DEGREE PROJECT IN VEHICLE ENGINEERING,
SECOND CYCLE, 30 CREDITS
*STOCKHOLM, SWEDEN 2019*

# Path planning and trajectory generation - model predictive control

**TOBIAS IDMAN**

**KTH ROYAL INSTITUTE OF TECHNOLOGY**
**SCHOOL OF ENGINEERING SCIENCES**

## Abstract

Autonomous vehicle technology is a rapidly expanding field that will play an important role in society in the future. The ambition of autonomous technology is to improve safety for drivers, passengers and pedestrians, reduce traffic congestion and lower fuel consumption. To reach these goals advanced driver-assistance systems and autonomous driving aid the driver or takes full control over the vehicle. With the help of computers, a more optimal driving style is implemented.

This thesis focuses on implementing lateral and longitudinal control for an autonomous vehicle with the help of model predictive control using artificial potential fields. The model predictive controller foresees the future for a finite time horizon using a mathematical model of the vehicle. Here a linearized and discretized kinematic bicycle model is used as the internal vehicle model in the controller. The model predictive controller avoids obstacles with artificial potential fields that are quadratically approximated using Taylor series with the help of Bézier curves. The control inputs are the vehicle's side slip angle and the longitudinal acceleration. The simulations take place on roads in the United Kingdom with left-hand traffic in urban environments. The simulated use-cases are limited to roadside parking scenarios with and without traffic. The goal is to follow the lane center while navigating around obstacles in a predictable way, respecting traffic laws and avoiding collisions or hazardous situations.

A simplistic decision-making module is used to determine the vehicle's next course of action. After the subsequent maneuver is decided a velocity profile and a lane center reference are created for the ego vehicle to follow. The model predictive controller solves the optimization problem as a quadratic programming problem that minimizes a cost function while satisfying a set of constraints. The cost function minimizes the error for the velocity, the lateral lane position, the control inputs, the control input's rate of change and the yaw angle. The artificial potential fields are also included in the cost function to guide the ego vehicle away from high cost regions.

The algorithms were built and simulated using MATLAB. The quadratic programming problem was solved using MATLAB's *quadprog* routine in the optimization toolbox, with a sample time of 0.1 seconds. The MPC developed good results with quick calculation times. The highest average calculation time was 0.0159 s with a maximum calculation time of 0.0549 s. The vehicle could perform overtaking maneuvers of two parked cars at 6 m/s in 8.4 s and in 17.2 s when overtaking from a standstill.

## Sammanfattning

Autonoma fordonssytem är ett snabbt växande område som kommer att spela en stor roll i vårt samhälle i framtiden. Ambitionen för autonoma fordonssytem är att förbättra säkerheten för förare och passagerare, reducera trafik samt sänka bränsleförbrukningen. För att nå dessa mål kan aktiva hjälpsystem och autonoma fordonssystem ta delvis eller full kontroll över fordonet. Datorer kan sedan beräkna en optimal kontrollstrategi som kan bidra till en bättre och säkrare körning.

Detta examensarbete fokuserar på att implementera lateral och longitudinell kontroll för ett autonomt fordon med hjälp av modell-prediktiv reglering (model predictive control) och artificiella potentiella fält. Där modell-prediktiv reglering förutspår framtiden för en finit tidshorisont med hjälp av en matematisk fordonsmodell. En linjäriserad och diskretiserad kinematisk cykelmodell används som den interna fordonsmodellen i regulatorn. Modell-prediktiv reglering undviker hinder genom artificiella potentiella fält som är kvadratiskt approximerade med Taylor approximering med hjälp av Bézier kurvor. Regulatorns styrsignaler är fordonets avdriftsvinkel och acceleration. Simuleringarna är baserade på vägar i Storbritannien med vänstertrafik i stadsmiljö. De simulerade scenariona är begränsade till väggrensparkering med och utan trafik. Där målet är att följa filens mittlinje samt att navigera runt hinder på ett tydligt sätt, respektera trafiklagarna samt undvika kollisioner och farliga situationer.

En enkel besluts-modul används för att bestämma fordonets nästa manöver. Efter manövern är bestämd skapas en hastighetsprofil samt en mittfilsreferens för fordonet att följa. Regulatorn löser optimeringsproblemet med hjälp av kvadratisk programmering som minimerar en kostnadsfunktion och samtidigt håller sig inom definierade gränsvärden. Kostnadsfunktionen minimerar differensen för hastigheten, laterala positionen, styrsignalerna, styrsignals ändringen samt girvinkeln. Även de artificiella potentiella fälten är inkluderade i kostnadsfunktionen för att leda fordonet från högkostnadsregioner.

Det kvadratiska programmeringsproblemet beräknades i MATLAB's *quadprog* rutin från optimerings verktyget med en samplingstid på 0,1 sekunder. Regulatorn visade goda resultat med snabba beräkningar. Det högsta medelvärdet på beräkningstiden var 0,0159 s med ett maximum på 0,0549 s. Fordonet kunde göra omkörningsmanövrar runt två på raken parkerade bilar med en hastighet på 6 m/s på 8,4 s och på 17,2 s när omkörningen gjordes från ett stillastående tillstånd.

## Acknowledgement

I would like to thank the people who have helped me in my work with my master thesis, without them this thesis would not have been possible. First, I would like to thank my supervisor at AVL Coventry, Huaji Wang, for taking the time to support me with my thesis. I enjoyed our weekly meetings, even though they were short and concise, they proved to be essential for the progress of my thesis. I would like to show my gratitude towards Nina Neubauer for our discussions about model predictive control. Although these conversations usually ended with us both being more confused they eventually led me to become more knowledgeable within the field.

At KTH I would like to acknowledge my supervisor Michael Nybacka. Despite that our contact was brief his literature recommendations and advice along the way was highly appreciated.

I would like to thank AVL for giving me the opportunity to do my Master thesis with them within autonomous vehicle control. I would also like to thank them for the possibility to visit Coventry, UK for two weeks during my thesis to work directly with the team in Coventry. At AVL I want to give a special thanks to: Jayesh, Anthony, Tommie and Manas for making the office a great place to work. At times they helped me take a step back from my work, which made it easier to see the solutions to my problems. They also made it easier to get through the grey and dull winter in Gothenburg.

Lastly but certainly not least I would like to give my heartfelt thanks to my family and friends for believing in me and supporting me on my journey. They motivated me to take the leap and move to Gothenburg to embark on a career within autonomous vehicles.

Tobias Idman

Gothenburg, August 2019

# Table of content

## List of abbreviations

| | |
|---|---|
| ADAS | Advanced Driver-Assistance Systems |
| ACC | Adaptive Cruise Control |
| LKAS | Lane Keeping Assistance System |
| EBA | Emergency Brake Assist |
| AD | Autonomous Driving |
| DARPA | Defense Advanced Research Projects Agency |
| GPS | Global Positioning System |
| MPC | Model Predictive Control |
| APF | Artificial Potential Field |
| RHC | Receding Horizon Control |
| QP | Quadratic Programming |
| PID | Proportional-Integral-Derivative |
| LP | Linear Programming |
| NLMPC | Non-Linear Model Predictive Control |
| EMPC | Explicit Model Predictive Control |
| UK | United Kingdom |
| PMM | Point Mass Model |
| CoG | Center of Gravity |

## List of Figures

## List of Tables

# 1 Introduction

Autonomous vehicles have been a highly researched topic in the last decade because of the potential benefits of the technology. In recent years the automotive industry has seen rapid growth of Advanced Driver-Assistance Systems (ADAS). Some examples of ADAS technologies include: Adaptive Cruise Control (ACC), Lane Keeping Assistance System (LKAS) and Emergency Brake Assist (EBA). The main purpose of ADAS is to reduce the number of accidents and to minimize fuel consumption, these benefits are not only for the individual but for society as well. In 1997 the cost of traffic congestion in the Netherlands was estimated to be 0.8 billion Euros and the cost of accidents were estimated to be around 8 billion Euros. It was concluded that the majority of the accidents does not happen on motorways but on the underlying road networks in urban environments [14]. Furthermore, ADAS technologies have shown a fuel saving potential [7], which will in turn reduce air pollution.

The next progression is to move towards Autonomous Driving (AD) which takes full control over the vehicle and removes the need for a driver. The research within AD and ADAS has been stimulated by competitions such as: the DARPA grand challenge and the Grand Cooperative Design Challenge. The beginning of the DARPA grand challenge was in 2004 in the United States, where the task was to autonomously drive from Los Angeles to Las Vegas [2]. In 2009 the Grand Cooperative Design Challenge started with testing taking place on specially equipped public roads [4]. But the early stages of autonomous vehicle could be seen back in the 1970s, where an automatic steering system was implemented on a Citroen DS 19 [1]. With the trend of embeded hardware getting smaller, less expensive and more powerful, the possibilities for AD and ADAS are expanding. This expansion is made possible because more complex computations can be made in real time.

Autonomous vehicles use several different hardwares and softwares to control the vehicle. Generally, the process can roughly be divided into five different groups: perception, decision-making, reference generation, control and vehicle actuation. A general system architecture for an autonomous vehicle can be seen in Figure 1.1.

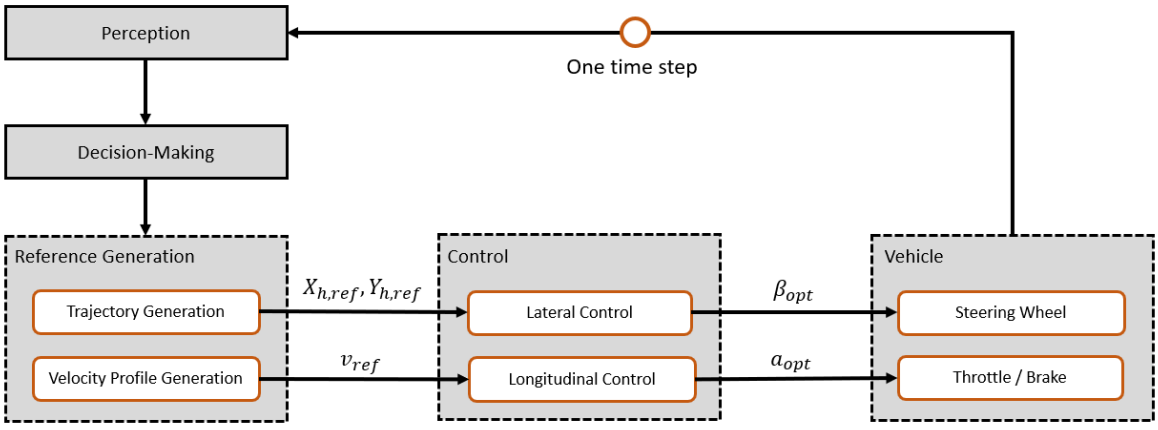


Figure 1.1: Autonomous vehicle system structure

With the help of sensor data such as: radar, lidar, GPS, cameras etc. an autonomous vehicle gets a perception of its environment and the surrounding obstacles. With this information the vehicle can predict where the obstacles will be in the future as well as seeing the roads curvature and boundaries up ahead. A decision can then be made about the host vehicle's (subsequently referred to as the ego vehicle) next course of action and the appropriate references to follow. The controller calculates the corresponding optimal control input to follow the desired references. The signal is sent to the steering wheel, throttle and brake for actuation and the loop starts over again at the perception block at each sampling time.

This thesis sets out to solve the problem of controlling an autonomous vehicle. It approaches the control problem by using Model Predictive Control (MPC) together with Artificial Potential Fields (APF). The main goal of this thesis is to investigate whether a linear MPC controller can be used for lateral and longitudinal control for low speed scenarios in urban environments.

## 1.1 Scope definition

This thesis mainly focuses on control, specifically MPC as a control strategy for autonomous driving. It investigates whether it is possible to use linear MPC for longitudinal and lateral control with efficient solving time and satisfying performance in urban environments. The controller will use a simple decision-making module to generate the desired maneuvers for the controller to track for the different use-cases. The driving takes place on low-speed British roads, with a maximum speed limit of 30 mph (48.2 km/h or 13.4 m/s), where the goal is to drive in a safe and comfortable manner. The control variables that is used to control the ego vehicle will be acceleration and slip angle. Several use-cases will be studied where the MPC controller must avoid obstacles in a predictive and safe manner.

An urban environment is more complex than highway driving with a higher likelihood of unpredictable events. Examples of such events could be: cyclists sharing the road, pedestrians jumping out in the road or busses leaving the bus station. There are also the infrastructure-based scenarios such as: roadside parking, traffic lights or narrow roads without centerline markings. Due to the complexity of urban driving it is decided that the controller needs to have a separate decision-making algorithm that decides the vehicle's general maneuver. For instance: staying in the original lane, overtaking a vehicle or braking to a complete stop.

To limit the scope of this thesis only a few selected use-cases will be studied, and they will be limited to roadside parking scenarios. The different use-cases are presented in detail in chapter 2.

## 1.2 Assumptions

This thesis makes some assumptions to limit the scope of the thesis. The limitations are listed below.

- This thesis will assume that the vehicle's current states and control input from sensor data and other vehicle systems are accurate and does not have any input delays.
- The driving takes place on urban roads where the speed does not exceed 30 mph, the roads are dry, and the weather condition is clear.
- The vehicle's control inputs, such as the throttle, brake and steering wheel angle are represented by the absolute acceleration and slip angle of the vehicle.
- The vehicle is assumed to only have front wheel steering.
- The control inputs are accurately actuated by the plant for each sampling time without delay.
- All the surrounding vehicles are detected by the ego vehicle and their movements predicted.
- The ego vehicle can detect the difference between a parked car and a moving vehicle.
- The road is straight, and the road boundaries and lane markings are known at all times.

## 1.3 Fundamentals of model predictive control

The Automotive industry is heavily invested in AD and ADAS, because of this MPC have had an increase in interest. Published research in the area have seen a considerable increase over the last decades and subsequently this has over time created a disconnect between theory and practice [6]. Model predictive control also known as Receding Horizon Control (RHC) is a method of process control that has been around since the late 1970s, where it became popular within the industrial processing industry. The advancements of digital computers made it possible to use MPC for stable systems with slow processes. The second generation of MPC that developed in the 1980s saw the use of Quadratic Programming (QP) for solving the optimization problem [18].

With embeded hardware getting smaller and faster MPC is making its way into vehicles. But MPC is not only used for AD and ADAS, it has been investigated for use in a number of different automotive control applications including: engine, transmission, emissions and suspension [8]. In AD and ADAS, MPC can be used to solve an array of different problems as well. It can be used for both motion planning and path planning. Motion planning is a set of maneuvers a vehicle should execute to get from one point to another, while considering constraints. These constraints could be: vehicle kinematics, collision avoidance or velocity, acceleration and jerk limits. Path planning usually refers to the ability to create a path that a vehicle should follow.

MPC is a method for optimal control by solving a finite time horizon open-loop control problem at every sampling time. In process control the sampling times are relatively long measuring in minutes or sometimes hours [8]. This provides a generous amount of time to solve the optimization in real time. For autonomous vehicles the sampling time is often just a fraction of a second where the calculations are measured in milliseconds. The control problem is solved through optimization by minimizing a cost function while considering a set of constraints. The MPC gets its predictive capabilities from using a mathematical model of the real system. The mathematical model is a state space model that is defined by differential equations. By using a model of the system, its future depending on the control inputs can be predicted. The accuracy of the model determines the accuracy of the future predictions, it is therefore important to have a good mathematical model of the system. Often 80% of the work of integrating a MPC controller consist of modelling and identifying the system model [18].

Proportional-Integral-Derivative (PID) controllers are widely known in the field of control theory. The PID controller is a control loop that continuously calculates the error between the actual signal and the desired signal of the system and applies the correct control input to minimize the error. The controller can only control one signal and it does not have information about the system it is controlling. The main difference between the MPC and the PID controller is that the MPC uses a mathematical model of the system to estimate its response to different control inputs. The difference in control response between MPC and PID can be seen in Figure 1.2.

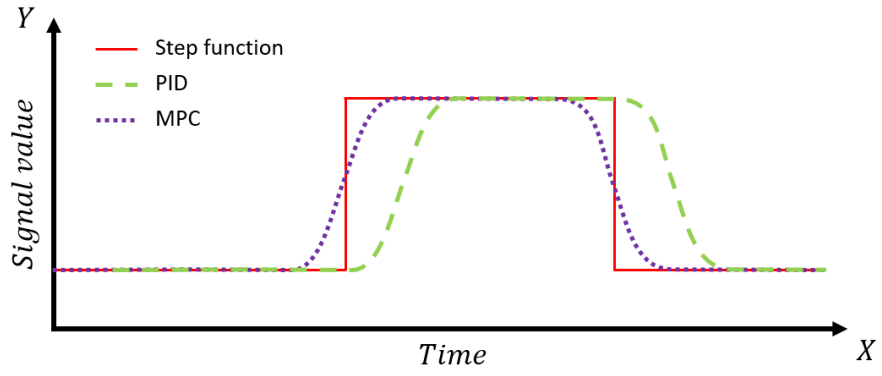


Figure 1.2: Step signal comparison between PID and MPC

The benefits with the MPC over a regular PID controller can be seen in the different response to a regular step signal. The PID controller has no predictive capabilities and can therefore only control the signal based on the error. The MPC has information about the upcoming step and can respond accordingly before the step in the signal. The predictive capabilities of the MPC are useful for autonomous vehicles as it allows the vehicle to foresee an upcoming curve or obstacle in the road. Another benefit is that the MPC can handle multiple input and multiple output systems with constraints [11]. Theoretically, this implies that hundreds of inter-dependent signals can be optimized in a single controller. The drawback is that the MPC is computationally heavy, and the calculation time increases with longer predictions. The MPCs horizons can be divided between the prediction horizon and the control horizon. The prediction horizon is the anticipated future states of the system while the control horizon is the corresponding control inputs.

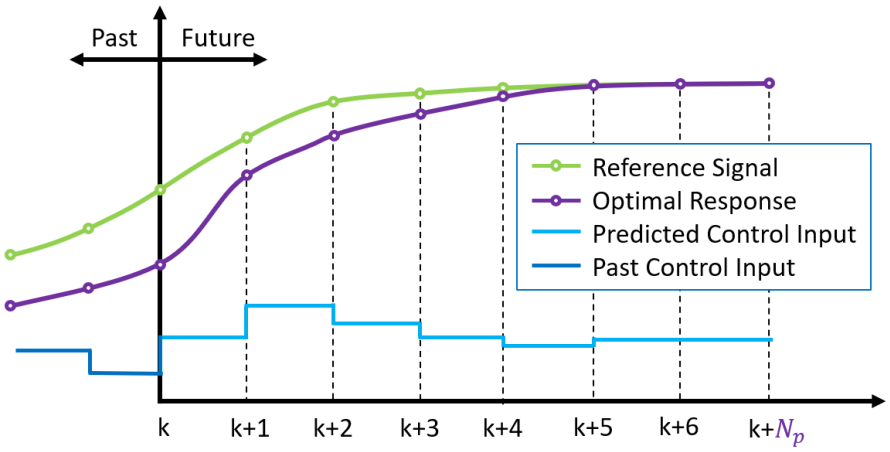


Figure 1.3: MPC control problem

Figure 1.3 shows a general example of a MPC and its procedure. Where $k$ is the timestep from k to the end of the horizon $k + N_p$. The optimal response is found by minimizing a cost function while considering the constraints. The cost function is defined based on the desired variables that wants to be minimized. To predict the future of the system the MPC needs a mathematical model that accurately represents the system. This is done with a state space model.

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{1.1}$$

The equation system in (1.1) is the general definition of a state space model. Where $A$ is the system matrix, $B$ is the input matrix, $C$ is the output matrix and $D$ is the feedthrough matrix. This state space model represents the physical system using differential equations with inputs u, outputs y and the states of the system x.

The optimal control strategy is found through mathematical optimization methods that differ depending on the complexity of the problem. To achieve predictive capabilities the MPC uses recursive calculations of the system's explicit discrete state space model. The state space model can calculate the future states of the system by applying an input. By using recursive calculations, the future for the system can be predicted for several steps into the future. This means that the calculation of the control input variables considers all possible outcomes during the optimization. The recursive calculations get more complex with a longer prediction horizon, to speed up the calculation of the MPC a common method is to introduce a control horizon. By allowing the control input to affect the system up to a certain point, called the control horizon, the optimization time can be minimized. This is because only the first optimal control input of the optimization problem is used for the next horizon. If the control horizon is shortened the equations after that point gets less complex and therefore reduces the computation time of the optimization problem [22].

### 1.3.1 Linear model predictive controller

The linear MPC is the simplest MPC configuration, it is easily solved with known optimization methods such as: Linear Programming (LP) or Quadratic Programming (QP) [9]. A LP problem is defined as

$$\min_u f^T u \ such\ that \begin{cases} A_{ineq}u \leq b_{ineq} \\ A_{eq}u = b_{eq} \\ lb \leq u \leq ub \end{cases} \tag{1.2}$$

Where $f$ is the linear system vector, $u$ is the optimization variable, $A_{ineq}$ and $b_{ineq}$ define the inequality constraints. $A_{eq}$ and $b_{eq}$ set the equality constraints and $lb$ and $ub$ define the lower and upper bounds. LP is defined by a linear system model, a linear cost function and linear constraints. The QP optimization problem is similar to LP with the difference being the capability to use a quadratic cost function. The QP problem is defined as

$$\min_u \frac{1}{2}u^T Hu + f^T u \ such\ that \begin{cases} A_{ineq}u \leq b_{ineq} \\ A_{eq}u = b_{eq} \\ lb \leq u \leq ub \end{cases} \tag{1.3}$$

Where $H$ is the Hessian matrix. Equation (1.3) creates a convex optimization problem that is easily solved with interior-point or active set methods [3].

The convex problem has a bowl like shape with a single minimum, this means that the local minimum is also the global minimum. Figure 1.4 illustrates the convex problem of a two-dimensional function. For higher dimensions the convex problem is impossible to illustrate but the same principles hold true.

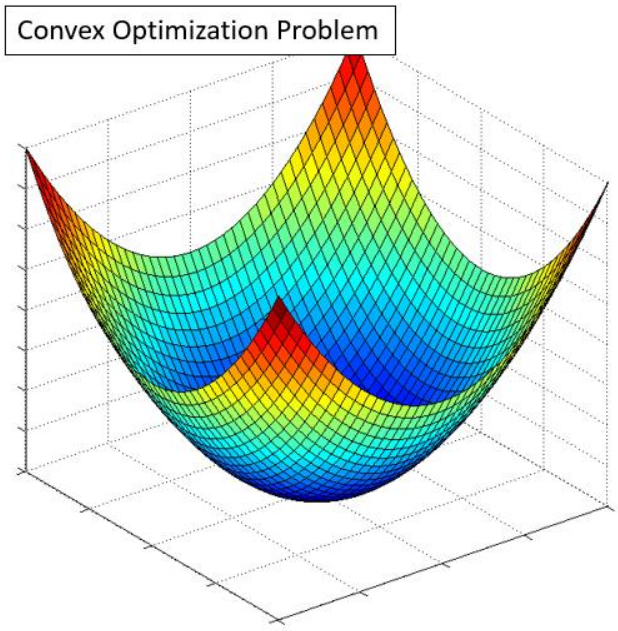


Figure 1.4: Convex optimization problem

Because a vehicle is a highly nonlinear system it must be linearized to be used in the linear MPC. The system is linearized around an operating point. If the controller deviates to much from this operating point the system becomes an inaccurate representation of the real vehicle. To get around this restriction a nonlinear MPC can be used that does not have to use linearized system models.

### 1.3.2 Nonlinear model predictive controller

The Non-Linear MPC (NLMPC) is characterized by having a nonlinear system model. The NLMPC can handle nonlinear: constraints, cost-function and system models. Therefore, it is the model that has the potential to give the most accurate predictions. Because of the non-convex nature of the NLMPC optimization problem it is very difficult and time consuming to solve. The NLMPC is non-convex and therefore has many different minimums and finding the global minimum is difficult as can be seen in Figure 1.5.

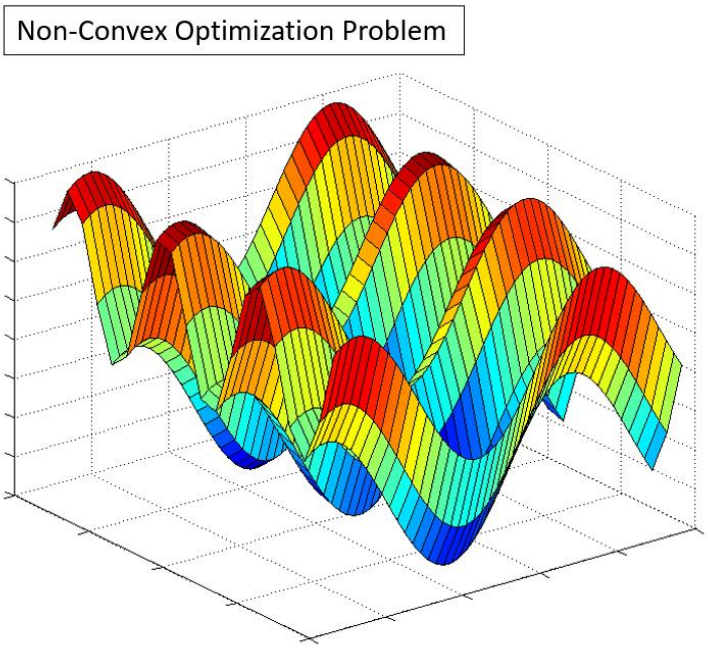


Figure 1.5: Non-convex optimization problem

NLMPC can be applied in process control where the sampling times are longer and where the computers are not restricted by physical space as they are in vehicles. The computational time is limited in a real-life scenario and with today's technology the NLMPC might not be ready for implementation in faster systems such as autonomous vehicles.

### 1.3.3 Explicit model predictive controller

The limiting factor with MPC is the large computational time, this is particularly true for systems with small sampling time. One method of reducing computational time is by using an Explicit MPC (EMPC) that does not need to optimize the control problem in real-time. EMPC is created by first building an implicit MPC and simulating the model in various conditions to create operating regions. The EMPC stores coefficients of the state space for each control region and creates a look up table for the MPC [16]. Each control region contains a sequence of the optimal control input for that particular scenario. The real-time control problem can then be reduced from a complex optimization problem to a simple function evaluation operation. EMPC does not require as much computational power and is therefore suited for fast dynamic systems with short sampling times.

## 1.4 Path generation

Path generation is used to find a suitable path for a vehicle that respects the vehicle's nonholonomic kinematics. In this thesis a path is generated to estimate the ego vehicle's path in the drivable space to approximate the APF. The MPC tracks the lane center reference, but because of the straight road assumption in this thesis it is represented by a constant value instead of a path.

The path for approximating the APFs should preferably consider the vehicle's initial yaw angle, velocity and position in space. One method of path generation is Dubins path which creates the shortest path between two points using 3 segments [12]. These segments can be a combination of straight and curved segments with a constant radius. Usually the path is constructed using a curve at the start and end of the path joined together with a straight line in between. This creates four different segment combinations which are illustrated Figure 1.6.

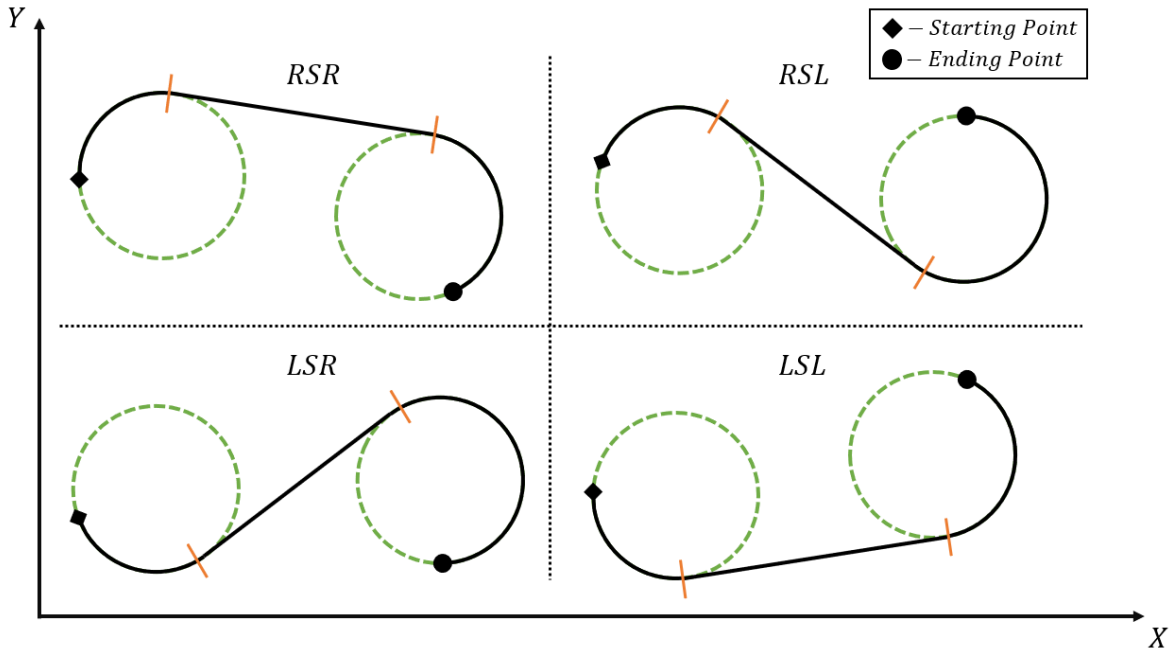


Figure 1.6: Dubins path segments, where R-right, L-left and S-straight

The combination of segments that produce the shortest path can be calculated with the Dubins path algorithm. This method is suitable if the traveled time or distance is of importance, however, in this thesis the main concerns are safety and comfort for the passengers of the autonomous vehicle.

To create smooth and comfortable curves a Bézier function can be used. The number of control points used in the Bézier curve defines the order of its polynomial, which can be seen in Equation (1.4).

$$P_0, \dots, P_n \; where \begin{cases} n = 1 & Linear \\ n = 2 & Quadratic \\ n = 3 & Cubic \end{cases} \tag{1.4}$$

The points in the Bézier curve defines the shape of the curve. More complex Bézier curves are usually a combination of cubic curves where the end points are collinearly joined together. Figure 1.7 shows the different Bézier curves.

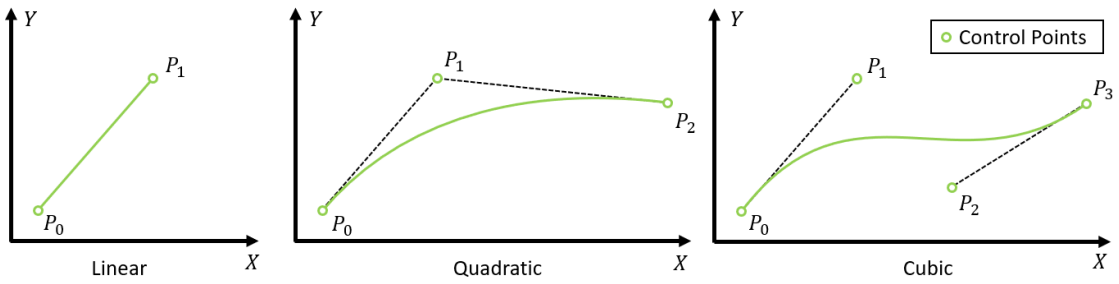


Figure 1.7: Bézier curves

A linear Bézier curve is simply a straight line between two points, if another control point is added a curve is created as a quadratic Bézier function. Whereas the cubic Bézier function can create s-shaped curves. The angle between $P_0$ and $P_1$ creates the initial angle and the angle between $P_2$ and $P_3$ creates the final angle at the end of the curve. The advantage of these curves is that they are easily manipulated to achieve a desired shape and the ends of the curve can be specified individually. The disadvantage is that the path does not take the vehicle's nonholonomic kinematics into consideration. The only way to ensure feasible paths for a vehicle is by positioning the control points in a way that limits sharp unnatural turns. Furthermore, the spacing between points in a discrete Bézier curve is not constant. For a vehicle traveling along the path this inconsistency in spacing causes a velocity variation along the curve. The inconsistent spacing between points is most prominent when curve have sharp bends. But it has a miniscule influence when the path is smooth and feasible for nonholonomic vehicles. The mathematical definition of the Bézier curves used in the thesis can be found in chapter 4.4.2.

# 2 Use-cases

In this chapter the different use-cases that are simulated are presented. The use-cases are limited to roadside parking on British roads, hence, the left-hand drive. All the surrounding moving vehicles travel at the 30 mph speed limit throughout the simulation unless otherwise stated.

## 2.1 Use-case 1

The first use-case is illustrated in Figure 2.1 and represents a typical scenario on urban roads in the UK. The ego vehicle approaches vehicles in its lane that are parked back-to-back at the edge of the lane.

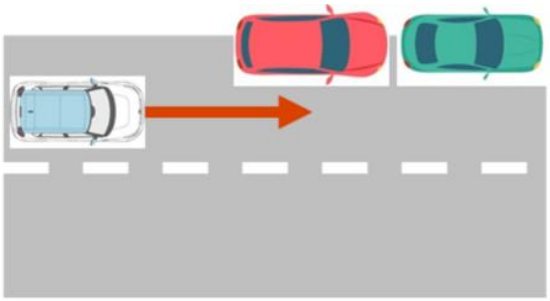

Figure 2.1: Use-case 1

The vehicle control must actuate the steering wheel to achieve a collision free trajectory along with the throttle and brake to comply with speed limits. When the ego vehicle approaches parked cars in its own lane it should slow down to 6m/s and keep this velocity while the vehicle is traveling in the opposing lane. This creates a safer overtaking maneuver because the ego vehicle's perception might not be able to detect pedestrians obscured by the parked cars or other unforeseen events.

This use-case tests the controller's ability to slow down before parked vehicles and complete an overtaking maneuver. The decision-making module is tested to see that it can accurately choose the correct references during the simulation to generate the desired maneuver. The overtaking should be made in a reasonable timeframe while keeping distance from surrounding obstacles. The vehicle's maneuver should be decisive and predictable to other drivers and the passengers in the vehicle.

## 2.2 Use-case 2

The second use-case is shown in Figure 2.2 and is an extension of use-case 1. The ego vehicle approaches the parked cars while there is traffic in the opposing lane. The moving vehicle is traveling in the middle of the opposite lane.

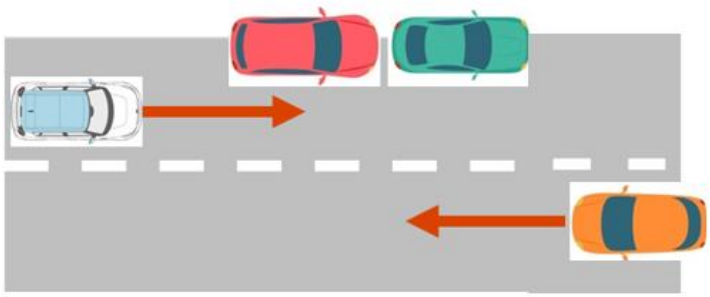

Figure 2.2: Use-case 2

Two scenarios are studied, the first one being a flying overtaking and the second one is a 'stop and go' scenario. If there is enough time for a safe overtaking the ego vehicle should actuate the steering wheel, throttle and brake to achieve a collision free overtaking. If there does not exist a safe path the ego vehicle should brake in a safe and smooth manner and wait behind the parked cars until it is safe to overtake. These scenarios test the controller's decision-making ability. The vehicle must be able to decide whether it should brake and wait behind the parked cars or if there is enough to overtake. If the vehicle decides to stop and wait behind the parked cars the vehicle should manage to come to a complete stop. When the road is clear or when there is sufficient time towards oncoming traffic the ego vehicle should accelerate and overtake the parked cars in a reasonable timeframe. During the overtaking the vehicle's speed limit is set to 6 m/s due to safety reasons.

The first variation of this use-case is almost identical to use-case 1 and will therefore only be simulated to verify that the ego vehicle can do a flying overtaking maneuver. The latter scenario will be studied in detail and compared to the overtaking maneuver from use-case 1.

## 2.3 Use-case 3

The third use-case is presented in Figure 2.3. This use-case does not contain any complex maneuvers instead it serves as a control scenario to evaluate the influence of other vehicles. The ego vehicle is traveling in the left lane and encounters traffic in the opposite lane.

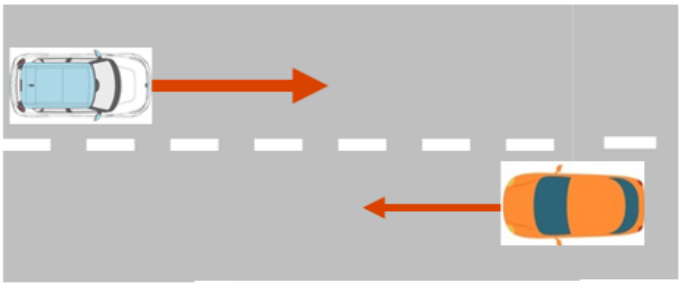

Figure 2.3: Use-case 3

Because the path is clear in the ego vehicle's lane it should not be influenced by the vehicles traveling in the opposite lane. The ego vehicle's decision-making should not change during the simulation and it must stay within its lane and show marginal deviation from the lane center. The acceleration should be smooth and uninterrupted by the traffic.

## 2.4 Use-case 4

The fourth use-case is illustrated in Figure 2.4. The ego vehicle approaches vehicles in the right lane that are parked back-to-back at the edge of the lane. The moving vehicle in the opposing lane does not respect the ego vehicle's right of way and starts to overtake the parked cars. This use-case tests the controller's ability to react to unusual situations where the traffic laws are not followed by other road users. The velocity of the opposing vehicle is 8 m/s and is constant throughout the simulation.

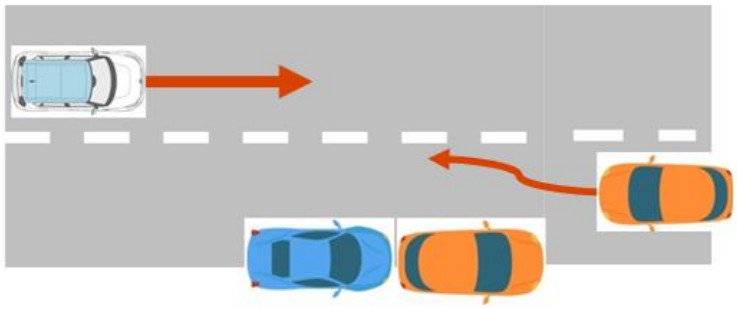

Figure 2.4: Use-case 4

During this maneuver the opposing vehicle occupies the ego vehicle's lane. For simplicity it is assumed that the ego vehicle predicts this maneuver from the opposing vehicle in advance. The ego vehicle must brake and if necessary actuate the steering to achieve a collision free trajectory. When the opposing vehicle has passed the ego vehicle it should be able to accelerate up to the speed limit while staying in the center of the desired lane.

# 3 Vehicle Model

Two vehicle models are used in this thesis, one for the internal MPC vehicle model and one for the plant model that represents the real vehicle in the simulations. This chapter starts with the definition of the coordinate system and then presents the different types of vehicle models. The chapter ends with a conclusion on the vehicle model choice for the MPC and the plant model.

## 3.1 Coordinate system

The coordinate system used in the thesis is a ground fixed cartesian coordinate system that is fixed at the start of the simulation. The origin is located at the start of the simulation between the left and right lane. The cartesian coordinate system was used because it makes it easier to implement vehicle models and path generation algorithms. In the simulations the Y-axis will represent the lateral movement and the X-axis will represent the longitudinal movement of the vehicle in relation to the road.

## 3.2 Point Mass Model

The point mass model (PMM) is a simple model that could be used to represent a vehicle [17]. It is representing a single point on a plane that can move freely in each direction. The PMM has the following set of differential equations

$$\begin{aligned} \dot{X}_h &= v_x \\ \dot{Y}_h &= v_y \\ \dot{v}_x &= a_x \\ \dot{v}_y &= a_y \end{aligned} \tag{3.1}$$

The equation system for the PMM in Equation (3.1) is already linear. The linear and discretized explicit state space model for the PMM can be written as

$$\begin{aligned} X_h(k+1) &= X_h(k) + v_x(k)T_s \\ Y_h(k+1) &= Y_h(k) + v_y(k)T_s \\ v_x(k+1) &= v_x(k) + a_x(k)T_s \\ v_y(k+1) &= v_y(k) + a_y(k)T_s \\ \forall k &= 1, \dots, N \end{aligned} \tag{3.2}$$

Where $T_s$ is the sampling time, the vehicle's states are defined as $\xi = \left[X_h, Y_h, v_x, v_y\right]^T$ and the control input as $u = \left[a_x, a_y\right]^T$. A vehicle's movement is nonholonomic meaning it cannot move laterally without moving longitudinally. In Equation (3.2) the vehicle's movement in the lateral and longitudinal direction is independent from each other, to mimic a vehicle's nonholonomic movement the following constraint can be implemented

$$\begin{aligned} -\beta v_x(k) \leq v_y(k) &\leq v_x(k)\beta \\ \xi_{min} \leq \xi(k) &\leq \xi_{max} \\ u_{min} \leq u(k) &\leq u_{max} \end{aligned} \tag{3.3}$$

Where $\beta$ is the vehicle's side slip angle and can be calculated with Equation (3.4)

$$\beta = tan^{-1}\left(\frac{v_y}{v_x}\right) \tag{3.4}$$

Equation (3.2) can be used to represent a vehicle if small angle approximation is used, this gives $-10° \leq \beta \leq 10°$ [17]. This limits the vehicle's lateral velocity based on the longitudinal velocity and creates a nonholonomic vehicle model.

This model works for generating paths for nonholonomic vehicles but is not suitable as an internal vehicle model in the MPC algorithm. This is because the model does not consider the vehicle's geometry and it does not have an input that depends on the wheel angle of the vehicle.

### 3.3 Kinematic bicycle model

The kinematic bicycle model, also called the single-track model is the simplest vehicle model that accurately represent the kinematic behavior of a vehicle. The kinematic bicycle model describes the vehicle's movement by considering its kinematic constraints, but it does not take the forces that cause these movements into consideration. This is the main drawback of the kinematic bicycle model, because of this the model suffers in accuracy during highly dynamic maneuvers. However, the kinematic bicycle model produces good results if the lateral acceleration is limited to $0.5g\mu$ [19], where $g$ is the earths gravity and $\mu$ is the friction coefficient of the road surface.

The benefit with the kinematic bicycle model is its simplicity, it provides simple equations and fast calculation times. The system's differential equations only need two vehicle specific parameters: the length between front and rear axles to the position of the center of mass. These parameters are easily obtained from any vehicle making a MPC algorithm with a kinematic bicycle model highly adaptable between different vehicles. Furthermore, because of the equations formulating the kinematic bicycle model can handle stop-and-go situations, which the dynamic bicycle model cannot, see Equation (3.11). These benefits makes the kinematic bicycle model a popular choice for MPC [19].

The kinematic bicycle model simplifies a vehicle by reducing the two wheels per axle to a single one. The model only has two wheels and it resemble a bicycle. The bicycle model is illustrated in Figure 3.1.

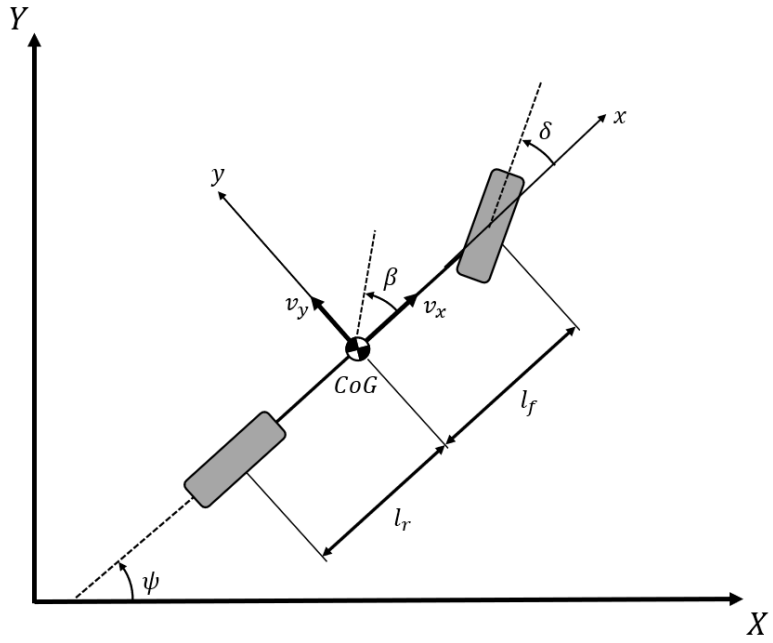


Figure 3.1: Kinematic bicycle model

Where $l_f$ and $l_r$ is the distance from the Center of Gravity (CoG) to the front respectively rear axle and $L$ is the total axle length. From Figure 3.1 it is possible to describe the kinematic bicycle model as differential equations as such

$$\begin{aligned} \dot{X}_h &= v\,cos(\psi + \beta) \\ \dot{Y}_h &= v\,sin(\psi + \beta) \\ \dot{\psi} &= \frac{v}{l_r} sin(\beta) \\ \dot{v} &= a \end{aligned} \tag{3.5}$$

Where $X_h$ and $Y_h$ are the ego vehicle's global coordinates, $v$ is the absolute velocity, $a$ is the absolute acceleration, $\psi$ is the yaw angle and $\beta$ is the side slip angle at the Center of Gravity (CoG). The slip angle has the following relationship with the wheel angle

$$\beta = tan^{-1}\left(\frac{l_r}{L} tan(\delta)\right) \tag{3.6}$$

Where $\delta$ is the front wheel angle in relation to the car body and $L$ is the wheelbase. The vehicle data can be seen in Table 3.1.

Table 3.1: Vehicle data

| Name | Symbol | Value | Unit |
|---|---|---|---|
| Length from rear axle to CoG | $l_r$ | 1.5 | $[m]$ |
| Length from front axle to CoG | $l_f$ | 1.05 | $[m]$ |
| Wheelbase | $L$ | 2.55 | $[m]$ |

### 3.4 Dynamic Bicycle Model

To get a more accurate representation of a vehicle, the dynamic bicycle model can be used. In comparison to the kinematic bicycle model, the dynamic bicycle model considers the tire forces, the vehicle's mass and the vehicle's inertia around the z-axis. Because of this it is a better representation of the vehicle during dynamic maneuvers. The dynamic bicycle model is depicted in Figure 3.2.

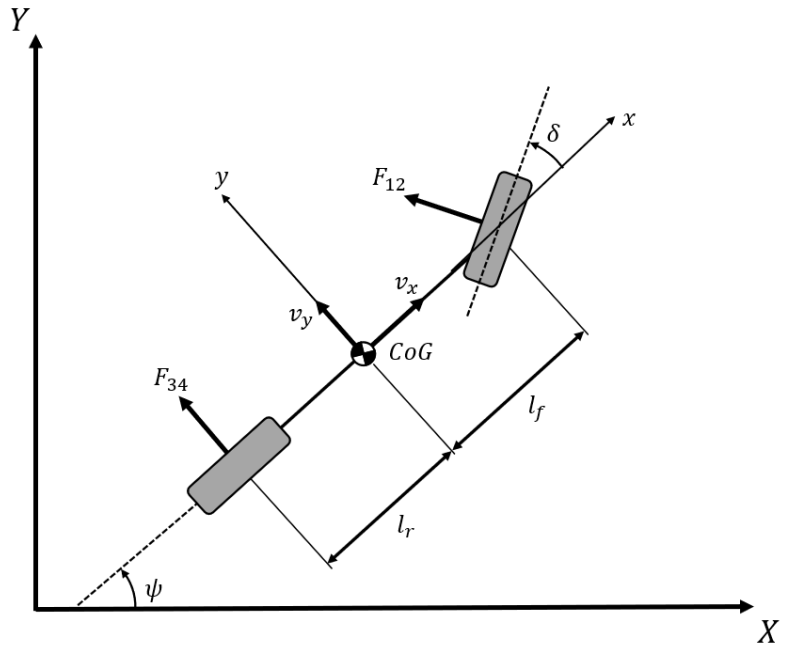


Figure 3.2: Dynamic bicycle model

The equations of motion for the dynamic bicycle model can be derived from Figure 3.2, by establishing force and torque equilibrium around the center of gravity (CoG)

$$\begin{aligned} \uparrow m(\dot{v}_x - \dot{\psi} v_y) &= -F_{12}\, sin(\delta) \\ \rightarrow m(\dot{v}_y - \dot{\psi} v_x) &= F_{34} + F_{12}\, cos(\delta) \\ \circlearrowleft J_z \ddot{\psi} &= l_f F_{12}\, cos(\delta) - l_r F_{34} \end{aligned} \tag{3.7}$$

Where $m$ is the vehicle mass, $v_x$ is the longitudinal velocity and $v_y$ is the lateral velocity. $F_{12}$ and $F_{34}$ are the tire forces on the front and rear axle respectively and $J_z$ is the inertia around the z-axis. The tire forces can be calculated using

$$\begin{aligned} F_{12} &= -C_{12}\alpha_{12} \\ F_{34} &= -C_{34}\alpha_{34} \end{aligned} \tag{3.8}$$

Where $C_{12}$ and $C_{34}$ are the tire stiffnesses for the front and rear wheels, $\alpha_{12}$ and $\alpha_{34}$ are the slip angles for the front and rear wheels respectively.

The slip angle for the front and rear axle respectively can be calculated with

$$\begin{aligned} \alpha_{12} &= tan^{-1}\left(\frac{v_y + \dot{\psi} l_f}{v_x}\right) - \delta \\ \alpha_{34} &= tan^{-1}\left(\frac{v_y - \dot{\psi} l_r}{v_x}\right) \end{aligned} \tag{3.9}$$

By inserting Equation (3.8) and (3.9) into (3.7) the equations of motion can be simplified to

$$\begin{aligned} m(Dv_y + \dot{\psi} v_x) &= -C_{34}\frac{v_y - \dot{\psi} b}{v_x} - C_{12}\left(\frac{v_y + \dot{\psi} l_f}{v_x} - \delta\right) \\ J_z D\dot{\psi} &= -l_f C_{12}\left(\frac{v_y + \dot{\psi} l_f}{v_x} - \delta\right) + l_r C_{34}\frac{v_y - \dot{\psi} l_r}{v_x} \end{aligned} \tag{3.10}$$

Where $D$ represents the derivative of the following variable. It can also be written in matrix form as such

$$\underbrace{\begin{bmatrix} mD + \frac{C_{12} + C_{34}}{v_x} & mv_x + \frac{l_f C_{12} - l_r C_{34}}{v_x} \\ \frac{l_f C_{12} - l_r C_{34}}{v_x} & J_z D + \frac{{l_f}^2 C_{12} + {l_r}^2 C_{34}}{v_x} \end{bmatrix}}_{A} \begin{bmatrix} v_y \\ \dot{\psi} \end{bmatrix} = \underbrace{\begin{bmatrix} C_{12} \\ l_f C_{12} \end{bmatrix}}_{b} \delta \tag{3.11}$$

Which can compactly be written as

$$A \begin{bmatrix} v_y \\ \dot{\psi} \end{bmatrix} = b\delta \tag{3.12}$$

As can be seen in Equation (3.11) the calculations in the $A$ matrix uses division with the longitudinal velocity. This means that the dynamic bicycle model does not work when $v_x = 0$.

## 3.5 Vehicle model choice

Two vehicle models are needed for the simulations, one as the internal vehicle model in the MPC and another as the plant model representing the real vehicle. For the MPC the vehicle model needs to be simple yet accurate for fast calculations with good performance.

The linear MPC needs to have a linear vehicle model that can handle 'stop and go' scenarios in an urban environment. Because the dynamic bicycle model does not work when $v_x = 0$ it is incompatible with the use-cases presented in chapter 2. The PMM is simpler than the kinematic bicycle model but because it does not consider the vehicle's geometry, wheel angle or steering wheel angle it is not a good choice for the MPC. Because of this the kinematic bicycle model is used as the internal vehicle model for the controller.

The plant model needs to be a close representation of the real vehicle. The dynamic bicycle model was implemented as a plant model under the assumption that the vehicle could simply be constrained to have an absolute velocity $v > 0$ to bypass the division by zero. In theory this works but when simulating in MATLAB the calculation times for the plant model rose exponentially when the vehicle's velocity reached zero. It was decided that this was not acceptable because of the severe calculation times. Furthermore, it was determined that the kinematic bicycle model would suffice as the plant model because of the low speed and low lateral acceleration scenarios of urban driving. Therefore, it would be little to no difference in the results between the kinematic and the dynamic bicycle model. Because of this the kinematic bicycle model is used as the plant vehicle model. For simplicity, the model is discretized with the same sample time as the MPC. The plant model does not need recursive calculations as it can use the actual velocity of the ego vehicle in its calculations. Because of this the regular state space model can be used for the plant model calculations.

# 4 Software architecture

This chapter explains the software architecture of the simulations in this thesis. The MPC structure is based on the linear MPC using QP formulation where the whole system is built and simulated in MATLAB. The software architecture design can be seen in Figure 4.1.

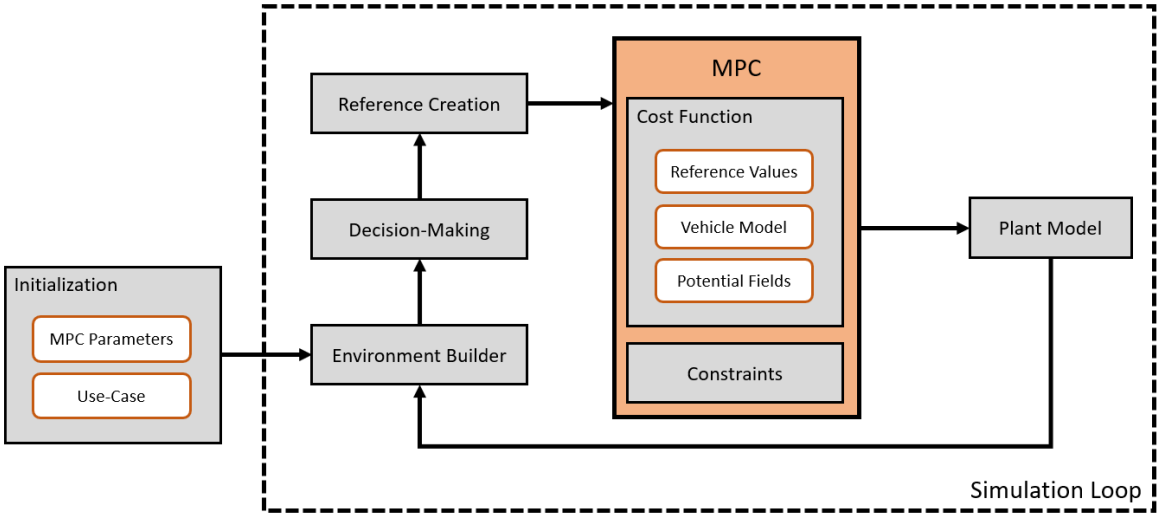


Figure 4.1: MPC software architecture

The parameters, vectors and matrices that keep constant throughout the simulation are initialized at the beginning of the code. The use-cases are built before the simulation loop which include the definition of the obstacle vehicle's positions and paths during every step of the simulation. By having these parameters outside the simulation loop the calculation time can be reduced. Because the environment is not changing throughout the simulation it could be built before the simulation loop. However, this is due to the assumption that the road is straight with a constant road width which is not valid outside the simulation environment. For a more representative solve time the environment builder is implemented in the simulation loop and it is updated every sample time.

At the beginning of the simulation loop the environment builder constructs nonlinear obstacle APFs for the path scouting in the decision-making. It also creates the approximated APFs for the road and the obstacles that is used in the MPC. The decision-making module uses the obstacle APF and the updated ego vehicle's states to decide the next maneuver. The decision-making decides the desired velocity and the preferred lane center that should be used as a reference. The velocity profile is constructed based on the current velocity and a constant acceleration or deceleration to the desired velocity. The lane center reference is a constant value of the desired lanes lateral position. Then the vehicle model's recursive calculations are made and the MPCs matrices and vectors are built on the QP form.

The constraints in the MPC limits the possible solutions to the optimization problem. The constraints can be divided between environmental constraints and vehicle constraints. The environmental constraints are defined by the road boundaries and the speed limit. The vehicle constraints are defined by the vehicle's performance or the desired limits on the vehicle's actuation such as: maximum slip angle or maximum acceleration.

The QP problem then solves the optimization problem and sends the first optimal control inputs to the plant model to be executed. The plant model calculates and updates the vehicle states and sends it back to the environment builder to start a new simulation loop for the next sample time. In a real-world scenario the updated vehicle states and environment information would come from sensors on the vehicle.

## 4.1 Optimization problem

The MPC in this thesis is a linear time-varying MPC with APFs. The MPC uses MATLAB's *quadprog* routine to solve the QP optimization problem defined in Equation (1.3). The cost function includes cost terms for the ego vehicle's states, control inputs and the APFs. The error between the states and their respective reference is minimized. The states in the cost function are the lateral position, yaw angle and absolute velocity. The references are the lane center lateral position, the yaw angle reference, which is zero, and the velocity reference from the velocity profile. Because the yaw angle reference is zero it is effectively penalized for large yaw angles. If the road would curve, the yaw angle reference would represent the angle of the road. The control inputs consist of the slip angle and absolute acceleration which controls the ego vehicle's steering, throttle and brake. These inputs are minimized in the cost function for smooth driving. The APFs gives a cost for being near surrounding obstacles and for deviating from the lane centers, this cost is minimized to guide the ego vehicle away from high cost regions, thus reducing the risk of collision.

## 4.2 Decision-making module

The general maneuver the ego vehicle should take is defined by a velocity profile and a reference lane center. The decision-making module's task is to decide the preferred lane center and the desired velocity. The velocity profile is a simple affine function between the current velocity and the desired velocity based on constant acceleration. When the velocity profile reaches the desired speed, it is kept constant for the rest of the prediction horizon. The reference lane center is represented by waypoints along either lane center, where the left lane is the default lane. If the ego vehicle encounters obstacles in the left lane and it is safe to overtake then the reference path should be changed to the right lane. For more details on how the references are created see chapter 4.4. The decision-making structure can be seen in Figure 4.2.

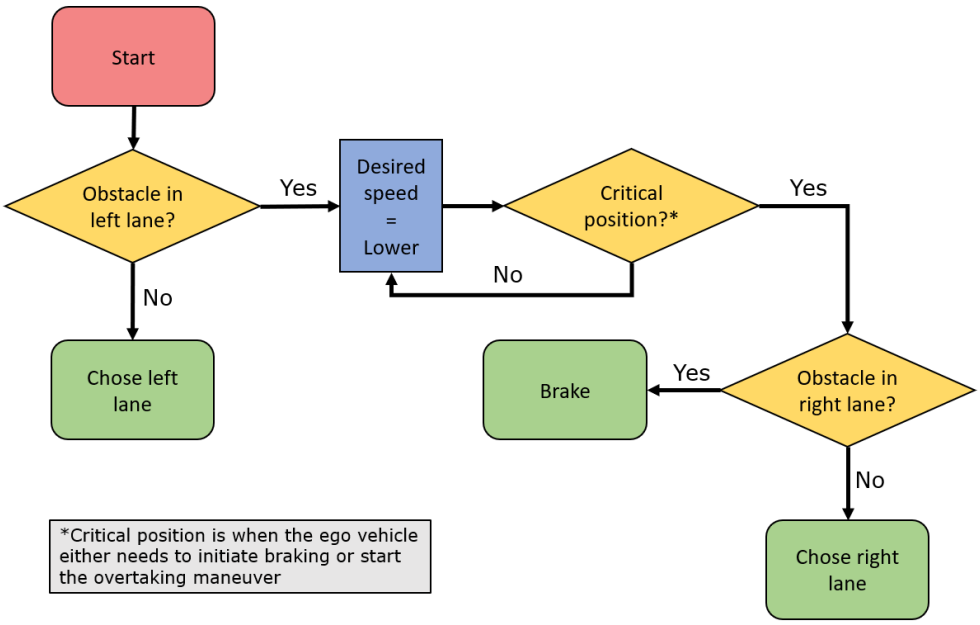


Figure 4.2: Decision-making flow chart

To make these decisions the decision-making module receives information about the surrounding obstacles and their predicted trajectories. To ensure a safe and comfortable overtaking maneuver the vehicle should slow down and keep a velocity of 6 m/s when it approaches obstacles in its own lane. This is partly because it is not favorable to overtake parked cars at the speed limit as this could both be dangerous and unsettling for any passengers. When approaching the parked cars, the ego vehicle should decide whether to initiate an overtaking maneuver or to brake and come to a stop. In this thesis this point is called the critical point. At this critical point, the vehicle evaluates if there is enough time for an overtaking maneuver. This is based on the ego vehicle's velocity, the obstacle vehicle's velocity and the distance to the oncoming vehicle in the other lane. If there is enough time it will initiate the overtaking maneuver by changing the reference path to the right lane center. Otherwise, the vehicle stays in the left lane and changes to a velocity profile that brakes the vehicle to a stop. When the vehicle has initiated an overtaking maneuver, it continuously checks whether it is safe to go back into the left lane and when it is safe to do so it changes back to using the left lane center as a reference.

Because of the high relative velocity between the ego vehicle and the oncoming vehicle the decision-making module needs to look far ahead. It does this by scouting for obstacles in its path, the path scouting is illustrated in Figure 4.3.

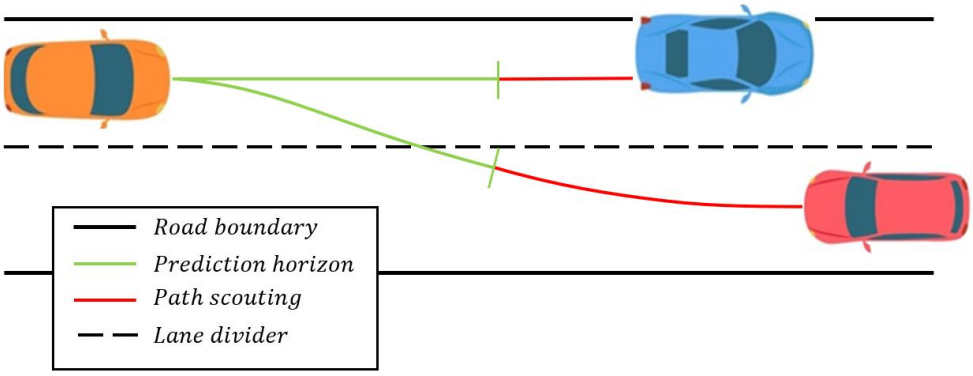


Figure 4.3: Path scouting

The scouting is done by two Bézier curves resembling the path the vehicle would take when favoring the left lane and right lane respectively. The scouting Bézier curves are used together with the obstacles APF and if the value from the APF reaches above a certain threshold it is registered as an obstacle.

## 4.3 Artificial potential fields

In this thesis the safety regions around the surrounding vehicles and other obstacles are created using APFs. APFs work like magnets where a higher value acts like a repulsive force and a lower value has an attracting force. To use the nonlinear APFs in the MPC they need to be converted to quadratic functions. This is done with quadratic Taylor series approximation, Taylor series is defined in chapter 5.3. Taylor series approximates a function around a point using polynomials, where a polynomial of the second degree is used to approximate the nonlinear APF to a quadratic function. The approximated APF compared to the nonlinear APF can be seen in Figure 4.4.

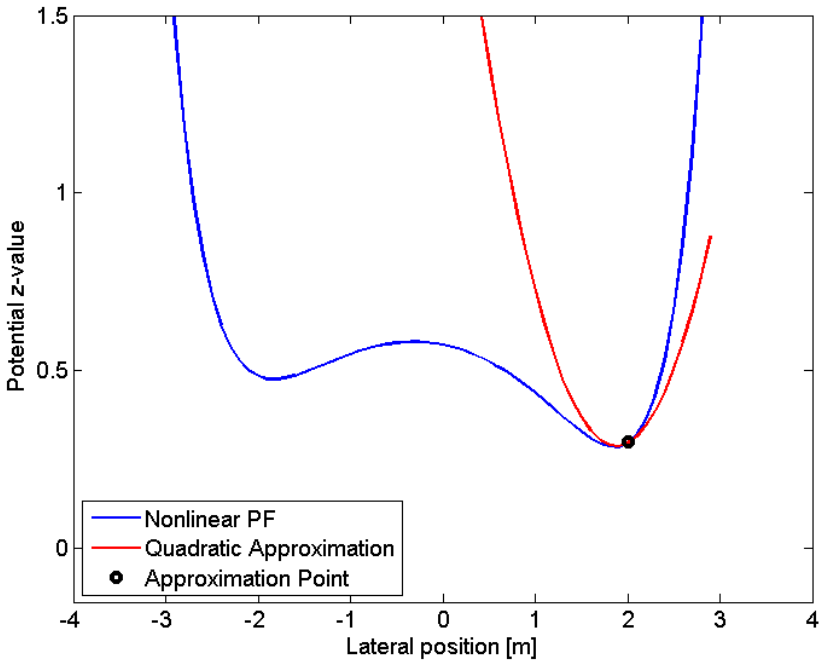


Figure 4.4: Cross section of quadratic Taylor approximated road APF

Figure 4.4 is a cross section of the APF and its quadratic approximation. For the vehicle this point is the estimated location of the vehicle during the prediction horizon. To estimate the path the vehicle will use smooth Bézier curves that are similar to the ones used for the decision-making modules obstacle scouting. The drawback with using approximated APFs is that when the approximation yields concave results the bounds go towards negative infinity instead of positive infinity. This occurs when the approximation point is in between the lanes. The impact of this behavior can be reduced by decreasing the hump in between the lanes, thus reducing the slope of the approximated APF.

### 4.3.1 Road Potential Field

The road APF rewards the ego vehicle to stay in the middle of a lane where the APF has local minimum. When the vehicle is in the middle of the lanes it rewards the ego vehicle to either turn left or right towards the lower cost regions. Between the lane centers and the road boundaries the APF function reach infinity towards the road boundaries to prohibit the vehicle from going off the road. Because the vehicle's position is represented by its center of mass the road APF needs to increase before the lane boundaries to factor in the width of the vehicle.

The equations for the road APF look the same throughout the prediction horizon because the road is assumed to be straight, uniform and unchanged over the horizon. The equations are based on the Morse potential for O-H bond interaction [5]. The Morse potential function for O-H bond interaction can be written as

$$E_{morse} = D_0\left(1 - e^{-a(r-r_0)}\right)^2 \tag{4.1}$$

Where $E_{morse}$ is Morse potential energy, $D_0$ is the depth of the curve, $a$ is the width of the well, $r$ is the distance between atoms and $r_0$ is the equilibrium bond distance. The Morse potential can be seen in Figure 4.5

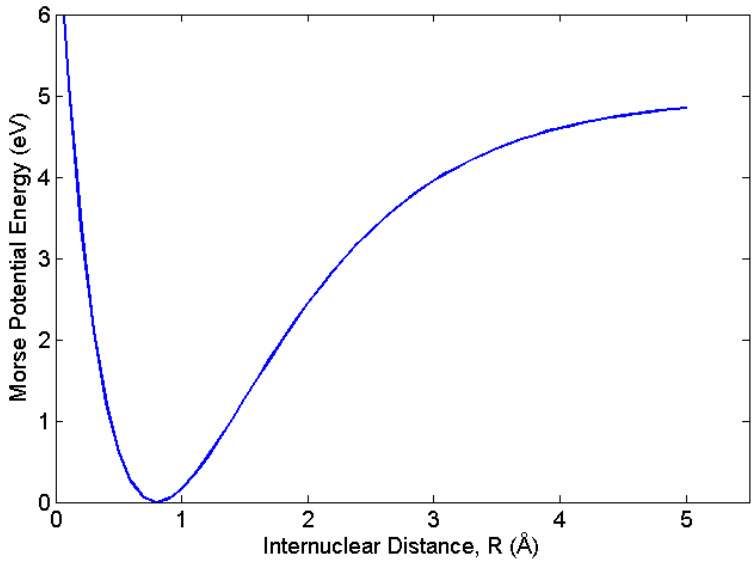


Figure 4.5: Morse potential

When using this function as a road APF then the original use is of little importance, the interest is within the shape of the curve. Equation (4.1) could be used to represent one lane of the road. When used to represent a two-lane road it has the following form

$$U_{road} = \underbrace{A_r\left(1 - e^{-b_r(Y_h - Y_{rlc})}\right)^2}_{Right\ lane\ PF} + \underbrace{A_l\left(1 - e^{b_r(Y_h - Y_{llc})}\right)^2}_{Left\ lane\ PF} \quad (4.2)$$

Where $U_{road}$ is the road potential z-value, $A_r$ and $A_l$ is the depth of the right and the left lane centers. $b_r$ is the width of the lane centers, $Y_h$ is the vehicle's optimal lateral position for the prediction horizon and $Y_{rlc}$ and $Y_{llc}$ is the right and the left lateral lane center position. The combined APF that resembles a road is illustrated in Figure 4.6.

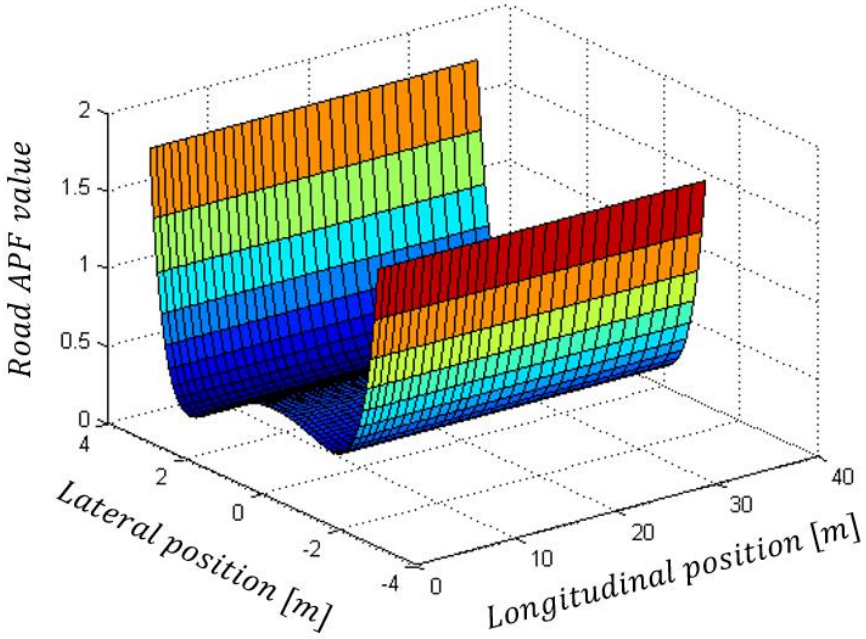


Figure 4.6: Road APF

The road APF has been tuned to have a higher local minimum for the right lane to put a higher cost for traveling in the opposite lane. Furthermore, the hump between the lanes is reduced in height to decrease the slope of the concave quadratic approximation, more information about the tuning process can be found in chapter 6.

### 4.3.2 Obstacle Potential Field

The objective of the obstacle APFs is to repel the ego vehicle away from obstacles and force the vehicle to favor the path with the lowest cost. The obstacle APF is based on Gaussian normal distribution surface function and has the following equation

$$U_{obs} = A_{obs} e^{-x_s(X_h + X_{obs})^2 - y_s(Y_h + Y_{obs})^2} \tag{4.3}$$

Where $U_{obs}$ is the obstacle potential z-value, $A_o$ is the height of the obstacle APF, $X_h$ is the vehicle's optimal lateral position for the prediction horizon and $X_{obs}$ and $Y_{obs}$ is the obstacle longitudinal and lateral position. $x_s$ and $y_s$ is the scaling factor for the obstacle APF in the longitudinal and the lateral direction respectively, where a lower coefficient value gives a larger region. The obstacle APF can be seen in Figure 4.7.

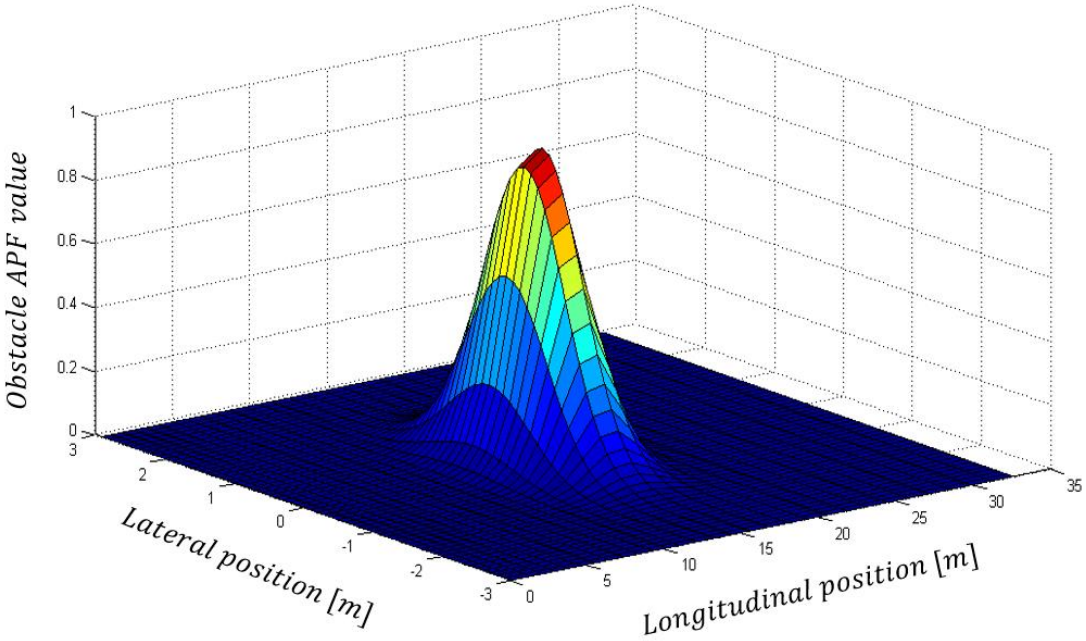


Figure 4.7: Obstacle APF contour map

The obstacle APF looks different for parked cars and moving vehicle. The parked cars have a smaller footprint on the x-axis while the moving vehicles takes up a larger area on the x-axis to compensate for its velocity. The moving vehicle's scaling coefficient for the x-axis is given by

$$x_{s,moving} = \left(1 - \frac{1}{10 v_{obs}}\right) x_{s,parked} \tag{4.4}$$

Where $x_{s,moving}$ and $x_{s,parked}$ is the APF longitudinal scaling factor for moving and parked obstacle vehicles. $v_{obs}$ is the velocity of the moving obstacle vehicle.

## 4.4 References

Four different references are used in this thesis. One reference is a velocity profile created by an affine function, whose shape is determined by the decision-making module. Two references are Bézier curves, one of which is used in the quadratic Taylor series approximation of the APFs and the other is used for scouting for obstacles in the decision-making module. The last reference is used as a path tracking reference in the MPC. This reference is the desired lane center lateral position. Because of the straight road assumption, the reference is a constant value for each respective lane.

### 4.4.1 Velocity Profile

The velocity profile is an affine function between the vehicle's initial velocity and the desired velocity and can be seen in Figure 4.8. Bézier curves were also tested as a velocity profile where the initial slope in the curve considered the initial acceleration. The Bézier curve velocity profile was abandoned because the smoothness of the curve created a slow response to changes in the system. The affine function provides shorter calculation times with a more distinct change in velocity that can be smoothed out by tuning the MPC.

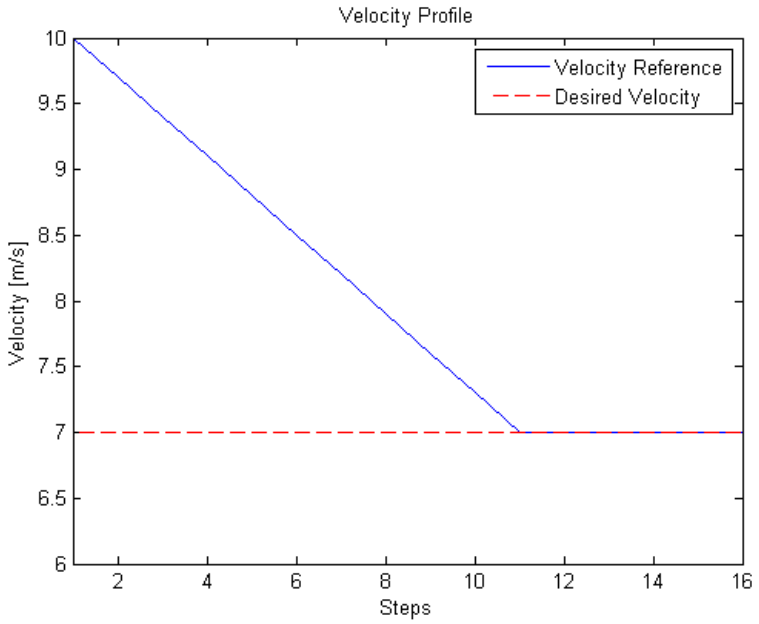


Figure 4.8: Velocity profile

The slope of the affine function is dictated by the desired maximum and minimum acceleration of the vehicle. From this function the corresponding velocity is retrieved for each step in the prediction horizon. If the velocity achieves the desired velocity within the prediction horizon then the rest of the values are kept constant at the desired speed for the rest of the horizon. This velocity profile is also used when creating the Bézier curves.

### 4.4.2 Bézier Curve

The reference path used in the APFs to estimate the vehicle's position for the prediction horizon is created using cubic Bézier curve. The cubic Bézier curve is used because it resembles a lane change maneuver. The cubic Bézier curve is defined by four control points and can be expressed in explicit form by

$$\boldsymbol{B}(t) = (1-t)^3\boldsymbol{P}_0 + 3(1-t)^2 t\boldsymbol{P}_1 + 3(t-1)t^2\boldsymbol{P}_2 + t^3\boldsymbol{P}_3 \quad 0 \leq t \leq 1 \tag{4.5}$$

Where $\boldsymbol{P}_0, \ldots, \boldsymbol{P}_3$ are the control points that contain $x$ and $y$ coordinates and $\boldsymbol{B}(t)$ is the Bézier function tracing the curves path in $x$ and $y$ coordinates. The first and last control points acts as anchors, creating the start and end of the curve. The Figure 1.7 demonstrates how a common cubic Bézier curve looks like

The line that is created between the first and second control points as well as between the last and second to last control point dictates the curves initial and final angle. The first angle will be determined by the initial angle of the vehicle while the last angle will be the angle of the road to ensure the vehicle ends up in the direction of the road. The Bézier curve lane change is illustrated in Figure 4.9.

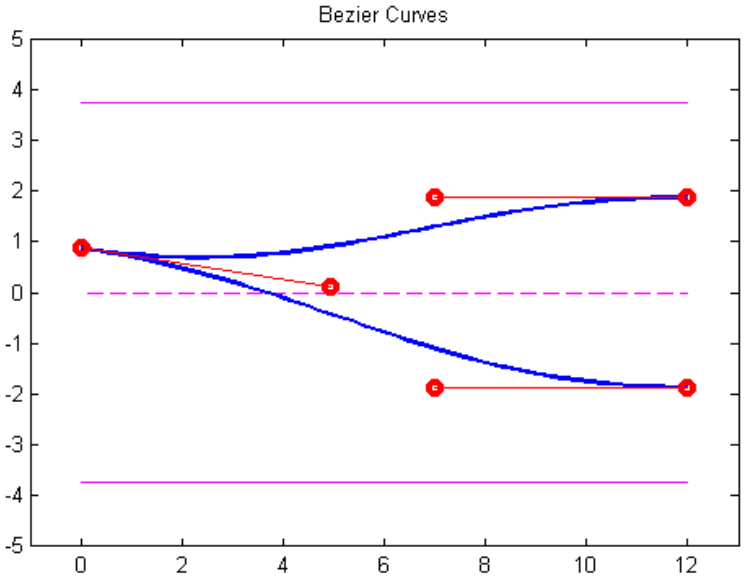


Figure 4.9: Bézier curve lane change

In Figure 4.9 the vehicle travels from left to right and creates two paths that both considers the vehicle's initial angle. The first path takes the vehicle back into the left lane and the second path takes the vehicle into the right lane. Both paths end in their respective lane centers and has the same angle as the road. The end control points are dependent on the vehicle's velocity as such

$$X_{end} = v_0 + X_{h,0} + 10 \tag{4.6}$$

Where $X_{end}$ is the longitudinal end point for the Bézier curve, $v_0$ is the initial velocity and $X_{h,0}$ is the initial longitudinal position for the ego vehicle. This is needed because the coordinate system does not follow the vehicle throughout the simulation. The absolute distance between the first and second as well as the last and second to last control points determine shape of the curve. For an overall smooth curve, the second and second to last control points are longitudinally placed in the middle of the start and end control points.

To use the Bézier curves in the MPC they need to be adjusted to fit within the MPCs time horizon. This is done by calculating how far the vehicle will travel along the curve during the prediction horizon. To do this, first the curve is created, and its length measured, then it is compared to the vehicle's total traveled distance over the prediction horizon based on the current velocity profile. The points are then spaced accordingly along the curve. If the vehicle travels further than the curve additional points are added that follow the appropriate lane center. This Bézier curve is then used when approximating the APFs.

# 5 Model predictive control matrix derivations

In this chapter, the matrices used to construct the MPC in MATLAB on QP form are derived. From the QP formulation (1.3), it can be seen that the optimization problem should only depend on the control input variable $u$. To achieve this all the calculations needs to be reformulated to meet this requirement. The QP problem can handle a quadratic cost function, linear system model and linear constraints. Because of this, the vehicle model specified in Equation (3.5) needs to be discretized and linearized.

## 5.1 Discretization and Linearization of Kinematic Bicycle Model

As the vehicle model equations in (3.5) are time-continuous, the kinematic bicycle model needs to be discretized and linearized to use it in the MPC controller. The discretized vehicle model can be formulated as

$$\begin{aligned} X_h(k+1) &= X_h(k) + v(k)\, cos\big(\psi(k) + \beta(k)\big)\, T_s \\ Y_h(k+1) &= Y_h(k) + v(k)\, sin\big(\psi(k) + \beta(k)\big)\, T_s \\ \psi(k+1) &= \psi(k) + \frac{v(k)}{l_r} sin\big(\beta(k)\big)\, T_s \\ v(k+1) &= v(k) + a(k) T_s \\ \forall k &= 1, 2, 3, \dots, N_p \end{aligned} \tag{5.1}$$

Where $T_s$ is the sample time and $k$ represents the steps in the prediction horizon where $N_p$ is the length of the prediction horizon. The kinematic bicycle model can be linearized by using small angle approximation

$$\begin{aligned} sin(\theta) &\approx \theta \\ cos(\theta) &\approx 1 \end{aligned} \tag{5.2}$$

By using the small angle approximation (5.2) in the discretized kinematic bicycle model (5.1) it can be linearized with

$$\begin{aligned} sin\big(\psi(k) + \beta(k)\big) &\approx \psi(k) + \beta(k) \\ cos\big(\psi(k) + \beta(k)\big) &\approx 1 \end{aligned} \tag{5.3}$$

This gives the following discretized and linearized kinematic bicycle model

$$\begin{aligned} X_h(k+1) &= X_h(k) + v(k) T_s \\ Y_h(k+1) &= Y_h(k) + v(k)\psi(k) T_s + v(k)\beta(k) T_s \\ \psi(k+1) &= \psi(k) + \frac{v(k)}{l_r}\beta(k) T_s \\ v(k+1) &= v(k) + a(k) T_s \end{aligned} \tag{5.4}$$

The formulation in Equation (5.4) is problematic for the MPC, since it uses recursive calculations of the state space model to predict the future. To create the recursive calculations the vehicle model needs to be pre-defined for each step in the prediction horizon before the optimization. Because Equation (5.4) is dependent on a time-varying velocity it cannot be written on a linear state space form. This is because the velocity is multiplied with the yaw angle and slip angle which in turn is also part of the vehicle's states. To solve this the velocity can be assumed to be constant over the prediction horizon, this is however not true in the chosen use-cases. To get around this problem the velocity in Equation (5.4) can be defined based on a velocity profile. The velocity profile is created after the decision-making and represents the desired velocity during the horizon. If the controller stays close to the velocity profile the vehicle model will be accurate. This assumption will be valid because the MPC will continuously try to minimize the error between the current velocity and the velocity profile. The explicit time-varying state space model is therefore given by

$$
\begin{aligned}
X_h(k+1) &= X_h(k) + v_{ref}(k)T_s \\
Y_h(k+1) &= Y_h(k) + v_{ref}(k)\psi(k)T_s + v_{ref}(k)\beta(k)T_s \\
\psi(k+1) &= \psi(k) + \frac{v_{ref}(k)}{l_r}\beta(k)T_s \\
v(k+1) &= v(k) + a(k)T_s
\end{aligned} \tag{5.5}
$$

Where $v_{ref}$ is the velocity profile over the prediction horizon. The state space equations can be written in matrix form ass

$$
\underbrace{\begin{bmatrix} X_h(k+1) \\ Y_h(k+1) \\ \psi(k+1) \\ v(k+1) \end{bmatrix}}_{x(k+1)} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & T_s \\ 0 & 1 & v_{ref}(k)T_s & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{A_d(k)} \underbrace{\begin{bmatrix} X_h(k) \\ Y_h(k) \\ \psi(k) \\ v(k) \end{bmatrix}}_{x(k)} + \underbrace{\begin{bmatrix} 0 & 0 \\ v_{ref}(k)T_s & 0 \\ \frac{v_{ref}(k)}{l_r}T_s & 0 \\ 0 & T_s \end{bmatrix}}_{B_d(k)} \underbrace{\begin{bmatrix} \beta(k) \\ a(k) \end{bmatrix}}_{u(k)} \tag{5.6}
$$

Which can be written in a compact form as

$$
x(k+1) = A_d(k)x(k) + B_d(k)u(k) \tag{5.7}
$$

Where $x$ is the vehicle state-vector, $u$ is the input vector and $A_d$ and $B_d$ is the discretized and linearized state space matrices.

## 5.2 Cost function

In the MPC the optimal control inputs are found by solving the QP optimization problem which minimizes a cost function. The cost function consists of cost terms of chosen variables that should be minimized for the prediction horizon $N_p$, the general cost function is defined as

$$
\begin{aligned}
J(k) = \sum_{i=1}^{N_p} & \underbrace{\Big(y(k+i) - y_{ref}(k+i)\Big)^T \lambda_y \Big(y(k+i) - y_{ref}(k+i)\Big)}_{Vehicle\ States} \\
& + \underbrace{\big(u(k+i-1)\big)^T \lambda_u \big(u(k+i-1)\big)}_{Input} \\
& + \underbrace{\big(\Delta u(k+i-1)\big)^T \lambda_{\Delta u} \big(\Delta u(k+i-1)\big)}_{Input\ Increment} + \underbrace{\lambda_{road} U_{road}(k+1)}_{Road\ PF} + \underbrace{\lambda_{obs} U_{obs}(k+1)}_{Obstacle\ PF}
\end{aligned} \tag{5.8}
$$

Where $J$ is the cost, $y = [Y_h, \psi, v]$ is a vector with the vehicle's states included in the cost function and $y_{ref} = \left[Y_{ref}, \psi_{ref}, v_{ref}\right]$ is a vector with reference values. The states that are optimized are the lateral position, yaw angle and absolute velocity where $u = [\beta, a]$ is a vector with the control inputs for slip angle and acceleration. $\lambda$ is the weighting matrix for each cost term. The weighting matrix weight each step in the horizon. However, to simplify the tuning process, the weighting is kept constant for each variable over the whole horizon.

The cost function in (5.8) needs to be written as QP form (1.3), the first step is to substitute the geometric sum with vector notations as such

$$
\begin{aligned}
J(k) = & \underbrace{\big(\tilde{y} - \tilde{y}_{ref}\big)^T \lambda_y \big(\tilde{y} - \tilde{y}_{ref}\big)}_{Vehicle\ States} + \underbrace{\tilde{u}^T \lambda_u \tilde{u}}_{Input} \\
& + \underbrace{\Delta\tilde{u}^T \lambda_{\Delta u} \Delta\tilde{u}}_{Input\ Increment} + \underbrace{\lambda_{road} \tilde{U}_{road}}_{Road\ PF} + \underbrace{\lambda_{obs} \tilde{U}_{obs}}_{Obstacle\ PF}
\end{aligned} \tag{5.9}
$$

Where the ~ notation is used for the vectors and matrices which represents the predicted values throughout the prediction horizon.

The cost function can only depend on the control inputs and stored values. To achieve this the vehicle's states, and the APFs cost terms needs to be rewritten. To express the vehicle's states as control inputs the vehicle model is used.

From Equation (5.6) the vehicle's future state can be calculated with stored values and control inputs. To calculate the vehicle's future states over the whole prediction horizon using stored values and the control inputs the vehicle model is used in recursive calculations. The recursive calculations are made in the following way

$$
\begin{aligned}
x(k+1) &= A_d(k)x(k) + B_d(k)u(k) \\
x(k+2) &= A_d(k+1)x(k+1) + B_d(k+1)u(k+1) \\
&= A_d(k+1)\big(A_d(k)x(k) + B_d(k)u(k)\big) + B_d(k+1)u(k+1) \\
&= A_d(k+1)A_d(k)x(k) + A_d(k+1)B_d(k)u(k) + B_d(k+1)u(k+1) \\
x(k+3) &= A_d(k+2)x(k+2) + B_d u(k+2) \\
&= A_d(k+2)\big(A_d(k+1)A_d(k)x(k) + A_d(k+1)B_d(k)u(k) + B_d(k+1)u(k+1)\big) \\
&+ B_d(k+2)u(k+2) \\
&= A_d(k+2)A_d(k+1)A_d(k)x(k) + A_d(k+2)A_d(k+1)B_d(k)u(k) \\
&+ A_d(k+2)B_d(k+1)u(k+1) + B_d(k+2)u(k+2) \\
&\vdots \\
x\big(k+N_p\big) &= A_d\big(k+N_p-1\big)\dots A_d(k)x(k) + A_d\big(k+N_p-1\big)\dots A_d(k+1)B_d(k)u(k) + \cdots + \\
&+ A_d\big(k+N_p-1\big)B_d\big(k+N_p-2\big)u\big(k+N_p-2\big) + B_d\big(k+N_p-1\big)u\big(k+N_p-1\big)
\end{aligned}
\tag{5.10}
$$

Which can be written in matrix form as

$$
\underbrace{\begin{bmatrix} x(k+1) \\ x(k+2) \\ \vdots \\ x\big(k+N_p\big) \end{bmatrix}}_{\tilde{x}} = \tilde{A}_d x(k) + \tilde{B}_d \underbrace{\begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u\big(k+N_p-1\big) \end{bmatrix}}_{\tilde{u}}
\tag{5.11}
$$

Where $x(k)$ is the vehicle's initial states and the $\tilde{\mathrm{A}}_\mathrm{d}$ and $\tilde{B}_\mathrm{d}$ matrices are defined as follows

$$
\tilde{A}_d = \begin{bmatrix} A_d(k) \\ A_d(k+1)A_d(k) \\ A_d(k+2)A_d(k+1)A_d(k) \\ \vdots \\ \prod_{i=1}^{N_p} A_d(k+i-1) \end{bmatrix}
\tag{5.12}
$$

$$
\tilde{B}_d = \begin{bmatrix} B_d(k) & 0 & \cdots & 0 \\ A_d(k+1)B_d(k) & B_d(k+1) & \ddots & 0 \\ A_d(k+2)A_d(k+1)B_d(k) & A_d(k+2)B_d(k+1) & \ddots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \prod_{i=2}^{N_p}\big(A_d(k+i-1)\big)B_d(k) & \prod_{i=3}^{N_p}\big(A_d(k+i-1)\big)B_d(k+1) & \cdots & B_d(k+i-1) \end{bmatrix}
\tag{5.13}
$$

Not all the vehicle's states are used in the MPC. To gather the right terms for the cost function, the following matrix is used

$$
\underbrace{\begin{bmatrix} y(k+1) \\ y(k+2) \\ \vdots \\ y\big(k+N_p\big) \end{bmatrix}}_{\tilde{y}} = \underbrace{\begin{bmatrix} C_d & 0 & \cdots & 0 \\ 0 & C_d & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & C_d \end{bmatrix}}_{\tilde{C}} \underbrace{\begin{bmatrix} x(k+1) \\ x(k+2) \\ \vdots \\ x\big(k+N_p\big) \end{bmatrix}}_{\tilde{x}}
\tag{5.14}
$$

Where

$$
C_d = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\tag{5.15}
$$

The complete calculation for $\tilde{y}$ in the cost function in Equation (5.9) can be written as

$$\begin{aligned} \tilde{y} &= \tilde{C}\tilde{x} \\ \tilde{y} &= \tilde{C}\left(\tilde{A}_d x(k) + \tilde{B}_d \tilde{u}\right) = \tilde{C}\tilde{A}_d x(k) + \tilde{C}\tilde{B}_d \tilde{u} \end{aligned} \tag{5.16}$$

The input cost function term is defined by

$$\underbrace{\begin{bmatrix} u(k|k) \\ u(k+1|k) \\ \vdots \\ u\left(k+N_p-1\middle|k\right) \end{bmatrix}}_{\tilde{u}} = \underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix}}_{I_{N_p}} \underbrace{\begin{bmatrix} u(k|k) \\ u(k+1|k) \\ \vdots \\ u\left(k+N_p-1\middle|k\right) \end{bmatrix}}_{\tilde{u}} \tag{5.17}$$

The input increments are the change between the control input values throughout the prediction horizon. The input increment cost function term has the following appearance

$$\underbrace{\begin{bmatrix} u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u\left(k+N_p-1\middle|k\right) \end{bmatrix}}_{\Delta\tilde{u}} = \underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 \\ -1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & -1 & 1 \end{bmatrix}}_{\tilde{A}_u} \underbrace{\begin{bmatrix} u(k|k) \\ u(k+1|k) \\ \vdots \\ u\left(k+N_p-1\middle|k\right) \end{bmatrix}}_{\tilde{u}} \tag{5.18}$$

The problem with this structure is that the initial input increment $\Delta u(k)$ is missing from Equation (5.18). Where $\Delta u(k)$ can be calculated as

$$\Delta u(k) = u(k) - u(k-1) \tag{5.19}$$

Where $u(k-1) = u_0$ is the vehicle's initial control input. Because $\Delta\tilde{u}$ is in vector form the initial control input needs to be added into the cost function as a vector that only affects the first-row value of the cost function. The cost function is modified to facilitate this change and is defined as follows

$$\begin{aligned} J(k) = &\underbrace{\left(\tilde{y} - \tilde{y}_{ref}\right)^T \tilde{\lambda}_y \left(\tilde{y} - \tilde{y}_{ref}\right)}_{Vehicle\ States} + \underbrace{\tilde{u}^T \tilde{\lambda}_u \tilde{u}}_{Input} \\ &+ \underbrace{(\Delta\tilde{u} - \tilde{u}_0^*)^T \tilde{\lambda}_{\Delta u} (\Delta\tilde{u} - \tilde{u}_0^*)}_{Input\ Increment} + \underbrace{\tilde{\lambda}_{road}\tilde{U}_{road}}_{Road\ PF} + \underbrace{\tilde{\lambda}_{obs}\tilde{U}_{obs}}_{Obstacle\ PF} \end{aligned} \tag{5.20}$$

The changes are marked with an asterisk in Equation (5.20) and can be written as

$$\tilde{u}_0 = [u_0, 0, \dots, 0]^T \tag{5.21}$$

Where $\tilde{u}_0$ is the same length as the prediction horizon. The initial value is the initial control input of the vehicle while the rest of the values are zero to not affect the rest of the horizon.

## 5.3 Approximated artificial potential fields

The nonlinear APFs are defined by the following equations

$$U_{road} = \underbrace{A_r\left(1 - e^{-b_r(Y_h - Y_{rlc})^2}\right)}_{Right\ lane\ PF} + \underbrace{A_l\left(1 - e^{b_r(Y_h - Y_{llc})^2}\right)}_{Left\ lane\ PF} \tag{5.22}$$

$$U_{obs} = A_o e^{-(X_h + X_{obs})^2 - (Y_h + Y_{obs})^2} \tag{5.23}$$

Because the QP problem has a quadratic cost function the APFs need to be approximated. This is done with a general two-dimensional quadratic Taylor series approximation which is defined as

$$\begin{aligned} Q_f(x,y) \approx\ & f(x_0,y_0) + f_x(x_0,y_0)(x - x_0) + f_y(x_0,y_0)(y - y_0) \\ &+ f_{xy}(x_0,y_0)(x - x_0)(y - y_0) \\ &+ \frac{1}{2} f_{xx}(x_0,y_0)(x - x_0)^2 + \frac{1}{2} f_{yy}(x_0,y_0)(y - y_0)^2 \end{aligned} \tag{5.24}$$

Where $Q_f$ is the approximated function around $(x_0, y_0)$ and $(x, y)$ is the actual point. To use (5.24) in the QP problem it is rewritten on vector and matrix form. The approximation is the same for the road as for the obstacle APF where the approximated APFs around a point $\big(X_h(k), Y_h(k)\big)$ for one step can be written as

$$U(k+1) \approx U_0(k) + U_1(k)\begin{bmatrix} X_h(k+1) \\ Y_h(k+1) \end{bmatrix} + \frac{1}{2}\begin{bmatrix} X_h(k+1) \\ Y_h(k+1) \end{bmatrix}^T U_2(k)\begin{bmatrix} X_h(k+1) \\ Y_h(k+1) \end{bmatrix} \tag{5.25}$$

And the $X_h(k+1)$ and $Y_h(k+1)$ can be extracted from the state vector in the following way

$$\underbrace{\begin{bmatrix} X_h(k+1) \\ Y_h(k+1) \end{bmatrix}}_{y_{XY}(k+1)} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{C_{XY}} \underbrace{\begin{bmatrix} X_h(k+1) \\ Y_h(k+1) \\ \varphi(k+1) \\ v(k+1) \end{bmatrix}}_{x(k+1)} \tag{5.26}$$

Which gives

$$U(k+1) \approx U_0(k) + U_1(k) y_{XY}(k+1) + \frac{1}{2} y_{XY}(k+1)^T U_2(k) y_{XY}(k+1) \tag{5.27}$$

Where

$$\begin{aligned} U_0(k) = {} & U\big(X_h(k), Y_h(k)\big) - U_{X_h}(k)X_h(k) - U_{Y_h}(k)Y_h(k) \\ & + \frac{1}{2}U_{X_hX_h}(k)X_h(k)^2 + U_{X_hY_h}(k)X_h(k)Y_h(k) + \frac{1}{2}U_{Y_hY_h}(k)Y_h(k)^2 \end{aligned} \tag{5.28}$$

$$U_1(k) = \begin{bmatrix} U_{X_h}(k) - U_{X_hX_h}(k)X_h(k) - U_{X_hY_h}(k)Y_h(k) \\ U_{Y_h}(k) - U_{X_hY_h}(k)X_h(k) - U_{Y_hY_h}(k)Y_h(k) \end{bmatrix}^T \tag{5.29}$$

$$U_2(k) = \begin{bmatrix} U_{X_hX_h}(k) & U_{X_hY_h}(k) \\ U_{Y_hX_h}(k) & U_{Y_hY_h}(k) \end{bmatrix} \tag{5.30}$$

By using vector notation $\tilde{U}_0, \tilde{U}_1$ and $\tilde{U}_2$ the general approximated APFs for the whole horizon can be determined

$$\tilde{U}_0 = \begin{bmatrix} U_0(k) & U_0(k+1) & \cdots & U_0\big(k+N_p\big) \end{bmatrix} \tag{5.31}$$

$$\tilde{U}_1 = \begin{bmatrix} U_1(k) & U_1(k+1) & \cdots & U_1\big(k+N_p\big) \end{bmatrix} \tag{5.32}$$

$$\tilde{U}_2 = \begin{bmatrix} U_2(k) & 0 & \cdots & 0 \\ 0 & U_2(k+1) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & U_2\big(k+N_p\big) \end{bmatrix} \tag{5.33}$$

The complete general APF calculation for the whole horizon can be written as

$$\tilde{U} \approx \tilde{U}_0 + \tilde{U}_1 \tilde{y}_{XY} + \frac{1}{2}{\tilde{y}_{XY}}^T \tilde{U}_2 \tilde{y}_{XY} \tag{5.34}$$

Where $\tilde{y}_{XY}$ is determined by

$$\underbrace{\begin{bmatrix} y(k+1) \\ y(k+1) \\ \vdots \\ y\big(k+N_p\big) \end{bmatrix}}_{\tilde{y}_{XY}} = \underbrace{\begin{bmatrix} C_{XY} & 0 & \dots & 0 \\ 0 & C_{XY} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & C_{XY} \end{bmatrix}}_{\tilde{C}_{XY}} \underbrace{\begin{bmatrix} x(k+1) \\ x(k+2) \\ \vdots \\ x\big(k+N_p\big) \end{bmatrix}}_{\tilde{x}} \tag{5.35}$$

The APF Equation (5.34) needs to depend on the control input. This can be made in the same way as the calculation in (5.16).

$$\begin{aligned} \tilde{y}_{XY} &= \tilde{C}_{XY}\tilde{x} \\ \tilde{y}_{XY} &= \tilde{C}_{XY}\big(\tilde{A}_d x(k) + \tilde{B}_d \tilde{u}\big) = \tilde{C}_{XY}\tilde{A}_d x(k) + \tilde{C}_{XY}\tilde{B}_d \tilde{u} \end{aligned} \tag{5.36}$$

By inserting Equation (5.36) in Equation (5.34) the terms that depend on the control input can be found. All the terms that does not contain the control input $\tilde{u}$ does not influence the optimization problem and can therefore be excluded from the calculation in Equation (5.37).

$$\begin{aligned}\tilde{U} &\approx \tilde{U}_0 + \tilde{U}_1\tilde{y}_{XY} + \frac{1}{2}\tilde{y}_{XY}{}^T\tilde{U}_2\tilde{y}_{XY} \\ &\approx \tilde{U}_0 + \tilde{U}_1\left(\tilde{C}_{XY}\tilde{A}_d x(k) + \tilde{C}_{XY}\tilde{B}_d\tilde{u}\right) + \frac{1}{2}\left(\tilde{C}_{XY}\tilde{A}_d x(k) + \tilde{C}_{XY}\tilde{B}_d\tilde{u}\right)^T\tilde{U}_2\left(\tilde{C}_{XY}\tilde{A}_d x(k) + \tilde{C}_{XY}\tilde{B}_d\tilde{u}\right) \\ &\approx \underbrace{\tilde{U}_0}_{No\ \tilde{u}} + \underbrace{\tilde{U}_1\tilde{C}_{XY}\tilde{A}_d x(k)}_{No\ \tilde{u}} + \tilde{U}_1\tilde{C}_{XY}\tilde{B}_d\tilde{u} + \underbrace{\frac{1}{2}\left(\tilde{C}_{XY}\tilde{A}_d x(k)\right)^T\tilde{U}_2\left(\tilde{C}_{XY}\tilde{A}_d x(k)\right)}_{No\ \tilde{u}} \\ &+ \left(\tilde{C}_{XY}\tilde{A}_d x(k)\right)^T\tilde{U}_2\left(\tilde{C}_{XY}\tilde{B}_d\tilde{u}\right) + \frac{1}{2}\left(\tilde{C}_{XY}\tilde{B}_d\tilde{u}\right)^T\tilde{U}_2\left(\tilde{C}_{XY}\tilde{B}_d\tilde{u}\right)\end{aligned} \tag{5.37}$$

The approximated APFs for road and obstacle respectively can be obtained by calculating the derivatives and partial derivatives of the respective equations. Because of the simplification of a straight road assumption the road APF does not change based on $X_h$ and thus the derivatives of $X_h$ are zero.

$$\begin{aligned}U_{road,X_h}(k) &= 0 \\ U_{road,Y_h}(k) &= b_r(A_r + A_l)\left(e^{-b_r\left(Y_{ref}(k)-Y_{rlc}\right)} - e^{-2b_r\left(Y_{ref}(k)-Y_{rlc}\right)}\right) \\ &\quad + b_r(A_r + A_l)\left(-e^{b_r\left(Y_{ref}(k)-Y_{llc}\right)} + e^{2b_r\left(Y_{ref}(k)-Y_{llc}\right)}\right) \\ U_{road,X_hX_h}(k) &= 0 \\ U_{road,X_hY_h}(k) &= 0 \\ U_{road,Y_hY_h}(k) &= b_r^2(A_r + A_l)\left(2e^{-2b_r\left(Y_{ref}(k)-Y_{rlc}\right)} - e^{-b_r\left(Y_{ref}(k)-Y_{rlc}\right)}\right) \\ &\quad + b_r^2(A_r + A_l)\left(2e^{2b_r\left(Y_{ref}(k)-Y_{llc}\right)} - e^{b_r\left(Y_{ref}(k)-Y_{llc}\right)}\right)\end{aligned} \tag{5.38}$$

And the obstacle derivatives for one timestep can be written as

$$\begin{aligned}U_{obs,X_h}(k) &= -2A_{obs}x_s\left(X_{ref}(k) - X_{obs}(k)\right)e^{-x_s\left(X_{ref}(k)-X_{obs}(k)\right)^2 - y_s\left(Y_{ref}(k)-Y_{obs}(k)\right)^2} \\ U_{obs,Y_h}(k) &= -2A_{obs}y_s\left(Y_{ref}(k) - Y_{obs}(k)\right)e^{-x_s\left(X_{ref}(k)-X_{obs}(k)\right)^2 - y_s\left(Y_{ref}(k)-Y_{obs}(k)\right)^2} \\ U_{obs,X_hX_h}(k) &= 2A_{obs}x_s\left(2x_s\left(X_{ref}(k) - X_{obs}(k)\right)^2 - 1\right)e^{-x_s\left(X_{ref}(k)-X_{obs}(k)\right)^2 - y_s\left(Y_{ref}(k)-Y_{obs}(k)\right)^2} \\ U_{obs,X_hY_h}(k) &= 4A_{obs}x_sy_s\left(X_{ref}(k) - X_{obs}(k)\right)\left(Y_{ref}(k) - Y_{obs}(k)\right) \\ &\quad \dots\ e^{-x_s\left(X_{ref}(k)-X_{obs}(k)\right)^2 - y_s\left(Y_{ref}(k)-Y_{obs}(k)\right)^2} \\ U_{obs,Y_hX_h}(k) &= 4A_{obs}x_sy_s\left(X_{ref}(k) - X_{obs}(k)\right)\left(Y_{ref}(k) - Y_{obs}(k)\right) \\ &\quad \dots\ e^{-x_s\left(X_{ref}(k)-X_{obs}(k)\right)^2 - y_s\left(Y_{ref}(k)-Y_{obs}(k)\right)^2} \\ U_{obs,Y_hY_h}(k) &= 2A_{obs}y_s\left(2y_s\left(Y_{ref}(k) - Y_{obs}(k)\right)^2 - 1\right)e^{-x_s\left(X_{ref}(k)-X_{obs}(k)\right)^2 - y_s\left(Y_{ref}(k)-Y_{obs}(k)\right)^2}\end{aligned} \tag{5.39}$$

## 5.4 Weighting matrices

Each variable in the cost function is weighted individually. For simplicity the weighting is kept constant for each variable throughout the horizon. The weighting matrix for the vehicle's states is defined as

$$\lambda_y = \begin{bmatrix}\lambda_{Y_h} & 0 & 0 \\ 0 & \lambda_\psi & 0 \\ 0 & 0 & \lambda_v\end{bmatrix}, \qquad \tilde{\lambda}_y = \begin{bmatrix}\lambda_y & 0 & \cdots & 0 \\ 0 & \lambda_y & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_y\end{bmatrix} \tag{5.40}$$

The weighting matrix for the control input and control input increment can be determined in a similar way

$$\lambda_u = \begin{bmatrix} \lambda_\beta & 0 \\ 0 & \lambda_a \end{bmatrix}, \qquad \tilde{\lambda}_u = \lambda_u I_{N_p} \tag{5.41}$$

$$\lambda_{\Delta u} = \begin{bmatrix} \lambda_{\Delta\beta} & 0 \\ 0 & \lambda_{\Delta a} \end{bmatrix}, \qquad \tilde{\lambda}_{\Delta u} = \lambda_{\Delta u} I_{N_p} \tag{5.42}$$

The weighting matrices for the APFs are simpler than the previous weighting matrices and can be written as

$$\tilde{\lambda}_r = \lambda_r I_{N_p}, \qquad \tilde{\lambda}_{obs} = \lambda_{obs} I_{N_p} \tag{5.43}$$

## 5.5 Combined cost function / Complete cost function

The cost function used in the MPC can now be retrieved. By inserting Equation (5.16), (5.18) and (5.34) into the modified cost function (5.20) the following equation is found

$$\begin{aligned} J(k) = & \underbrace{\left(\tilde{C}\tilde{A}_d x(k) + \tilde{C}\tilde{B}_d \tilde{u} - \tilde{y}_{ref}\right)^T \tilde{\lambda}_y \left(\tilde{C}\tilde{A}_d x(k) + \tilde{C}\tilde{B}_d \tilde{u} - \tilde{y}_{ref}\right)}_{Vehicle\ States} + \underbrace{\tilde{u}^T \tilde{\lambda}_u \tilde{u}}_{Input} \\ & + \underbrace{\left(\tilde{A}_u \tilde{u} - \tilde{u}_0\right)^T \tilde{\lambda}_{\Delta u} \left(\tilde{A}_u \tilde{u} - \tilde{u}_0\right)}_{Input\ Increment} \\ & + \underbrace{\tilde{\lambda}_{\text{road}} \left( \tilde{U}_{1,\text{road}} \tilde{C}_{XY} \tilde{B}_d \tilde{u} + \left(\tilde{C}_{XY}\tilde{A}_d x(k)\right)^T \tilde{U}_{2,\text{road}} \left(\tilde{C}_{XY}\tilde{B}_d \tilde{u}\right) + \frac{1}{2}\left(\tilde{C}_{XY}\tilde{B}_d \tilde{u}\right)^T \tilde{U}_{2,\text{road}} \left(\tilde{C}_{XY}\tilde{B}_d \tilde{u}\right) \right)}_{Road\ APF} \\ & + \underbrace{\tilde{\lambda}_{obs} \left( \tilde{U}_{1,\text{obs}} \tilde{C}_{XY} \tilde{B}_d \tilde{u} + \left(\tilde{C}_{XY}\tilde{A}_d x(k)\right)^T \tilde{U}_{2,\text{obs}} \left(\tilde{C}_{XY}\tilde{B}_d \tilde{u}\right) + \frac{1}{2}\left(\tilde{C}_{XY}\tilde{B}_d \tilde{u}\right)^T \tilde{U}_{2,\text{obs}} \left(\tilde{C}_{XY}\tilde{B}_d \tilde{u}\right) \right)}_{Obstacle\ APF} \end{aligned} \tag{5.44}$$

Which can be simplified as

$$\begin{aligned} J(k) = & \underbrace{\left(\tilde{C}\tilde{A}_d x(k)\right)^T \tilde{\lambda}_y \left(\tilde{C}\tilde{A}_d x(k)\right)}_{Vehicle\ States,\ No\ \tilde{u}} + 2\underbrace{\left(\tilde{C}\tilde{A}_d x(k)\right)^T \tilde{\lambda}_y \left(\tilde{C}\tilde{B}_d \tilde{u}\right)}_{Vehicle\ States} - 2\underbrace{\left(\tilde{C}\tilde{A}_d x(k)\right)^T \tilde{\lambda}_y \left(\tilde{y}_{ref}\right)}_{Vehicle\ States,\ No\ \tilde{u}} \\ & + \underbrace{\left(\tilde{C}\tilde{B}_d \tilde{u}\right)^T \tilde{\lambda}_y \left(\tilde{C}\tilde{B}_d \tilde{u}\right)}_{Vehicle\ States} - 2\underbrace{\left(\tilde{C}\tilde{B}_d \tilde{u}\right)^T \tilde{\lambda}_y \left(\tilde{y}_{ref}\right)}_{Vehicle\ States} + \underbrace{\left(\tilde{y}_{ref}\right)^T \tilde{\lambda}_y \left(\tilde{y}_{ref}\right)}_{Vehicle\ States,\ No\ \tilde{u}} \\ & + \underbrace{\tilde{u}^T \tilde{\lambda}_u \tilde{u}}_{Input} + \underbrace{\left(\tilde{A}_u \tilde{u}\right)^T \tilde{\lambda}_{\Delta u} \left(\tilde{A}_u \tilde{u}\right)}_{Input\ Increment} - 2\underbrace{\left(\tilde{A}_u \tilde{u}\right)^T \tilde{\lambda}_{\Delta u} \left(\tilde{u}_0\right)}_{Input\ Increment} + \underbrace{\left(\tilde{u}_0\right)^T \tilde{\lambda}_{\Delta u} \left(\tilde{u}_0\right)}_{Input\ Increment,\ No\ \tilde{u}} \\ & + \underbrace{\tilde{\lambda}_{\text{road}} \left( \tilde{U}_{1,\text{road}} \tilde{C}_{XY} \tilde{B}_d \tilde{u} + \left(\tilde{C}_{XY}\tilde{A}_d x(k)\right)^T \tilde{U}_{2,\text{road}} \left(\tilde{C}_{XY}\tilde{B}_d \tilde{u}\right) + \frac{1}{2}\left(\tilde{C}_{XY}\tilde{B}_d \tilde{u}\right)^T \tilde{U}_{2,\text{road}} \left(\tilde{C}_{XY}\tilde{B}_d \tilde{u}\right) \right)}_{Road\ APF} \\ & + \underbrace{\tilde{\lambda}_{obs} \left( \tilde{U}_{1,\text{obs}} \tilde{C}_{XY} \tilde{B}_d \tilde{u} + \left(\tilde{C}_{XY}\tilde{A}_d x(k)\right)^T \tilde{U}_{2,\text{obs}} \left(\tilde{C}_{XY}\tilde{B}_d \tilde{u}\right) + \frac{1}{2}\left(\tilde{C}_{XY}\tilde{B}_d \tilde{u}\right)^T \tilde{U}_{2,\text{obs}} \left(\tilde{C}_{XY}\tilde{B}_d \tilde{u}\right) \right)}_{Obstacle\ APF} \end{aligned} \tag{5.45}$$

The terms that does not contain the control input $\tilde{u}$ can be removed as they do not influence the optimization problem in Equation (5.45).

The Hessian matrix $H$ and the linear vector $f$ can be calculated from Equation (5.45) by retrieving the terms that depend on the control input $\tilde{u}$. The Hessian matrix is constructed from quadratic terms while the linear vector is created from the linear terms.

$$
\begin{aligned}
H &= 2\underbrace{\left(\tilde{C}\tilde{B}_d\right)^T\tilde{\lambda}_y\left(\tilde{C}\tilde{B}_d\right)}_{Vehicle\ States} + 2\underbrace{\tilde{\lambda}_u I_{N_p}}_{Input} + 2\underbrace{\left(\tilde{A}_u\right)^T\tilde{\lambda}_{\Delta u}\left(\tilde{A}_u\right)}_{Input\ Increment} \\
&+ \underbrace{\tilde{\lambda}_{\text{road}}\left(\left(\tilde{C}_{XY}\tilde{B}_d\right)^T\tilde{U}_{2,\text{road}}\left(\tilde{C}_{XY}\tilde{B}_d\right)\right)}_{Road\ APF} + \underbrace{\tilde{\lambda}_{obs}\left(\left(\tilde{C}_{XY}\tilde{B}_d\right)^T\tilde{U}_{2,\text{obs}}\left(\tilde{C}_{XY}\tilde{B}_d\right)\right)}_{Obstacle\ APF} \\
f &= 2\underbrace{\left(\tilde{C}\tilde{A}_d x(k)\right)^T\tilde{\lambda}_y\left(\tilde{C}\tilde{B}_d\right)}_{Vehicle\ States} - 2\underbrace{\left(\tilde{C}\tilde{B}_d\right)^T\tilde{\lambda}_y\left(\tilde{y}_{ref}\right)}_{Vehicle\ States} - 2\underbrace{\left(\tilde{A}_u\right)^T\tilde{\lambda}_{\Delta u}\left(\tilde{u}_0\right)}_{Input\ Increment} \\
&+ \underbrace{\tilde{\lambda}_{\text{road}}\left(\tilde{U}_{1,\text{road}}\tilde{C}_{XY}\tilde{B}_d + \left(\tilde{C}_{XY}\tilde{A}_d x(k)\right)^T\tilde{U}_{2,\text{road}}\left(\tilde{C}_{XY}\tilde{B}_d\right)\right)}_{Road\ APF} \\
&+ \underbrace{\tilde{\lambda}_{obs}\left(\tilde{U}_{1,obs}\tilde{C}_{XY}\tilde{B}_d + \left(\tilde{C}_{XY}\tilde{A}_d x(k)\right)^T\tilde{U}_{2,obs}\left(\tilde{C}_{XY}\tilde{B}_d\right)\right)}_{Road\ APF}
\end{aligned}
\tag{5.46}
$$

Where the Hessian terms need to be multiplied with 2 to write the cost function (5.45) on QP form (1.3).

## 5.6 Constraints

The MPC uses a variety of different constraints to limit the solutions from the controller. The constraints are implemented because of comfort reasons, such as: acceleration or jerk, or because of safety and legal reasons such as: road boundaries and absolute velocity. All the constraints and their respective values are listed in Table 5.1.

Table 5.1: Constraints

| Name | Symbol | Maximum | Minimum | Unit |
|---|---|---|---|---|
| Road boundaries | $Y$ | 3.75 | -3.75 | $[m]$ |
| Yaw angle | $\psi$ | 0.78 | -0.78 | $[rad]$ |
| Absolute velocity | $v$ | 13.4 | 0 | $\left[\frac{m}{s}\right]$ |
| Side slip angle | $\beta$ | 0.0524 | -0.0524 | $[rad]$ |
| Acceleration | $a$ | 2 | -3 | $\left[\frac{m}{s^2}\right]$ |
| Side slip angular velocity | $\Delta\beta$ | 0.03 | -0.03 | $\left[\frac{rad}{s}\right]$ |
| Jerk | $\Delta a$ | 0.25 | -0.25 | $\left[\frac{m}{s^3}\right]$ |
| Side Slip angular acceleration | $\partial\beta$ | 0.002 | -0.002 | $\left[\frac{rad}{s^2}\right]$ |
| Snap | $\partial a$ | 0.03 | 0.03 | $\left[\frac{m}{s^4}\right]$ |

The MPC controllers' constraints constructs boundaries for the environment and the vehicle. All the constraints for the MPC controller are inequality constraints which will create a region between two definite values. The state constraints are defined by the following inequality constraints

$$\begin{aligned} Y_{rrb} \leq Y_h &\leq Y_{lrb} \\ -\psi_{max} \leq \psi &\leq \psi_{max} \\ v_{min} \leq v &\leq v_{max} \end{aligned} \tag{5.47}$$

The states are constrained through the road boundaries, the maximum yaw angle and the ego vehicle's minimum and maximum absolute velocity. The maximum yaw angle is included in the system to keep the vehicle from ending up in the wrong direction in relation to the road. The maximum velocity is the speed limit of the road and the minimum velocity is zero. This allows the vehicle to come to a complete stop but not allow the controller to find solutions that rely on negative velocities. The control input constraints are defined as

$$\begin{aligned} -\beta_{max} \leq \beta &\leq \beta_{max} \\ a_{min} \leq a &\leq a_{max} \\ -\Delta\beta_{max} \leq \Delta\beta &\leq \Delta\beta_{max} \\ \Delta a_{min} \leq \Delta a &\leq \Delta a_{max} \\ -\partial\beta_{max} \leq \partial\beta &\leq \partial\beta_{max} \\ -\partial a_{max} \leq \partial a &\leq \partial a_{max} \end{aligned} \tag{5.48}$$

The control inputs are constrained for slip angle, slip angular velocity, slip angular acceleration, acceleration, jerk and snap. The slip angle constraint is set to keep the validity of the small angle approximation [20]. The acceleration constraints are based on the typical human driving style [10]. From [13] the wheel angular velocity for comfortable driving is defined as 0.5 rad/s. With the relationship between slip angle and wheel angle in Equation (3.6) the slip angular velocity can be determined by derivation of both sides

$$\frac{d}{dt}\beta = \frac{d}{dt}\tan^{-1}\left(\frac{l_r}{L}\tan(\delta)\right) \tag{5.49}$$

$$\dot{\beta} = \frac{\frac{l_r}{L}\dot{\delta}}{\left(\frac{l_r}{L}\right)^2 \sin^2(\delta) + \cos^2(\delta)} \tag{5.50}$$

Where $\dot{\beta}$ is the slip angular velocity and $\dot{\delta}$ is the wheel angular velocity. The corresponding slip angular velocity constraint is at its lowest when the wheel angle is zero. This gives the following relationship

$$\dot{\beta} = \frac{l_r}{L}\dot{\delta} \tag{5.51}$$

To determine the slip angular velocity constraint the wheel angular velocity is set to 0.5 rad/s. The relationship in Equation (5.51) is valid for zero radian wheel angles and define the slip angular velocity constraint for all wheel angles.

In [21] the comfort levels for lateral acceleration as well as the longitudinal and lateral jerk are defined. The longitudinal jerk from [21] is used as a constraint in the MPC while the lateral acceleration and jerk is used in the evaluation of ride comfort. The slip angular acceleration and snap are arbitrarily set to reduce noise in the system.

The constraints are rewritten to fit the QP problem form (1.3). The state constraints are written in vector form as

$$\underbrace{\begin{bmatrix} Y_h(k+1) \\ -Y_h(k+1) \\ \psi(k+1) \\ -\psi(k+1) \\ v(k+1) \\ -v(k+1) \end{bmatrix}}_{y_{ineq}(k+1)} \leq \underbrace{\begin{bmatrix} Y_{lrb}(k+1) \\ -Y_{rrb}(k+1) \\ \psi_{max}(k+1) \\ \psi_{max}(k+1) \\ v_{max}(k+1) \\ -v_{min}(k+1) \end{bmatrix}}_{f_{ineq}(k+1)} \tag{5.52}$$

The state constraints need to be rewritten to depend on the control inputs, similarly to the cost function the state constraints are constructed from the state vector

$$\underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix}}_{C_{ineq}} \underbrace{\begin{bmatrix} X_h(k+1) \\ Y_h(k+1) \\ \varphi(k+1) \\ v(k+1) \end{bmatrix}}_{x(k+1)} \leq \underbrace{\begin{bmatrix} Y_{lrb}(k+1) \\ -Y_{rrb}(k+1) \\ \psi_{max}(k+1) \\ \psi_{max}(k+1) \\ v_{max}(k+1) \\ -v_{min}(k+1) \end{bmatrix}}_{f_{ineq}(k+1)} \tag{5.53}$$

The constraints are then repeated for each step in the prediction horizon

$$\underbrace{\begin{bmatrix} C_{ineq} & 0 & \cdots & 0 \\ 0 & C_{ineq} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & C_{ineq} \end{bmatrix}}_{\tilde{C}_{ineq}} \underbrace{\begin{bmatrix} x(k+1) \\ x(k+2) \\ \vdots \\ x(k+N_p) \end{bmatrix}}_{\tilde{x}} \leq \underbrace{\begin{bmatrix} f_{ineq}(k+1) \\ f_{ineq}(k+2) \\ \vdots \\ f_{ineq}(k+N_p) \end{bmatrix}}_{\tilde{f}_{ineq}} \tag{5.54}$$

The above equation is rewritten to depend on the control input in the following way

$$\begin{aligned} \tilde{C}_{ineq}\tilde{x} &\leq \tilde{f}_{ineq} \\ \tilde{C}_{ineq}\left(\tilde{A}_d x(k) + \tilde{B}_d \tilde{u}\right) &\leq \tilde{f}_{ineq} \\ \underbrace{\tilde{C}_{ineq}\tilde{B}_d}_{\tilde{A}_{x,ineq}} \tilde{u} &\leq \underbrace{\tilde{f}_{ineq} - \tilde{C}_{ineq}\tilde{A}_d x(k)}_{\tilde{b}_{x,ineq}} \end{aligned} \tag{5.55}$$

Which can be formulated on compact form as

$$\tilde{A}_{x,ineq}\tilde{u} \leq \tilde{b}_{x,ineq} \tag{5.56}$$

The control input constraints are written in vector form as

$$\underbrace{\begin{bmatrix} \beta(k+1) \\ -\beta(k+1) \\ a(k+1) \\ -a(k+1) \\ \Delta\beta(k+1) \\ -\Delta\beta(k+1) \\ \Delta a(k+1) \\ -\Delta a(k+1) \\ \partial\beta(k+1) \\ -\partial\beta(k+1) \\ \partial a(k+1) \\ -\partial a(k+1) \end{bmatrix}}_{u_{ineq}(k+1)} \leq \underbrace{\begin{bmatrix} \beta_{max}(k+1) \\ \beta_{max}(k+1) \\ a_{max}(k+1) \\ -a_{min}(k+1) \\ \Delta\beta_{max}(k+1) \\ \Delta\beta_{max}(k+1) \\ \Delta a_{max}(k+1) \\ -\Delta a_{min}(k+1) \\ \partial\beta_{max}(k+1) \\ \partial\beta_{max}(k+1) \\ \partial a_{max}(k+1) \\ \partial a_{max}(k+1) \end{bmatrix}}_{b_{u,ineq}(k+1)} \tag{5.57}$$

Where (5.57) needs to be written on QP form (1.3). To do this it needs to be rewritten to depend on the control input. For the first derivative this is done in the following way

$$\begin{aligned} \Delta\beta(k) &= \beta(k) - \beta(k-1) \\ \Delta\beta(k+1) &= \beta(k+1) - \beta(k) \\ \Delta\beta(k+2) &= (k+2) - \beta(k+1) \\ &\vdots \\ \Delta\beta(k+N_p) &= \beta(k+N_p) - \beta(k+N_p-1) \end{aligned} \tag{5.58}$$

Where $\beta(k)$ is the optimal slip angle for the current timestep and $\beta(k-1)$ is the initial slip angle. The second derivative can be determined as

$$
\begin{aligned}
\partial\beta(k) &= \Delta\beta(k) - \Delta\beta(k-1) \\
&= \big(\beta(k) - \beta(k-1)\big) - \big(\beta(k-1) - \beta(k-2)\big) \\
&= \beta(k) - 2\beta(k-1) + \beta(k-2) \\
\\
\partial\beta(k+1) &= \Delta\beta(k+1) - \Delta\beta(k) \\
&= \big(\beta(k+1) - \beta(k)\big) - \big(\beta(k) - \beta(k-1)\big) \\
&= \beta(k+1) - 2\beta(k) + \beta(k-1) \\
\\
\partial\beta(k+2) &= \Delta\beta(k+2) - \Delta\beta(k+1) \\
&= \big(\beta(k+2) - \beta(k+1)\big) - \big(\beta(k+1) - \beta(k)\big) \\
&= \beta(k+2) - 2\beta(k+1) + \beta(k) \\
&\vdots \\
\partial\beta\big(k+N_p\big) &= \Delta\beta\big(k+N_p\big) - \Delta\beta\big(k+N_p-1\big) \\
&= \beta\big(k+N_p\big) - 2\beta\big(k+N_p-1\big) + \beta\big(k+N_p-2\big)
\end{aligned}
\tag{5.59}
$$

Where $\beta(k-2)$ is the previous prediction horizons initial slip angle. The initial slip angle $\beta(k-1)$ and the previous loops initial slip angle $\beta(k-2)$ is not present in the control input vector as can be seen in Equation (5.17). These values are stored values and are incorporated into the constraints when the they are created. The constraints depending on the control input can be written on matrix form with the following expressions

$$
A_{u,ineq,1} = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}, \quad A_{u,ineq,2} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ -2 & 0 \\ 2 & 0 \\ 0 & -2 \\ 0 & 2 \end{bmatrix}, \quad A_{u,ineq,3} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}
\tag{5.60}
$$

These three matrices can be combined to construct the complete control input matrix for the whole prediction horizon

$$
\tilde{A}_{u,ineq} = \begin{bmatrix}
A_{u,ineq,1} & 0 & 0 & \cdots & & \cdots & 0 \\
A_{u,ineq,2} & A_{u,ineq,1} & 0 & & & & \\
A_{u,ineq,3} & A_{u,ineq,2} & A_{u,ineq,1} & & \ddots & \ddots & \vdots \\
0 & A_{u,ineq,3} & A_{u,ineq,2} & \ddots & & & \\
0 & 0 & A_{u,ineq,3} & & A_{u,ineq,1} & 0 & 0 \\
\vdots & \ddots & & \ddots & A_{u,ineq,2} & A_{u,ineq,1} & 0 \\
0 & \cdots & & \cdots & A_{u,ineq,3} & A_{u,ineq,2} & A_{u,ineq,1}
\end{bmatrix}
\tag{5.61}
$$

To include the stored values from the previous prediction horizons $b_{u,ineq}$ is divided into three parts. Where $b_{u,ineq,1}$ is the inequality constraints of the control input for the first step, $b_{u,ineq,2}$ is the second step and $b_{u,ineq,3}$ describes the control input constraints for the rest of the prediction horizon. The stored values from the equations in (5.58) and (5.59) are subtracted from the right side and added to the left side. This gives the following vectors

$$b_{u,ineq,1} = \underbrace{\begin{bmatrix} \beta_{max}(k) \\ \beta_{max}(k) \\ a_{max}(k) \\ -a_{min}(k) \\ \Delta\beta_{max}(k) + \beta(k-1) \\ \Delta\beta_{max}(k) - \beta(k-1) \\ \Delta a_{max}(k) + a(k-1) \\ -\Delta a_{min}(k) - a(k-1) \\ \partial\beta_{max}(k) + 2\beta(k-1) - \beta(k-2) \\ \partial\beta_{max}(k) - 2\beta(k-1) + \beta(k-2) \\ \partial a_{max}(k) + 2a(k-1) - a(k-2) \\ \partial a_{max}(k) - 2a(k-1) + a(k-2) \end{bmatrix}}_{b_{u,ineq}(k)} \tag{5.62}$$

$$b_{u,ineq,2} = \underbrace{\begin{bmatrix} \beta_{max}(k+1) \\ \beta_{max}(k+1) \\ a_{max}(k+1) \\ -a_{min}(k+1) \\ \Delta\beta_{max}(k+1) \\ \Delta\beta_{max}(k+1) \\ \Delta a_{max}(k+1) \\ -\Delta a_{min}(k+1) \\ \partial\beta_{max}(k+1) - \beta(k-1) \\ \partial\beta_{max}(k+1) + \beta(k-1) \\ \partial a_{max}(k+1) - a(k-1) \\ \partial a_{max}(k+1) + a(k-1) \end{bmatrix}}_{b_{u,ineq}(k+1)} \tag{5.63}$$

$$b_{u,ineq,3} = \underbrace{\begin{bmatrix} \beta_{max}(k+1) \\ \beta_{max}(k+1) \\ a_{max}(k+1) \\ -a_{min}(k+1) \\ \Delta\beta_{max}(k+1) \\ \Delta\beta_{max}(k+1) \\ \Delta a_{max}(k+1) \\ -\Delta a_{min}(k+1) \\ \partial\beta_{max}(k+1) \\ \partial\beta_{max}(k+1) \\ \partial a_{max}(k+1) \\ \partial a_{max}(k+1) \end{bmatrix}}_{b_{u,ineq}(k+2) \to b_{u,ineq}(k+N_p)} \tag{5.64}$$

Where the complete $\tilde{b}_{u,ineq}$ can be written as

$$\tilde{b}_{u,ineq} = \begin{bmatrix} b_{u,ineq,1} \\ b_{u,ineq,2} \\ b_{u,ineq,3} \\ \vdots \\ b_{u,ineq,3} \end{bmatrix} \tag{5.65}$$

The total inequality matrix that will be used in the MPC QP problem can be formulated as

$$\underbrace{\begin{bmatrix} \tilde{A}_{x,ineq} \\ \tilde{A}_{u,ineq} \end{bmatrix}}_{\tilde{A}_{ineq}} \tilde{u} \leq \underbrace{\begin{bmatrix} \tilde{b}_{x,ineq} \\ \tilde{b}_{u,ineq} \end{bmatrix}}_{\tilde{b}_{ineq}} \tag{5.66}$$

# 6 Simulation

This chapter presents the results and tuning process of the MPC controller. The results from each use-case is analyzed based on safety, decision-making and smoothness. The controller's priority is to obtain collision free trajectories that keep an adequate safety distance towards its surrounding obstacles. The controller should make maneuvers that are predictable to other drivers as well as to the passengers of the ego vehicle. The vehicle's maneuvers should also be smooth and comfortable for the passengers. The simulations were running on a Dell Precision 7510 with an Intel core i7-6820HQ at 2.7 GHz using MATLAB 2010 with the optimization toolbox.

For all simulations the ego vehicle starts the simulation traveling at 8 m/s in the center of the left lane. This is faster than the vehicle's low speed during overtaking maneuvers to test the vehicle's ability to decelerate and settle at a constant speed. Use-case 3 and 4 does not include a lane change and are not analyzed on their respective lane change characteristics. Instead the ego vehicle's deviation from the lane center is measured to examine the obstacle vehicle's influence on the ego vehicle.

The tuning is divided between the controller tuning and the environment tuning. The MPC controller is tuned with the weighting parameters in the cost function. While the environmental tuning is composed of altering the appearance of the road and obstacle APFs.

## 6.1 Controller Tuning

The tuning of the MPC is done through the weighting parameters. A higher weight gives a higher value which in turn gives a higher cost. When tuning the weights, the tuned variables size must be considered, this is because the variables will produce values of different magnitudes. The controller prioritizes cost terms with higher weights as these will affect the total cost the most. There does not exist a standardized tuning process for MPCs as there does with PID controllers. The MPC controller is tuned through trial and error from changing one variable at the time and observing the result. The weighting parameters and their final tuning weight for the different cost function terms are specified in Table 6.1.

Table 6.1: Weighting parameters

| Name | Symbol | Weight |
|---|---|---|
| Lateral position | $\lambda_{Y_h}$ | 1 |
| Yaw angle | $\lambda_{\psi}$ | 35 |
| Absolute velocity | $\lambda_{v}$ | 10 |
| Slip angle | $\lambda_{\beta}$ | 1 |
| Acceleration | $\lambda_{a}$ | 2 |
| Side slip angular velocity | $\lambda_{\Delta\beta}$ | 5000 |
| Jerk | $\lambda_{\Delta a}$ | 20 |
| Road APF | $\lambda_{road}$ | 20 |
| Obstacle APF | $\lambda_{obs}$ | 20 |

The chosen states to be tuned are $[Y_h, \psi, v]$. The lateral position is the desired lane center, a higher weight gives a larger urge to minimize the lateral error. The yaw angle is included to keep the vehicle from gaining too high yaw angles in relation to its velocity resulting in an overshoot of the lateral position. The overshoots come from a combination of strict slip angle constraints and a high weight on the slip angle rate. The velocity weight is tuned to follow the velocity reference close enough to provide short braking distances and short overtaking times while keeping the jerk to a minimum.

The control inputs $[\beta, a]$ are tuned to keep the slip angle and the acceleration to a minimum. A high weight brings down the respective peak values. The chosen weight for the controller is low and does not significantly affect the peak values while smoothing out the control inputs. The input increment was heavily tuned to smooth out the slip angle and the acceleration. Putting a higher weight on the input increment cost term gives smoother results with the downside of having a slower response.

Increasing the weight on the APFs in the cost function makes the controller favor the local minimums of the APFs over the lane center reference. If the cost is too high the vehicle could get stuck in a APF local minimum and if it is to low the vehicle won't be affected by the APFs. There needs to exist a balance between the lane center reference and the APF weights to encourage the vehicle to overtake while still being affected by the road and its surrounding obstacles.

The results are analyzed based on the safety and comfort of the vehicle's trajectory. The main objective is to have collision free maneuvering while being able to overtake roadside parked cars. To evaluate the comfort, the lateral and longitudinal acceleration and jerk are analyzed. Furthermore, the lane change characteristics are evaluated, this is done with the method described in Figure 6.1.

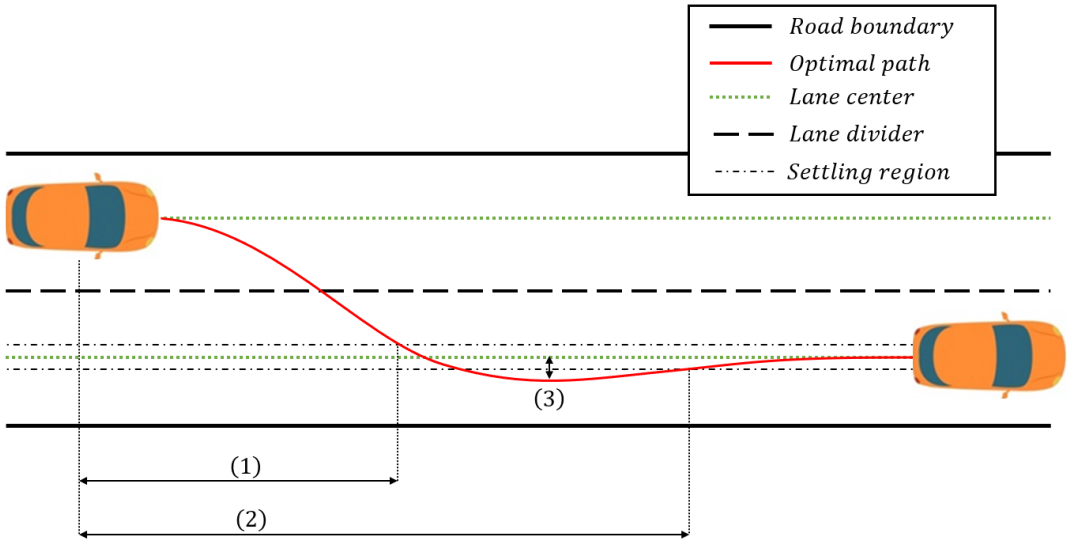


Figure 6.1: Lane change characteristics: 1) Rise time, 2) Settling time, 3) Overshoot

Where the rise time is the time it takes to rise from 5% to 95% of the steady-state response. The steady-state response is the conditions the system reaches a long time after an excitation. The settling time is the time it takes for the lateral error to settle within 5% of the final value and the overshoot is the percentage the controller goes above the final value. When calculating the settling time, the initial lane center is considered the origin. All lane change characteristics are measured from the instance the lane change maneuver is initiated. The lane change duration ends when the ego vehicle has reached its settling time.

## 6.2 Artificial potential field tuning

The APF for the road and obstacles influences the controller's behavior and can be modified to tune the controller's results. The APF weights in the MPC changes the importance of the APF compared to the other cost function terms. The road APF is tuned with the coefficients defined in Table 6.2.

Table 6.2: Road APF coefficients

| Name | Symbol | Weight |
|---|---|---|
| Right lane depth | $A_r$ | 0.2 |
| Left lane depth | $A_l$ | 0.3 |
| Width of dip | $b_r$ | 1 |

By changing the appearance of the APFs it changes how the vehicle interacts with them. The road APF can be modified to have deeper lane center dips, a higher lane divider hump or a larger difference in lane center heights. If the lane divider hump is to aggressive the approximated APF becomes excessively negative within the road boundaries which is undesired when tuning. Therefore, the road APF lane depths are tuned so that the lane divider hump does not create function values below zero within the boundaries of the road. The tuned road APF can be seen in Figure 6.2.

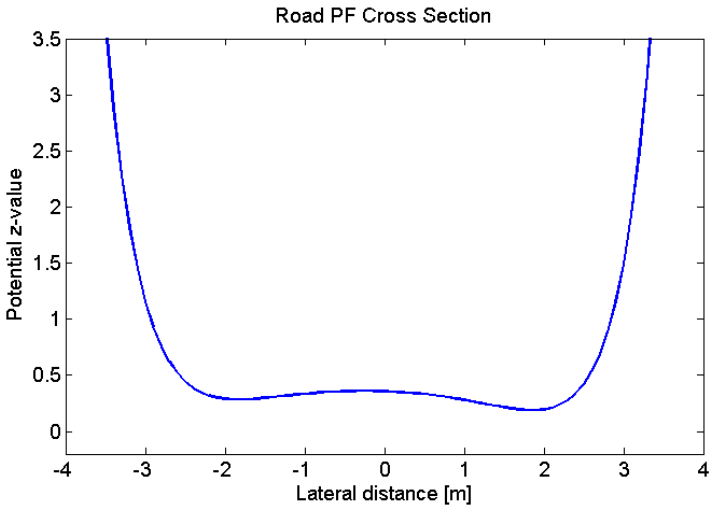


Figure 6.2: Tuned road APF

The road lane depths are asymmetrical to give a higher cost driving on the opposite side of the road, while giving a lower cost to drive on the correct side of the road. A lower value for the width of the dip created a wider channel for the lane, reduced the slope of the road boundaries and decreased the width of the lane divider hump. The width of the dip was kept at 1, as this proved to be a good middle ground.

Furthermore, the obstacle APFs peak value can be changed to have a greater emphasis on the obstacles compared to the road, this can however be tuned in the MPC and is therefore kept unchanged. By changing the scaling of the APF, the size in the lateral and longitudinal direction can be adjusted. The scaling factor effect can be seen in the contour map in the Figure 6.3.

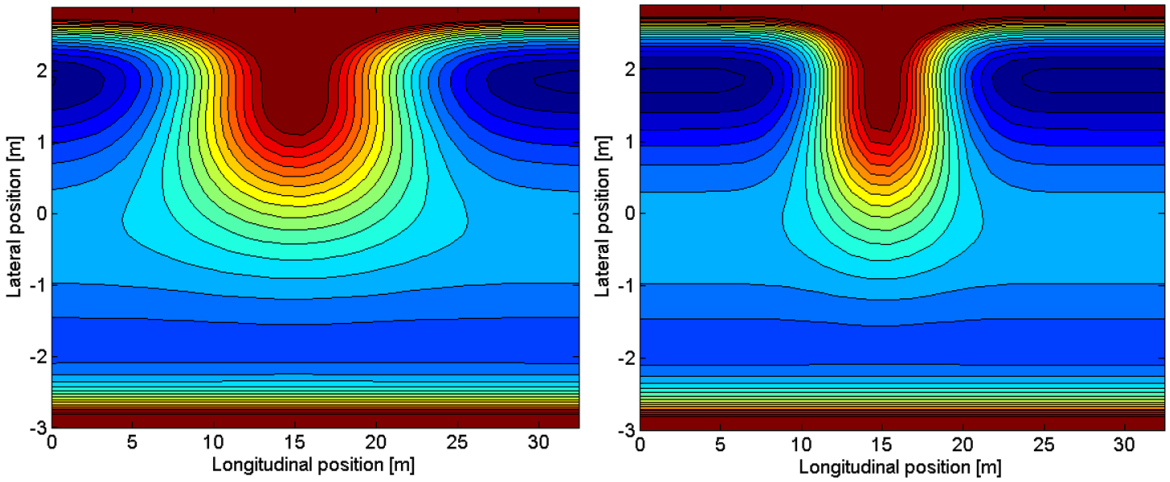


Figure 6.3: Obstacle APF tuning

Where the colors are a heatmap of high cost regions, where red is high cost and blue is low cost. An elongated obstacle APF will increase the ego vehicle's safety distance to parked cars, it also gives a higher cost when being in a moving obstacles trajectory. The obstacle APF can also be moved in the obstacle vehicle's longitudinal direction to skew the cost of being in front or behind the obstacle vehicle.

# 7 Results

The ego vehicle's different maneuvers and optimal paths for the use-cases can be seen in Figure 7.25. The overshoot when overtaking is measured from the right and left lane respectively. Use-case 3 and 4 does not have an overtaking maneuver. Instead the overshoot represents the deviation from the lane center to the left or to the right in the ego vehicle's direction of travel. However, the lane change characteristics are not analyzed for those use-cases. The results from all the use-cases can be seen in Table 7.1.

Table 7.1: Result, values for all use-cases

| Name | Use-case 1 | Use-case 2 | Use-case 3 | Use-case 4 | Unit |
|---|---|---|---|---|---|
| Use-case duration | 18 | 27 | 10 | 15 | $[s]$ |
| Average calculation time | 0.0146 | 0.0150 | 0.0119 | 0.0155 | $[s]$ |
| Max. calculation time | 0.0549 | 0.0541 | 0.0338 | 0.0248 | $[s]$ |
| Min. calculation time | 0.0065 | 0.0060 | 0.0055 | 0.0062 | $[s]$ |
| Right lane change rise time | 3.6 | 5.4 | - | - | $[s]$ |
| Right lane change settling time | 3.6 | 5.4 | - | - | $[s]$ |
| Left lane change rise time | 3.0 | 2.9 | - | - | $[s]$ |
| Left lane change settling time | 3.0 | 2.9 | - | - | $[s]$ |
| Lane change duration | 8.4 | 17.2 | - | - | $[s]$ |
| Lateral overshoot left lane | 0.0839 | 0.0857 | 0 | 0.0165 | $[m]$ |
| Lateral overshoot right lane | -0.0551 | -0.028 | 0 | 0 | $[m]$ |
| Max. longitudinal acceleration | 1.8671 | 1.8671 | 1.8643 | 1.8673 | $\left[\frac{m}{s^2}\right]$ |
| Min. longitudinal acceleration | -1.4354 | -2.6926 | -0.03 | -2.7906 | $\left[\frac{m}{s^2}\right]$ |
| Max. lateral acceleration | 0.1544 | 0.1663 | 0.0016 | 0.0026 | $\left[\frac{m}{s^2}\right]$ |
| Min. lateral acceleration | -0.1725 | -0.1720 | -0.0033 | -0.0031 | $\left[\frac{m}{s^2}\right]$ |

All simulations had low average calculation times with maximum calculation times that were faster than the sampling time. Where the calculation times are measured during each sampling period. The lane change maneuver in use-case 1 and 2 were successful and was executed with a smooth slip angle input as can be seen in Figure 7.1 and Figure 7.7. The maneuvers from all the use-cases can be seen in Figure 7.25.

## 7.1 Use-case 1

This use-case has the ego vehicle drive along a straight road with roadside parked cars in its lane without any oncoming traffic. This use-case was tested with different starting positions for the obstacle vehicle to test the ego vehicle's decision-making. In one of the simulations the ego vehicle came to a standstill and waited for the traffic to pass. The starting position of the obstacle vehicle was then changed until the ego vehicle decided the gap towards the oncoming vehicle was large enough to overtake the parked cars without stopping. The resulting maneuver can be seen in Figure 7.25. The data from that simulation is not included in the results as it is identical to use-case 1 that displayed a simple overtaking maneuver. The results from use-case 1 can be seen in Table 7.2.

Table 7.2: Results of use-case 1

| Name | Value | Unit |
|---|---|---|
| Average calculation time | 0.0146 | $[s]$ |
| Max. calculation time | 0.0549 | $[s]$ |
| Min. calculation time | 0.0065 | $[s]$ |
| Right lane change rise time | 3.6 | $[s]$ |
| Right lane change settling time | 3.6 | $[s]$ |
| Left lane change rise time | 3.0 | $[s]$ |
| Left lane change settling time | 3.0 | $[s]$ |
| Lane change duration | 8.4 | $[s]$ |
| Lateral overshoot left lane | 0.0839 | $[m]$ |
| Lateral overshoot right lane | -0.0551 | $[m]$ |
| Max. longitudinal acceleration | 1.8671 | $\left[\frac{m}{s^2}\right]$ |
| Min. longitudinal acceleration | -1.4354 | $\left[\frac{m}{s^2}\right]$ |
| Max. lateral acceleration | 0.1544 | $\left[\frac{m}{s^2}\right]$ |
| Min. lateral acceleration | -0.1725 | $\left[\frac{m}{s^2}\right]$ |

The lane change was efficient and completed in a reasonable timeframe. The lateral overshoot for the left and right lane changes were respectively small. The vehicle's path could easily be made overdamped with no overshoots at the expense of slower lane changes. During the tuning process a compromise was made between a smooth and quick overtaking maneuver. The control inputs and corresponding absolute velocity and yaw angle of the ego vehicle can be seen in Figure 7.1.

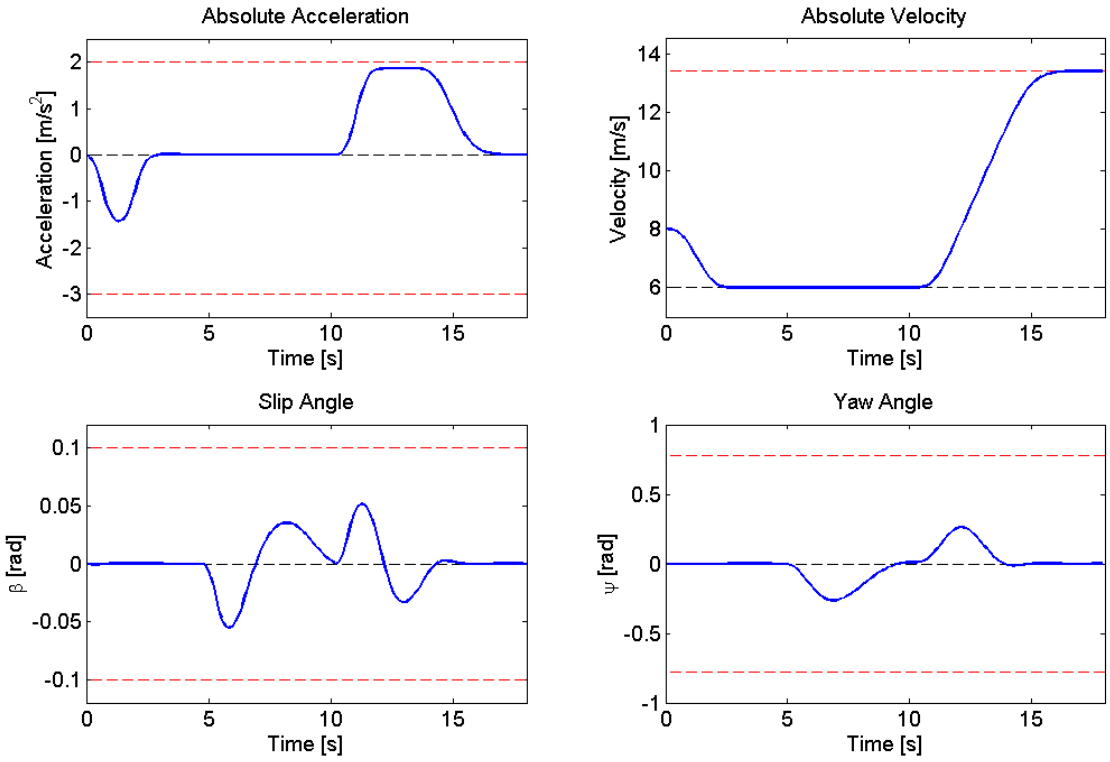


Figure 7.1: Use-case 1, control input and vehicle states

The red dashed lines in the plot above represents the constraints for each respective variable. The black dashed lines in the absolute velocity plot represents the vehicle's slow speed. The results are smooth and satisfactory while providing fast response. The acceleration almost reaches its constraint, by putting a higher weight on the acceleration the maximum and minimum acceleration could be brought closer to zero. However, this was not wanted as the constraints are based on a typical driving behavior.

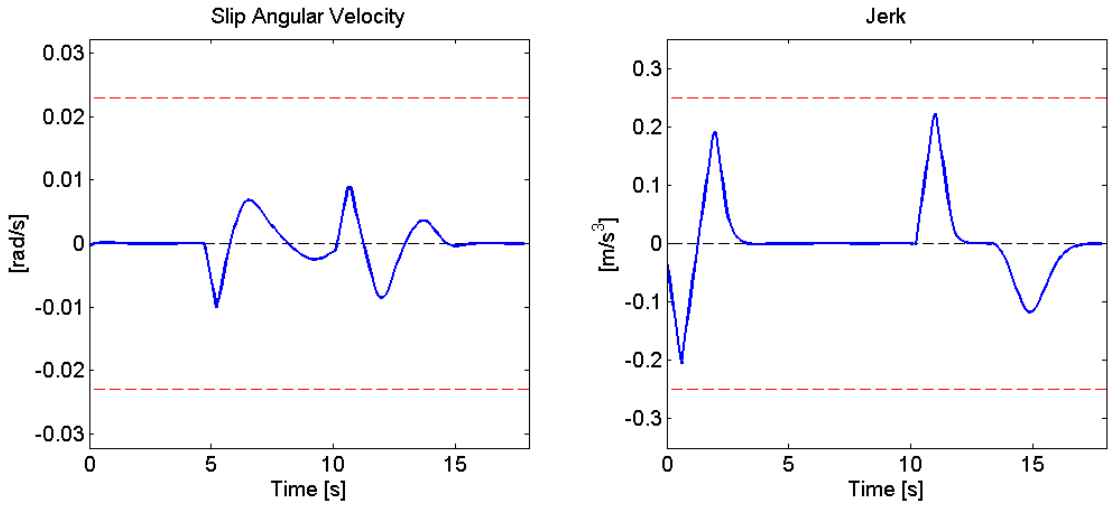


Figure 7.2: Use-case 1, control input change

The slip angular velocity and jerk are within the constraints and are smooth during most of the simulation. Some fast changes can be seen in both the slip angular velocity and the jerk, this is also reflected by the slip angular acceleration and the snap, presented in Figure 7.3. This behavior could be tuned out, but it would in turn give a slower response for the slip angle and the acceleration which introduced overshoots in the vehicle's path and velocity. The slip angular acceleration, snap and their constraints can be seen in Figure 7.3.

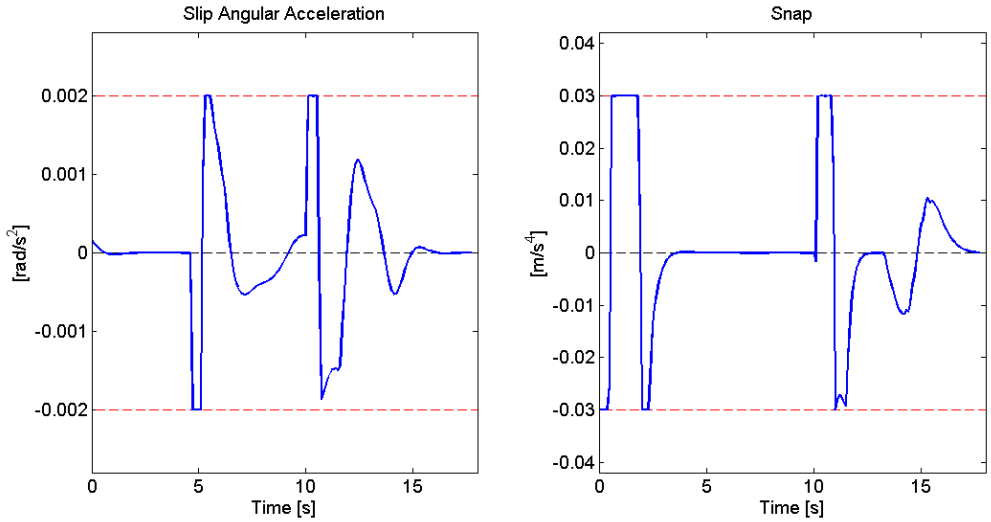


Figure 7.3: Use-case 1, control input noise

The slip angular acceleration and the snap reach their respective constraints. One of the reasons for this is that neither of the variables are included in the cost function and therefore larger values are not penalized. These constraints are included to reduce the change in slip angular velocity and the jerk. This in turn reduces the hesitation and noise in the control input signal. To evaluate comfort the lateral velocity, acceleration and jerk is analyzed. The lateral results in all the use-cases are from the ego vehicle fixed coordinate system. The results from use-case 1 is presented in Figure 7.4.

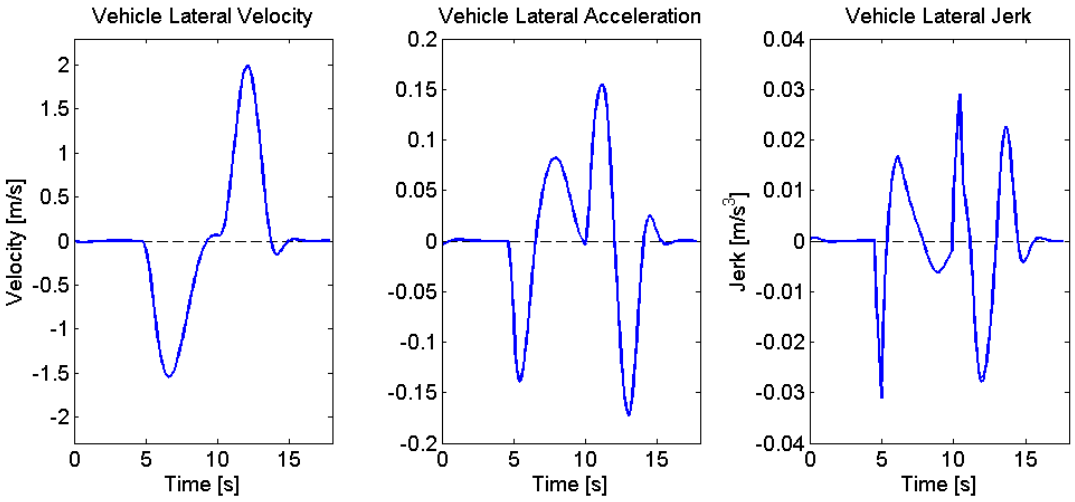


Figure 7.4: Use-case 1, lateral result

The lateral acceleration does not exceed the comfort level of 0.25 m/s². It is smooth except for the two spikes at roughly 5 s and 10 s. These spikes most likely come from the change in lane center reference, as can be seen in Figure 7.5. The decision-making is made through the selection of the preferred lane center and velocity profile to use as references for the MPC. The velocity profile is created based on a desired velocity that is either the speed limit, the low speed setting or zero velocity. For zero velocity the ego vehicle should brake until it reaches a complete stop. The decision-making result from use-case 1 can be seen in Figure 7.5.

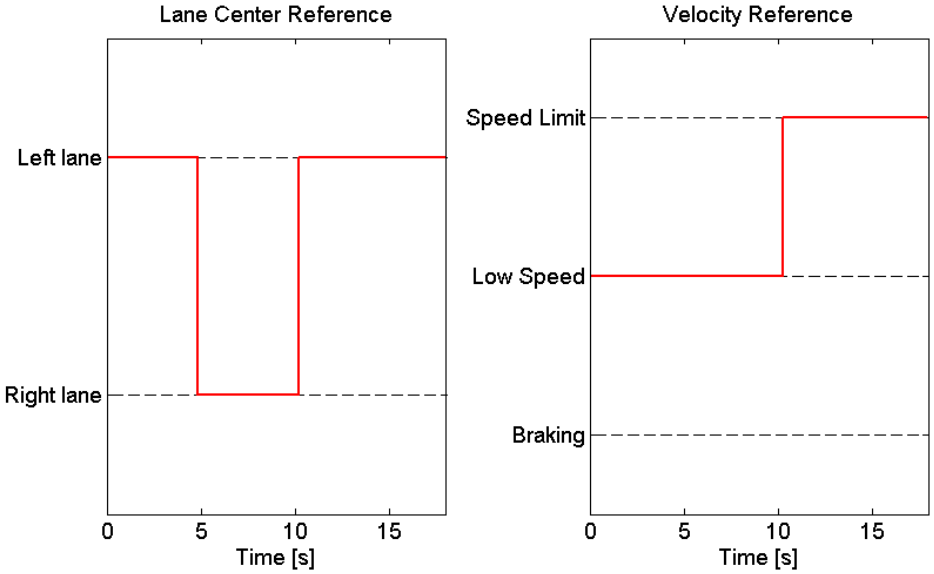


Figure 7.5: Use-case 1, decision-making

The decision-making is decisive without any hesitation. The speed limiter selection only limits the ego vehicle's velocity and does not affect the desired velocity. High speed is the roads speed limit while low speed is the vehicle's desired velocity when approaching and overtaking parked cars. When braking the desired velocity is zero while the set speed limit might be higher. The MPC loop calculation time for each sample time is displayed in Figure 7.6.

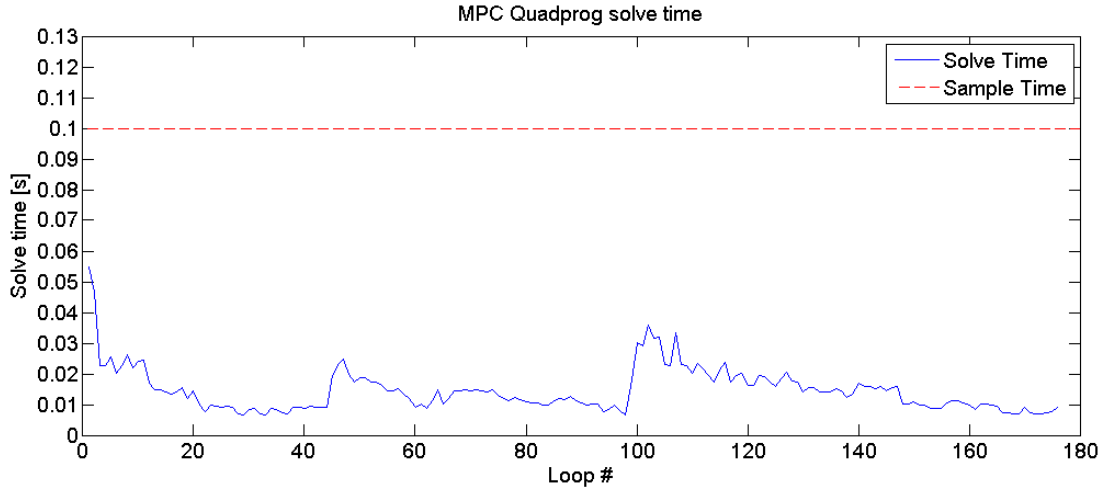


Figure 7.6: Use-case 1, MPC solve time

The first 3-5 loops from all simulations had much higher calculation times. Because this behavior could be seen in all simulations the first 5 calculations times were removed from the results. The spike at roughly loop number 100 is consistent with the change of the lane center and the desired velocity.

## 7.2 Use-case 2

This use-case has the ego vehicle drive along a straight road with roadside parked cars in its lane and an oncoming vehicle in the opposite lane. The results from use-case 2 can be seen in Table 7.3.

Table 7.3: Results of use-case 2

| Name | Value | Unit |
|---|---|---|
| Average calculation time | 0.0150 | $[s]$ |
| Max. calculation time | 0.0541 | $[s]$ |
| Min. calculation time | 0.0060 | $[s]$ |
| Right lane change rise time | 5.4 | $[s]$ |
| Right lane change settling time | 5.4 | $[s]$ |
| Left lane change rise time | 2.9 | $[s]$ |
| Left lane change settling time | 2.9 | $[s]$ |
| Lane change duration | 17.2 | $[s]$ |
| Lateral overshoot left lane | 0.0857 | $[m]$ |
| Lateral overshoot right lane | -0.028 | $[m]$ |
| Max. longitudinal acceleration | 1.8671 | $\left[\frac{m}{s^2}\right]$ |
| Min. longitudinal acceleration | -2.6926 | $\left[\frac{m}{s^2}\right]$ |
| Max. lateral acceleration | 0.1663 | $\left[\frac{m}{s^2}\right]$ |
| Min. lateral acceleration | -0.1720 | $\left[\frac{m}{s^2}\right]$ |

The results in Table 7.3 show similar outcomes as in Table 7.2 in use-case 1 . This could be explained by the fact that the maneuver in this use-case is very similar to use-case 1 except that the vehicle must stop for oncoming traffic. The control inputs and corresponding absolute velocity and yaw angle of the ego vehicle can be seen in Figure 7.7.

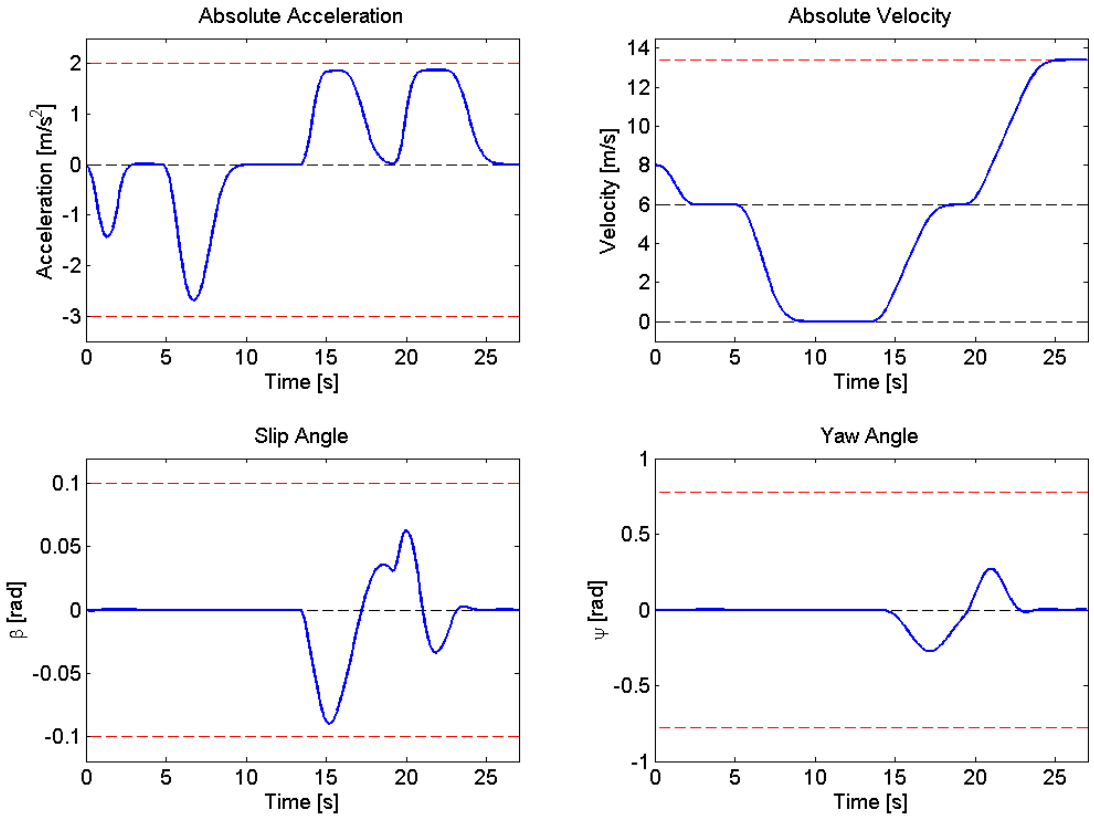


Figure 7.7: Use-case 2, control input and vehicle states

Because of the 'stop and go' scenario the result is a deceleration which results in the vehicle coming to a complete stop. After the stop the vehicle first accelerates up to the low-speed speed-limit before changing the limit to the roads speed limit when the decision-making determines to go back to the left lane. When the simulations were carried out with lower acceleration constraints the vehicle did not reach the low speed limit during the overtaking maneuver. This resulted in a more continuous acceleration which could be perceived as more comfortable. However, it was determined that it was better to reduce the time spent in the opposing lane rather than having a continuous acceleration region. The slip angular velocity, the jerk and the corresponding constraints can be seen in Figure 7.8.

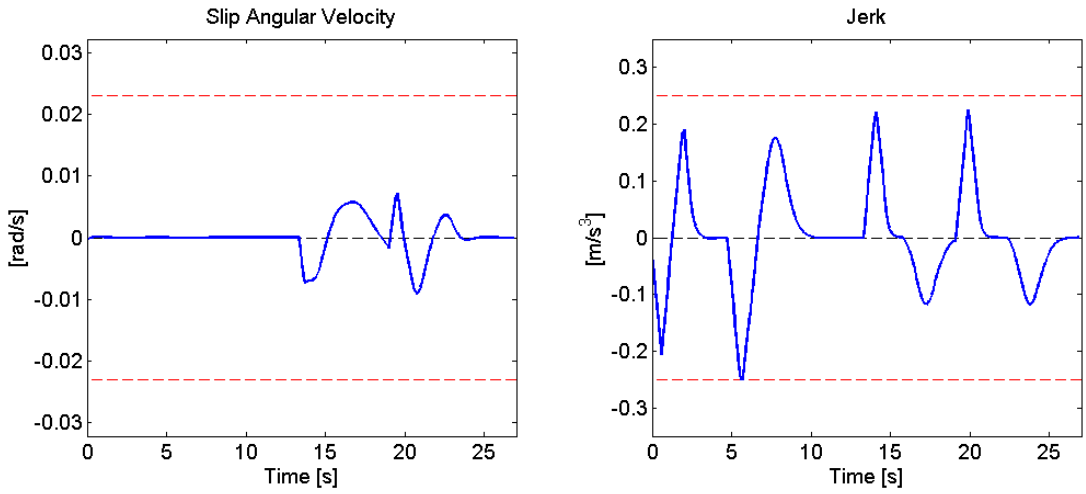


Figure 7.8: Use-case 2, control input change

Because of the 'stop and go' scenario the jerk has a few sharp changes during the simulations. This is because there are more changes for the ego vehicle's reference velocity. The jerk reaches its minimum constraint during the deceleration part of the simulation. The slip angular acceleration, snap and their constraints can be seen in Figure 7.9.

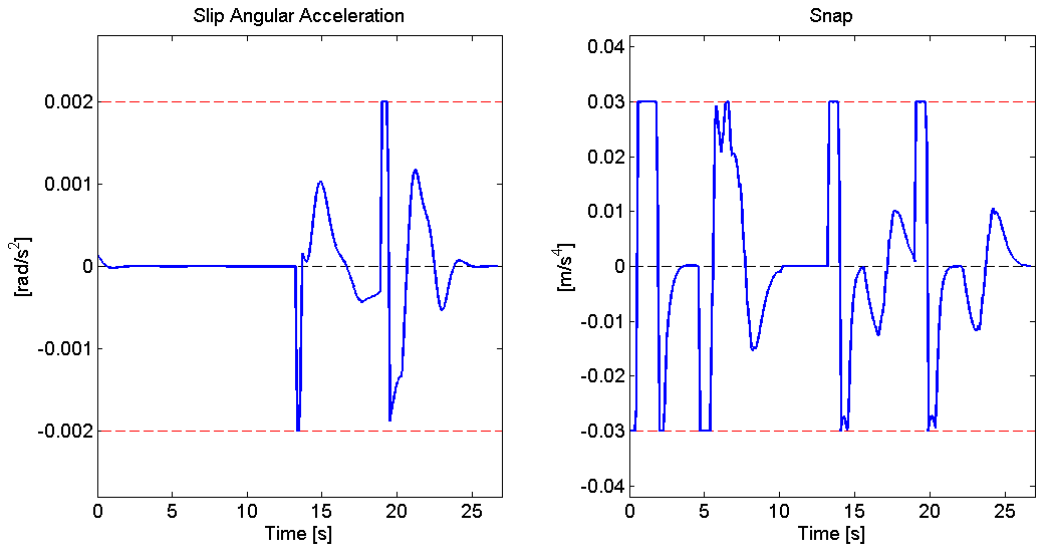


Figure 7.9: Use-case 2, control input noise

Since the lane change is initiated by a standstill, this use-case differs from the previous one. This can be seen while comparing the slip angular acceleration of the two use-cases. Both have reached the low-speed speed-limit during the overtaking and have similar lateral positions. This results in an almost identical result for the last half of the overtaking maneuver. The first half of the simulations does not only have a differentiation in velocity, but the initial longitudinal position is different, as can be seen in Figure 7.25. When the lane change maneuver is started closer to parked cars the obstacle APF will have a larger influence on the controller. The lateral results from use-case 2 is presented in Figure 7.10.

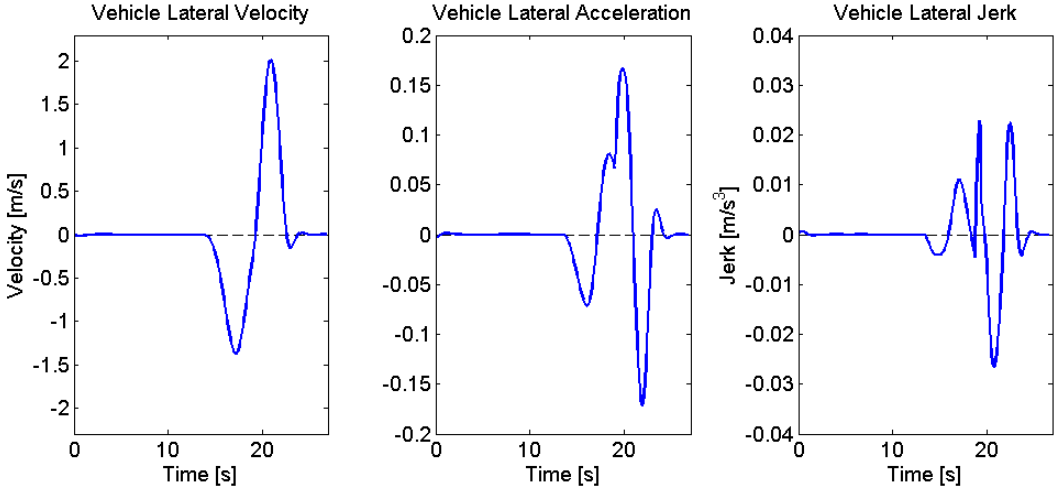


Figure 7.10: Use-case 2, lateral result

The lateral velocity, acceleration and jerk all show smooth results. The lateral acceleration decreases slightly at roughly 18 s before rising again. This can also be seen in the slip angle in Figure 7.7 and is consistent with the decision to change the lane center reference from the right to the left lane center. Because the slip angle has not reached zero from handling the lane center lateral overshoot before the decision is made a spike shows up in the lateral jerk.

The decision-making in use-case 2 can be seen in Figure 7.11.

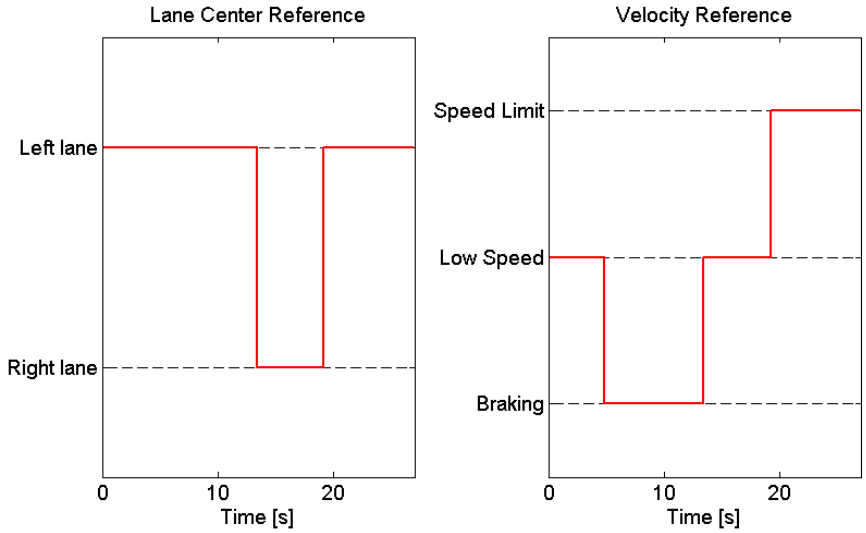


Figure 7.11: Use-case 2, decision-making

The decision-making is decisive and predicable to both the passengers of the ego vehicle but also for the driver of the oncoming vehicle. It does not hesitate between stopping and overtaking. In Figure 7.25 the ego vehicle waits until the traffic has passed completely before it initiates the overtaking maneuver. The MPC loop calculation time for each sample time is displayed in Figure 7.12.

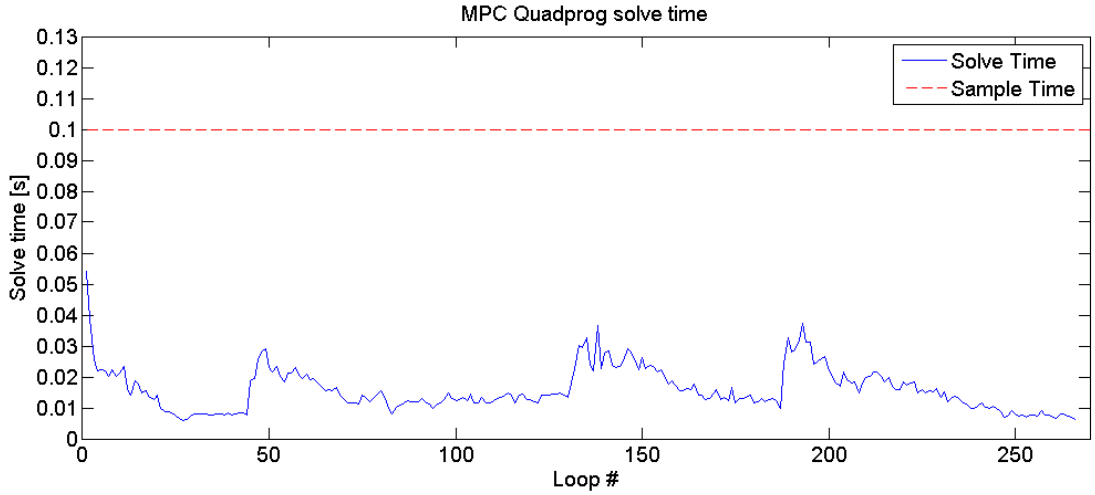


Figure 7.12: Use-case 2, MPC solve time

The calculation times are still fast with maximum calculation times, faster than the specified sampling time. This use-case uses three obstacle vehicles, two for the parked cars and one for the oncoming traffic. Despite having the greatest number of obstacles in its environment it has very similar calculations times as the rest of the use-cases. The increase in calculation times are consistent with the decision-making instances.

## 7.3 Use-case 3

This use-case has the ego vehicle drive along a straight road without any roadside parked cars and an oncoming vehicle in the opposite lane. The results from use-case 3 can be seen in Table 7.4.

Table 7.4: Results of use-case 3

| Name | Value | Unit |
|---|---|---|
| Average calculation time | 0.0119 | $[s]$ |
| Max. calculation time | 0.0338 | $[s]$ |
| Min. calculation time | 0.0055 | $[s]$ |
| Right lane change rise time | - | $[s]$ |
| Right lane change settling time | - | $[s]$ |
| Left lane change rise time | - | $[s]$ |
| Left lane change settling time | - | $[s]$ |
| Lane change duration | - | $[s]$ |
| Lateral overshoot left lane | 0 | $[m]$ |
| Lateral overshoot right lane | 3.7326 | $[m]$ |
| Max. longitudinal acceleration | 1.8643 | $\left[\frac{m}{s^2}\right]$ |
| Min. longitudinal acceleration | -0.03 | $\left[\frac{m}{s^2}\right]$ |
| Max. lateral acceleration | 0.0016 | $\left[\frac{m}{s^2}\right]$ |
| Min. lateral acceleration | -0.0033 | $\left[\frac{m}{s^2}\right]$ |

This use-case is simulated while taking the oncoming traffic into account. The ego vehicle did not deviate from the lane center, and thus is not affected by the opposing vehicles. The control inputs and corresponding absolute velocity and yaw angle of the ego vehicle can be seen in Figure 7.13.

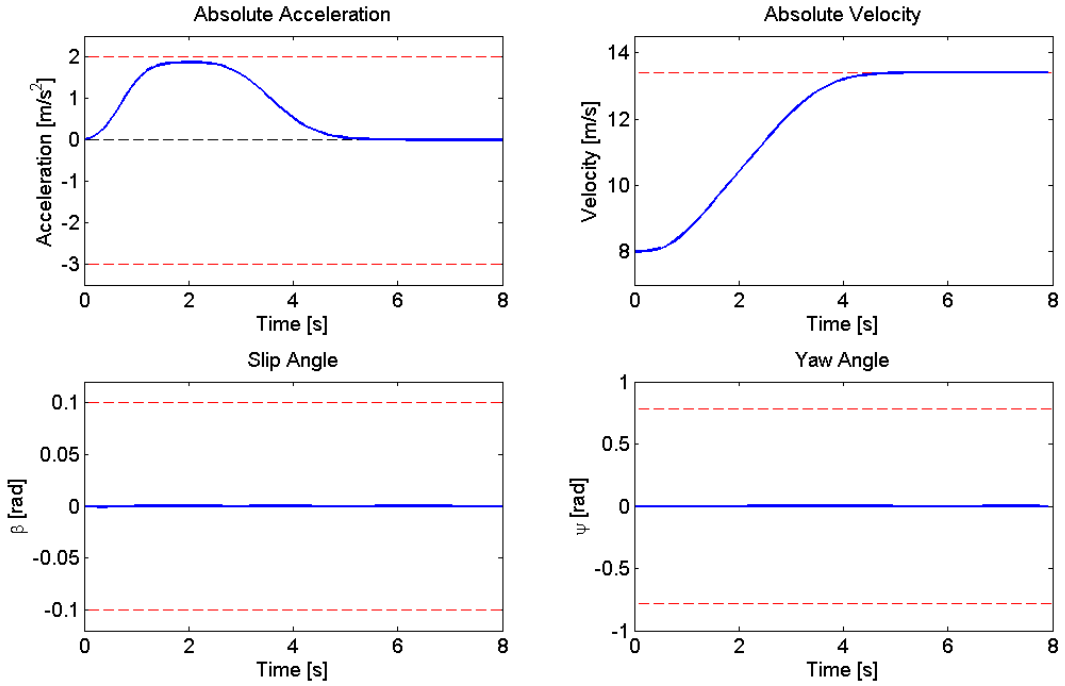


Figure 7.13: Use-case 3, control input and vehicle states

The ego vehicle is free to accelerate up to the speed limit of the road. The acceleration is smooth and quick, and the ego vehicle does not overshoot the speed limit. The slip angular velocity, the jerk and the corresponding constraints can be seen in Figure 7.14.

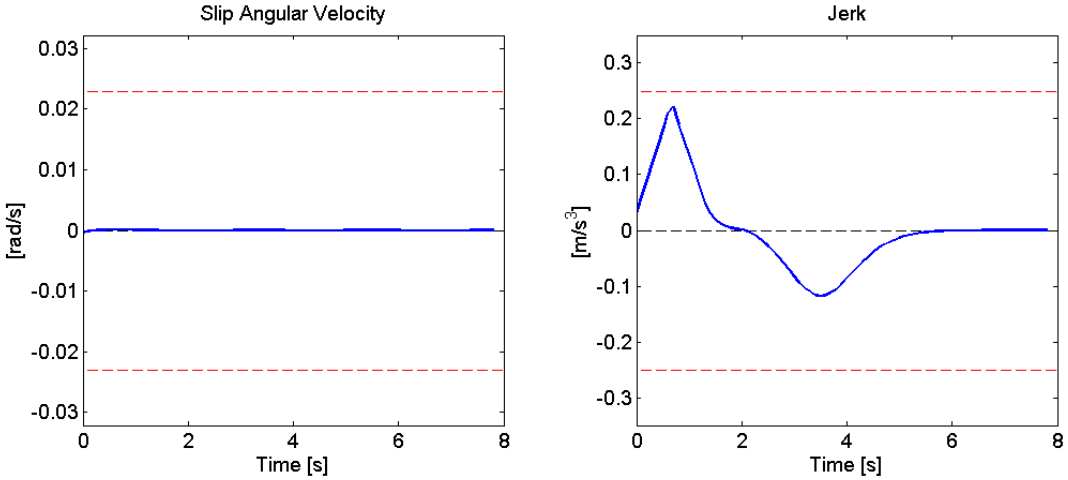


Figure 7.14: Use-case 3, control input change

The slip angular velocity is close to zero while the jerk shows similar tendencies as previous use-cases during acceleration scenarios. The slip angular acceleration, snap and their constraints can be seen in Figure 7.15.

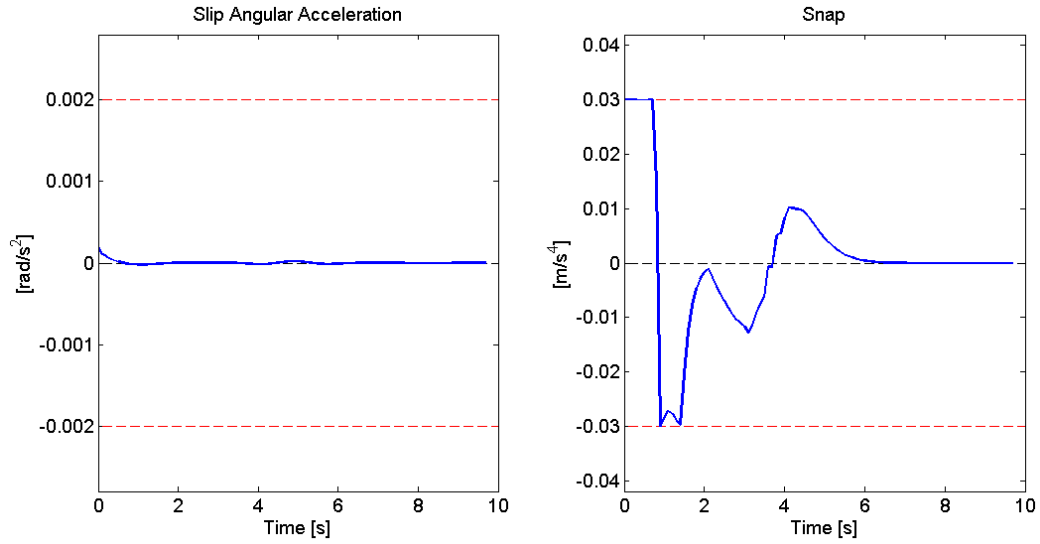


Figure 7.15: Use-case 3, control input noise

The slip angular acceleration is zero which shows that there is no induced noise in the control from the oncoming traffic. The snap has some stutter between 3 s and 4 s which happens when the ego vehicle reduces its acceleration towards zero. The lateral results from use-case 3 is presented in Figure 7.16.

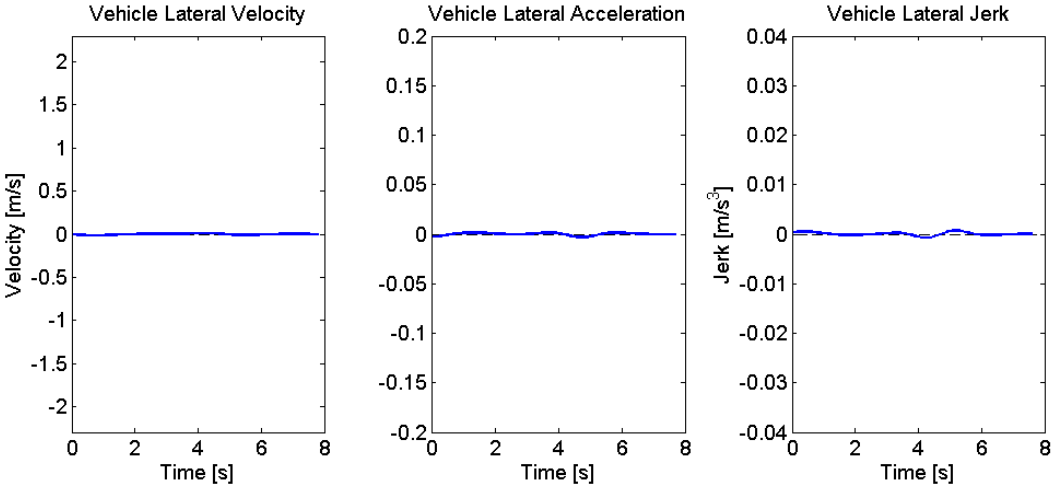


Figure 7.16: Use-case 3, lateral result

Despite the small slip angle, slip angular velocity and slip angular acceleration the lateral acceleration and lateral jerk show some small oscillations. The decision-making in use-case 2 can be seen in Figure 7.17.

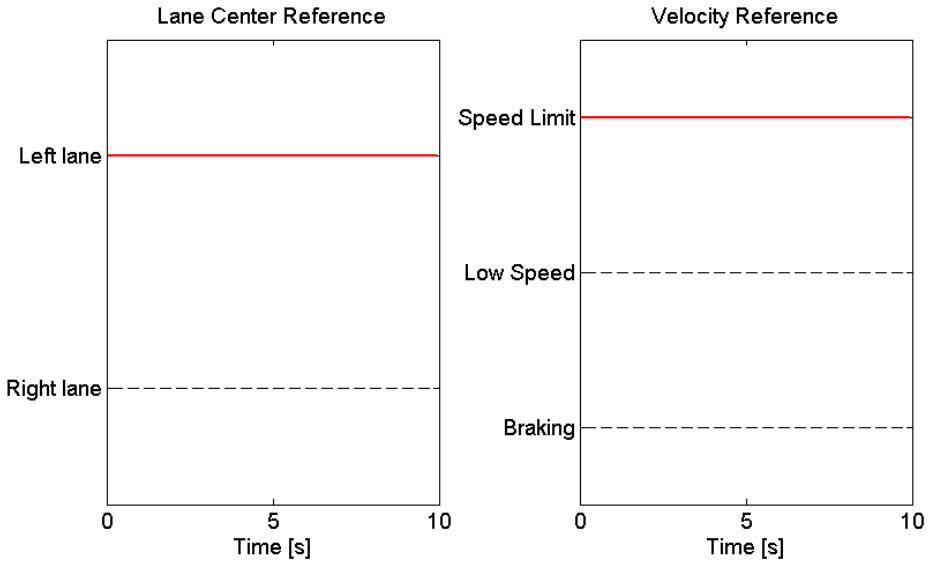


Figure 7.17: Use-case 3, decision-making

Because the ego vehicle is not affected by the oncoming traffic there is no decision-making during this simulation. The controller therefore uses the same references for the whole simulation. The MPC loop calculation time for each sample time is displayed in Figure 7.18.

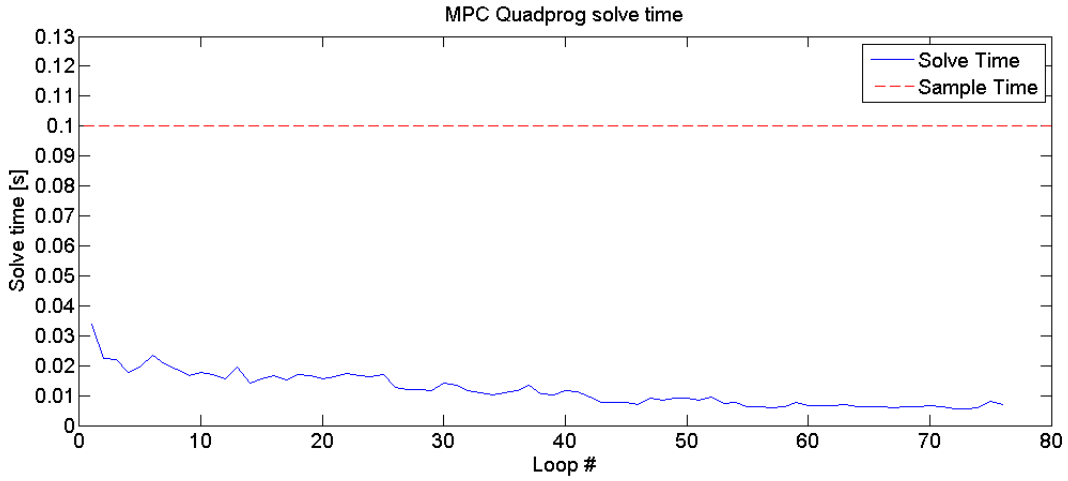


Figure 7.18: Use-case 3, MPC solve time

This use-case has the fastest calculation times of all the use-cases. Because there is no decision-making during the simulation there are no apparent increases in calculation time during the simulation.

## 7.4 Use-case 4

This use-case has the ego vehicle drive along a straight road without any roadside parked cars and an oncoming vehicle in the opposite lane. The results from use-case 4 can be seen in Table 7.5.

Table 7.5: Results of use-case 4

| Name | Value | Unit |
|---|---|---|
| Average calculation time | 0.0155 | $[s]$ |
| Max. calculation time | 0.0248 | $[s]$ |
| Min. calculation time | 0.0062 | $[s]$ |
| Right lane change rise time | - | $[s]$ |
| Right lane change settling time | - | $[s]$ |
| Left lane change rise time | - | $[s]$ |
| Left lane change settling time | - | $[s]$ |
| Lane change duration | - | $[s]$ |
| Lateral overshoot left lane | 0.0165 | $[m]$ |
| Lateral overshoot right lane | 3.7326 | $[m]$ |
| Max. longitudinal acceleration | 1.8673 | $\left[\frac{m}{s^2}\right]$ |
| Min. longitudinal acceleration | -2.7906 | $\left[\frac{m}{s^2}\right]$ |
| Max. lateral acceleration | 0.0026 | $\left[\frac{m}{s^2}\right]$ |
| Min. lateral acceleration | -0.0031 | $\left[\frac{m}{s^2}\right]$ |

This use-case differs from the other use-cases in the way that the ego vehicle must react to an unusual situation. Even though the oncoming vehicle's movement is predicted, the controller's decision-making does not include this scenario. Because of this it is of interest to see the controller's behavior during this unstructured scenario. The average calculation time is fast, and the maximum calculation time is the quickest of all the simulations. The control inputs and corresponding absolute velocity and yaw angle of the ego vehicle can be seen in Figure 7.19. Even though the slip angle has a small change the lateral overshoot is insignificant.

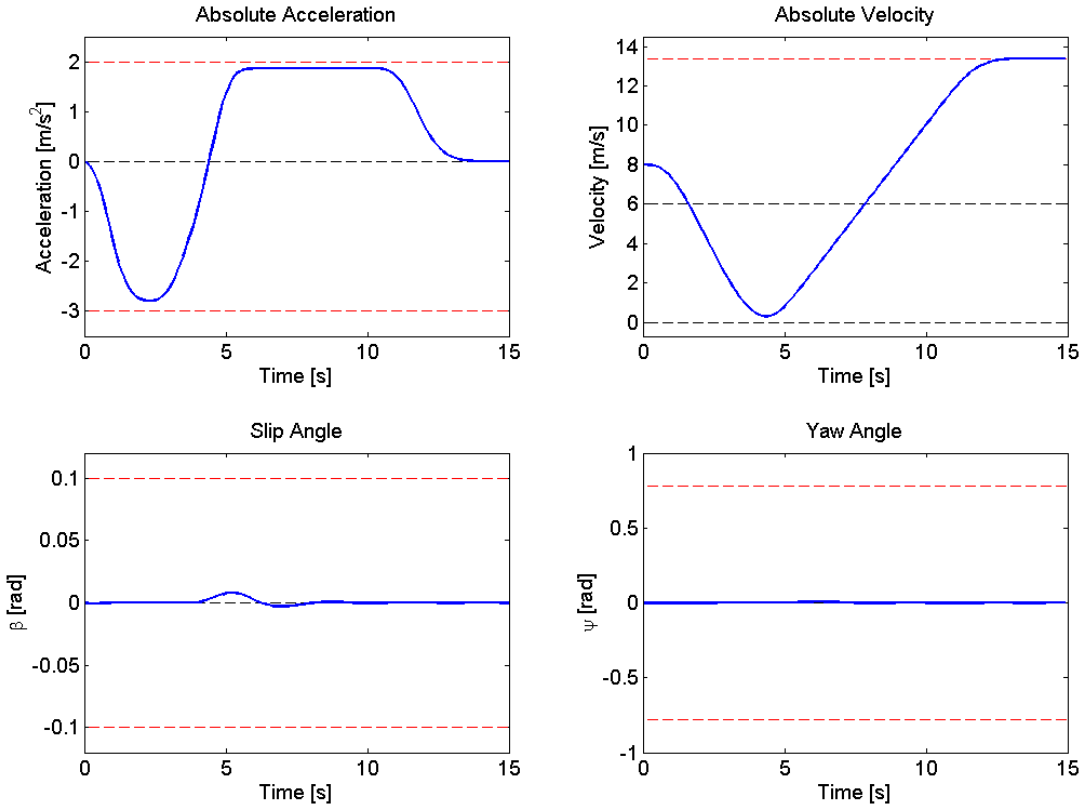


Figure 7.19: Use-case 4, control input and vehicle states

The slip angle sees a slight change at around 4 s. At the same time the controller decides there is no obstacle in the ego vehicle's lane and changes the reference from braking to accelerating. The slight change in the slip angle is influenced by the obstacle vehicle crossing back into the right lane. The path scouting of the decision making has a clear path in the left lane, but the ego vehicle is still affected by the obstacle vehicle's APF. The slip angular velocity, the jerk and the corresponding constraints can be seen in Figure 7.20.

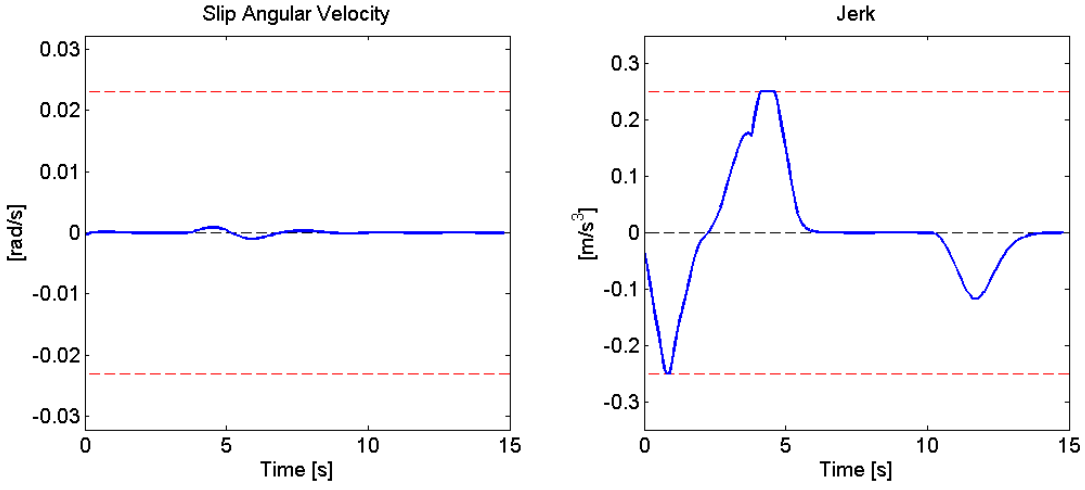


Figure 7.20: Use-case 4, control input change

The jerk has a small hesitation at around 4 s which again is due to the decision-making changing the velocity profile from braking to acceleration. The controller did not achieve a complete stop, at 4 s before the decision, the controller is reducing the jerk to obtain zero acceleration. When the decision changes the jerk is then increased to achieve a positive acceleration and to increase the ego vehicle's velocity. The slip angular acceleration, snap and their constraints can be seen in Figure 7.21.

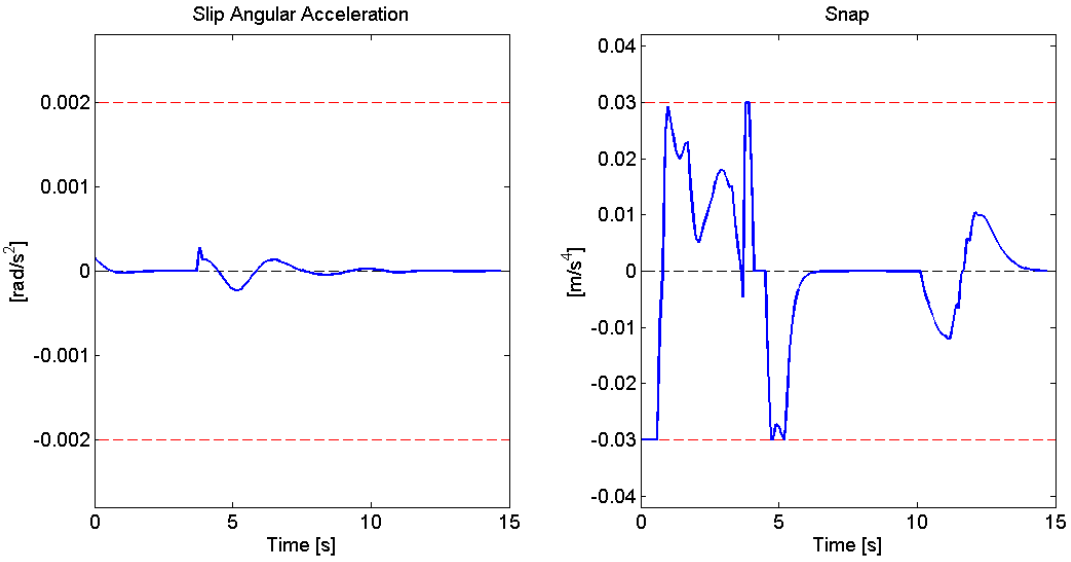


Figure 7.21: Use-case 4, control input noise

At roughly 12 s the snap displays similar stutter as Figure 7.15 in use-case 3. The lateral results from use-case 4 is presented in Figure 7.22.

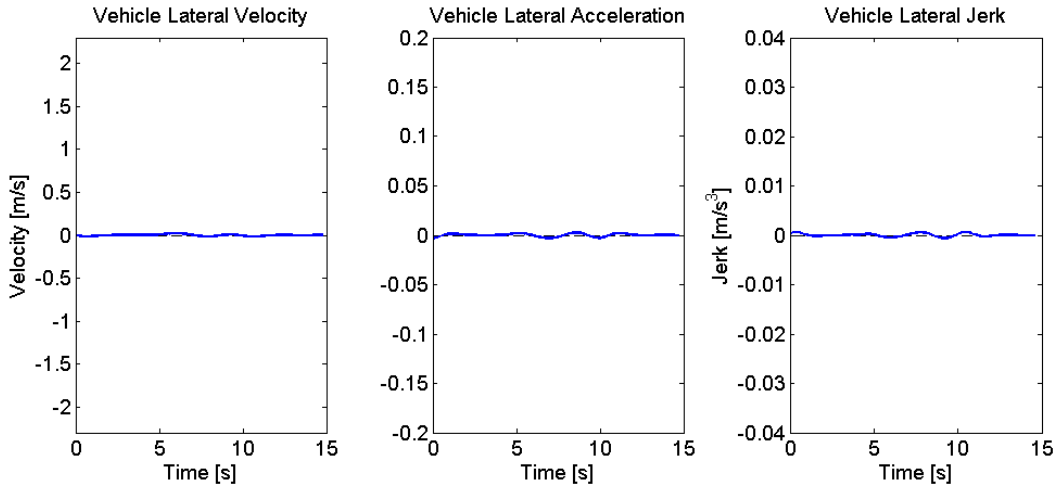


Figure 7.22: Use-case 4, lateral result

The lateral velocity, acceleration and jerk have similar oscillations as use-case 3, shown in Figure 7.16. These oscillations in both cases show up after the ego vehicle has passed the obstacles and is most likely due to small corrections to keep the vehicle in the center of the lane. The decision-making in use-case 2 can be seen in Figure 7.23.

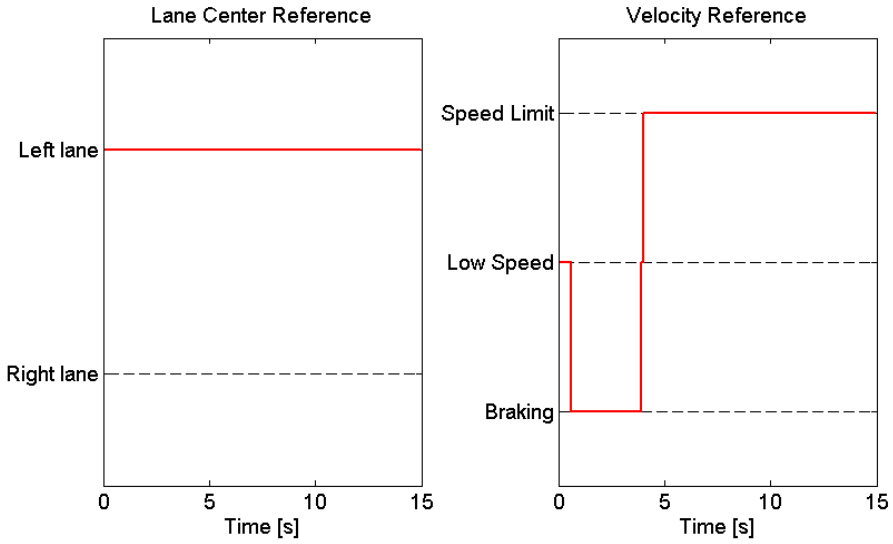


Figure 7.23: Use-case 4, decision-making

Because the ego vehicle predicts the oncoming vehicle's maneuver the decision to brake is made before the oncoming vehicle is in the ego vehicle's lane. The MPC loop calculation time for each sample time is displayed in Figure 7.24.

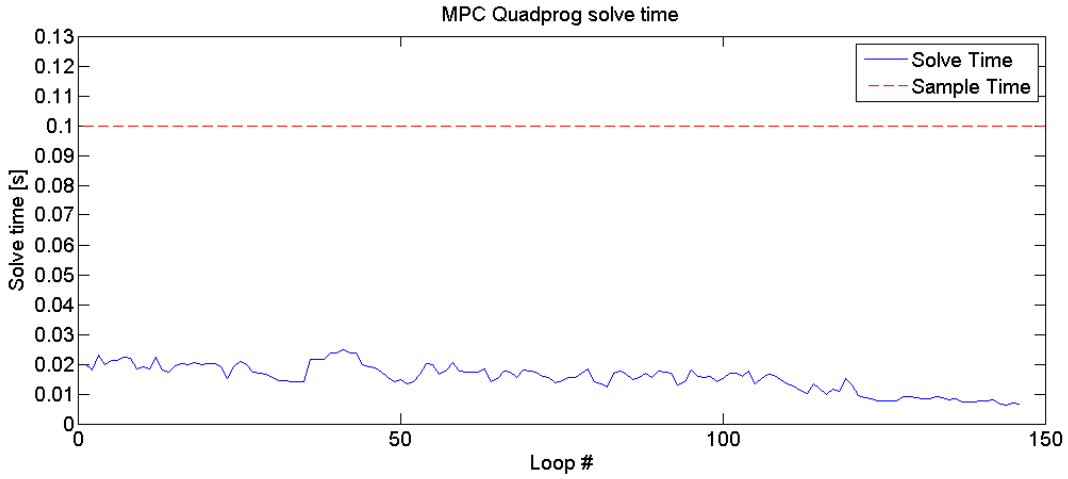


Figure 7.24: Use-case 4, MPC solve time

The calculation times are consistently low during the simulation. This indicates that the simulation time is predominantly influenced by the change of the velocity profile and lane center reference. This can be seen in Figure 7.6 and Figure 7.12 where the calculation times increase after the change of references and slowly decrease when the controller is settling.

The ego vehicle's path for each use-case can be seen in Figure 7.25, where the use-cases are displayed in descending order starting with use-case 1. The blue line is the ego vehicle's path during the simulation. The red boxes represents the parked cars and the oncoming traffic. The unmarked cars are the parked vehicles while the marked car is the moving vehicle. The numbering for the vehicle's positions is at an interval of 2.5 s.

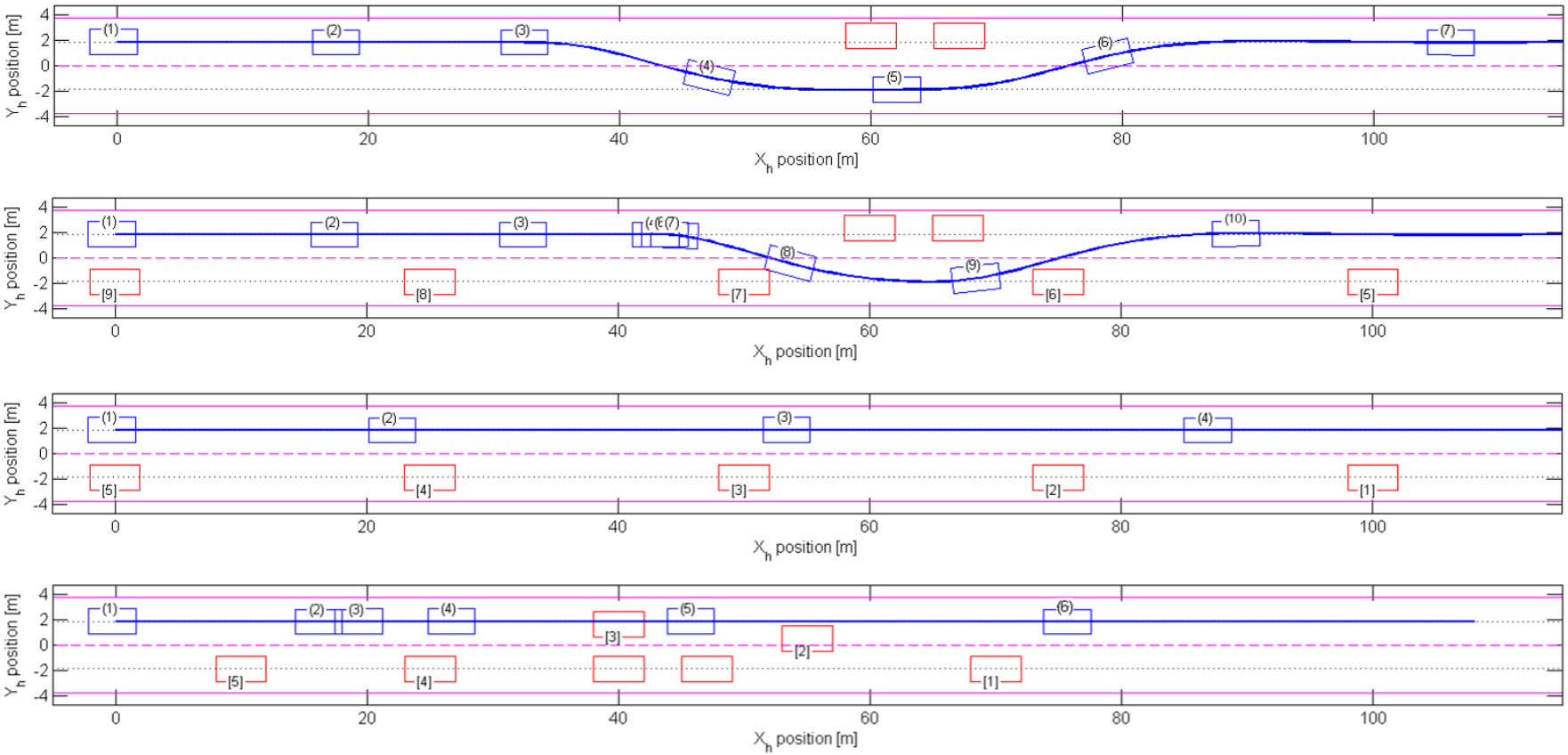


Figure 7.25: Use-case simulations

# 8 Conclusion

This thesis set out to create a longitudinal and lateral controller for autonomous vehicles in urban environments using MPC. The controller showed satisfying results and managed to find paths and guide the vehicle in a safe and controlled manner. The controller was quick with a maximum average calculation time of 0.0159 s. However, in use-case 1,2 and 4 the maximum calculation times were sometimes the double compared to the simulation average. This is most likely correlated to the decision-making, the spikes in calculation can be seen at the same time when the decision-making changes the desired velocity profile or the preferred lane center. Both these references have abrupt changes depending on the decision which could be the cause for the higher calculation times.

Because the decision-making is done at every loop the MPC cannot predict the future decision. This is most apparent during overtaking maneuvers. In Figure 7.7 the MPC first accelerates up to the low speed limit during overtaking and when it decides to go back into its lane it accelerates again. If the MPC would include information about this decision it might try to reduce the on and off acceleration section by smoothing out the acceleration curve.

The decision-making module demonstrated that it could produce safe and predictable decision without hesitation. The driving behavior was not considered during the tuning process, this is partly since it could easily be adjusted in the decision-making process. Furthermore, the desired driving behavior is subjective, and it does not reflect the controller's performance.

At the beginning of the thesis it was investigated whether it would be possible to create a MPC controller without approximating the APF. This would make it possible for the vehicle to fully find its own path in the APF without the influence of reference paths that the controller needs to track. However, this proved to be impossible with the nonlinear APFs. A NLMPC would need to be used which was determined to be to complex and computationally heavy for the application in autonomous vehicles. After having transitioned to the path tracking approach with a separate decision-making module it was questioned whether the APF served a purpose in the MPC. The reason to keep the APF is the MPC was twofold, the first reason was safety and the second one was tuneability. When the ego vehicle was near other vehicles and affected by its APF it would deviate from the lane center reference to avoid collision. The road APF also provided better results when tuning. The lane hump reduced the time the ego vehicle was in between lane centers and the drastic increase in APF potential value at the road boundaries helped reduce the lateral overshoot.

It was also investigated whether it is possible to couple lateral and longitudinal control. The current controller has uncoupled control which means it needs to follow the predefined velocity profile to accurately represent the vehicle model. This was problematic with the kinematic bicycle model. This is because the kinematic model needed a constant velocity assumption or a velocity profile to include both lateral and longitudinal control in the MPC. If coupled control could be implemented the controller could optimize its velocity based on the surrounding obstacles APF instead of having to rely on the velocity profile.

Despite this, the linearized kinematic bicycle model proved to be a good vehicle model for the MPC. It had a good balance between accuracy and fast calculation times. The dynamic bicycle model could not be used as a plant model because of the 'stop and go' scenarios. However, the kinematic model should still be valid as a plant model as long as the lateral acceleration does not exceed $0.5\mu g$ [19]. From Table 7.1 the maximum lateral acceleration in the simulations was 0.1663 $m/s^2$. This makes the kinematic bicycle model valid as a plant model even for snow covered roads according to [23].

When first simulating the MPC the yaw angle was not included as a reference in the cost function. As a result, it was found that the MPC did not consider the vehicle's final states of the horizon. This was particularly problematic at low velocity lane change maneuvers where the vehicle did not complete a full lane change within the prediction horizon. In these scenarios the optimal path could end up at the new lane center but at a high yaw angle. Instead of creating an s-shaped lane change the controller would favor a continuous turn. If this maneuver was followed by a vehicle it would end up outside the road boundaries. As the prediction horizon receded and the vehicle came closer to the new lane center the optimal path began to settle. At this point in time the aggressive continuously turning trajectory initialized by the previous horizons created an overshoot for the lateral control.

This problem was partly solved by introducing the yaw angle cost function term that penalized large yaw angles. This reduced overshoots and at the same time reduced the lateral acceleration. The road APF lane divider hump height was also reduced to create more shallow slopes. The prediction horizon was also lengthened to increase the vehicle's traveled distance.

At first the references were created using Bézier curves. The path tracking was a s-shaped curve like the one used in the path scouting. The velocity reference was based on the same principle where the initial acceleration would determine the initial slope of the velocity profile. When simulating it was difficult to achieve quick lane change maneuvers and accelerations. This is because the MPC is trying to create smooth maneuvers with smooth references. The controller got better results when the references were simplified which in turn simplified the code and subsequently reduced the calculation time.

# 9 Future Work

Future work on this project can be focused on real life implementation. To do this the work needs to be adapted to go along with other systems in the software architecture of an autonomous vehicle. This means adapting the controllers internal coordinate system to follow the vehicle or be vehicle fixed instead of ground fixed. The code needs to be converted to C-code to work outside of MATLAB, currently MATLAB's *quadprog* is not C-code compatible. A good substitute for the *quadprog* routine is MATLAB's *mpcqpsolver*, which is part of MATLAB's MPC toolbox. This routine is both C-code compatible and better suited for real-time implementation as it has more efficient calculations. The *mpcqpsolver* uses the same structure as *quadprog* with the difference being that it uses a lower-triangular Cholesky decomposition of the Hessian matrix [15].

Depending on the previous modules in the software architecture the APFs would either be provided or would need to be created from existing environment data from the environmental builder module. The controller would also need to depend on the curvature of the road as the road cannot be assumed to be straight, even for short prediction horizons. To accommodate this the road boundaries and lane centers need to take the road curvature into account.

Before testing is done in a demo vehicle the MPC controller would preferably be simulated in a more complex simulation software such as: CarMaker or SimScape. The simulations were done on a portable laptop, for more realistic simulation times the algorithm would need to be simulated in the hardware that is more likely to end up in an autonomous vehicle. There would also need to be a study to determine whether it is necessary to include signal delays into the model, if so the controller would need to be modified to accommodate this.

## Bibliography


[1] Cardew, K. H. F. (1970) The Automatic Steering of Vehicles: An Experimental System Fitted to a DS 19 Citroen Car. Available online at https://trl.co.uk/sites/default/files/LR340.pdf.

[2] DARPA The DARPA Grand Challenge. Ten Years Later. DARPA. Available online at https://www.darpa.mil/news-events/2014-03-13.

[3] Freund, Robert M. Solution Methods for Quadratic Optimization.

[4] Grand Cooperative Driving Challenge GCDC - Grand Cooperative Driving Challenge. GCDC. Available online at https://www.saferresearch.com/projects/gcdc-grand-cooperative-driving-challenge.

[5] Greathouse, Jeffery A.; Durkin, Justin S.; Larentzos, James P.; Cygan, Randall T. (2009) Implementation of a Morse potential to model hydroxyl behavior in phyllosilicates. In *The Journal of chemical physics*. DOI: 10.1063/1.3103886.

[6] Han, Jixia; Hu, Yi; Dian, Songyi (Eds.) (2018) The State-of-the-art of Model Predictive Control in Recent Years (428).

[7] Hellström, Erik; Ivarsson, Maria; Åslund, Jan; Nielsen, Lars Look-ahead control for heavy trucks to minimize trip time and fuel consumption. Linköping University (2).

[8] Hrovat, D.; Di Cairano, S.; Tseng, H. E.; Kolmanovsky, I. V. (Eds.) (2012) The development of Model Predictive Control in automotive industry: A survey, October 3-5. Dubrovnik, Croatia.

[9] Johanssen, Tor Introduction to Nonlinear Model Predictive Control and Moving Horizon Estimation. Available online at http://folk.ntnu.no/torarnj/nonlinear.pdf.

[10] Karjanto, Juffrizal; Md. Yusof, Nidzamuddin; Terken, Jacques; Delbressine, Frank; Hassan, Muhammad Zahir; Rauterberg, Matthias (Eds.) (2017) Simulating autonomous driving styles: Accelerations for three road profiles (90).

[11] Kong, Jason; Pfeiffer, Mark; Schildbach, Georg; Borrelli, Francesco Kinematic and dynamic vehicle models for autonomous driving control design. University of California, Berkeley.

[12] L.E. Dubins (1957) On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents, pp. 497–516. Available online at http://www.jstor.org/stable/2372560.

[13] Law, Che Kun; Dalal, Darshit; Shearrow, Stephen Robust Model Predictive Control for Autonomous Vehicle/Self-Driving Cars.

[14] Louwerse, W.J.R.; Hoogendoom, S. P. (2004) ADAS safety impacts on rural and urban highways. University of Parma.

[15] MATLAB MPC Toolbox. mpcqpsolver. Available online at https://se.mathworks.com/help/mpc/ref/mpcqpsolver.html.

[16] Morari, Manfred; Raimondo, Davide Martino; Thoma, Manfred; Allgöwer, Frank; Magni, Lalo. A Survey on Explicit Model Predictive Control, in Nonlinear Model Predictive Control, Online-ausg ed. Berlin, Heidelberg (2009), ch. 29, pp. 346–369.

[17] Nilsson, Julia; Falcone, Paolo; Ali, Mohammad; Sjöberg, Jonas (2014) Receding horizon maneuver generation for automated highway driving.

[18] Nunes, Giovani (2001) Design and analysis of multivariable predictive control applied to an oil-water-gas seperator. A polynomial approach. University of Florida.

[19] Polack, Philip; Altche, Florent; d'Andrea-Novel, Brigitte; La Fortelle, Arnaud de The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? University of California, Berkeley.

[20] Snapper, E. Y. (2018) Model-based Path Planning and Control for Autonomous Vehicles using Artificial Potential Fields. Master of Science. Delft University of Technology, Delft. Faculty of Mechanical, Maritime and Materials Engineering.

[21] Usieto, Gerard Ferrer (2017) Trajectory generation for Autonomous Highway Driving using Model Predictive Control. Graz University of Technology, Graz. Institute of Automation and Control.

[22] Zhao, Shiquan; Maxim, Anca; Liu, Sheng; Keyser, Robin de; Ionescu, Clara (2018) Effect of Control Horizon in Model Predictive Control for Steam/Water Loop in Large-Scale Ships (12).

[23] Zhao, You-Qun; Li, Hai-Qing; Lin, Fen; Wang, Jian; Ji, Xue-Wu (2015) Estimation of Road Friction Coefficient in Different Road Conditions Based on Vehicle Braking Dynamics. Chinese Mechanical Engineering Society (4).

TRITA -SCI-GRU 2019:
363
ISSN 1651-7660