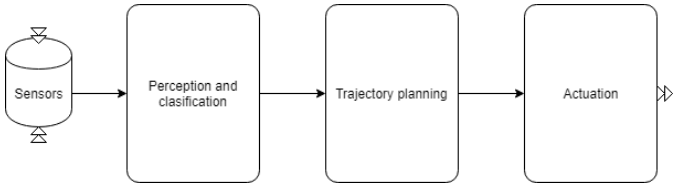

Figure 2: Self Driving car architecture

1.3.2 Formula Student

Formula Student is an international engineering design competition that was founded in 1998. In its initial years it was a competition with solely combustion engine cars. A coupe of years later the electric class was introduced and in the summer of 2017 the first Driverless competition was held at Formula Student Germany. The Formula Student class is based on a framework of rules [3] which all participating teams has to obey, independent of class. Resulting in cars looking fairly similar, the car of KTH Formula Student for the season 2018, eV14, is shown in Figure 3.

Figure 3: eV14 from KTH Formula Student

The competition is then based on six events for the Driverless participants. Cost and Manufacturing, Business Plan Presentation and Engineering Design are the static events where the car is not rolling at all. Skidpad, Acceleration and Trackdrive is the dynamic events and the most challenging of those events is the Trackdrive. In the Trackdrive shall the car be turned on out on the track without any knowledge of the track or the environment. From then it shall drive as

fast as possible 10 laps around the track. The track is bounded by blue cones on the left, yellow cones on the right and large orange cones at the start/finish line, see Figure 4.

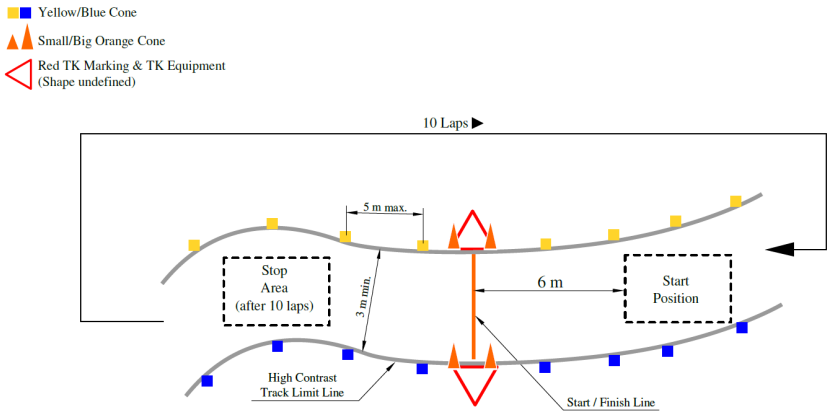

Figure 4: Trackdrive Layout

Creating something prone to succeed with such a task the students researches possible setups and designs and end up with sensors and systems much alike the industry and researchers use. The setup used by TU Wien [4], AMZ from ETH Zurich [5], KTH Formula Student [6] and other Formula Student Driverless teams also consists of multiple cameras and a LiDAR. Together with wheel speed sensors, Inertial Measurement Units (IMU) and Global Positioning System (GPS) it is assumed to be sufficient to drive around the track. With a sufficient set of sensors and a processing unit the hardware is there to be able to calculate the optimal race line, minimizing the lap times and winning the race.

1.3.3 From Cruise Control to Autonomous Racing

Cruise Control The automated driving started with the introduction of the cruise controller [7] and has been developed since. In the beginning a control system simply minimizing the difference between the actual speed of the vehicle and the reference speed set by the driver to later include more complex models as fuzzy logic controllers [8]. The cruise control system was later advanced such that the vehicle not only held a predefined speed but also adapted the speed in accordance to the vehicle in front, this is called Adaptive Cruise Control (ACC) and was first introduced on the Japanese market after achievements by Lingyun Xiao and Feng Gao [9] in 1995. Also there the development have continued by developing more sophisticated PID feedback loops, balance based adaptive controllers and even Nonlinear Model Predictive Controllers [10]. The benefit of those systems has in theory been clear, that they would increase safety and reduce the congestion on highways. But the human machine interaction might destroy the predicted benefits of these ADAS systems. For example might the increased help from the cruise control result in a more relaxed driver and therefore a driver that is not as observant about

what is going on in its surroundings. In the case of an unusual event will this state of the driver prolong the handling time and an accident might occur. These effects together with the traffic flows on highways are discussed in reports like [11] and [12]. The final conclusion is though that these systems do increase safety and high way efficiency, but would be even more efficient if the systems reached a higher level of automation. Making them able to handle non-ideal situations would reduce the need of a fallback on a driver and thus fully enable the advantages of ADAS systems or AD systems.

Lane following The ACC systems is controlling the longitudinal motion of the vehicle. The next step in the automotive industry, when it comes to ADAS systems, was the lateral positioning control. Often referred to as Lane Keeping Assist (LKA), Lane Departure Warning (LDW) or Lane Keeping System (LKS). The new dimension for the vehicles is that they now need to be able to see the colors of the road in order to detect the lane markers. Using a camera and lane segmentation algorithms like D.O. Cualain et al. in [13] the lanes can be detected and a warning can be raised to make the driver alert again. Increasing the level of autonomy by developing controllers that can steer the car within the lanes is the natural next step, as the fuzzy logic controller in [14] and the proportional controller described in [15]. Chang Mook Kang et al. [16] also presents three methods of lane following control to reduce the impact of short system failure and reduce the ripple.

Autonomous Drive The difference between having those two systems, Cruise Control and Lane following, running simultaneously and the more advanced trajectory planning algorithms is the ability of the car to choose its own way. A general purpose structure of such trajectory planner is proposed in [17]. A Model Predictive Controller (MPC) based trajectory planner in the world fixed global coordinate system is presented in [2] and successfully managed to complete the 103km Bertha-Benz-Memorial-Route. Other ways to do it, is to generate minimal jerk polynomials in the path following Frenet coordinate frame in order to generate smooth trajectories as is done in [18] for the dynamic driving case and in [19] for time critical maneuvers.
Once the trajectory is generated there has to be a low level controller following it. Different MPC based trajectory followers are presented in [20], [21], [22], [23], where the last two are using a Contouring Control. There the objective are a combination of minimizing the contour error while still maximize the progress along the curve. Another approach is taken by the team behind [24] where human control inputs, e.i. steering and throttle/braking, is taken as a trajectory reference and then tracked by the underlying MPC.

Instead of dividing the work into trajectory planning and following there is the possibility of solving both problems at once. Such a system for highway scenarios is presented in [25] and [26]. Pedro Lima proposes a MPC method based on clothoids [27], a smooth geometrical shape often used to construct comfortable roads, to generate smooth steering control. Trajectory generation and following for cooperative driving is discussed in [28]. A really good report to highlighten in this nest of valuable work is the report done by Brian Paden et al. [29], where they have studied 165 papers to get an overview of what is the state of the art in motion planning and control techniques for self-driving urban vehicles.

1.3.4 Models and Autonomous Racing

Discussed in almost all of these papers is the dynamic model embedded in the MPC formulation. This is the core of the controller and the more alike the model is the real world the better control input will be. But a more world-like model will inevitable increase the computational time of the algorithm and therefore reduce the controllers performance. A good balance between the model complexity and computational power has to be found in every application. The most commonly used model is the kinematic bicycle model thanks to its simplicity and still usable results. A kinematic bicycle model in the road following Frenet coordinate frame is developed in [26] and one in the global coordinate frame is discussed in [30]. The Master Thesis report [31] by S. Van Koutrik extends this one track model into the two track model which allows for wheel load distribution changes at accelerations. The kinematic bicycle model assumes no slip at all conditions, which is fine for low speeds, but at higher accelerations the dynamics of the tyre is nonlinear and this has to be simulated in order to assure stability. To allow for more precise modeling at higher slip angles [30], [31] and [32] introduces tyre models, linear and non linear. The last article, [32], is part of the very extensive research [33] presented by Krisada Kritayakirana on vehicle handling at the limits.

Driving at the limits is seldom happening for vehicles in the every day driving. An exception might be when an unexpected event is occurring and the avoiding of an potential accident has to take place. At those instances in every-day driving and at every second on a race track shall the optimal trajectory be calculated and the maximum acceleration be utilized to achieve the wanted outcome, avoiding accidents or wining races. When it comes to racing optimizing the parameters of the cars as in [34] or offline calculate the optimal race line as shown in [35] are two ways of cutting the lap times. A time optimal approach is described in [36] and a optimal race line with respect to the vehicle dynamics is presented in [37]. A successful approach to this problem is also developed in the report [38], where Florian Curinga proposes a MPC method that is working in the Frenet frame and tries to maximize the distance traveled along the road following axis. Methods like that can then be improved by letting the MPC learn from several iterations as done in [39] where the parameters of the vehicle dynamics is tuned to achieve the best results in simulations. Maximilian Brunner et al. presents in [40] another way to guarantee recursive feasibility while the algorithm is learning. This was then tested and verified on RC cars. Alexander Liniger and his colleagues in [41] also uses RC cars to verify that their single layer and double layer MPC algorithms are successfully in both driving at the limits but also avoid dynamic obstacles on the track, such as other racing cars. Finally ways to guarantee robustness is presented in [42], which would motivate the choice of algorithm.

1.4 Background

In this section basic concepts and notions will be presented that are the core of this project. First the different coordinate systems used, second a brief description of racing theory, third a comment on classical control theory and finally some Model Predictive Control fundamentals.

1.4.1 Different frames, the same world

There are several ways to express a position in space. The Global Positioning System (GPS) is widely used to get a longitudinal, lateral and altitude on the planet earth. For this purpose, where the curvature of the earth can be assumed to be zero, a fixed two dimensional Cartesian coordinate system, with a x and y axis, will be used for the positioning of the cones and the center line of the track. This will be transformed to a path following Frenet coordinate frame, as seen in the Figure 5.

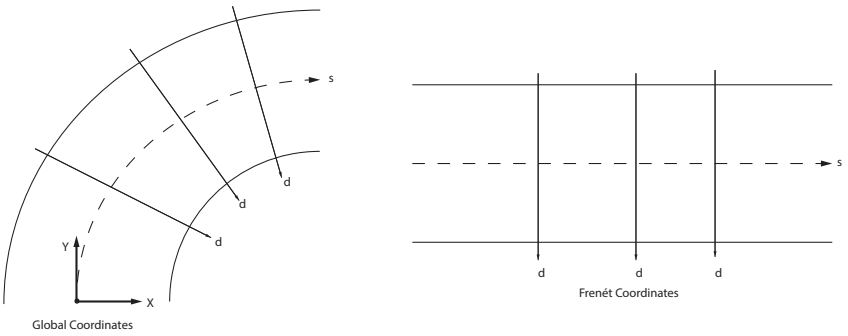

Figure 5: Global and Frenet frame

The main reason, as described in [18], is that it asserts invariant tracking performance under the action of the special Eucledian group $SE(2) \coloneqq SO(2) \times \mathbb{R}^2$, which reduces the computational time of the optimization algorithm.

1.4.2 The theory of racing

No matter if you are at Pikes Peak going from the bottom of the mountain to the top or if you are at pole position at Monaco's GP with 19 cars behind you, racing is all about going from the start to the finish line as fast as possible. In order to succeed the driver should maximize the acceleration at every single instant and follow a line that would allow the highest average speed. The race line is an individual line for every car type, because it is dependent on the vehicle dynamics, power, grip etc. An example for a 90 degree corner is shown in Figure 6.

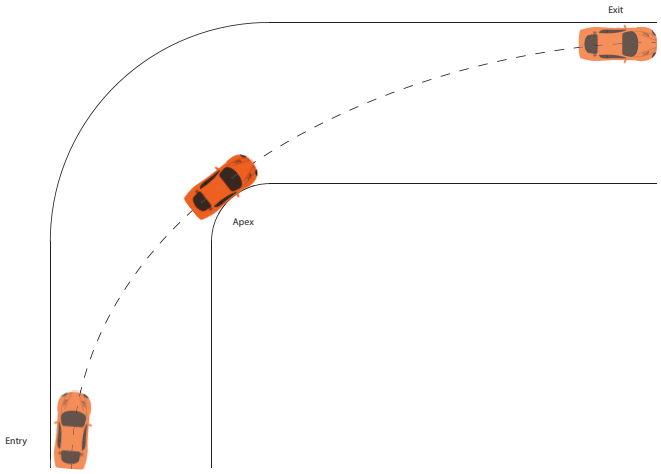

Figure 6: Race line

To achieve the maximal average speed over a section, like the curve in Figure 6, the driver or algorithm should choose the maximal radius possible. Entering the corner on one border of the track, hitting the other border, the apex, in the middle of the turn and then exit on the same side as entered. Once the corners are connected and of other types of radii a compromise has to be done making it harder to analyse. It gets even more tricky when there are more drivers on the track and their driving behaviour has to be taken into account. Luckily the Formula Student Rules [3] states that there is only one car at the time out on the track leaving that problem outside of the scope of this thesis.

1.4.3 Classical Control

To control a dynamic system, for example a car, a model of the system has to be established. By using the fundamental laws of physics the behaviour of the model, to some extent comparable to the real system, can be described by differential equations. The model or the real system is often referred to as the plant and is included in the analysis. How a control signal affects the dynamic system can be analysed by the input-output behaviour of the whole feedback system. A typical feedback system is depicted in Figure 7.

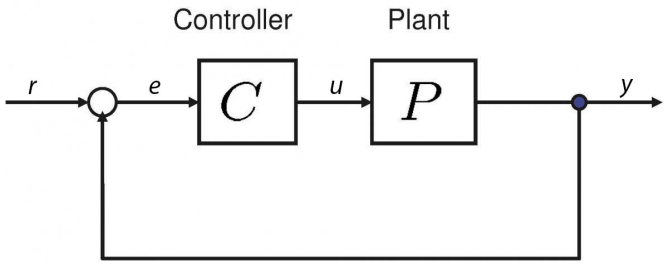

Figure 7: A simple feedback system

Analysis on a system like this has in general been done in the frequency domain after having transformed the differential equations with the Laplace transform, techniques like Bode diagrams, Nyquist plots and Pole Zero maps were used. Another, time domain, technique that is prominent is the state space representation, where the change of the plant is a function of the control inputs u and the previous states x. Simply put as

$$\dot{x} = f(x, u), \tag{1}$$

where $u \in \mathbb{R}^m$ is the input to the plant, $x \in \mathbb{R}^n$ is the states of the plant. More commonly the plant can be visualized in its matrix from

$$\begin{aligned} \dot{x} &= \mathbf{A}x + \mathbf{B}u \\ y &= \mathbf{C}x + \mathbf{D}u \end{aligned} \tag{2}$$

the so called state space representation. The matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ describes the dynamics of the system, $\mathbf{B} \in \mathbb{R}^{n\times m}$ the influence the control input has on the system, $\mathbf{C} \in \mathbb{R}^{n\times n}$ the measurable states and $\mathbf{D} \in \mathbb{R}^{n\times m}$ the influence of the control input directly on the measurable output y.

To control the plant the control signal u has to be determined. Error feedback is the most common, where the error signal $e(t) = r(t) - y(t)$ is manipulated in a favorable way. Proportional Integral Derivative (PID) controllers is the most common way to construct a controller $C(e(t)) \rightarrow u(t)$ but are limited by the fact that they only work as controllers for Single Input and Single Output (SISO) systems or decoupled Multiple Input Multiple Output (MIMO) systems. To handle more complicated coupled MIMO systems the best way to construct a classical controller is by using a Linear Quadratic Regulator (LQR). This controller is not based on the error feedback as the PID but instead a state feedback. The control signal is a function of the states $u(t) = f(x(t), t)$. Where the minimal cost of a quadratic cost function, Equation 3,

$$J(x(t), u(t)) = \int_0^\infty \left(x(t)^T Q x(t) + u(t)^T R u(t)\right) dt \tag{3}$$

has to be found, here $Q = Q^T \in \mathbb{R}^{n\times n}, Q \geq 0$ is the weight on the states and $R = R^T \in \mathbb{R}^{m\times m}, R > 0$ is the weight on the control signal. This equation is solved by solving the Riccati equation

$$\Phi \cdot B \cdot R^{-1} \cdot B^T \cdot \Phi - \Phi \cdot A - A^T \cdot \Phi - Q = 0 \tag{4}$$

which will make it possible to find the state feedback control matrix $K \in \mathbb{R}^{m\times n}$ as in Equation

5.

$$K = R^{-1} \cdot B^T \cdot \Phi \tag{5}$$

That would give the optimal control signal

$$u(t) = -K \cdot x(t) \tag{6}$$

The feedback is optimal over the infinite horizon, but it does not take into account the physical limitations of the system that is going to be controlled. Restructuring this LQR to an optimization problem with constraints and solving this at each time step is taking care of this problem and the method is called Model Predictive Control or MPC.

1.4.4 Model Predictive Control

MPC is essentially a receding horizon LQR with constraints and its usage is accelerating thanks to the performance of microprocessors. The logic is fairly simple, the controller uses the dynamic model of the system to compute the optimal control input for a certain time into the future with respect to the constraints. But the process is redone at a much higher rate than the horizon it looks into the future with. This means that only the first few of the control inputs of each optimization will be used before new ones are calculated. This increases the robustness and increases the disturbance rejection. Figure 8 from [43] shows the process.

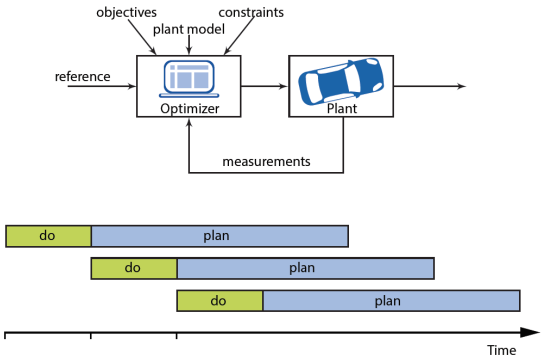

Figure 8: MPC scheme

For a control system to run on an embedded computer the logic has to be written in Discrete

Time (DT) which yields the following MPC formulation.

$$\begin{aligned} Minimize \ & \sum_{t=0}^{N-1} x_t^T Q x_t + u_t^T R u_t \\ Subject\ to: \ & x_{t+1} = Ax_t + Bu_t \qquad t = 0, ..., N-1 \\ & x_t = x(t) \\ & x_t \leq \mathcal{X} \qquad t = 0, ..., N \\ & u_t \leq \mathcal{U} \qquad t = 0, ..., N-1 \end{aligned} \tag{7}$$

where Q and R are the weight matrices and x and u are the state vector and control input vector, just as before in the LQR. The constraints consists of first the linear dynamic describing how the state of the car evolves into the future and then the constrained regions allowed for x and u. N is the prediction horizon and is a control parameter that can be chosen. A longer horizon is usually to prefer but it comes at a higher computational cost. The measured state $x(t)$ is assigned to x_t as the actual state, providing feedback. The solution to this optimization problem gives us the optimal control input $u_t^* \in [t = 0, ..., N-1]$ which is applied to the system until the next u^* is calculated.

1.5 Controller and Thesis structure

Since this thesis is a comparison between a multi layer controller and a single layer controller an introduction to the two systems are in place.

The multi layer controller is depicted in Figure 9 where the given center line of the track and the initial position is fed to the MPC and the MPC outputs the optimal trajectory in Frenet coordinates. Those are then transformed into global coordinates that the lower level controllers, the Pure Pursuit and the PID controller shall follow. The low level controllers output steering angle and longitudinal acceleration, i.e. throttle or brake. Those signals are used in the vehicle model that is representing the real world car, which in turn calculates the new states that are measured and sent back to the MPC as feedback.

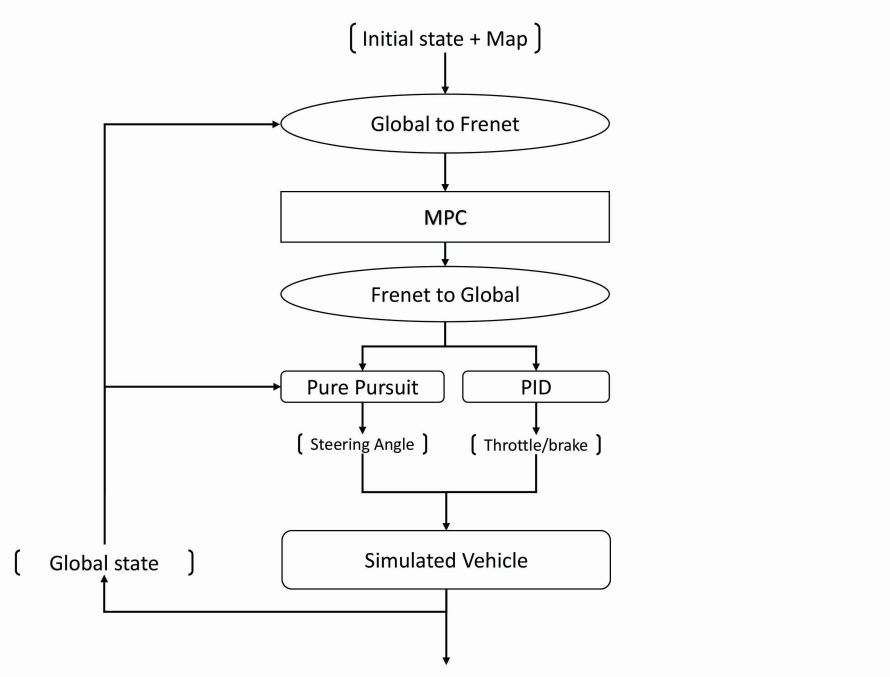

Figure 9: Multi layer control structure

The single layer control structure is only using a MPC controller to calculate the optimal steering angle and longitudinal acceleration directly. The system is shown in Figure 10.

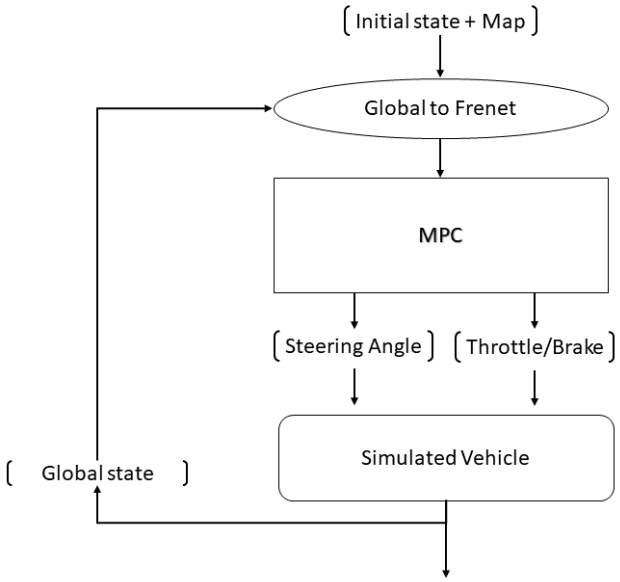

Figure 10: Single layer control structure

This report is arranged in the same way as the structure of the overall systems. It is initiated by going through the derivation and idea behind the coordinate transformations in section 2. That is followed by section 3 which describes the models used within the Model Predictive Controllers, both the multi layer and single layer version. The same chapter also describes the model used within the simulation environment to represent the real vehicle. Then the MPC framework is discussed with its cost functions, constraints and weights. Section 4 describes the single layer MPC and Section 5 the multi layer MPC with its low level controllers. Then all parts of the system is described and the final part is the comparison and conclusion which is in section 6.

2 Coordinate Transformation

The MPC could operate either in the Global coordinate frame or in the Frenet frame. The Frenet frame simplifies the structure of the MPC, much thanks to that the road-width constraints become constant, but problems occurs somewhere else instead. Since the car is operating in the global coordinate frame transformations are needed first from Frenet to global and then from global to Frenet. In this chapter the transformations are derived and described.

2.1 Frenet to Global

The simple MPC controller is calculating the optimal trajectory within the Frenet frame. The output is containing $[s, d, \dot{s}, \dot{d}, \ddot{s}, \ddot{d}]$ with several components of each dimension, dependent of the horizon length N. Each point into the future of the trajectory is then transformed to global coordinates $[X, Y, v, \dot{v}, \psi, \dot{\psi}]$ in incremental steps based on the curvature of the track at that instance. The position along s is connected to a x and y coordinate in the global frame from when the track was created and hence uses this information to map the position in Frenet to Global. Since the perpendicular deviation is know from the d-coordinates the actual global state of the car can be acquired easily together with the yaw angle of the track.

The yaw angle of the track is derived as a discrete derivative Δx and Δy between the known global positions along the center line and the yaw angle of the car trajectory is calculated in the same fashion but with the global positions of the car trajectory.

$$yaw = \arctan\left(\frac{\Delta y}{\Delta x}\right) \tag{8}$$

The curvature of the trajectory of the car κ is often denoted as $\kappa = 1/R$ but can also be derived as a function of the change in yaw angle over the change in s, as shown in Figure 11...

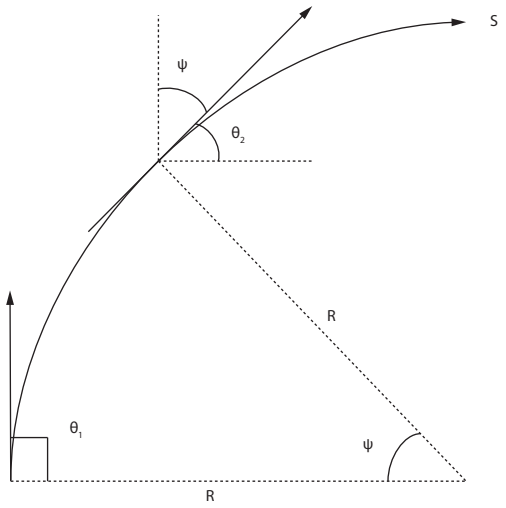

Figure 11: Derivation of the curvature as the change of yaw angle, used in the coordinate transformations

and with the notions $\dot{\psi} = \omega$, $s = R\psi$ and $\theta = \frac{\pi}{2} - \psi$. The change in θ over the change in s is,

$$\frac{\partial\theta}{\partial s} = \frac{\partial\left(\frac{\pi}{2} - \psi\right)}{\partial\left(R\psi\right)} = \frac{1}{R}\frac{\partial\left(\frac{\pi}{2} - \psi\right)}{\partial\left(\psi\right)} = \frac{1}{R} \cdot -1 \tag{9}$$

The discrete equivalent of this equation would be,

$$\kappa = \frac{\psi - \psi_s}{\Delta s}. \tag{10}$$

Having the path described with the derived states (x, y, ψ, κ) the planned trajectories can be visualised and also used for the lower level controllers as reference to follow.

2.2 Global to Frenet

When the states are measured from the real car they correspond to translation and rotational position, velocity and acceleration in the global frame. To be able to have a feedback to the MPC those states has to be expressed in Frenet coordinates.

The middle line of the track is the central reference point for the controllers. When this one is created by the Simultaneous Localization And Mapping (SLAM) algorithm every point in the global frame gets mapped to the corresponding Frenet frame point. This means that when the car starts at the starting line it places the $s = 0$ coordinate in the global frame and all incremental center line points are linked to that one and has its global corresponding point. This is then extended by creating a look up table for all incremental points over the whole track

which is visualised in Figure 12.

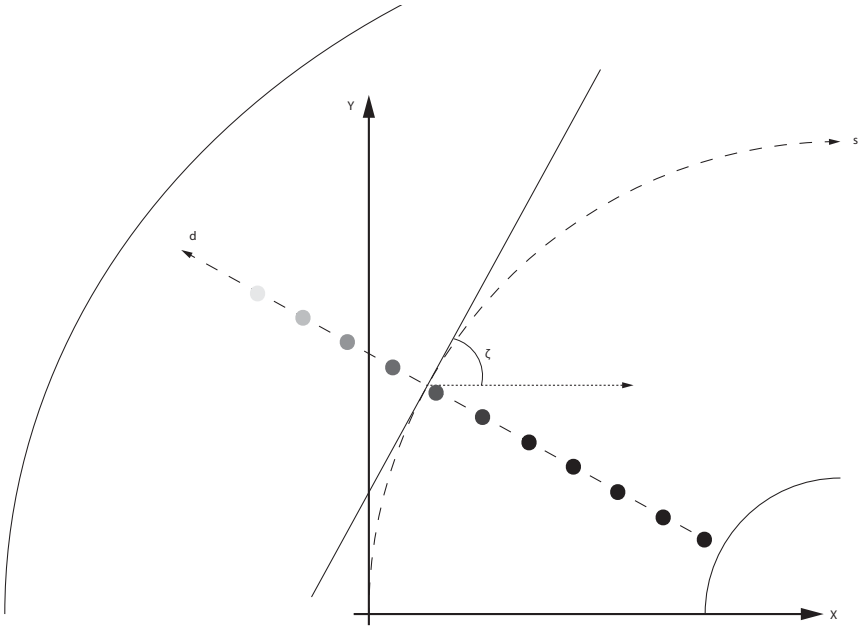

Figure 12: The creation of a look up table

Since the width of the track is known after the first mapping lap the look up table creation algorithm simply uses that as constraints in the d direction. Knowing the global and Frenet point of the center line at any s makes it possible to derive the rest of the points in the following manner.

With this look up table it is possible to go directly from the measured global coordinates to the corresponding Frenet coordinates. Once the position is found the velocity and acceleration has to be transformed as well, according to the geometry in Figure 13.

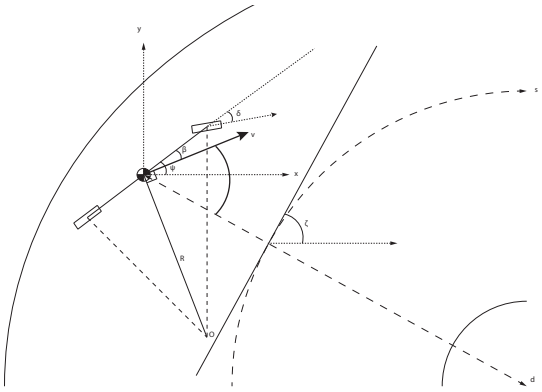

Figure 13: Transformation from global to Frenet coordinates

From the simulated vehicle model the body slip angle β can be derived as a function of the steering wheel angle δ as in [30]. With the assumptions that the vehicle is neutral steer and no slip.

$$\beta = \arctan(\frac{l_r}{l_r + l_f} \tan(\delta)) \tag{11}$$

and the full derivation can be found in section 3. The yaw angle of the car is given from the integration of the yaw-rate $\dot{\psi}$ which is derived as

$$\dot{\psi} = \frac{v}{l_r} \sin \beta(\delta). \tag{12}$$

The yaw angle of the track ζ is derived as the track is built up and known for each element along the s-axis. With those three angles the transformation from global to frenet can be calculated. The velocities in the s and d direction are expressed as

$$\dot{s} = v \cdot \sin\left((\frac{\pi}{2} - \zeta) + (\psi - \beta(\delta))\right) \tag{13}$$

and

$$\dot{d} = v \cdot \cos\left((\frac{\pi}{2} - \zeta) + (\psi - \beta(\delta))\right). \tag{14}$$

The longitudinal and lateral acceleration can be measured from the accelerometer and together is a acceleration vector derived.

$$\dot{v} = a_{lat} \cos\left(\beta(\delta)\right) + a_{long} \sin\left(\frac{\pi}{2} - \beta(\delta)\right) \tag{15}$$

With the general acceleration the accelerations in the Frenet frame can be expressed as

$$\ddot{s} = \dot{v} \sin((\frac{\pi}{2} - \zeta) + (\psi - \beta)) \tag{16}$$

and in the perpendicular direction

$$\ddot{d} = \dot{v} \cos((\frac{\pi}{2} - \zeta) + (\psi - \beta)) \tag{17}$$

This gives the full transformation from $[X, Y, v, \dot{v}, \psi, \dot{\psi}] \rightarrow [s, d, \dot{s}, \dot{d}, \ddot{s}, \ddot{d}]$.

3 Vehicle Modeling

3.1 The purpose of models

A dynamic model of the system that is controlled is essential in MPC, hence the name. By having a dynamic model that can approximate the behaviour of the system with respect to some inputs makes it possible to test for several different inputs, like steering angle and throttle/brake, and then evaluate the outcome. This is exactly what is done in a MPC and the better the model is the closer will the predicted outcome be the behaviour of the physical system. The downside though of having extremely detailed models is that they become computationally heavy, which either increase the computational time, or implies the need of a stronger and more expensive computer. In this chapter three different models are proposed. One extremely simple point mass model, one intermediate kinematic bicycle, or one track model, and one dynamic bicycle model including simple tyre dynamics.

3.2 Point Mass Model

One of the most simple models that can be used is the one that approximates the whole car as a single tyre with linear dynamics that assumes perfect grip at all times and the center of gravity in the road plane. This means that no nonholonomic physics are modeled, e.i. the model does not take into consideration that the car only steer by it's front wheels and this implies a certain travel for the rest of the car. There is also no dynamic load transfer (such as roll and pitch motion). It is basically a point mass in the ground plane that can translate the input acceleration and steering into the desired movements. The derived model in the state space representation becomes

$$x_{k+1} = \left(I + T_s \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \right) x_k + \left(T_s \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \right) u_k \tag{18}$$

where the states and inputs are

$$x = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} \quad u = \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} \tag{19}$$

The derivatives are with respect to time, i.e. the states contains the position and velocity in the two orthogonal dimensions and the inputs are its corresponding accelerations.

3.3 The kinematic bicycle model

The kinematic bicycle model is widely used within vehicle modeling, [30], [31] and [5] just to name a few. The idea is that the front and rear axle are merged into a single tyre and the center of gravity is in the ground plane. A schematic version of this is shown in Figure 14.

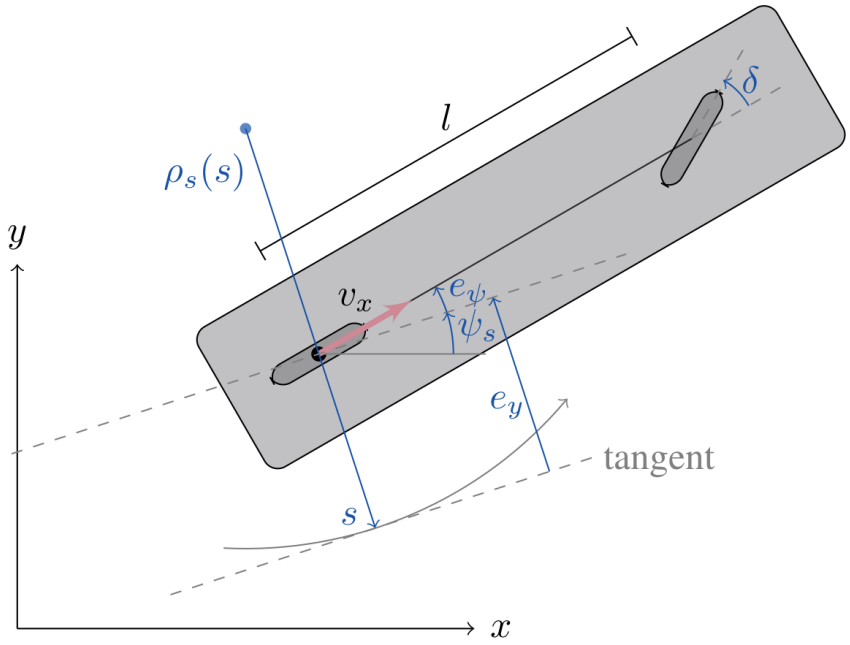

Figure 14: The bicycle model in a Global and Frenet coordinate system

The assumption that the center of gravity is in the rear axle is made and that the body slip is non existing. That means the tyres are seen as having perfect grip at all times and the resulting velocity of the vehicle is always aligned with the heading of the car itself. The global coordinate system is given by x, y and the road aligned Frenet coordinate system is denoted as s along the center line of the track and e_y is the deviation from it. The difference in heading between the car and the track is given by $e_\psi = \psi - \psi_s$ where ψ_s is the heading of the track. The steering angle of the front wheel is denoted as δ and $\rho_s(s) = 1/\kappa_s$ is the radius of the track at a specific point s. The time domain equations of this nonholonomic model in a global coordinate system

are given by

$$\begin{aligned} \dot{x} &= v_x \cos\psi, & \dot{y} &= v_x \sin\psi, \\ \dot{v}_x &= a_x, & \dot{\psi} &= v_x \kappa, \end{aligned} \tag{20}$$

where (x, y, ψ) represent the vehicle position and direction, v_x and a_x the longitudinal velocity and acceleration and κ the curvature of the vehicles path. The curvature has a direct geometrical connection to the steering angle as $\kappa = \frac{\tan\delta}{l}$ where l is the wheelbase of the vehicle.

With the knowledge and geometrical relationship the dynamics can be described with respect to the Frenet Frame

$$\begin{aligned} \dot{e}_y &= v_x \cos e_\psi, & \dot{e}_\psi &= \dot{\psi} - \dot{\psi}_s, \\ \dot{v}_x &= a_x, & \dot{s} &= \frac{v_x \cos e_\psi}{1 - e_y \kappa_s}, \end{aligned} \tag{21}$$

where κ_s is the curvature of the track, e.i. the inverse of the radius of the track ρ_s, the radius is also assumed to be larger than e_y at all times.

Using the road aligned coordinate system the propagation in each variable can be expressed as a function of the change in s. This spatial derivative is denoted with the prime notation e.g. e'_y and can be obtained using the chain rule of time derivatives. The expression can then be altered as,

$$\frac{(\cdot)}{ds} = \frac{(\cdot)}{dt}\frac{dt}{ds} = \frac{(\cdot)}{dt}\frac{1}{\dot{s}}. \tag{22}$$

Following this formulation the space-based representation can written as

$$e'_\psi = \frac{(1 - e_y\kappa_s)\kappa}{\cos e_\psi} - \dot{\psi}_s, \qquad e'_y = (1 - e_y\kappa_s)\tan e_\psi, \qquad v'_x = \frac{(1 - e_y\kappa_s)}{v \cos e_{psi}} a_x. \tag{23}$$

3.4 The nonlinear bicycle model

The kinematic model has really good performance with respect to its simplicity. The model is better at low speeds and when the tyres are in their linear, adhesion-region of friction, which can be seen in the slip-friction curve in Figure 15.

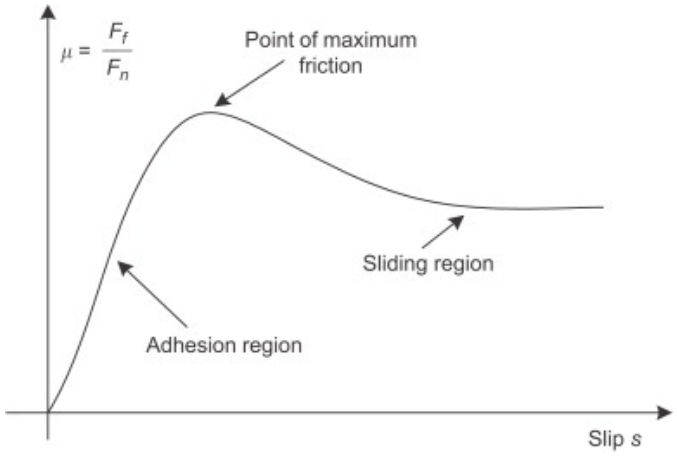

Figure 15: A schematic tyre slip versus friction curve illustrating the nonlinear tyre dynamics

However, for a race car the optimal point to consistently be at is the point of maximum friction. Here maximum acceleration is guaranteed either forward, backwards or sideways. To be able to accelerate with a car the wheels need to rotate faster than the vehicle is moving. When this happen the tyres can be seen as a spring that is extended, the force increases as the spring is extended, just as the linear region in Figure 15. Increased slip gives increased forward force F_f, with μ as the adhesion constant and F_n is the normal force on the tyre coming from the weight of the car and the load transfer of a roll or pitch motion. The spring has a spring constant k that maps the extension and force, for tyres are the corresponding constant the cornering stiffness C_α. The famous Fiala tyre model assumes that the lateral and longitudinal cornering stiffness are the same. The slip is often denoted σ and is calculated in the following way,

$$\sigma = \sqrt{\sigma_x^2 + \sigma_y^2}, \tag{24}$$

where σ_x and σ_y denotes the longitudinal and lateral slip, respectively. The dimensionless slip are calculated in the following fashion,

$$\begin{aligned} \sigma_x &= \frac{r\omega_w - v_x}{r\omega_w}, \\ \sigma_x &= \frac{r\omega_w - v_x}{v_x}, \\ \sigma_y &= \frac{v_x}{r\omega_w} \tan\alpha, \end{aligned} \tag{25}$$

with the first equation is used while accelerating and the second one during deceleration. ω_w being the rotational speed of the wheel, r the radius of the wheel and α the slip angle of the tyre. The Fiala tyre models then uses two expressions to express the force created by the tyre. The first one is valid during slip values less than the slip at the maximum friction σ_{max},

$$F = \mu F_z(3\theta\sigma - 3(\theta\sigma)^2 + (\theta\sigma)^3). \tag{26}$$

When the wheels exceed the slip at the maximum friction the force is better approximated with,

$$F = \mu F_z, \tag{27}$$

with $\theta = 1/\sigma_{max} = \frac{C_\alpha}{3\mu F_Z}$ from Equation 26. The longitudinal and lateral forces at the tyres can then be computed as

$$\begin{aligned} F_x &= \frac{\sigma_x}{\sigma} F, \\ F_y &= \frac{\sigma_y}{\sigma} F. \end{aligned} \tag{28}$$

In steady state cornering only lateral forces appears which gives $\sigma_x = 0$ and $\sigma_y = \tan\alpha$. If the car instead accelerates longitudinaly $\sigma_y = 0$.

With the tyre forces from the Fiala model the dynamic bicycle (or single track) model can be derived. Following the derivation in [31] together with the notation in Figure 16,

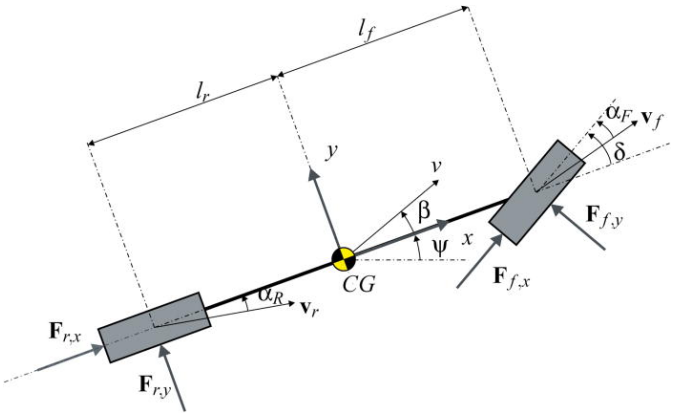

Figure 16: The dynamic bicycle model

the following equations are given.

$$\begin{aligned} \dot{X} &= v_x \cos\psi - v_y \sin\psi, \\ \dot{Y} &= v_x \sin\psi + v_y \cos\psi, \\ \dot{\psi} &= \omega, \\ \dot{v}_x &= \frac{1}{m}(F_{r,x} - F_{f,y}\sin\delta + m v_y \omega), \\ \dot{v}_y &= \frac{1}{m}(F_{r,y} - F_{f,y}\cos\delta - m v_x \omega), \\ \dot{\omega} &= \frac{1}{I_z}(F_{f,y} l_f \cos\delta - F_{r,y} l_r), \end{aligned} \tag{29}$$

with the assumption that the car is rear wheel driven, as is the case of eV14, $F_{f,x} = 0$. The rotation around the positive z-axis is denoted as ω and ψ is the yaw angle of the vehicle. The state is given by the vector $x = (X, Y, \psi, v_x, v_y, \omega)^T$. This vehicle model is the one being used as

the simulated vehicle while the two first models are used in the two different MPC formulations.

The body slip angle β is used during the coordinate transformation described in section 2. The derivation of this, is as follows: In every instant the trajectory of the vehicle path has a curvature which is the inverse of the radius from the point around the vehicle is turning marked as turning centre in Figure 17, but it will be referred to as O.

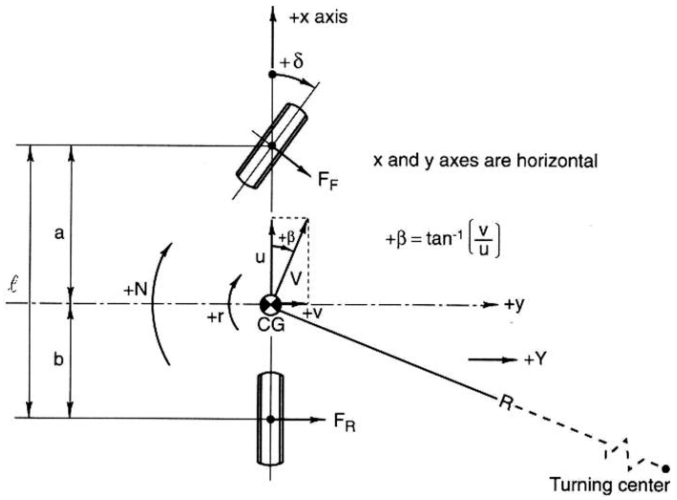

Figure 17: The dynamic bicycle model, around its turning center

The velocity vector of the vehicle can be assumed to always be perpendicular to the line, R long, from O to the center of gravity of the car, assuming neutral steer and no slip. Knowing that and that the point O around which the car is turning has a geometrical relationship to the front δ and rear ($\delta_{rear} = 0$) steering angle the body slip angle can be derived as follows. Using trigonometry the relations,

$$\frac{l_f}{\sin(\delta - \beta)} = \frac{R}{\sin(\frac{\pi}{2} - \delta)}, \tag{30}$$

and simplifying it to,

$$\frac{l_f}{\sin(\delta - \beta)} = \frac{R}{\cos\delta}. \tag{31}$$

Writing $R = \frac{l_r}{\sin\beta}$ the relationship can be rewritten as,

$$\frac{l_f}{\sin(\delta - \beta)} = \frac{l_r}{\cos\delta \sin\beta}. \tag{32}$$

Seeing that the trigonometric rule $\sin(\delta - \beta) = \sin\delta\cos\beta - \cos\delta\sin\beta$ can be applied to rewrite it once again as,

$$l_f = l_r\left(\frac{\sin\delta\cos\beta - \cos\delta\sin\beta}{\sin\beta\cos\delta}\right) = l_r\left(\frac{\sin\delta\cos\beta}{\sin\beta\cos\delta} - 1\right) = l_r\tan\delta\tan^{-1}\beta - l_r. \tag{33}$$

A final rearrangement of equation 33 gives the connection between the steering angle δ and the

body slip angle β as,

$$\beta = \tan^{-1}\left(\frac{l_r}{l_f + l_r}\tan\delta\right), \tag{34}$$

which is a fundamental relationship in vehicle modeling.

4 Single Layer MPC

Integrating a kinematic bicycle model in the MPC framework makes it possible to create a single layer controller structure, taking vehicle states as inputs and with respect to the previous predicted optimal solutions calculating the new optimal controller inputs in a receding horizon fashion. The idea was to use the exact same controller as in [38], but as some small but yet fundamental faults were detected was the model changed a bit thanks to input from Pedro Lima and [27]. Though the performance shall be similar.

4.1 Maximizing Progression

The idea of this controller is to maximize the progression along s the road following coordinate system. As described in section 3 the progression along the center line is described as

$$s = T_s \frac{v \cos e_\psi}{1 - e_y \kappa_s} \tag{35}$$

Which is a function of the vehicle states $x = (e_\psi, e_y, v)$ where T_s is the sampling time. To visualize this Figure 18 is created.

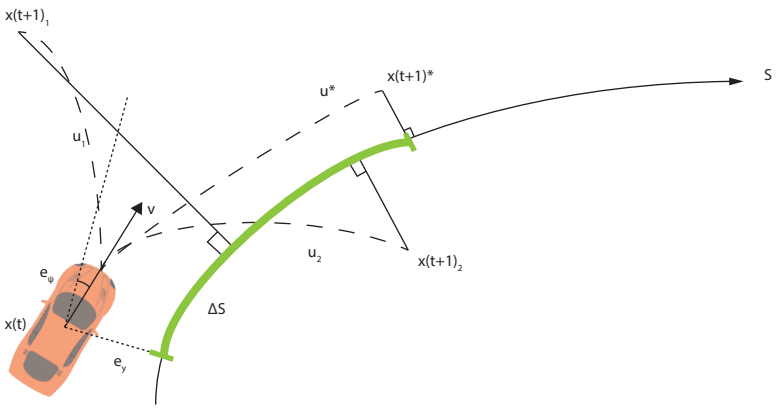

Figure 18: The progression along the center line

The state of the car is described with respect to the center line of the track, the difference in heading angle of the car compared to the tangent of the track e_ψ, the perpendicular deviation from the center line e_y and the speed in the direction of the vehicle v. Information about the

track such as the curvature κ_s in each point is known from beforehand. The objective of the MPC is to find the optimal input u^* such that the progression is the largest between each iteration x_t and x_{t+1}.

4.2 General nonlinear MPC formulation

With the idea of how the controller shall find the optimal solution a MPC formulation is put together as

$$\begin{aligned} & Maximize \sum_{k=0}^{N-1} \frac{v \cos e_\psi}{1 - e_y \kappa_s} \\ Subject\ to: \ & x_{t+1} = f(x_k, u_k) \qquad t = 0, ..., N-1 \\ & x_0 = x(t) \\ & x_k \leq \mathcal{X} \qquad t = 0, ..., N \\ & u_k \leq \mathcal{U} \qquad t = 0, ..., N-1 \end{aligned} \tag{36}$$

where $f(x_k, u_k)$ is the nonlinear dynamics of the system described in the vehicle model section 3, x_0 is the measured initial state of the iteration and $\mathcal{X}$, $\mathcal{U}$ are state and input constraints respectively. To optimize a problem with nonlinear cost function and nonlinear dynamics is very costly and might threaten the real time applications. Therefore the problem formulation is linearised in the following sections, creating a convex optimization problem.

4.3 Cost function of the single layer MPC

The cost function of the PM-MPC maximizes the progress along the center line s, which is equivalent to maximize $\dot{s}$, or minimize $\frac{1}{\dot{s}} = \frac{1 - e_y \kappa}{v \cos e_\psi}$. Approximating the non convex function $\frac{1}{\dot{s}}$ with a Taylor expansion will make it a convex problem. Since the Taylor expansion of cos is consisting of only even terms the approximation of $\frac{1}{\dot{s}}$ with respect to e_ψ is taken to the second derivative to not impose any errors. The two remaining states e_y and v can though be approximated with the first degree terms, which yields the following results. The first derivatives:

$$\frac{\partial \frac{1}{\dot{s}}}{\partial e_\psi} = \frac{(1 - e_y \kappa) \tan e_\psi \sec e_\psi}{v} \tag{37}$$

$$\frac{\partial \frac{1}{\dot{s}}}{\partial e_y} = -\frac{\kappa}{v \cos e_\psi} \tag{38}$$

$$\frac{\partial \frac{1}{\dot{s}}}{\partial v} = -\frac{(1 - e_y\kappa)}{v^2 \cos e_\psi} \tag{39}$$

Following with the second derivative of $\frac{1}{\dot{s}}$,

$$\frac{\partial^2 \frac{1}{\dot{s}}}{\partial e_\psi^2} = \frac{(1 - e_y\kappa) \sec e_\psi (\tan^2 e_\psi + \sec^2 e_\psi)}{v} \tag{40}$$

which yields the following result for the linearization points $e_\psi = 0$, $e_y \neq 0$ and $v \neq 0$, since $\sec 0 = 1$ and $\tan 0 = 0$.

$$\frac{\partial^2 \frac{1}{\dot{s}}}{\partial e_\psi^2}|_{e_\psi=0} = \frac{(1 - e_y\kappa)}{v}, \quad \frac{\partial \frac{1}{\dot{s}}}{\partial e_y}|_{e_\psi=0} = -\frac{\kappa}{v}, \quad \frac{\partial \frac{1}{\dot{s}}}{\partial v}|_{e_\psi=0} = -\frac{(1 - e_y\kappa)}{v^2} \tag{41}$$

Finally the time varying cost function is

$$Minimize : \frac{(1 - e_y\kappa)}{v} e_\psi^2 - \frac{\kappa}{v} e_y - \frac{(1 - e_y\kappa)}{v^2} v \tag{42}$$

4.4 The linearized and discretiziced dynamics of the system

The nonlinear dynamics given by the previously derived bicycle model in the frenet frame derived with respect to the spatial domain are

$$\begin{pmatrix} e'_\psi \\ e'_y \\ v' \\ \kappa' \\ a \end{pmatrix} = x' = f(x,u) = \begin{pmatrix} f_1(x,u) \\ f_2(x,u) \\ f_3(x,u) \\ f_4(x,u) \\ f_5(x,u) \end{pmatrix} = \begin{pmatrix} \frac{(1-e_y\kappa_s)\kappa}{\cos e_\psi} - \psi'_s \\ (1 - e_y\kappa_s) \tan e_\psi \\ \frac{(1-e_y\kappa_s)}{v \cos e_\psi} a \\ \frac{(1-e_y\kappa_s)}{v \cos e_\psi} C \\ \frac{(1-e_y\kappa_s)}{v \cos e_\psi} J \end{pmatrix} \tag{43}$$

By using the chain rule $\frac{d(\cdot)}{ds} = \frac{d(\cdot)}{dt}\frac{dt}{s} = \frac{d(\cdot)}{dt}\frac{1}{\dot{s}}$ those derivatives can be obtained. The linearisation has to be made around a valid equilibrium point. If the goal was to follow the center line it would make sense to linearise around that. But in the racing case the car is rarely at the center

line, hence other points has to be chosen for the system to be valid. In this case the previous optimal solution is used to linearise the LTV system around. This is illustrated in Figure 19.

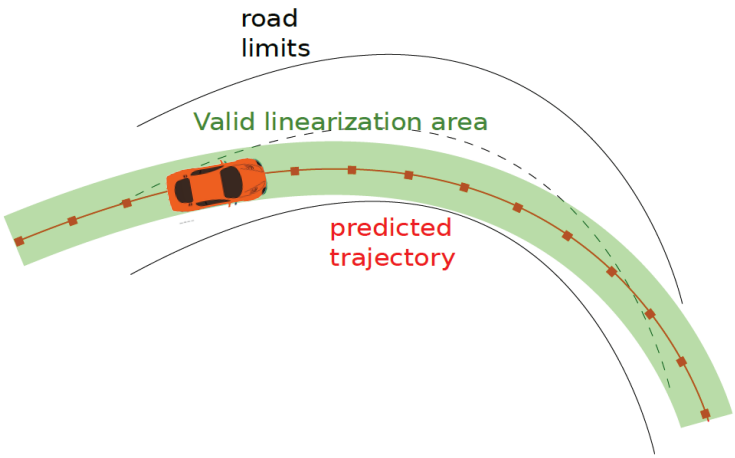

Figure 19: The linearisation points and the area where the linearisation is a valid approximation of the nonlinear system

First the system is linearised around the predicted optimal states from the last iteration $(\bar{x}, \bar{u})$. Following the general formulation

$$\dot{x} = \begin{bmatrix} \frac{\partial f_1}{\partial e_\psi} & \frac{\partial f_1}{\partial e_y} & \frac{\partial f_1}{\partial v} & \frac{\partial f_1}{\partial \kappa} & \frac{\partial f_1}{\partial a} \\ \frac{\partial f_2}{\partial e_\psi} & \frac{\partial f_2}{\partial e_y} & \frac{\partial f_2}{\partial v} & \frac{\partial f_2}{\partial \kappa} & \frac{\partial f_2}{\partial a} \\ \frac{\partial f_3}{\partial e_\psi} & \frac{\partial f_3}{\partial e_y} & \frac{\partial f_3}{\partial v} & \frac{\partial f_3}{\partial \kappa} & \frac{\partial f_3}{\partial a} \\ \frac{\partial f_4}{\partial e_\psi} & \frac{\partial f_4}{\partial e_y} & \frac{\partial f_4}{\partial v} & \frac{\partial f_4}{\partial \kappa} & \frac{\partial f_4}{\partial a} \\ \frac{\partial f_5}{\partial e_\psi} & \frac{\partial f_5}{\partial e_y} & \frac{\partial f_5}{\partial v} & \frac{\partial f_5}{\partial \kappa} & \frac{\partial f_5}{\partial a} \end{bmatrix} (x - \bar{x}) + \begin{bmatrix} \frac{\partial f_1}{\partial C} & \frac{\partial f_1}{\partial J} \\ \frac{\partial f_2}{\partial C} & \frac{\partial f_2}{\partial J} \\ \frac{\partial f_3}{\partial C} & \frac{\partial f_3}{\partial J} \\ \frac{\partial f_4}{\partial C} & \frac{\partial f_4}{\partial J} \\ \frac{\partial f_5}{\partial C} & \frac{\partial f_5}{\partial J} \end{bmatrix} (u - \bar{u}) + f(\bar{x}, \bar{u}) \tag{44}$$

The three by three matrix is denoted the A matrix and the three by two matrix is commonly denoted as the B matrix in a state space representation.

The system is linearised but it is still continuous time (CT) which is the way a physical system is denoted, but it is not possible to implement on a micro controller that works in discrete time

(DT). To discretize the problem equation 44 has to be rewritten as,

$$\dot{x} = \begin{bmatrix} \frac{\partial f_1}{\partial e_\psi} & \frac{\partial f_1}{\partial e_y} & \frac{\partial f_1}{\partial v} & \frac{\partial f_1}{\partial \kappa} & \frac{\partial f_1}{\partial a} \\ \frac{\partial f_2}{\partial e_\psi} & \frac{\partial f_2}{\partial e_y} & \frac{\partial f_2}{\partial v} & \frac{\partial f_2}{\partial \kappa} & \frac{\partial f_2}{\partial a} \\ \frac{\partial f_3}{\partial e_\psi} & \frac{\partial f_3}{\partial e_y} & \frac{\partial f_3}{\partial v} & \frac{\partial f_3}{\partial \kappa} & \frac{\partial f_3}{\partial a} \\ \frac{\partial f_4}{\partial e_\psi} & \frac{\partial f_4}{\partial e_y} & \frac{\partial f_4}{\partial v} & \frac{\partial f_4}{\partial \kappa} & \frac{\partial f_4}{\partial a} \\ \frac{\partial f_5}{\partial e_\psi} & \frac{\partial f_5}{\partial e_y} & \frac{\partial f_5}{\partial v} & \frac{\partial f_5}{\partial \kappa} & \frac{\partial f_5}{\partial a} \end{bmatrix} x + \begin{bmatrix} \frac{\partial f_1}{\partial C} & \frac{\partial f_1}{\partial J} \\ \frac{\partial f_2}{\partial C} & \frac{\partial f_2}{\partial J} \\ \frac{\partial f_3}{\partial C} & \frac{\partial f_3}{\partial J} \\ \frac{\partial f_4}{\partial C} & \frac{\partial f_4}{\partial J} \\ \frac{\partial f_5}{\partial C} & \frac{\partial f_5}{\partial J} \end{bmatrix} u + \underbrace{f(\bar{x},\bar{u}) - \begin{bmatrix} \frac{\partial f_1}{\partial e_\psi} & \frac{\partial f_1}{\partial e_y} & \frac{\partial f_1}{\partial v} & \frac{\partial f_1}{\partial \kappa} & \frac{\partial f_1}{\partial a} \\ \frac{\partial f_2}{\partial e_\psi} & \frac{\partial f_2}{\partial e_y} & \frac{\partial f_2}{\partial v} & \frac{\partial f_2}{\partial \kappa} & \frac{\partial f_2}{\partial a} \\ \frac{\partial f_3}{\partial e_\psi} & \frac{\partial f_3}{\partial e_y} & \frac{\partial f_3}{\partial v} & \frac{\partial f_3}{\partial \kappa} & \frac{\partial f_3}{\partial a} \\ \frac{\partial f_4}{\partial e_\psi} & \frac{\partial f_4}{\partial e_y} & \frac{\partial f_4}{\partial v} & \frac{\partial f_4}{\partial \kappa} & \frac{\partial f_4}{\partial a} \\ \frac{\partial f_5}{\partial e_\psi} & \frac{\partial f_5}{\partial e_y} & \frac{\partial f_5}{\partial v} & \frac{\partial f_5}{\partial \kappa} & \frac{\partial f_5}{\partial a} \end{bmatrix} \bar{x} - \begin{bmatrix} \frac{\partial f_1}{\partial C} & \frac{\partial f_1}{\partial J} \\ \frac{\partial f_2}{\partial C} & \frac{\partial f_2}{\partial J} \\ \frac{\partial f_3}{\partial C} & \frac{\partial f_3}{\partial J} \\ \frac{\partial f_4}{\partial C} & \frac{\partial f_4}{\partial J} \\ \frac{\partial f_5}{\partial C} & \frac{\partial f_5}{\partial J} \end{bmatrix} \bar{u}}_{g(\bar{x},\bar{u})} \tag{45}$$

The function $g(\bar{x},\bar{u})$ is constant and therefore not directly part of the discretization. To discretize the system with the fixed step length ΔS the exact discretization can approximated as,

$$A_d = e^{A\Delta S} \approx (I + \Delta S A) \quad and \quad B_d = A^{-1}(e^{A\Delta S} - I)B \approx \Delta S B, \tag{46}$$

as long as the A matrix is nonsingular, i.e. it is invertable. Otherwise exact discretization has to be done in the exact manner with matrix exponential. The matrix M is constructed as

$$M = \exp\left(\Delta S \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix}\right) = \begin{bmatrix} A_d & B_d \\ 0 & 0 \end{bmatrix} \tag{47}$$

from the matrix exponential of M the discretize A and B matrix can be extracted. This gives the linearized and discretize system as

$$x_{k+1} = \left(I + \Delta S \begin{bmatrix} \frac{\partial f_1}{\partial e_\psi} & \frac{\partial f_1}{\partial e_y} & \frac{\partial f_1}{\partial v} & \frac{\partial f_1}{\partial \kappa} & \frac{\partial f_1}{\partial a} \\ \frac{\partial f_2}{\partial e_\psi} & \frac{\partial f_2}{\partial e_y} & \frac{\partial f_2}{\partial v} & \frac{\partial f_2}{\partial \kappa} & \frac{\partial f_2}{\partial a} \\ \frac{\partial f_3}{\partial e_\psi} & \frac{\partial f_3}{\partial e_y} & \frac{\partial f_3}{\partial v} & \frac{\partial f_3}{\partial \kappa} & \frac{\partial f_3}{\partial a} \\ \frac{\partial f_4}{\partial e_\psi} & \frac{\partial f_4}{\partial e_y} & \frac{\partial f_4}{\partial v} & \frac{\partial f_4}{\partial \kappa} & \frac{\partial f_4}{\partial a} \\ \frac{\partial f_5}{\partial e_\psi} & \frac{\partial f_5}{\partial e_y} & \frac{\partial f_5}{\partial v} & \frac{\partial f_5}{\partial \kappa} & \frac{\partial f_5}{\partial a} \end{bmatrix} \right) x_k + \left(\Delta S \begin{bmatrix} \frac{\partial f_1}{\partial C} & \frac{\partial f_1}{\partial J} \\ \frac{\partial f_2}{\partial C} & \frac{\partial f_2}{\partial J} \\ \frac{\partial f_3}{\partial C} & \frac{\partial f_3}{\partial J} \\ \frac{\partial f_4}{\partial C} & \frac{\partial f_4}{\partial J} \\ \frac{\partial f_5}{\partial C} & \frac{\partial f_5}{\partial J} \end{bmatrix} \right) u_k + g(\bar{x}_k, \bar{u}_k) \tag{48}$$

if A is nonsingular. Otherwise the

$$x_{k+1} = A_d x_k + B_d u_k + g(\bar{x}_k, \bar{u}_k) \tag{49}$$

has to be used. In this case it is seen that this A matrix is singular and represented as equation 49, but in the simpler point mass model, presented later the approximation method is used, since that A matrix is nonsingular. The matrices describing the dynamic system are

$$A = \begin{bmatrix} \frac{(1-\bar{e}_y\kappa_S)\bar{\kappa}\tan\bar{e}_\psi}{\cos\bar{e}_\psi} & -\frac{\bar{\kappa}\kappa_s}{\cos\bar{e}_\psi} & 0 & \frac{(1-\bar{e}_y\kappa_s)\bar{\kappa}}{\cos\bar{e}_\psi} & 0 \\ \frac{(1-\bar{e}_y\kappa_s)}{\tan^2\bar{e}_\psi+1} & -\kappa_s\tan\bar{e}_\psi & 0 & 0 & 0 \\ \frac{(1-\bar{e}_y\kappa_S)\bar{a}\tan\bar{e}_\psi}{\bar{v}\cos\bar{e}_\psi} & -\frac{\kappa_s\bar{a}}{\bar{v}\cos\bar{e}_\psi} & -\frac{(1-\bar{e}_y\kappa_s)\bar{a}}{\bar{v}^2\cos\bar{e}_\psi} & 0 & \frac{(1-\bar{e}_y\kappa_s)}{\bar{v}\cos\bar{e}_\psi} \\ \frac{(1-\bar{e}_y\kappa_S)\bar{C}\tan\bar{e}_\psi}{\bar{v}\cos\bar{e}_\psi} & -\frac{\kappa_s\bar{C}}{\bar{v}\cos\bar{e}_\psi} & -\frac{(1-\bar{e}_y\kappa_s)\bar{C}}{\bar{v}^2\cos\bar{e}_\psi} & 0 & 0 \\ \frac{(1-\bar{e}_y\kappa_S)\bar{J}\tan\bar{e}_\psi}{\bar{v}\cos\bar{e}_\psi} & -\frac{\kappa_s\bar{J}}{\bar{v}\cos\bar{e}_\psi} & -\frac{(1-\bar{e}_y\kappa_s)\bar{J}}{\bar{v}^2\cos\bar{e}_\psi} & 0 & 0 \end{bmatrix} \tag{50}$$

$$B_d = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{(1-\bar{e}_y\kappa_s)}{\bar{v}\cos\bar{e}_\psi} & 0 \\ 0 & \frac{(1-\bar{e}_y\kappa_s)}{\bar{v}\cos\bar{e}_\psi} \end{bmatrix} \tag{51}$$

and the state and input vectors are

$$x = \begin{bmatrix} e_\psi \\ e_y \\ v \\ \kappa \\ a \end{bmatrix} \quad u = \begin{bmatrix} C \\ J \end{bmatrix} \tag{52}$$

where $\kappa = \frac{\tan\delta}{l}$ and δ is the steering angle of the front wheels. The change jerk is denoted J and the change in curvature, sharpness, is denoted C.

4.5 Constraints

The system that is controlled has physical limitations in it self, for example, it is not possible to steer the front wheel to any angle, the power from the motors or the grip limits the system acceleration and the car shall not leave the track since that would be dangerous and in the content of Formula Student Driverless would result in penalties. Therefore constraints are included in the optimization problem such that the optimal solution is also feasible with respect to the system that should be controlled. The physical constraints used are measured from the system, in this case the KTH Formula Student car eV14 where the max acceleration is $a_{max} = 10$ and the max deceleration is $a_{min} = -12[m/s^2]$. The steering is symmetric in left to right and the max steering angle is $\delta_{max} = \pi/4[rad]$ which in turn gives the maximum possible curvature $\kappa = \frac{\tan\delta_{max}}{l}$. The final physical constraint is on the speed of the vehicle. Since the electrical machines has a maximum rpm and a single gear gearbox is used the maximum speed is limited to $v_{max} = 41,6[m/s] = 150[km/h]$ the speed is also constrained by the fact that the car shall not go backwards $v_{min} = 0$. The Formula Student track is a tight track but to study the behaviour of the

trajectory planners a width of 6 meters is chosen which gives the constraint in lateral deviation from the center line of $e_{ymax} = -e_{ymin} = 3[m]$. To win the race the car has to propagate along the race track as fast as possible, in the right direction, that is why the constraint on the car to not go backwards is implemented as well, as $e_{\psi max} = -e_{\psi min} = \pi/4$.

4.6 Soft constraints

Imposing hard constraints on the state of the system will reduce the feasible region even for stable systems and therefore it is often better to soften these constraints such that a small violation does not terminate the controller. This is exactly the case for the Formula Student car, it is better to be outside the track in one instance and get around the track than to make the car stop as soon as it gets outside the track. The implementation of this is done by including a slack variable $\sigma \geq 0$ which is also included in the cost function with a heavy weight on it to push it down towards zero at all times, because the hard constraints are not violated as long as the slack variable is zero.

4.7 Final formulation

Including all changes to make the control system work as well as possible the final formulation is shown in equation 53.

$$
\begin{aligned}
Minimize \quad & \sum_{k=0}^{N-1} \frac{(1-e_y\kappa)}{v} e_\psi^2 - \frac{\kappa}{v} e_y - \frac{(1-e_y\kappa)}{v^2} v + \eta \sigma^T \sigma \\
Subject\ to: \quad & x_{t+1} = A_k x_k + B_k u_k + g(\bar{x}_k, \bar{u}_k) \\
& x_0 = x(t) \\
& e_{\psi min} - \sigma_{e_\psi} \leq e_\psi \leq e_{\psi max} + \sigma_{e_\psi} \\
& e_{ymin} - \sigma_{e_y} \leq e_y \leq e_{ymax} + \sigma_{e_y} \\
& v_{min} \leq v \leq v_{max} \\
& \kappa_{min} \leq \kappa \leq \kappa_{max} \\
& a_{min} \leq a \leq a_{max} \\
& C_{min} \leq C \leq C_{max} \\
& J_{min} \leq J \leq J_{max} \\
& \sigma_{e_\psi} \geq 0, \sigma_{e_y} \geq 0 \\
& k = 0, ..., N-1
\end{aligned}
\tag{53}
$$

where the decision variables that are being optimized are s, u and σ. The initial state of the vehicle x_0 is given by the current vehicle state $x(t)$, which are measured at time t. The penalizing factor η is tuneable to a value such that the slack variables are pushed towards zero as much as

possible. Some practicalities include the iteration of the track dependent variables κ_s and ψ_s, which are found by using the described look-up table in section 2. Calculating the propagation along s keeps track of where on the track the car is and makes it possible to know the index of the position along the center line. This index then gives the information about the curvature κ_s and angle ψ_s of the track in the given position. The initialization of the simulation is done by a so called warm start. Properly guessed start vectors ensures a smooth start for the recursive optimization.

5 Double layer MPC

5.1 A simple model

The idea behind this report was to study how a super simple model would behave in comparison to a more sophisticated model. While reading reports it is noted over and over again that such is the case, a more detailed model is always better, but it is hard to find documentation of simple models and their performance. That is what is going to be tested with this model. Will a simple point mass model be enough to control a vehicle around a race track in a satisfactory manner?

5.2 Model and Constraints

The kinematic bicycle model that are used in most MPC formulations is a really good approximation of the car at low speeds and no slip angles. But as the speed increases the approximation is getting worse. The idea is that simulating the car as just a point mass and instead use the g-g diagram to describe its handling abilities could be almost as good. The model is derived in section 3 and is written as,

$$x_{k+1} = \left(I + T_s \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \right) x_k + \left(T_s \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \right) u_k. \tag{54}$$

The maximum force tyres can produce is limited by the friction between them and the ground. A common way to describe this is the so called g-g diagram and is shown in Figure 20.

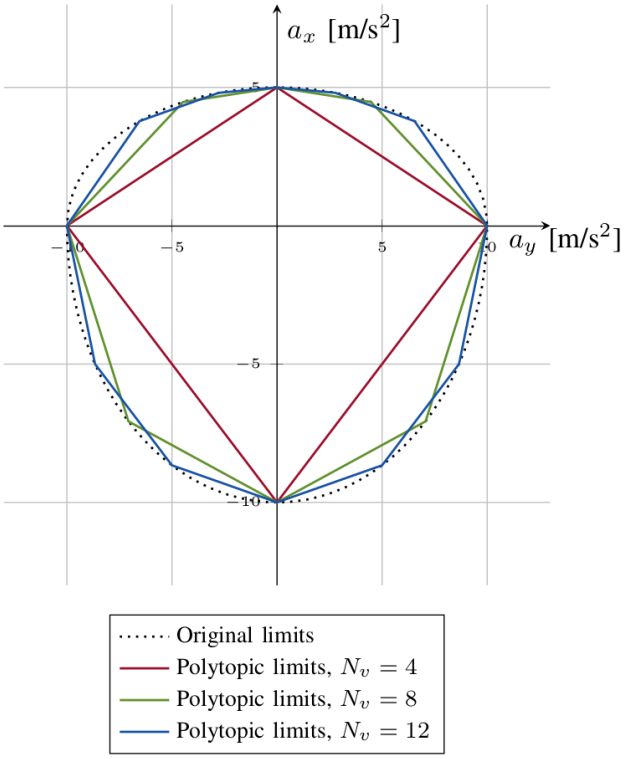

Figure 20: The g-g diagram with some polytope approximations

The upper part of the g-g diagram, e.i. the acceleration is typically distorted compared to the deceleration because of limiting factors of the vehicle. These are because limiting factors like engine power at higher speeds. To be able to constrain the point mass model to what the vehicle actually can handle a approximation of the g-g diagram is created as an inscribed polytope with N_v vertices, whom are also depicted in Figure 20. Using these the acceleration limits can be modeled as N_v linear constraints. The polytope are on the form $a_x = p_{m,j}a_y + p_{b,j}$ where the $p_{m,j}$ is the slope and $p_{b,j}$ is the y-intercept of j-th edge, and $j = 0, ..., N_v$. Since the orthogonal accelerations are inputs to the model the expression limited by these constraints are,

$$a = \sqrt{a_x^2 + a_y^2}. \tag{55}$$

To be able to implement it into the MPC formulation and still have a convex problem the expression has to be linearised. The lateral acceleration is

$$v_{x,max} = \sqrt{\frac{a_{y,max}}{|\kappa|}}. \tag{56}$$

Linearising the expression gives

$$v_{x,max} \approx -\frac{\sqrt{a_{y,max}} \cdot \bar{\kappa}}{2|\bar{\kappa}|^{\frac{5}{2}}} \cdot \kappa + \sqrt{\frac{a_{y,max}}{|\bar{\kappa}|}} + \frac{\sqrt{a_{y,max}} \cdot \bar{\kappa}^2}{2|\bar{\kappa}|^{\frac{5}{2}}} \tag{57}$$

The constraint limiting the length of the acceleration vector is chosen by investigating the $sign(a_x)$ and $sign(a_y)$. The polytope used in this report is made with $N_v = 4$.

5.3 Cost Function

The cost function method of maximizing the progress along the center line, from section 4, is intuitive and together with the quadratic term of the Taylor expansion in e_ψ gives a convex cost function and therefore computationally effective ways of finding a maximum exists.

$$Minimize: \frac{(1 - e_y\kappa)}{v} e_\psi^2 - \frac{\kappa}{v} e_y - \frac{(1 - e_y\kappa)}{v^2} v \tag{58}$$

5.4 Low Level Control

The trajectory that is generated from the MPC is used as a reference for the low level control system that is supposed to track it using a lateral and a longitudinal controller.

5.4.1 Pure Pursuit

One of the oldest and most used version of lateral control of vehicles is the so called pure pursuit and is described in [29]. It has been proven several times to deliver satisfactory tracking and among those are two cars that finished the DARPA Grand Challenge and three cars in the DARPA Urban Challenge [1]. The controller is designed to draw a circle segment from the vehicles current position to the point on the track that it is tracking. A schematic illustration of this is given in Figure 21.

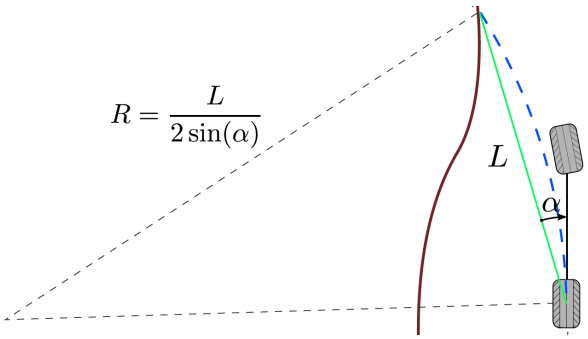

Figure 21: The logic behind the pure pursuit controller, the circle segment is fit between the car position and the tracking point one lookahead distance L away along the reference path.

The reference point is chosen to be one lookahead distance L in front of the car. The circle segment that is drawn is always tangent to the car's heading and the curvature is given by,

$$\kappa_c = \frac{2\sin\alpha}{L}. \tag{59}$$

The change in heading can be written as,

$$\omega = \frac{2v_r \sin\alpha}{L}, \tag{60}$$

for a certain vehicle speed v_r. The lookahead distance L is a control parameter that can be chosen to a fixed value or set to a function dependent on other parameters. A short lookahead distance can impose oscillations on a straight road and a too long lookahead distance will make the car "cutting corners". A common way to solve this is to have the lookahead distance dependent on the speed.
Once the lookahead distance is decided the tracking point on the reference has to be find. This is done by finding the $x_{ref}(s)$ and $y_{ref}(s)$ points that satisfy equation 61,

$$\|(x_{ref}(s), y_{ref}(s)) - (x, y)\| = L. \tag{61}$$

where s is the coordinate along the track and x and y are the global coordinates of the vehicle. There is usually at least 2 points where equation 61 is fulfilled and to be sure to choose the point that is ahead the one with the highest value of s is taken.
Once the reference point is found the angle α is found from the relationship,

$$\alpha = \arctan\left(\frac{y_{ref} - y}{x_{ref-x}}\right) - \psi, \tag{62}$$

The steering angle can finally be derived as,

$$\delta = \arctan(\kappa_c L), \tag{63}$$

which is sent to the simulated vehicle.

5.4.2 Proportional Integral Control

The change in speed is controlled by a controller type that is used in a majority of all systems that needs to be controlled, namely the family of PID controllers. The basic feedback loop is shown in Figure 22.

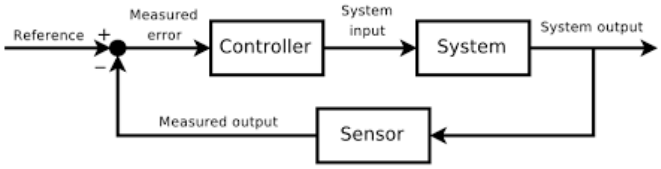

Figure 22: The feedback loop of the speed controller

The measured output is the speed of the Formula Student car or the speed from the dynamic bicycle model in the simulations. This one is compared to the speed wanted at this instant which is calculated in the MPC optimization and becomes the measured error, hence the name error feedback. The error is then sent to the controller which is built up as,

$$u_{speed} = K_p e_k + T_S K_i (e_{k-1} + e_k) \tag{64}$$

which is a PI-controller in discrete form.

5.5 Final Formulation of the 2-layer MPC

Putting the system together the top level MPC controller is written as,

$$\begin{aligned}
Minimize\ & \sum_{k=0}^{N-1} \frac{(1-e_y\kappa)}{v} e_\psi^2 - \frac{\kappa}{v} e_y - \frac{(1-e_y\kappa)}{v^2} v + \beta\sigma^T\sigma \\
Subject\ to: & \ x_{t+1} = A_k x_k + B_k u_k + g(\bar{x}_k, \bar{u}_k) \\
& \ x_0 = x(t) \\
& \ y_{min} - \sigma_y \leq y \leq y_{max} + \sigma_y \\
& \ v_{min} \leq v \leq v_{max} \\
& \ \text{if } sign(a_x) \leq 0,\ a_x \leq (p_{m,j} a_y + p_{b,j})_{max} \\
& \ \text{if } sign(a_x) \geq 0,\ -a_x \leq (p_{m,j} a_y + p_{b,j})_{max} \\
& \ \text{if } sign(a_y) \leq 0,\ a_y \leq (p_{m,j} a_y + p_{b,j})_{max} \\
& \ \text{if } sign(a_y) \geq 0,\ -a_y \leq (p_{m,j} a_y + p_{b,j})_{max} \\
& \ \sigma_y \geq 0 \\
& \ k = 0, ..., N-1
\end{aligned} \tag{65}$$

with the state and input vectors,

$$x = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} \quad u = \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} \tag{66}$$

The optimized trajectory is then sent to the pure pursuit controller described in 5.4.1 for lateral control and to the PI-controller for longitudional control. In the next chapter this 2-level controller is compared to the 1-level MPC controller derived in section 4.

6 Results

In this section the way the controllers were analysed will be described to make it understandable how the structure of the simulations is built up. Then the MPC's from the two different controllers will be compared to analyse their strengths and weaknesses. Following is the comparable performance of the full control logic.

6.1 Experimental setup

6.1.1 Hardware and software

To simulate the behaviour of the vehicle subject to the control inputs a simulation environment had to be developed. The environment was chosen to be programmed in Python because implementation in ROS is a future project in mind. All simulations were run on a laptop with a Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz, 8GB memory and using Pycharm as the environment. To solve the trajectory optimization problem the Python embedded modeling language for convex optimization CVXPY [44] was used. It enables the user to specify the MPC problem in an analytic way instead of having to learn the specific way a certain solver wants the problem to be depicted in. CVXPY offers several different solvers and the one used is SCS [45] [46] which is a is a numerical optimization package for solving large-scale convex cone problems.

6.1.2 The track

The rule framework in Formula Student is extensive and also include constraints on the competition track. The track has to be built according to these points:

- 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.
- The minimum track width is 3.5 m
- The length of one lap is approximately 200 m to 500 m

A complete such a track was generated and is shown in Figure 23 in order to prove that the vehicle could handle the track conditions necessary.

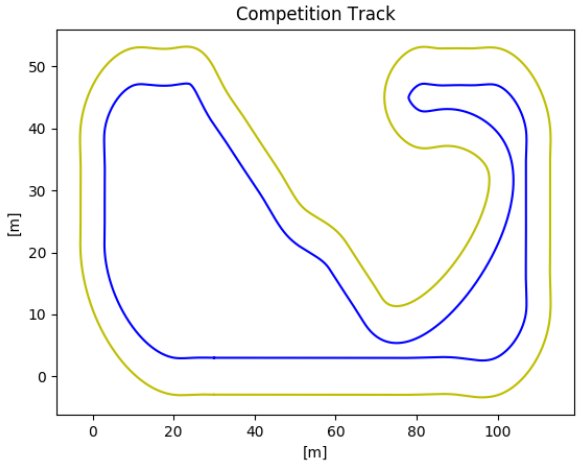

Figure 23: A complete Formula Student competition track with all properties needed to test. The length of the track is 375 [m]

To be able to analyse the behaviour and compare in a more precise manner only small sections will be used. The two sections chosen was a 90 degree corner and double u-turn. Both seen in Figure 24 and Figure 25, also following the regulations of the Formula Student rules.

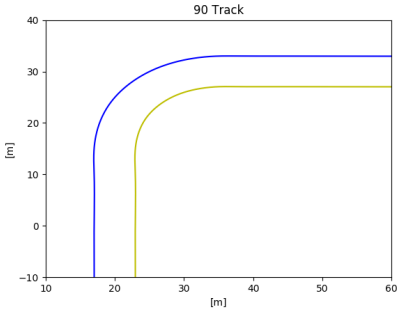

Figure 24: The 90 degree turn

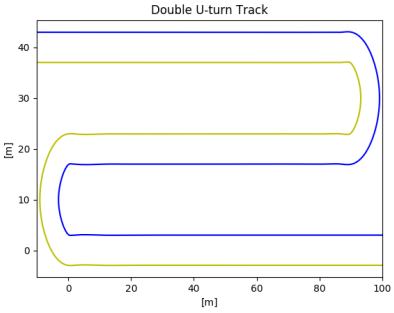

Figure 25: The double U-turn track

The blue and yellow color of the track boundaries corresponds to the fact that the Formula Student Driverless Track is bounded by blue cones on the left side and yellow cones on the right side.
The track is created and a function is interpolating the user defined points creating a matrix with the information about corresponding Global and Frenet coordinates as well as the curvature and heading angle of the track in each point. This is made with a step length of 0.1 [m].

6.1.3 The simulated vehicle

The simulated vehicle is the KTH Formula Student car eV14 seen in Figure 3 with the vehicle parameters depicted in Table 1.

Table 1: The simulated car eV14 in figures

m [kg]	I_z [kgm^2]	L [m]	λ	l_r [m]	l_f [m]	$C_{\alpha f}$ [N/rad]	$C_{\alpha r}$ [N/rad]	Width [m]
212	281	1.530	0.52	0.7344	0.7956	74537	62385	1.304

It is important to note that the model used as the simulation model of the actual Formula Student car is based on the nonlinear dynamic bicycle model presented in section 3. Since the prediction model in the MPC is the kinematic bicycle model, there will be a difference and an analysis of how the controller performs with respect to the error is possible. Other parameters defined by the vehicle performance that are used as constraints in the optimization problem are shown in Table 2.

Table 2: Vehicle performance parameters and physical constraints

$a_{xmax/min}$ [m/s^2]	$a_{ymax/min}$ [m/s^2]	$\kappa_{max/min}$ [m^{-1}]	$e_{ymax/min}$ [m]
$\pm$ 10	$\pm$ 12	$\pm$ 0.15	$\pm$ 3
$e_{\psi max/min}$ [rad]	v_{max} [km/h]	$J_{max/min}$ [m/s^3]	C_{max} [ms^{-1}]
$\pm\pi/4$	150	20	0.1

The simulations will be ran with a horizon length of $N = 20$ and for the single layer MPC that is using a fixed step length that length is set to $\Delta S = 4$[m]. The double layer MPC is running with a fixed time step that is set to match the fixed step at top speed $\frac{\Delta S}{v_{max}} \approx 0.1 = T_s$.

6.1.4 Code structure

To start the model a so called warm start is used, which means that feasible state and input predictions are given to the MPC. After the first iteration the effects of this warm start is gone and the simulation runs as planned. In every iteration step is the predicted states and inputs (N and $N-1$ by nature) saved and used as the linearisation points in the next iteration. Since the travel of the car will take it to roughly the position of the first predicted state, the best estimation is if the predictions are shifted one step and the last entries of both the states and input vectors are appended to increase the length of the respective vector. This works since all states are relative measures not dependent on where the car is on the track, except for the first two states in the simple model, there an extension is calculated based on the previous velocity and sampling time.
The single layer MPC gets its inputs directly from the MPC and the simulated vehicle applies the control input for one iteration. The resulting state is then taken in the global coordinate

frame and derived back to the road following Frenet frame in the manner presented in section 2 where the MPC uses it for the following iteration. The double layer MPC is feeding the planned trajectory to the Pure Pursuit controller which is calculating the steering input and the PI controller which gives the acceleration input. As in the single layer MPC the control input is sent to the simulation model and the new states back to the MPC for the next iteration.

6.2 MPC layer comparison

To be able to track down the performance of every layer of the controller the two trajectory planners will be compared. Even though one gives the direct input signals to the actuators and the other needs a trajectory following layer it is of interest to study if the hypothesis that a simpler model will reduce the computational times and since it is stated in several reports [38], [41], [31] that the kinematic bicycle model is only valid at low speeds it might be as good to use the properties of the center of gravity of the car as constraints and model instead of mimicking some of the dynamics.

6.2.1 The 90 degree turn comparison

Below are the two controllers compared side by side in a 90 degree turn. The green color indicates that the trajectory planner is requesting acceleration and red indicates that braking is initiated.

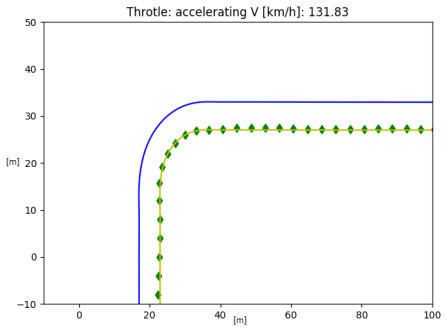

Figure 26: The 90 degree turn with the single layer MPC

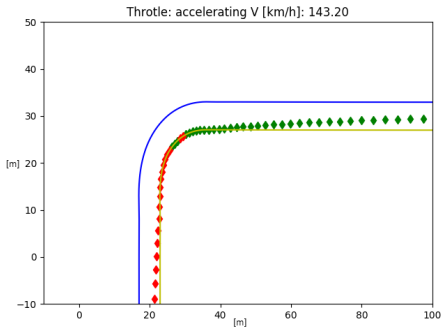

Figure 27: The 90 degree turn with the double-layer controller

Both x- and y-axis of the figures are in meters. The simulated trajectory planners are starting at the center line outside of the figure window. It is possible to see that the 1 layer MPC, Figure 26, with it's fixed step length moves to the inside of the track as soon as it recognizes a right turn is coming up. And since the model is based on the kinematic bicycle model is the road-tyre friction assumed to be perfect at all times. This means that even though it mimics the handling

dynamics it does not take in consideration the tyre limits and therefore not the limitations on the lateral accelerations.
This is though observed in the point mass model in Figure 27, it keeps a line that is almost inner line all the time but since the point is only constrained by the G-G diagram it realises that the speed it is going to enter the curve with is too high to continue on the course and therefore chooses to brake. As soon as it exits the curve it accelerates again.
Figure 28 and Figure 29 show the current prediction horizon of the two controllers on the top, for the velocities and the longitudinal accelerations, below is the velocity and acceleration plotted over the traveled distance. The green dots are the previous predictions and therefore the linearization points, the black ones are the current predictions.

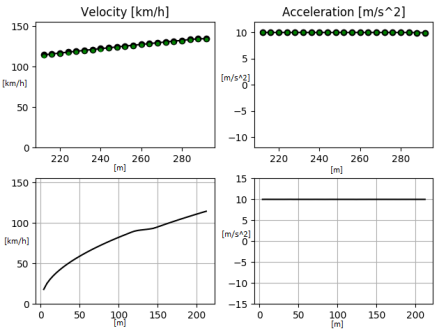

Figure 28: The 90 degree turn with the single layer MPC

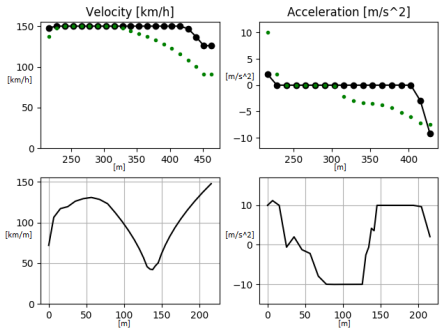

Figure 29: The 90 degree turn with the double-layer controller

Here it is possible to observe the constant increase in speed for the single layer MPC. The double layer MPC shows the more expected behaviour of braking into the turn. The oscillatory behaviour of the controller before the entrance of the turn is due to the fact that the step length is varying, and the time is constant. When the prediction horizon reaches the turn it realises that it has to brake so much that the next prediction does not reach the turn at all, hence it has fixed time steps, giving higher optimal speed again. This goes on until the predicted horizon is enough to see the corner at each update.

Figure 30 and Figure 31 shows the absolute value of the total acceleration of the vehicle.

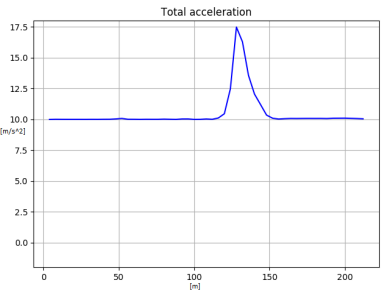

Figure 30: The 90 degree turn with the single layer MPC

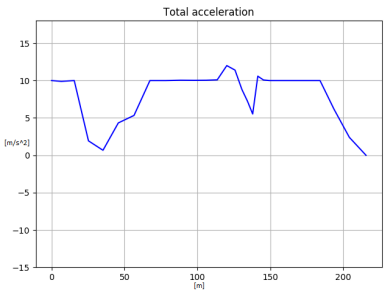

Figure 31: The 90 degree turn with the double-layer controller

In Figure 30 the single layer MPC maximizes the longitudinal acceleration to around 10 [m/s^2] at all times and therefore when cornering the acceleration reaches a physically infeasible 17.5 [m/s^2]. This is due to that the model does not have any constraints on the lateral acceleration. The total acceleration of the double-layer controller with the point mass model is shown in Figure 31. The model fulfills the constraints at almost every instant except at one point where the slack is active.
In Figure 32 the longitudinal, lateral and absolute total acceleration is depicted on top of each other for the double layer controller.

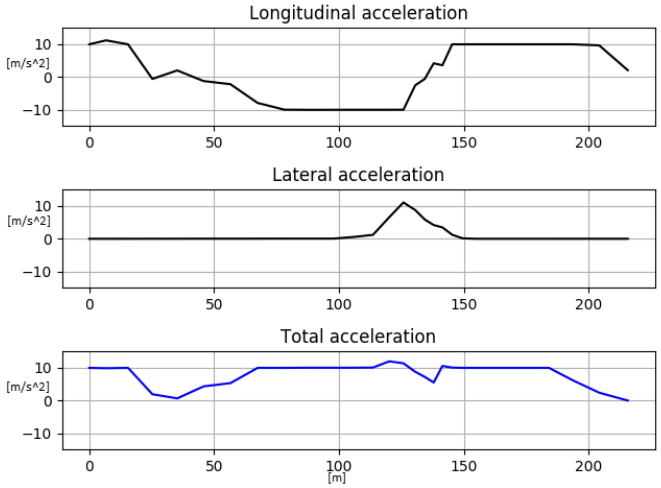

Figure 32: Longitudinal, lateral and absolute total acceleration over the 90 degree turn

It is possible to see that the lateral acceleration increases in the turn and at the same time the longitudinal acceleration is close to zero.

6.2.2 The double U-turn

The two trajectory planners are put to the test in the double U-turn track in Figure 33 and Figure 34. The red and green diamonds are deceleration points and acceleration points respectively and the round connected red dots are the horizon of the MPC predictions.

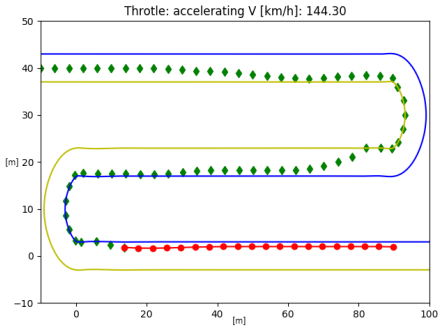

Figure 33: The double U-turn with the single layer MPC

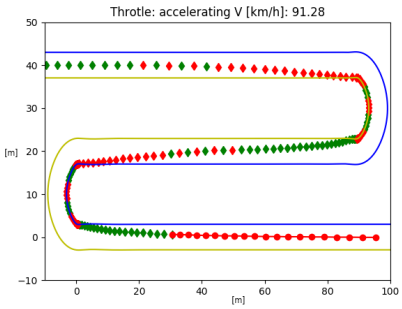

Figure 34: The double U-turn with the double-layer controller

Similar performance analysis can be done in this case as well. The single layer kinematic bicycle model in Figure 33 is accelerating at all times but the constrained point mass model in Figure 34 is deceleration when entering the turn and accelerating out of the turn. Note that the red circular dots with connecting lines in the end of the straight is the prediction horizon and not deceleration!
The corresponding acceleration and velocity predictions (top) and states over the full traveled distance (bottom), for the single layer MPC is presented in Figure 35 and for the MPC in the double layer controller in Figure 36.

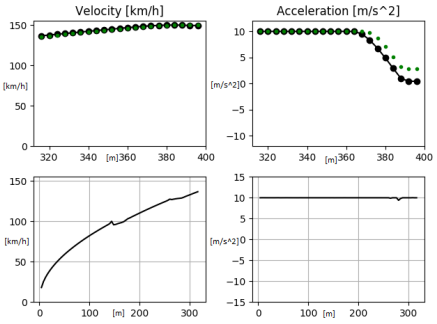

Figure 35: The double U-turn with the single layer MPC

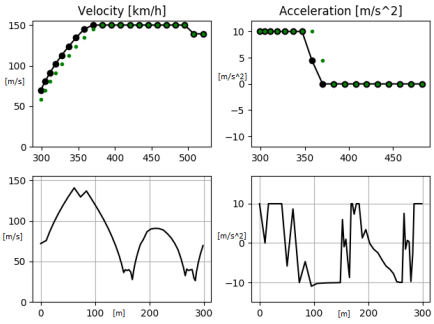

Figure 36: The double U-turn with the double-layer controller

Figure 37 and Figure 38 shows the total accelerations of the double U-turn of the corresponding models.

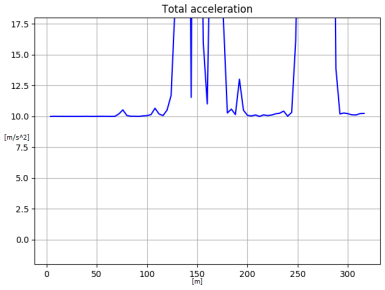

Figure 37: The double U-turn degree turn with the single layer MPC

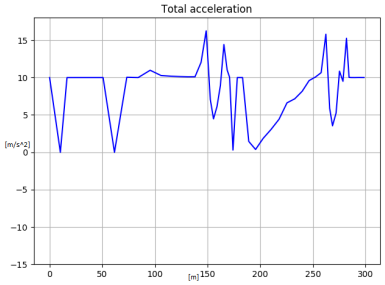

Figure 38: The double U-turn turn with the double-layer controller

Figure 39 shows the longitudinal, lateral and total acceleration of the double-layer controller.

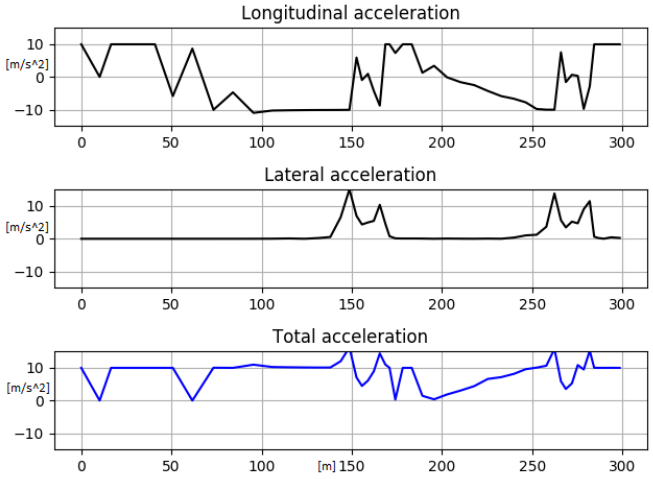

Figure 39: Longitudinal, lateral and absolute total acceleration over the double U-turn for the double-layer controller

6.3 Full controller performance comparison

To study if the controllers were able to control a simulated vehicle the control input was sent to a dynamic vehicle model. In Figure 40 and Figure 41 the result of the single layer controller is shown.

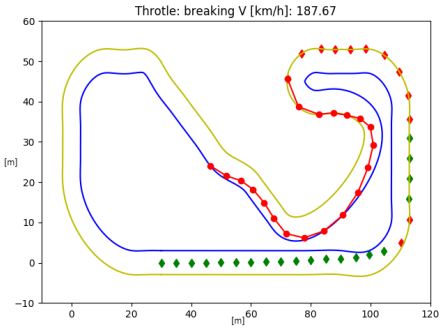

Figure 40: The single layer MPC trying to control the simulated Formula Student vehicle

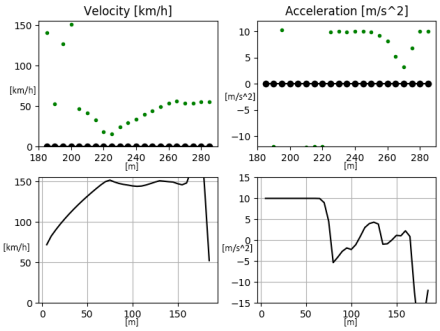

Figure 41: Velocity and acceleration of the single layer MPC trying to control the simulated Formula Student vehicle

Since the controller does not take lateral acceleration into account the controller is not slowing down until it is too late. The simulated vehicle is contrary to the prediction model, and in accordance to a real vehicle, in need of lower velocities to handle the curves of the track.
The two layer MPC on the other hand uses the G-G constraints to limit the acceleration when necessary. Figure 42 shows a full lap around the track with the two layer controller. Beside in Figure 43 is the velocity and acceleration analysis, showing that the controller keeps the vehicle within the constraints at all times.

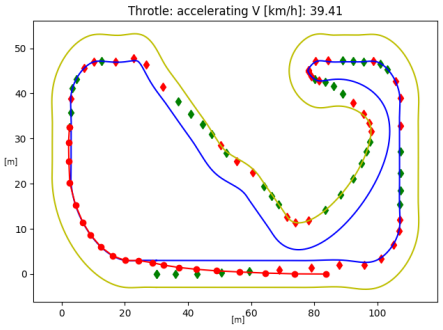

Figure 42: The two layer MPC controlling the simulated Formula Student vehicle

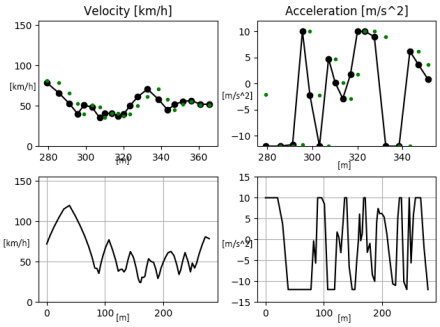

Figure 43: Velocity and acceleration of the two layer MPC controlling the simulated Formula Student vehicle

7 Conclusions and future work

With the intention of testing if a simple model could work for controlling of a vehicle performing highly dynamic tasks a two layer controller was developed. To prove its performance a published controller was taken as reference. This was a one layer MPC with 5 states and 2 inputs. It has to be noted that the document providing the theory behind the one layer MPC needed small corrections but the controller used in this report was designed with the initial idea kept in mind.

After analysing the simulation results it is hard to believe that the single layer MPC would work in the racing case without adding constraints on the lateral acceleration. Right now the kinematic bicycle model allows the predictions to run at speed way above whats physically possible. This model is according to these results too simple to handle the case at all.

On the other hand was the point mass model with only constraints on acceleration proven to work. It chose more of a shortest path than the optimal racing line, but it follows the track in an efficient manner and stays stable. The ability to increase the radius of corners and choose a more racing like line was expected. There might be errors embedded in the way the code is structured but after several iterations nothing could be found. For exact details the code can be found and downloaded at [47] and [48]. To better have prevented possible faults like this would greater knowledge within different programming languages and code structure be of importance. An external code reviewer should also be in place in future work as that is seen as common practice in the programming industry but was not utilised in this report.

To lower the computational times at still acceptable results was one of the main reasons behind this report. But as it was shown that one controller could not handle it, the time comparison was omitted. During the work it could though be seen that the times were very similar. The reason is believed to be that the simpler point mass model had more complex constraints instead. The take-away of this is that just because the elimination of the complexity at one place does not mean that the problem is gone, more likely the problem is moved to somewhere else.

A more optimal solution would though be to combine the kinematic movements of the bicycle model and the constraints on the accelerations into one single MPC, that is to combine the best of the two controllers developed in this thesis. This would eliminate the lower control layer and is still efficient.

The simulations were ran with the CVXPY library that has to setup the problem formulation in each iteration. This makes the computations take around half a second per iteration. There is a version of this library which is called CVXGEN that generates the problem once and then the values in the different matrices are only shifted, making it possible to reduce the computational times with one order of magnitude. This this is of course necessary to make the system run real time.

All in all the developed control system seems to perform one step closer to a optimal racing line compared to following the center-line but further development regarding the completion of a full model that has both the kinematics of a car as the bicycle model and the acceleration constraints of a g-g diagram and future implementation in the real Formula Student vehicle would be interesting to follow.

References

[1] M. Buehler and K. Iagnemma, *The DARPA Urban Challenge*. Springer, Berlin, Heidelberg, 2009, vol. 56.

[2] J. Ziegler and P. Bender, "Trajectory planning for bertha-a local, continuous method," *IEEE Intelligent Vehicles Symposium*, vol. IV, pp. 450–457, June 2014, dearborn, Michigan, USA.

[3] F. S. Germany, "Formula student rules 2018 v1.1," pp. 0–130, 2018.

[4] M. Zelinger and R. Hauk, "Design of an autonomous race car for the formula student driverless (fsd)," 2017.

[5] M. Valls and H. Hendrikx, "Design of an autonomous racecar: Perception, state estimation and system integration," *ROSCON*, vol. 1, pp. 1–8, April 2018.

[6] P. Rawat, "Vehicle dynamics project course part i and ii," Project Course Report, The Royal Institute of Technology, January 2018, pp. 237-255.

[7] P. A. Ioannou, "Autonomous intelligent cruise control," *IEEE Transactions on Vehicular Technology*, vol. 42, no. 4, pp. 657–672, November 1993.

[8] K. Osman and M. F. Rahmat, "Modelling and controller design for a cruise control system," *International Colloquium on Signal Processing and Its Applications*, no. 5, pp. 254–258, 2009.

[9] L. Xiao and F. Gao, "A comprehensive review of the development of adaptive cruise control systems," *Vehicle System Dynamics,*, vol. 48, no. 10, pp. 1167–1192, October 2010.

[10] P. Shakouri and J. Czeczot, "Simulation validation of three nonlinear model-based controllers in the adaptive cruise control system," *J Intell Robot Syst*, no. 80, pp. 207–229, October 2015.

[11] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 3, pp. 143–153, September 2003.

[12] Z. L. Ye Li and H. Wang, "Evaluating the safety impact of adaptive cruise control in traffic oscillations on freeways," *Accident Analysis and Prevention*, vol. 104, pp. 137–145, May 2017.

[13] D. Cualain and C. Hughes, "Automotive standards-grade lane departure warning system," *IET Intelligent Transport Systems*, vol. 6, no. 1, pp. 44–57, March 2012.

[14] N. C. Basjaruddin and Kuspriyanto, "Lane keeping assist system based on fuzzy logic," *International Electronics Symposium*, vol. 1, pp. 110–113, April 2015.

[15] J.-Y. Hsu and K.-L. Ku, "Integration and implementation of a lane keeping system with vehicle dynamics control," Report, Automotive Research and Testing Center, Changhua County, TAlWAN, 2016.

[16] C. M. Kang and J. Leel, "Lateral control for autonomous lane keeping system on highways," *International Conference on Control, Automation and Systems*, vol. 15, pp. 1728–1733, October 2015.

[17] M. Nolte and M. Rose, "Model predictive control based trajectory generation for autonomous vehicles an architectural approach," Report, Technische UniversitÃ¤t Braunschweig, Germany, 2017.

[18] M. Werling and J. Ziegler, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," *International Conference on Robotics and Automation*, pp. 987–993, May 2010.

[19] M. Werling and S. Kammel, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 346–359, 2011.

[20] L. del Re and F. Allgower, *Automotive Model Predictive Control, Models, Methods and Applications*. Springer-London, 2010, vol. 402.

[21] P. Belven, "Implementation of model predictive control for path following with the kth research concept vehicle," Master Thesis TRITA EE 2015-77 ISSN 1653-5146, KTH, Stockholm, Sweden, 2015.

[22] J. Fredlund and K. Sulejmanovic, "Autonomous driving using model predictive control methods," Master Thesis, LTH, Department of Automatic Control, Lund University, Sweden, 2017.

[23] D. Lam and C. Manzie, "Model predictive contouring control," *Conference on Decision and Control*, vol. 49, pp. 6137–6142, December 2010.

[24] T. Weiskircher and B. Ayalew, "Predictive adas: A predictive trajectory guidance scheme for advanced driver assistance in public traffic," *European Control Conference*, vol. 1, pp. 3402–3407, July 2015.

[25] U. Rosolia and S. D. Bruyne, "Autonomous vehicle control: A nonconvex approach for obstacle avoidance," *Transactions on Control Systems Technology*, vol. 25, no. 2, pp. 469–484, March 2017.

[26] X. Qian and A. D. L. Fortelle, "A hierarchical model predictive control framework for on-road formation control of autonomous vehicles," *IEEE Intelligent Vehicle Symposium*, vol. 4, pp. 376–381, June 2016.

[27] P. F. Lima, "Predictive control for autonomous driving, with experimental evaluation on a heavy-duty construction truck," Licenciate Thesis, KTH, Stockholm Sweden, May 2016.

[28] B. Nordell, "Trajectory planning for autonomous vehicles and cooperative driving," Master Thesis TRITA EE 2016:119 ISSN 1653-5146, KTH, Stockholm Sweden, 2016.

[29] B. Paden and M. Cap, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, April 2016.

[30] P. Polack and F. Altche, "The kinematic bicycle model: a consistent model for planning feasible trajectories for autonomous vehicles?" *IEEE Intelligent Vehicles Symposium*, vol. 4, pp. 812 – 818, June 2017.

[31] S. van Koutrik, "Optimal control for race car minimum time maneuvering," Master of Sience Thesis, TUDelft, March 2015.

[32] V. Laurense and J. Goh, "Path-tracking for autonomous vehicles at the limit of friction," *American Control Conference*, pp. 5586–5591, May 2017.

[33] A. V. C. at the Limits of Handling, "Krisada kritayakirana," Doctoral Thesis, Stanford, June 2012.

[34] G. Perantoni and D. Limebeer, "Optimal control for a formula one car with variable parameters," *Vehicle System Dynamics*, vol. 52, no. 5, pp. 653–678, 2014.

[35] R. Vesel, "Racing line optimization @ race optimal," *ACM SIGEVOlution*, vol. 7, no. 2-3, pp. 12–20, 2015.

[36] D. Kellya and R. Sharp, "Time-optimal control of the race car: a numerical method to emulate the ideal driver," *Vehicle System Dynamics*, vol. 48, no. 12, pp. 1461 – 1474, December 2010.

[37] A. Rucco and G. Notarstefano, "An efficient minimum-time trajectory generation strategy for two-track car vehicles," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 4, pp. 1505–1519, July 2015.

[38] F. Curinga, "Autonomous racing using model predictive control," Master Thesis TRITA 2017:175 ISSN 1653-5146, KTH, Stockholm, Sweden, January 2018.

[39] U. Rosolia and A. Carvalho, "Autonomous racing using learning model predictive control," *American Control Conference (ACC)*, pp. 5115 – 5120, May 2017.

[40] M. Brunner and U. Rosolia, "Repetitive learning model predictive control: An autonomous racing example," *IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 2545 – 2550, December 2017.

[41] A. Liniger and A. Domahidi, "Optimization-based autonomous racing of 1:43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, pp. 628–647, July 2014.

[42] E. Thilen, "Robust model predictive control for autonomous driving," Master Thesis TRITA EE 2017:090 ISSN 1653-5146, KTH, Stockholm Sweden, 2017.

[43] M. Johansson, "El2700 model predictive control," Lecture Notes, KTH, 2017.

[44] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[45] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, "Conic optimization via operator splitting and homogeneous self-dual embedding," *Journal of Optimization Theory and Applications*, vol. 169, no. 3, pp. 1042–1068, June 2016. [Online]. Available: http://stanford.edu/~boyd/papers/scs.html

[46] ——, "SCS: Splitting conic solver, version 2.0.2," https://github.com/cvxgrp/scs, Nov. 2017.

[47] M. Ahlberg, "1 layer mpc code," https://github.com/maxahlberg/MPC_1Layer_5x2u, Jun. 2018.

[48] ——, "2 layer mpc code," https://github.com/maxahlberg/MPC_2Layer_4x2u, Jun. 2018.