Model Complexity and Coupling of Longitudinal and Lateral Control in Autonomous Vehicles Using Model Predictive Control

CHRISTIAN OLSSON
ACKNOWLEDGEMENTS

A big thanks to my supervisor Jonas Mårtensson who introduced the thesis topic to me and gave invaluable feedback during our meetings. The biggest thanks to my co-supervisor Pedro Lima for assisting me throughout the whole thesis and giving feedback on my work.
ABSTRACT

Autonomous vehicles and research pertaining to them have been an important topic in academia and industry in recent years. Developing controllers that enable vehicles to perform path and trajectory following is a diverse topic where many different control strategies are available. In this thesis, we focus on lateral and longitudinal control of autonomous vehicles and two different control strategies are considered: a standard decoupled control and a new suggested coupled control.

In the decoupled control, the lateral controller consists of a linear time-varying model predictive controller (LTV-MPC) together with a PI-controller for the longitudinal control. The coupled controller is a more complex LTV-MPC which handles both lateral and longitudinal control. The objective is to develop both control strategies and evaluate their design and performance through path following simulations in a MATLAB environment.

When designing the LTV-MPC, two vehicle models are considered: a kinematic model without tyre dynamics and a dynamic bicycle model with tyre forces derived from a linear Pacejka model. A research on how model complexity affects tracking performance and solver times is also performed. In the end, the thesis presents the findings of the different control strategies and evaluate them in terms of tracking performance, solver time, and ease of implementation.
## CONTENTS

1 Introduction .............................................. 5

2 Background ............................................. 7

3 Problem Formulation ................................. 12

4 Vehicle Modelling .................................... 13
   4.1 Kinematic Vehicle Model ..................... 13
   4.2 Dynamical Vehicle Model ................. 14

5 Vehicle Control ........................................ 18
   5.1 System Overview ............................. 18
   5.2 Lateral Control ............................. 18
   5.3 Longitudinal Control ...................... 19
   5.4 Decoupled Control ......................... 19
   5.5 Coupled Control ........................... 20
   5.6 Model Predictive Control ............... 21
      5.6.1 Introduction to MPC ................. 21
      5.6.2 MPC for tracking ...................... 22
      5.6.3 MPC for Path Following .......... 23
      5.6.4 Linear MPC ............................ 25
      5.6.5 Turn MPC into Quadratic Programming 26
      5.6.6 Tuning ................................ 28
      5.6.7 Coupled Path Following MPC ......... 30
   5.7 Speed Profiling ............................... 30

6 Simulation Setup ...................................... 34
   6.1 System Overview ............................. 34
   6.2 Waypointing - Creating the References .. 35
   6.3 Simulation Path .............................. 36
   6.4 Performance Measurement and Tuning .. 36
   6.5 MPC Solver Evaluation ...................... 38
   6.6 Tuning the Controllers .................... 39
CHAPTER 1

INTRODUCTION

The cruise control in vehicles is far older than people believe, dating back to early 1900's and was an adaptation from James Watt and Matthew Boulton's steam engine control. The modern version of a cruise control was invented 1948 by the mechanical engineer Ralph Teetor and it was capable of keeping the speed of a vehicle constant on any road [1]. The cruise controller can be seen as the first step towards autonomous, or self driving, cars. The cruise controller is an example of a longitudinal controller, affecting the acceleration and deceleration of a car. The next step in autonomous vehicle control would be lateral control, which is how to make the vehicle steer itself.

In 2005, a group of engineers at Google won the 2005 DARPA Grand Challenge for their work on Google's self driving car [2]. Using a wide range of sensors, the car had both longitudinal and lateral control and could successfully navigate on public roads in the USA. This autonomous Toyota Prius caught the public's attention and after 2005 the world has seen a rise in autonomous vehicle technology. More recently, German car manufacturer Audi has produced a autonomous R7 race car which has shown incredibly performance on race tracks, producing lap times on the same level as professional racing drivers [3]. When autonomous vehicles were first introduced, it was a mere proof of concept that worked on closed roads and highly supervised by engineers, but today there are several car manufacturers who wish to introduce autonomous vehicle systems to the public, including Mercedes, Volvo and Scania.

Scania leads a project called iQmatic which aims to introduce autonomous trucks and heavy duty vehicles in closed environments, such as mines. In this vast project lies topics of vehicle control, path finding, obstacle avoidance, and many more which all together will soon produce a fully autonomous Scania truck. A part of this project involves the use of model predictive controllers (MPC) for lateral control of the truck. The benefit of using MPC is that it is a very fast optimization based control that has proved to be suitable for task such as path and trajectory following. The use of MPC in autonomous vehicles is a blooming research field and many aspects of control,
stability, implementation, among others, are being researched every day.

A specific research topic pertains to vehicle modelling and control. The majority of mathematical vehicle models are coupled, meaning that trying to change one state of the vehicle is likely going to affect another state. A common control strategy is to have separate control for lateral and longitudinal dynamics, where the most common being a stand-alone lateral controller together with an ordinary cruise controller. This way, an engineer can control what speed the vehicle will travel and let the lateral controller handle the steering. A more advanced version of this is using one lateral controller and one longitudinal controller that can control speed, acceleration and deceleration of the vehicle. These two controllers, when they work independently are called decoupled control and it is a common control strategy for autonomous vehicles.

In this thesis, we investigate if the fact that the vehicle models are coupled can be used to create a coupled control strategy. The idea being that a combination of lateral and longitudinal control can work together and be beneficial for the autonomous system. We will also research how model complexity in MPC affects performance and if it is possible to perform path following with lower model complexity in the MPC than in the controlled vehicle.
CHAPTER 2

BACKGROUND

The road to a completely autonomous vehicle crosses a vast amount of engineering fields that all need to be integrated into one complete system. In the article on Mercedes' autonomous car "Bertha", the authors give a detailed view of what parts are needed to complete the system [4]. An overview of the system is shown in Figure 2.0.1 and the Bertha system has here been simplified into four layers. The top layer contains hardware and sensors. Bertha has a great amount of camera and radar sensors that gives the vehicle a situational awareness and helps the car interact with its environment. On the inside, there are vehicle sensors which measures and observes the internal states of the car, including engine, steering and throttle control.

The second layer contains all tools for performing localization and interpreting the sensor data, or perception. A high resolution GPS keeps track of the global position of the car that together with a detailed digital map tells the driving system where it is and what restrictions the current position imposes on the vehicle. Examples of information contained in the detailed digital map are road conditions, speed limits and traffic rules. In the bottom layers reside motion planning and trajectory control. These fields are responsible for controlling the longitudinal and lateral dynamics of the car. The motion planning layer is one of the most important layers as it works as a link between sensors and actuators. The goal of the motion planning is to output waypoints, path or trajectory to the control layer. The control layer then creates control inputs to the vehicle in order to perform path or trajectory following.

To design a controller that will control an autonomous vehicle one needs to have a mathematical model of the vehicle. In [5] the authors use a simple nonholonomic vehicle model which consists of basic kinematic equations. Using this model, the authors were able to design and test their control system and perform a simple path following. The drawbacks with the simple model is that it is not accurate with the real behaviour of a vehicle. In papers [6][7][8], the authors use what is called a "bicycle model". This model accounts for vehicle dynamics and tyre forces and behaves more closely to a real vehicle. The bicycle model is still a simplification of a real vehicle as it only contains two wheels in the model. The two front and the two rear wheels have been combined to one front and one rear wheel on the center
Figure 2.0.1: Sketch of the layers in the system architecture for an autonomous vehicle.

An introduction to how moving controllable objects can be controlled to perform trajectory tracking using MPC is given in paper [5]. The authors wish to control a nonholonomic wheeled robot and have it track a predetermined trajectory. To predict the motion of the robot, the authors use a simple three state kinematic vehicle model which tracks the vehicle's global $X, Y$ position as well as its orientation. The vehicle is controlled using its linear and angular velocity where the latter is governed by the steering angle of the single front wheel. An initial observation is that the model used in the MPC is not an exact derivation of the model of the vehicle, but
rather a simplified and less complex model where assumptions such as no wheel slip are made. The authors explain how to formulate a linear MPC problem and transforming it into a quadratic programming (QP) while imposing constraints on the control inputs. The authors however mention very little about tuning the controller and how to create the references that are sent to the robot. The result of the paper is an MPC that successfully controls a kinematic vehicle model.

F. Borrelli is one of the leading researches in the field of MPC for autonomous vehicles, and several papers by him have been used, including [10, 7, 9, 8]. In [10], the authors discuss two different solver approaches for path following using MPC. The first approach involves a nonlinear vehicle model based MPC that requires a nonlinear optimization solver. This approach proves to be unsuitable for a vehicle as it needs several calculations per second and the nonlinear solver cannot output control inputs quick enough. The second control approach is to use a linearized model with time varying parameters (LVP-MPC) which is a simplified version of the nonlinear vehicle model but with some degree of adaptation. The linear problem can be solved quick enough for vehicle applications and simulations show good lane keeping and obstacle avoidance performance for the LPV-MPC. The discussion of the paper presents the advantages of reducing the model complexity for MPC and how it can be modified to yield good performance.

In [7], the authors introduce an alternative reference tracking approach together with corresponding tuning. Here they have two different approaches for the reference trajectory generation. The first approach involves a method called “receding horizon trajectory replanning”, which aims to replan the path of the vehicle in order to always have an optimal path to travel down. This replanned path is then passed to the MPC controller. The second approach uses the initial desired path and passes it directly to the MPC without reworking it online. The results of their simulation conclude that both approaches solve the path following satisfactory but the second approach is less complex and less computational heavy. The horizon for the second approach is nearly double that of the first, and since the results are very similar it raises a question regarding whether or not increased system complexity yields better simulations.

In [9], the authors show an MPC controller which is based on an extended bicycle model which uses four wheels in the models instead of the standard two. The use of a more complex model has given a rise in state constraints as the number of states has increased. The authors have also made a couple of assumptions on the model in order to limit the computational complexity. Initially, there is an assumption that aerodynamical forces are negligible when computing the load transfer. Secondly, the authors assume the rotational inertia of the wheels is negligible, meaning that the wheels are assumed to be equally easy to rotate using wheel torque computations. Thirdly, the low level longitudinal controller distributes the forces evenly on all wheels and that braking actions of the wheels are assumed constant over the prediction horizon. The test of the controller is performed on a typical low curvature road scenario, meaning high speeds and light corners. The heading angle of the ve-
hicle is hence assumed small and the necessary steering angle to keep the vehicle along the road is small. The increased model complexity gives way for additional constraints in their MPC including the tyre slip angles and the vehicle's position on the road in order to perform lane keeping. The final simulation is performed in an obstacle avoidance scenario and the results from their simulations are very promising.

The research presented in [8], gives a very clear and structured overview of how to utilize the bicycle model and design a MPC for lateral control. All simulations are done with constant speed which shows that the authors have focused their attention to the steering part of autonomous vehicles. In this paper, the authors have chosen to use the full version of the Pacejka model, explained in [11], and explain how the tyre forces are generated and how they affect the vehicle. When constructing their linear time-variant MPC (LTV-MPC) the authors have used what appears to be a industry standard cost function with different prediction horizon and control horizon length, where prediction horizon being longer than control horizon. Using the Pacejka model, the authors argue that a constraint on the tyre slip angle has to be introduced as a linear MPC control is able to predict the change of the slope in tyres' characteristics. More precisely, the tyre slip angle has been limited with a minimum and a maximum value which it has to remain within during the whole prediction horizon. For the final simulations they have used a sampling time of 0.05 seconds together with a prediction horizon of 25 and control horizon 10. According to their results, the average solver time is equal the sampling time which proves the practical use of their control strategy if it were to be implemented in a real system. An interesting remark is done regarding linear and non-linear MPC solvers. The performance using a nonlinear MPC solver is great in terms of path following and constraint abiding, however the mean solver time is in the dimension of seconds, while the linear MPC solver is able to match the desired sample time.

The idea of coupled and decoupled longitudinal and lateral dynamics have been dealt with in two prominent papers, [12] and [13]. In [12], the authors present a very detailed and advanced modelling of an entire vehicle, including engine and power train modelling. Even though this vehicle model very advanced, they introduce a clean control architecture which includes several of the necessary components needed for trajectory tracking and path following. An illustration of this system is shown in Figure 2.0.2a.

A strategy for coupling the longitudinal and lateral control is presented in [13] where the idea is to use a nonlinear MPC and a longitudinal controller that are both governed by a constraint on the side slip angle of the vehicle. Their control architecture is shown in Figure 2.0.2b. The industry practice for path following MPC is to use a control system that decouples the longitudinal and lateral dynamics. This practice has been shown in [12] and [9]. Decoupling the control means that we use one controller for the lateral dynamics, for example an MPC, and another controller for the longitudinal dynamics which, for simplicity, can be a simple PID controller. In [12], the authors have suggested a control approach where they couple the lateral and
longitudinal dynamics by combining them into one control system. The system still has two separate controllers for longitudinal and lateral dynamics but exchange information between the two while working under similar constraints on the tyre slip angle. The authors bring up important topics regarding lateral stability and what kind of constraints to use when controlling both lateral and longitudinal dynamics.

In this thesis, we will research the areas of lateral and longitudinal control as well as investigate how model complexity affects the performance of an MPC. On the topic of coupled and combined control, we design an MPC controller that couples the lateral and longitudinal control of a vehicle. This control strategy is compared to a decoupled control system where lateral and longitudinal controllers are separate. The longitudinal control is able to act as a cruise controller to maintain constant speed, as well as track a reference speed. We will use the kinematic vehicle model and the slightly more complex bicycle model and design lateral MPC controllers based on these two. As part of the model complexity investigation, we will investigate if it is reasonable to use an MPC based on a kinematic model to control a bicycle model.
The first topic of this thesis aims to investigate how MPC can be implemented in autonomous vehicle control and explore how longitudinal and lateral control can be achieved. In this thesis, it is suggested for vehicle path following to design a completely coupled controller which handles both longitudinal and lateral dynamics in one controller. The idea is to use one MPC controller which controls both steering angle and acceleration of the vehicle through one QP optimization problem. The expectation of coupled control is to be able to assure the vehicle does not violate any constraints and that instability does not occur due to controller mismatch. An example of this would be to adjust the vehicle speed based on its lateral dynamics and not only the longitudinal. The coupled controller will be designed based on different vehicle models and evaluated in terms of design and performance. The thesis also focuses on designing decoupled controllers and compare these to the coupled controllers.

The second topic of this report aims to analyse model complexity when designing the MPC. A model of a vehicle can be kept at low complexity using a simple kinematic model as in [14]. However, there are some advanced vehicle models, such as the bicycle model used in [7], [8] and [10], and the even more realistic four wheel extended bicycle model used in [9]. The goal is to see whether or not one can use low complex models on high complexity systems, and if so, find what are the limitations and constraints of doing this. It will also look into possible advantages of such a control strategy by simulating several MPC controllers on a bicycle model.
CHAPTER 4

VEHICLE MODELLING

In general, a simple vehicle model with few states can be evaluated and predicted quickly, while a more complex model will yield more accurate state predictions but at the cost of computational complexity. In this thesis, two different types of models are tested: nonholonomic kinematic model for wheeled vehicles and a dynamical bicycle model. The two models will henceforth be referred to as the kinematic model and the dynamic model, respectively.

4.1 KINEMATIC VEHICLE MODEL

The kinematic model consists of four states which track a vehicle’s speed, coordinates in a global coordinate system and its orientation within it. A graphical representation of the kinematic model is shown in Figure 4.1. The states are $z = [XYθv]$ and the governing kinematic equations are:

$$
\begin{align*}
\dot{X} &= v \cos(θ) \\
\dot{Y} &= v \sin(θ) \\
\dot{θ} &= \frac{v}{D} \tan(δ) \\
\dot{v} &= a,
\end{align*}
$$

where $v$, $D$ and $δ$ is vehicle speed, vehicle length and the angle of the steering wheel, or steering angle respectively. The control input vector $u$ consists of the steering angle $δ$ and an acceleration $a$. This model assumes that the vehicle has perfect grip with the ground and never slips in any direction and the orientation of the vehicle changes with the steering angle. The simplifications in this model make it easy to implement in the MPC but makes it less accurate in terms of predicting the state evolutions. Studying the equations in (4.1.1) one sees that the longitudinal control input, $a$, only affects the speed of the vehicle, but the speed of the vehicle affects the lateral control. The kinematic equations and their states have some coupling which is important to consider during the controller design part of the thesis.
4.2  DYNAMICAL VEHICLE MODEL

To represent a vehicle more accurately, one needs to model vehicle dynamics. This helps predicting and accounting for behaviours such as understeering, oversteering, side-slipping and friction [15]. The key difference between the kinematic model and the bicycle model is the modelling of tyre forces. This means that not only the vehicle kinematics affect where the vehicle is headed, but there will as well be new forces affecting the overall handling of the vehicle. The bicycle model is built upon a set of differential equations that are presented in (4.2.1).

In Figure 4.2.1 we see the simplified dynamical model of a vehicle where the four wheels have been reduced to two in the middle of the vehicle axis so that it resembles a bicycle. This model also assumes that the center of gravity (CoG) is just above the ground, i.e. the vehicle cannot tip over. The derivations of the vehicle dynamics are inspired by [15], where one firstly considers the acceleration of the CoG of a vehicle using the derivative of the velocity vector \( \vec{\mathbf{v}} = v_x \hat{x} + v_y \hat{y} \), which gives

\[
\ddot{\mathbf{r}} = \dot{v}_x \hat{x} + \dot{v}_y \hat{y} + \omega \times (v_x, v_y, 0) = (\dot{v}_x - \dot{\phi} v_y) \hat{x} + (\dot{v}_y + \dot{\phi} v_x) \hat{y}.
\]

Considering the forces in Figure 4.2.1, the equations of motion become:

\[
\begin{align*}
mx &= m(\dot{v}_x - \dot{\phi} v_y) = -F_{12} \sin \delta \\
my &= m(\dot{v}_y + \dot{\phi} v_x) = F_{34} + F_{12} \cos \delta \\
J\dot{\phi} &= f F_{12} \cos \delta - b F_{34},
\end{align*}
\]

where \( f, b, J \) are distance from CoG to front tyre, distance from CoG to rear tyre and the inertia of the vehicle respectively. To calculate the forces on the tyres, \( F_{12} \) and
There exists a formula called "Pacejka's Formula", or more colloquially known as "The Magic Formula" \cite{11}. The forces are therefore usually formulated as

\[
\begin{align*}
F_l &= f_l(\alpha, s, \mu, F_z) \\
F_c &= f_c(\alpha, s, \mu, F_z),
\end{align*}
\]

where $F_l$ and $F_c$ are the longitudinal force component on the wheel and the lateral, or cornering, force component respectively. The function $f_l$ and $f_c$ are highly nonlinear and based on a semi-empirical model which depends on friction, $\mu$, side slip angle, $\alpha$, slip ratio, $s$ and vehicle weight, $F_z$ \cite{10}. A plot of the non-linear behaviour of tyre forces can be seen in Figure \ref{fig:tyre_forces}.

One thing to notice in Pacejka's formula is that the tyre forces are close to linear in a small region around zero slip. When the speed of a vehicle is kept low, the slip angle and ratio stay within these bounds. Instead of using the nonlinear version of Pacejka's formula, one can estimate the forces with a linear function in a region around $\alpha = 0$. This assumes that the side slip and slip angles stay relatively low, which is a fair assumption in this case where we want to control a vehicle and are not interested in racing techniques such as drifting and sliding. Based on \cite{15}, we can approximate the forces acting on the front and rear wheels in the bicycle model with:

\[
\begin{align*}
F_{12} &= -C_{12}\alpha_{12} \\
F_{34} &= -C_{34}\alpha_{34},
\end{align*}
\]
Figure 4.2.2: The nonlinear behavior of Pacejka’s tyre formula [10].

where $C_{12}$ and $C_{34}$ are the cornering stiffness of the front and rear wheel respectively. The slip angles $\alpha_{12}$ and $\alpha_{34}$ are defined as:

$$
\alpha_{12} = \arctan\left(\frac{v_y + \dot{\phi} f}{v_x}\right) - \delta, \quad \alpha_{34} = \arctan\left(\frac{v_y - \dot{\phi} b}{v_x}\right).
$$

For the coupled MPC, we will add a second control input in the form of an additional force on the CoG of the vehicle. The force, $F_a$ stems from the engine for speeding up or the breaks for slowing down the vehicle as shown in Figure 4.2.3.

For practical implementation reasons, the control input will be an acceleration. Using newton’s third law of motion, $F_a = ma$, yields the equations of motion:

$$
f(z, u) = \begin{cases} 
m \ddot{x} = m(\dot{v}_x - \dot{\phi} v_y) = -F_{12} \sin \delta + ma \\
m \ddot{y} = m(\dot{v}_y + \dot{\phi} v_x) = F_{34} + F_{12} \cos \delta \\
J \ddot{\phi} = f F_{12} \cos \delta - b F_{34}, \end{cases}
$$

(4.2.1)

where the control inputs are $u = [\delta, a]$. Putting it all together, we have six states $z = [X, Y, \theta, \dot{\phi}, v_x, v_y]$ and $u = \delta$ for the decoupled control, and $u = [\delta, a]$ for the coupled control.
Figure 4.2.3: Bicycle model with added $F_a$ force [15].
CHAPTER 5

VEHICLE CONTROL

5.1 SYSTEM OVERVIEW

When setting up an autonomous or automatic vehicle control system one needs multiple layers of sensing and computational technology, and controllers that govern the movement of the vehicle: lateral control and longitudinal control [4]. The lateral controller generally controls the steering angle of the vehicle while monitoring lateral forces that arise from cornering manoeuvres. The goal of the lateral controller is, simply put, to keep the vehicle on the road and to follow a path or trajectory. The longitudinal controller governs all the dynamics that involve the speed of the vehicle. It can be as simple as a cruise controller which keeps a vehicle at a set speed or more advanced in terms of actively adapting the speed according to given road conditions [13]. The lateral and longitudinal controllers can be seen as independent systems in situations when vehicle tyre forces are small. On real roads with real vehicles this assumption cannot be done due to factors such as road friction and wheel torque [16].

5.2 LATERAL CONTROL

The objective of the lateral controller is to output a steering angle for the vehicle. The steering angle is the measured angle between the vehicle’s x-axis and the direction of the steering wheels. By issuing steering angle commands, the lateral controller aims to keep the vehicle on a reference path or reference trajectory. The lateral controller may also try to reduce tyre slip and excessive lateral forces when the vehicle is cornering [8, 10]. When setting up the lateral controller one needs to know what vehicle constraints are necessary and how the control signal should be limited.

Given a situation where a vehicle is following a path, the main objective is to keep the vehicle as close to the path as possible. Any deviations from the path are to be avoided and when the lateral controller outputs steering angles it is important that these control commands do keep the vehicle on the path for a close future [9]. It is
hence important that the lateral controller is a closed-loop control which can compensate for previous control errors. An important aspect of the lateral controller is the overall vehicle stability, poor control inputs may cause the vehicle to lose lateral stability. The vehicle kinematics and dynamics are therefore important to consider in the lateral controller.

In terms of constraints, the lateral controller has to consider the physical limitations of the vehicle. Two common limitations for the lateral controller are steering rate and maximum steering angle. The steering rate is how fast one can turn the wheels of the vehicle and is often mechanically limited [12]. Performance wise, one would like to avoid control actions that cause the vehicle to swivel on the road, and hence one tends to limit the steering rates in a conservative way [10]. On a normal path following the steering rate limitation is rarely reached while in other scenarios such as obstacle avoidance the vehicle may operate close to the maximum steering rate of the vehicle [9]. The maximum steering angle is based on how much the wheels of the vehicle can turn and is often limited to 45\degree.

5.3 Longitudinal Control

To control the speed and acceleration of a vehicle, one needs to introduce a longitudinal controller. From the vehicle modelling one knows that the change in speed $v$ is given by the acceleration $a$. A relation on the form $\dot{v} = a$ is modelled by what is called an integrator in field of control theory, and can be controlled by a PID-controller [17]. The acceleration of the vehicle is often constrained due to mechanical performance and comfort, and therefore the longitudinal controller should be designed with an input saturation [12]. The maximum acceleration, $a_{\text{max}}$, ensures a smooth and not too aggressive acceleration while the minimum acceleration, $a_{\text{min}}$, decides how quickly the vehicle may slow down. Usually, $|a_{\text{max}}| < |a_{\text{min}}|$ given that slowing down a vehicle is done more quickly than speeding it up and in a road safety perspective, stopping a vehicle is more important than speeding it up [15]. The control input is calculated based on reference speeds that are fed to the system and the corresponding control action is an acceleration of the vehicle. In order to control the longitudinal dynamics, it is assumed that one can control the throttle response of the vehicle, causing it to accelerate instantly. The act of accelerating the vehicle gives rise to an additional force, $F_a$, in the dynamic vehicle model illustrated in Figure 4.2.1.

5.4 Decoupled Control

The conventional theory on decoupling control describes how a system with multiple inputs and outputs (MIMO) can have a controller that isolates the inputs and outputs so that one input affects only one output instead of multiple. The act of doing so is called decoupling [18]. In vehicle control, the vehicle models have coupled
kinematic and dynamics which means that lateral and longitudinal dynamics may affect each other. Decoupled control of a vehicle means that the longitudinal and lateral dynamics of the vehicle are handled separately by two different controllers. In [16], the authors state that the dynamical forces of a vehicle must be kept low in order to have lateral and longitudinal dynamics controlled separately. This means that the decoupled control may be limited by the dynamics of the vehicle.

In practice, a decoupled control strategy uses two different control approaches for the lateral and longitudinal dynamics. Generally, the longitudinal control will be first to optimize the acceleration and speed of the vehicle before it feeds the lateral controller with the current speed of the vehicle. The lateral controller then uses the information about the speed in order to output a suitable steering angle [12].

A simplified drawing of the decoupled control system is shown in Figure 5.4.1.

![Figure 5.4.1: Longitudinal and lateral control diagram for the decoupled strategy.](image)

In this thesis, the decoupled control strategy will use a PI-controller for the longitudinal controller and an MPC controller for the lateral control. To handle longitudinal constraints, the PI-controller will be subject to control input saturation to avoid too large vehicle accelerations.

### 5.5 Coupled Control

The coupled control strategy attempts to benefit on the fact that the lateral and longitudinal dynamics of the vehicle are coupled in both vehicle models. In the decoupled control strategy the lateral and longitudinal controllers are separated and the lateral controller cannot send any information to the longitudinal controller, which may lead to unstable longitudinal control inputs [16]. In coupled control, the lateral and longitudinal control are combined into one big control problem where acceleration and steering angle are outputted simultaneously. The act of coupling the two controllers will make the control problem more complex in terms of computational
burden. For example, in the decoupled strategy, the longitudinal controller only has to consider longitudinal dynamics and constraints before outputting a control action. In the coupled strategy, each longitudinal control output will also be evaluated in terms of lateral constraints and the two controllers will be able to exchange information between them, as shown in Figure 5.5.1.

In this thesis, the coupled control strategy consists of one complex MPC controller that optimizes steering angle and acceleration while trying to abide both lateral and longitudinal constraints. Compared to the lateral MPC in the decoupled strategy, the coupled MPC the same lateral controller in the decoupled control, but an extended version to account for longitudinal control as well.

5.6 Model Predictive Control

5.6.1 Introduction to MPC

Model predictive control, (MPC), is an optimization based control technique that emerged in the 1960’s in chemical processing, but today finds itself in numerous industries and applications [19]. The applications are versatile and range from computer control on nanosecond scale to industry control and production planning with time spans of minutes to weeks [19]. Unlike PID controllers, the MPC is based on a formulation of an optimization problem, but unlike linear quadratic control, the MPC can handle various constraints on state and control inputs [19]. A basic MPC formulation is shown in (5.6.1).
minimize \( u_i \) \[ z_{H_p}^T P z_{H_p} + \sum_{i=1}^{H_p-1} z_i^T Q z_i + u_i^T R u_i \]

subject to \( z_0 = z(0) \) measurement \( (5.6.1) \)
\( z_{i+1} = A z_i + B u_i \) linear system model
\( C z_i + D u_i \leq b \) constraints
\( R \succeq 0, Q \succeq 0 \) performance weights,

where \( u_i = \{u_0, u_1, \ldots, u_{H_p-1}\} \) is a sequence of optimal control inputs that minimizes the cost function, and \( z \) is the state of the system we are trying to control. The summation is done over a number of iterations, \( H_p \), called horizon, which means that the MPC does not only calculate the next control input but also plans ahead for future control inputs by trying to predict the behaviour of the system.

The four core parts of an MPC are its cost function, the system model, the constraints and the state measurements. These measurements can come from either a state observer of physical measurements. The cost function has three parameters, or weight matrices, \( Q \), \( R \) and \( P \). These matrices are used to penalize the states and control inputs during the optimization and tuning them is essential for the performance of an MPC. The \( Q \) matrix is most commonly a diagonal matrix of size \((n \times n)\) and the \( R \) matrix is of size \((m \times m)\) where \( n \) equals the number of states and \( m \) the number of control inputs.

The matrices \( C \) and \( D \) help select the states and control inputs that are to be constrained by \( b \). A common use of \( D \) is to set limits on the control signal \( u \) on the form \( u \in [u_{\text{min}}, u_{\text{max}}] \). The matrix \( P \) which appears together with the last predicted state, or terminal state, \( z_{H_p} \), is called terminal state cost. The purpose of the terminal cost is to emphasize the importance of the final state to reach the optimal state. This is normally done by designing the elements of \( P \) so that they are larger than the elements of \( Q \).

5.6.2 MPC FOR TRACKING

The MPC formulated in \((5.6.1)\) is a linear MPC, used when one wants to bring a system to the origin, but one can can rewrite \((5.6.1)\) so that it can track a constant reference \( r \neq 0 \). Using theory from [20], we assume that there exists a state \( z_s \) and control input \( u_s \) that keeps the system at reference \( r \) as in the linear time-invariant (LTI) system \((5.6.2)\).

\[ \begin{cases} z_{t+1} = A z_t + B u_t = z_s \\ y = C z_s = r. \end{cases} \]  

\( (5.6.2) \)

The "s" stands for "steady state", but for our application with tracking and path following, we will use the notation is \( z_{\text{ref}} \) and \( u_{\text{ref}} \). What we now want to minimize is not \( z \) but \((z - z_{\text{ref}})\) and \((u - u_{\text{ref}})\). This is suitably done by introducing two new states:
\( \tilde{z} = (z - z_{\text{ref}}) \) and \( \tilde{u} = (u - u_{\text{ref}}) \). We can now rewrite (5.6.1) to adjust for the new variables.

\[
\begin{align*}
\text{minimize} & \quad \tilde{z}^T H_p \tilde{z} + \sum_{i=1}^{H_p-1} \tilde{z}_i^T Q \tilde{z}_i + \tilde{u}_i^T R \tilde{u}_i \\
\text{subject to} & \quad z_0 = z(0) \\
& \quad \tilde{z}_i = (z_i - z_{\text{ref}}) \\
& \quad \tilde{u}_i = (u_i - u_{\text{ref}}) \\
& \quad \tilde{z}_{i+1} = A \tilde{z}_i + B \tilde{u}_i \\
& \quad C \tilde{z}_i + D \tilde{u}_i \leq b.
\end{align*}
\]

(5.6.3)

The optimal control input \( u^* \) is taken from \( u^* = \tilde{u}^* + u_{\text{ref}} \). The value of \( z_{\text{ref}} \) can be taken directly from \( y = Cz_{\text{ref}} = r \), meanwhile \( u_{\text{ref}} \) can be acquired in a few different ways. The more analytical way of doing it is knowing the reference \( r \) that we want to track, and then find the steady state \( z_{\text{ref}} \) through \( Cz_{\text{ref}} = r \). To remain in steady state, there has to exist a control input \( u_{\text{ref}} \) which keeps us in that state when time tends to infinity.

From (5.6.2) we get that \( z_{\text{ref}} = Az_{\text{ref}} + Bu_{\text{ref}} \), and from here we can solve for \( u_{\text{ref}} \) by constructing the inverse of \( B \) and writing everything as

\[
\begin{align*}
u_{\text{ref}} &= B^{-1} (I - A) z_{\text{ref}}. \tag{5.6.4}
\end{align*}
\]

Note however that \( B^{-1} \) in (5.6.4) is only valid if \( B \) is square and its determinant is well defined and not zero \([21]\). In practical applications, when this is not the case, the analytical pseudo-inverse can be used to approximate \( u_{\text{ref}} \). In later chapters of this thesis it will be shown that one can achieve good tracking with suboptimal \( u_{\text{ref}} \) if you tune \( Q \) and \( R \) cleverly. In our path following scenario, the information from the waypoints prove useful for determining the steady state control input \( u_{\text{ref}} \). Discretizing \( \dot{\theta} \) in (4.1.1) yields

\[
\dot{\theta}_{t+1} = \dot{\theta}_t + \frac{v T_s}{L} \tan(\delta)
\]

and from this we can find the reference steering angle \( \delta_{\text{ref}} = \delta_{\text{ref}} \) that will bring our vehicle to \( z_{\text{ref}} \) by solving for \( \delta_{\text{ref}} \)

\[
\begin{align*}
\delta_s &= \arctan \left( \frac{L}{T_s v} (\theta_{t+1} - \theta_t) \right)
\end{align*}
\]

5.6.3 MPC FOR PATH FOLLOWING

In this thesis we aim to use an MPC tracking algorithm, but unlike (5.6.2) we are not interested in tracking one constant reference, but rather a time-varying reference. As the vehicle moves along the path it needs multiple references to follow to ensure that the vehicle maintains itself on the path. This puts a lot of demand on the references and the MPC tracking algorithm. As the vehicle travels along the path the
waypoint update algorithm has to work in real-time to supply the MPC with reference points to track and if it fails to deliver valid references the path following may fail.

To solve this problem we introduce multiple references in control horizon so that the vehicle tracks a new reference every iteration. Using this algorithm, one needs to update $z_{\text{ref}}$ and $u_{\text{ref}}$ at each sampling time. Formally, this changes our reference control input to $u_{\text{ref}} = (u_{s0}, u_{s1}, \ldots, u_{sH_p-1}, u_{sH_p})$ as well as our reference states become $z_{\text{ref}} = (z_{s0}, z_{s1}, \ldots, z_{sH_p-1}, z_{sH_p})$. This method ensures that the vehicle tracks the reference well while avoiding to deviate from the path assuming the reference are reachable and valid for the vehicle, see more in section 6.2.

A general path following case is depicted in Figure 5.6.1 where the current state $z_0$ is the small black circle and ahead of it are four waypoints. Around each waypoint is a larger circle that signifies the points where the MPC is allowed to be, the reachable set. These sets have a radius of $\epsilon$ and the state constraints of the MPC will try to ensure that the MPC is contained within these sets. Each waypoint has a corresponding iteration number $i$ which means that after the first iteration, the states will be somewhere in set $i = 1$, in iteration two it will be in set $i = 2$ and so on. The euclidean distance between the waypoints is chosen so that the vehicle can reach each waypoint with its current velocity. The rate of change in control input $u$ is limited by

![Figure 5.6.1: MPC path following with waypoints and reachable sets.](image)
setting a maximum control input change between iterations, and maximum control input is constrained using absolute value of $u$.

The final MPC formulation for the lateral controller for solving the path following in Figure 5.6.1 is summarized in equation (5.6.5).

$$
\begin{align*}
\text{minimize} & \quad \tilde{z}_i^T P \tilde{z}_i + \sum_{i=1}^{H_p-1} \tilde{z}_i^T Q \tilde{z}_i + \sum_{i=0}^{H_c} \tilde{u}_i^T R \tilde{u}_i \\
\text{subject to} & \quad z_0 = z(0) \\
& \quad \tilde{z}_i = (z_i - z_{ref}) \\
& \quad \tilde{u}_i = (u_i - u_{ref}) \\
& \quad \tilde{z}_{i+1} = A_i \tilde{z}_i + B_i \tilde{u}_i \\
& \quad |\tilde{z}_i| \leq \epsilon \\
& \quad |u_{i+1} - u_i| \leq \Delta u \\
& \quad |u_i| \leq u_{max}.
\end{align*}
$$

(5.6.5)

5.6.4 Linear MPC

The vehicle models presented in Section 4 contain nonlinear terms and the bicycle model in (4.2.1) is highly nonlinear after introducing tyre forces [15]. There are nonlinear optimization solvers capable of solving the MPC problems but they are all very time consuming [22]. Nonlinear optimization solvers have been used for MPC and vehicle path following, but averaging 2 - 3 seconds for each iteration when the solving time should equal the sampling time between 0.01 and 0.05 seconds, the nonlinear approach is often deemed unsuitable [8].

A possible approach is to use a linear MPC approach which in this application of vehicle path following means that one has to linearize the vehicle models before solving the MPC. The idea of linearizing a nonlinear system is to find a corresponding linear system that behaves roughly the same as the original. In path and trajectory following, the nonlinear model of the vehicle changes too much during the horizon which means that one has to linearize at each sample time [8]. From equation (5.6.5) we can rewrite the system model with a new notation:

$$
\begin{align*}
\tilde{z}_{i+1}/i & = A_i \tilde{z}_i/i + B_i \tilde{u}_i/i
\end{align*}
$$

(5.6.6)

Since the vehicle model will be linearized for each iteration, the $A$ and $B$ matrices will receive a new notation to show which iteration they belong to. The notation $z_{i+1}/i$ means the state $z$ at iteration $i + 1$, based on predictions from iteration $i$.

Given a system on the form $\dot{z} = f(z,u)$, we need to calculate two matrices, $A$ and $B$, so that we can write the system as (5.6.6). Since linearisation is only valid in small regions, we need to linearise around every point we want the vehicle to pass. We denote such a point $(z_{ref}, u_{ref})$. Given that $z_{i+1}/i = z_i + f(z_i/i, u_i/i)$, we get that:
\[ A_i = \frac{\partial g(z_i, u_i)}{\partial z} \bigg|_{z_i = z_{ref} \atop u_i = u_{ref}} \quad B_i = \frac{\partial g(z_i, u_i)}{\partial u} \bigg|_{z_i = z_{ref} \atop u_i = u_{ref}}. \]

What we now have is actually a system on the form:

\[ z_{i+1} | i - z_{ref} | i + 1 | i = A_i (z_i - z_{ref}) + B_i (u_i - u_{ref}), \]

since we want to investigate the behaviour around \((z_i, u_i)\). This will prove to be useful in the case of tracking MPC where the system model is written using tilde formulation where \(\tilde{z} = z - z_{ref}\). For each iteration and each horizon step, the system model needs to be relinearized in order to have as good as possible approximation.

In the case of the kinematic vehicle model given in (4.1.1), where \(z_{i+1} = g(z, u)\), a linearization around \(z_{ref} = [x_s, y_s, \theta_s, v_s]\) and \(u_{ref} = [\delta_s]\) would yield:

\[
A = \frac{\partial g(z, u)}{\partial z} = \begin{bmatrix}
1 & 0 & -v_s \sin \theta_s & \cos \theta_s \\
0 & 1 & v_s \cos \theta_s & \sin \theta_s \\
0 & 0 & 1 & \tan \delta_s \\
0 & 0 & 0 & 1
\end{bmatrix},
\]

\[
B = \frac{\partial g(z, u)}{\partial u} = \begin{bmatrix}
0 & 0 \\
0 & 0 \\
0 & v_s \sec^2 \delta_s \\
0 & 1
\end{bmatrix}.
\]

5.6.5 Turn MPC into Quadratic Programming

The MPC problem stated in (5.6.5) can be solved by transforming the MPC formulation into QP [14]. Given a discrete-time system model

\[ z_{i+1} = A_i z_i + B_i u_i, \]

and a standard MPC formulation as the one given in (5.6.1), the states of the system will be predicted up till the horizon end \(H_p\), where

\[ z(H_p | i) = A(H_p - 1 | i) z(H_p - 1 | i) + B(H_p - 1 | i) u(H_p - 1 | i). \]

This approach will lead to \(H_p\) amount of states and inputs which would mean that one needs to solve for \(H_p\) control inputs and predict \((H_p \times n)\) states [5]. Rewriting this, we can create a QP problem which only depends on the initial state \(z_0\) and control inputs \(u\).

In order to track how the states change during the MPC iterations, we introduce two vectors:

\[
\tilde{z}_{i+1} = \begin{bmatrix}
\tilde{z}_{i+1} | i \\
\tilde{z}(i + 2 | i) \\
\vdots \\
\tilde{z}(i + H_p | i)
\end{bmatrix},
\]

\[
\tilde{u}_{i+1} = \begin{bmatrix}
\tilde{u}_{i+1} | i \\
\tilde{u}(i + 2 | i) \\
\vdots \\
\tilde{u}(i + H_p | i)
\end{bmatrix},
\]

using these two vectors we can write one big system that accounts for the evolutions of the MPC iterations over the horizon, hence create something on the form
\[
\tilde{z}_{i+1|i} = \tilde{A}_{i|i} \tilde{z}_{i|i} + \tilde{B}_{i|i} \tilde{u}_{i|i}.
\]

The matrix \( \tilde{A}_{i|i} \) has one column and \( H_p \) number of rows, with a product sum of \( A_{i|i}, A_{i+1|i}, \ldots, A(i + H_p)i \) matrices. The matrix \( \tilde{B}_{i|i} \) is a lower triangular matrix containing both \( A \) and \( B \) matrices. The complete structure of the two matrices, as seen in [5] is

\[
\tilde{A}_{i|i} = \begin{bmatrix}
A_{i|i} \\
A_{i+1|i} \\
\vdots \\
a(i, 1, 0)
\end{bmatrix},
\]

\[
\tilde{B}_{i|i} = \begin{bmatrix}
B_{i|i} & 0 & 0 & \ldots & 0 \\
A_{i|i} B_{i|i} & B_{i+1|i} & 0 & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
a(i, 2, 1) B_{i|i} & a(i, 2, 2) B_{i+1|i} & \ldots & \ldots & 0 \\
a(i, 1, 0) B_{i|i} & a(i, 1, 2) B(i - 1|i) & \ldots & \ldots & B(i + H_p - 1|i)
\end{bmatrix},
\]

where \( a(i, j, l) \) is defined as

\[
a(i, j, l) = \prod_{i = l}^{N - j} A(i + i|i).
\]  

(5.6.8)

Constructing new matrices \( \tilde{Q} = \text{diag}(Q, \ldots, Q) \) and \( \tilde{R} = \text{diag}(R, \ldots, R) \) we can rewrite the cost function in (5.6.5)

\[
\Phi_i = \tilde{z}^T(i + 1) \tilde{Q} \tilde{z}(i + 1) + \tilde{u}^T(i) \tilde{R} \tilde{u}.
\]  

(5.6.9)

The general QP problem is aimed to solve a problem on the following form [22]:

\[
\text{minimize } \frac{1}{2} u^T H u + f^T u + d \quad \text{subject to } A u \leq b.
\]  

(5.6.10)

For the MPC formulated as a QP, one wishes to solve for the control inputs \( u \) that minimizes the cost function in (5.6.5). Changing the variable \( u \) in (5.6.10) with control input \( \tilde{u} \) we get a QP cost function on the form:

\[
\phi_i = \frac{1}{2} \tilde{u}^T \tilde{H} \tilde{u} + c^T \tilde{u} + d.
\]

To construct the matrices \( H, f \) and \( d \) we look at (5.6.9) and use that \( \tilde{z}(i + 1) = A_{i|i} \tilde{z} + B_{i|i} \tilde{u} \). From this we collect the quadratic and linear \( \tilde{u} \) terms as well as the constant term not containing \( \tilde{u} \) and then we identify from these \( H, c \) and \( d \), presented in (5.6.11).
Now that the optimization problem and the cost function have been constructed, the next items to add are the state and input constraints used in MPC. Since the variable one wishes to optimize in the MPC is $\bar{u}$, the state constraints are rewritten as $D\bar{u} \leq c$. Depending on the amount of control inputs $m$, the size of $\bar{u}$ will be one column and $m \times H_p$ rows. What we wish to constrain with $D\bar{u} \leq c$ is $u_{\text{min}} \leq u_i \leq u_{\text{max}}$. To put all this into matrix form, we use the same method as in [5] where they construct $D$ as a diagonal matrix with first half of diagonal being equal to 1 and the second part equal to $-1$. Since $\bar{u}_i = u_i - \text{u}_\text{ref}$, we need to compensate for the reference inputs in the $d$ vector. The $d$ vector is then upper half consisting of $u_{\text{max}} - \text{u}_\text{ref}$ values and lower half $u_{\text{min}} + \text{u}_\text{ref}$ values so that one gets $u - \text{u}_\text{ref} \leq u_{\text{max}} - u_{\text{ref}}$ and $-(u - \text{u}_\text{ref}) \leq u_{\text{min}} + u_{\text{ref}}$. Note that even if $u_{\text{min}} < 0$, when constructing $d$ one has to use the absolute values of $u_{\text{max}}$ and $u_{\text{min}}$ as the sign shift in $D$ makes sure that the constraints are correctly represented, as shown in (5.6.12).

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}, \quad c = \begin{bmatrix} a_{\text{max}} - a_{\text{ref}} \\ \delta_{\text{max}} - \delta_{\text{ref}} \\ -a_{\text{min}} + a_{\text{ref}} \\ -\delta_{\text{min}} + \delta_{\text{ref}} \end{bmatrix}. \quad (5.6.12)$$

If one wishes to impose a constraint on the vehicle states $z$, it is necessary to write the constraint so that it is imposed on $\bar{u}$ since it is the optimization variable. Assume we wish to constrain the vehicle in terms of deviation from the reference path, $||\bar{z}_i|| \leq \epsilon \forall i$, we have to rewrite this in terms of $\bar{u}$. Using the fact that $\bar{z}(i+1) = [\bar{z}(i+1)||\bar{z}(i+2)||...||\bar{z}(N+1)]$ and $\bar{z}(i+1) = \bar{A}z + \bar{B}_i \bar{u}_i$ we get:

$$D\bar{z}(i+1) \leq c \Leftrightarrow D(\bar{A}z + \bar{B}_i \bar{u}_i) \leq c \Leftrightarrow D\bar{u}_i \leq c, \quad (5.6.13)$$

where $D$ is the same as in (5.6.12) and

$$c = \bar{B}_i^{-1}(\epsilon - \bar{A}_i \bar{z}_i).$$

### 5.6.6 Tuning

In standard PID control theory there are several methods for tuning the controller in order to affect its performance [17]. In PID control there are gains and tuning parameters which will effectively change its behaviour which makes it highly customizable and is partly one of the reasons why it has been in such great use in the industry [18]. In the case of MPC, which is a mathematical optimization problem, the tuning is done slightly differently and most of it comes down to the cost function and the optimization constraints [20].
Taking a look at the cost function, there are three matrices involved that are neither states nor control inputs: \( Q \), \( R \) and \( P \). Starting with the \( Q \) matrix, it is used in combination with the system states and in the cost function they together create a quadratic cost \( \tilde{z}^T Q \tilde{z} \). What \( Q \) effectively does is penalizing the solver for any deviations in the states \[{19}\]. If one wants to keep all three states equally close to their references, the weights would be chosen to be identical. However, if one or two states are more important to track than the other, the weights for these states would be greater than those for the less important. Practically this means that any deviation in the more important states will be greater penalized, leading to the MPC will try to find solutions which minimize the deviation in the important states while the deviation in the less important states is allowed to be larger. The size of the weights have to be in relation of the quantity it is penalizing. For example, penalizing meters and radians with the same weight means that one meter path deviation is equally penalized as one radian orientation deviation.

In vehicle path following, the \( x \) and \( y \) coordinates are deemed the most important states as one wants to stay on the road at all times \[{8}\]. The weights for these two states are then often the largest. The orientation of the vehicle is not as equally important since tyre slip angle and the effective heading angle of the vehicle might not be equal to the physical orientation.

The \( R \) matrix appears together with the \( \tilde{u} \) and serves as a weight matrix for the control inputs. In the tilde formulation it penalizes deviations from the reference control inputs and states. As in the case of the \( Q \) matrix, the elements of \( R \) are chosen so that the control inputs can penalized differently. Since the cost function summarises the penalties of the states and control inputs, the interaction between \( Q \) and \( R \) is of equal importance. If the elements of \( Q \) is greater than those of \( R \) it would mean that deviations in states are more important to avoid and the control strategy would be to minimize the states even if it would mean having large deviations in control inputs. Therefore, the elements of \( R \) are usually chosen to be smaller than those of \( Q \) in the lateral controller \[{12}\].

The terminal cost matrix \( P \) has a very dominant effect on the MPC, even tough it only penalizes one instance of the states, namely the terminal state \( z_{H_p} \) \[{19}\]. Since the MPC predicts over a horizon, the iterations between the initial state \( z_0 \) and the final state \( z_{H_p} \) can be seen as steps on a path from the starting point to a goal. In standard non-path following MPC, what one is interested in is not as much how it travels to the goal but rather that it reaches that point and does it well. The elements of \( P \) are almost always greater than those of \( Q \) to emphasize that the final state is more important than the path to it. In path following MPC, the path is equally important to the goal and the terminal cost serves as a stabilizing and recovering weight which is discussed more in Section\[{6.6}\].
5.6.7 Coupled Path Following MPC

In our specific case, coupled control means that there will be one MPC controller that will be responsible for two control inputs, steering angle and acceleration, at each iteration. Looking at the bicycle model shown in (4.2.1), the steering angle has a direct effect on the tire slip angle which in itself affects the lateral, longitudinal and yaw rate of the vehicle. These three together affect the position and orientation of the vehicle. The control inputs affect each other and each state of the vehicle, and the idea of letting one MPC optimize both inputs is to better predict the evolution the states of the vehicle and improve stability.

The suggested MPC formulation for coupled control will include constraints on both control inputs and the states for both the kinematic and dynamic models. The cost function of the coupled MPC will remain the same as in the lateral control presented in (5.6.5), but, with more states to track, the matrices Q, P and R will have additional elements. The terminal cost matrix P is tuned in accordance with the theory stated in Section 5.6.6 and will follow the same structure as Q but with larger elements. The R matrix for the coupled system will be \( R = \text{diag}(r_1, r_2) \) where tracking reference accelerations and steering angles are of similar importance.

For the bicycle model there are two more states to consider into the cost function, namely \( \dot{\phi} \) and \( v_y \). This adds two more elements in Q and P but they can both be ignored and set to either zero or one. The reason for this is that even though \( \dot{\phi} \) and \( v_y \) affect the handling of the vehicle, one can still perform path following without tracking these two variables. This also means that we do not need to generate any references for these states.

The control strategy used here therefore suggests that one does not focus on the evolution of \( \dot{\phi} \) and \( v_y \). One can however add constraints on these two in terms of different stability criteria. A suggested way of creating references for \( v_y \) is to set \( v_{y,t,i} = v_{y,t,i-1} \), or in words the future reference of the lateral speed is the lateral speed from previous iteration. This way, \( v_y \) is allowed to grow but very quick or sudden growths will be penalized and hopefully avoided.

5.7 Speed Profiling

With the strategies suggested in section 5.6.7 we have now introduced a second control input in form of an accelerating force which acts upon the vehicle and effectively alters its speed. To find the optimal acceleration we introduce an additional block in our system called speed profiler. The goal of this block is to produce an optimal reference speed \( v_{\text{ref}} \) for each point on the path (12). The definition of \( v_{\text{ref}} \) will from here on be equal to the maximum speed a vehicle can have without slipping on the road surface (12). To find \( v_{\text{ref}} \) one utilizes information about the road and vehicle conditions at the given time. A simple formula for determining the maximum speed on a road is given by
\[ v_{\text{max}} = \sqrt{\frac{g \mu}{\rho_r}}, \quad (5.7.1) \]

where \( g, \mu \) and \( \rho_r \) are gravity, friction coefficient and road curvature respectively. In [12] the authors use an extended formula which introduces road chamber angle, \( \phi_r \), which explains how much the road banks, or, how level the road is. For the applications of this thesis, all of the tested roads have zero bank and hence equation (5.7.1) suffices. The curvature \( \rho_r \) is defined as \( 1/r \) where \( r \) is the radius of the road curve. According to [23], the curvature can be estimated using only three points one a road, \( p_{i-1}, p_i \) and \( p_{i+1} \). Constructing two vectors \( v_1 = p_i - p_{i-1}, v_2 = p_{i+1} - p_i \) and the curvature at point \( p_i \) is given by

\[
\rho_{r,i} = \frac{2 \sin \left( \frac{\theta}{2} \right)}{\sqrt{||v_1|| \cdot ||v_2||}} \quad \text{with} \quad \theta = \arccos \left( \frac{v_1 \cdot v_2}{||v_1|| \cdot ||v_2||} \right). \quad (5.7.2)
\]

Using (5.7.1) and (5.7.2) one can now determine \( v_{\text{max}} \) for each waypoint on the path. The formula in (5.7.1) will diverge when \( \rho_r \to 0 \), and hence straight roads will be limited to a predetermined maximum value.

Using this information one can create the speed profile for any road. Using the path in Figure 5.7.1a, plotting \( v_{\text{max}} \) for each point of the path results in the speed profile in Figure 5.7.1b. Here, \( v_{\text{max}} = 10 \text{ m/s} \) on the straight segments of the path.

Using the speed profile we can feed the reference speeds to the longitudinal controller and the system tries to follow the path as well as adapt its speed during the drive. An issue with the speed profile in Figure 5.7.1b is that if the speed of the vehicle before it enters the corner, entry speed, is much larger than the maximum speed allowed inside the corner, the deceleration needed to suit the new \( v_{\text{max}} \) might be too large for the vehicle to handle. A safe limit for deceleration is set to \( a_{\text{max}} = -1 \text{ m/s}^2 \).
Instead of applying unnecessary large deceleration, it would make more sense to create a speed profile that takes the time to decelerate into account and makes sure that the vehicle is prepared for corners before they arrive at them. In summary, one would like to achieve the maximum allowed speed for a corner before entering it.

To create this improved speed profile one has to consider three special points on the path: \( p_{\text{entry}} \), \( p_{\rho_{r,\max}} \) and \( p_{\text{exit}} \). The point \( p_{\text{entry}} \) is simply the waypoint where the curve on the path begins and can be found as the first waypoint where \( \rho_r \neq 0 \). The second point \( p_{\rho_{r,\max}} \) is the point where the curvature is the highest. The last point \( p_{\text{exit}} \) is where the curve ends and the curvature \( \rho_r \to 0 \). The goal of the new profile is to have reached the \( v_{\min} \) found in \( p_{\rho_{r,\max}} \) when we arrive at \( p_{\text{entry}} \). Taking the speed the vehicle travels on the straight path before the corner, one has to calculate the distance from \( p_{\text{entry}} \) where the vehicle needs to start decelerating with \( a_{\max} \).

This is done with simple kinematic equations (5.7.3), where first one finds the time to decelerate from \( v_{\max} \) to \( v_{\min} \), \( T_a \), and then the distance \( S_a \) the vehicle needs to decelerate.

\[
T_a = \sqrt{\frac{v_{\max} - v_{\min}}{a_{\max}}} \\
S_a = v_{\max}T_a + \frac{a_{\max}T_a^2}{2}.
\]  

(5.7.3)

The new speed profile is shown in Figure 5.7.2 together with the old profile previously shown in Figure 5.7.1b. The dashed lines show where \( p_{\text{entry}} \) and \( p_{\text{exit}} \) respectively occur.

![Figure 5.7.2: Comparison between the old speed profile and the new improved one for the path in Figure 5.7.1a](image)

For the general road where there might not be any straight segments, the new speed profiling algorithm will use the old speed profile in order to generate the new one.
Knowing the waypoint spacing, one can measure the distance between the last point of maximum speed and the point with the slowest speed. From this, the speed profiler can calculate the appropriate point to start decelerating in order to bring the vehicle speed down to a minimum well before entering the road segment with lowest allowed top speed.
CHAPTER 6

SIMULATION SETUP

In this section the different controllers and the complete system is evaluated using a simulation environment in MATLAB. The primary goal is to evaluate the different controllers separately at first using different tuning options and performance measurements. In the last part of this section all controllers that have been tested on the dynamical bicycle model vehicle will be compared in a final test in order to find data which will be used in the discussion and ultimately serve as base for answering the questions posed in the problem formulation in Section 3. The complete system that will be used in the simulation and evaluation is a combination of all systems mentioned in Section 5. An overview of the full system is given in Section 6.1 and shown in Figure 6.1.1.

6.1 SYSTEM OVERVIEW

The simulation setup is shown in Figure 6.1.1. The closed loop system consists of three major blocks: path_generator(), controller() and vehicle(). The first block path_generator() will take a continuous version of the path and output a discretized version in form of a set of waypoints W. This set is stored in the computer and is constantly used through the path following. The second block controller() can be broken down into three smaller blocks: find_waypoints(), find_input() and MPC(). The first block find_waypoints() will use the current position of the vehicle and the set of waypoints in order to select the waypoints that will be used for the next iterations. The find_input() block will use these waypoints to create reference control inputs which are then passed down to the MPC() block which performs the actual optimization problem and outputs control inputs. The MPC() block uses the reference waypoints to first linearize the system around each point. Finally, it solves the MPC problem and the control inputs are fed to the block vehicle(). The updated states of the vehicle are then fed back into the controller() and the loop is closed. For the simulations with longitudinal control, the path_generator() will include a speed profile element which together with the set of waypoints store the optimal speed for the entire path. These speed profile val-
ues are then passed along with the discrete path to the controller(·) block.

![Diagram of control system](image)

Figure 6.1.1: An overview of the control system used for path following.

### 6.2 Waypointing - Creating the References

In a 2D space one can track a vehicle using two coordinates \( x, y \) and an orientation \( \theta \) which is described by the vehicle models in Section 4. To perform path following the vehicle needs reference points to follow, or waypoints. The waypoints describe a position in the 2D space as well as an orientation. A waypoint is henceforth defined as a vector \( z_{\text{ref}} \) containing at least information about \( x, y \) and \( \theta \), and any path in this space can be described using a vector of waypoints \( W \). When constructing \( W \) from a continuous path, it is built using multiple discrete waypoints \( [z_{\text{ref}}^0, z_{\text{ref}}^1, \ldots, z_{\text{ref}}^N] \).

Given an MPC horizon of \( H_p \), we would need to input at least \( H_p \) waypoints which is ordered as \( z_{\text{ref}} = [z_{\text{ref}}^0, z_{\text{ref}}^1, z_{\text{ref}}^2, \ldots, z_{\text{ref}}^{H_p-1}, z_{\text{ref}}^{H_p}] \). The waypoint \( z_{\text{ref}}^0 \) is the waypoint closest to the initial state of the vehicle \( z(0) \). This first waypoint \( z_{\text{ref}}^0 \) is defined as the first reachable waypoint when moving from the initial state \( z(0) \), which is found using \( T_s \) and \( v \). The vehicle will travel \( T_s v \) meters per sampling time, hence the euclidean distance between \( z_{\text{ref}}^0 \) and \( z_{\text{ref}}^1 \) have to equal to this. The waypoint in \( W \) that best matches this criteria is chosen as \( w_1 \). The following waypoint for the MPC horizon is chosen in the same fashion using the same speed and sampling time.

After choosing appropriate waypoints for the path following, the MPC will then try and minimize \( \tilde{z} \) for each iteration and bring the vehicle state \( z_i \) close to its respective waypoints \( z_{\text{ref}}^i \). Since the waypoints are highly dependent on the vehicle’s current state, they have to be calculated online at each sampling time. The algorithm for finding appropriate waypoints is shown in Algorithm 1.
Data: \( z(0), v, T_s, H_p, W \)

Result: Extract \( H_p \) number of waypoints

Find \( z_{\text{ref}0} \in W \) such that \( ||z(0) - z_{\text{ref}0}|| \) is minimized.

Find \( S_s \) using \( S_s = vT_s \);

while \( i \leq H_p \) do

Choose \( z_{\text{ref}i} \in W \) such that;

\[
||z_{\text{ref}i} - z_{\text{ref}i-1}|| - S_s \text{ is minimized;}
\]

\( i++ \);

end

Return \( z_{\text{ref}} = [z_{\text{ref}0}, \ldots, z_{\text{ref}H_p}] \);

**Algorithm 1:** Find waypoints

### 6.3 Simulation Path

The path where the controllers are tested on is a long > 180 degree corner that has a 30 meter straight leading into the curve and a 30 meter straight leading out of it. The road friction is set to \( \mu = 0.7 \) which is less slippery than \( \mu = 0.3 \) used in [9], but more than \( \mu = 1 \) used in [13]. The path along with its boundaries is showed in Figure 6.3.1. The maximum speed of the path is chosen to 10 m/s but is considerably lower inside the corner. The dashed line shows the center line of the road which serves the purpose of path for the path following. The width of the road is 2 meters and if a controller fails to keep the vehicle within these boundaries the controller will be given a failure status in the later simulation tables. However it is allowed to deviate from the center line and the amount it deviates will be presented in the performance tables.

### 6.4 Performance Measurement and Tuning

In this section of the thesis a thorough presentation of the different controllers that were developed is done and each one is evaluated through simulations in MATLAB. To evaluate each control strategy and their respective controllers, several performance numbers summarized in Table 6.4.1 will be used.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_{\text{sol}} )</td>
<td>MPC solver time</td>
<td>Average time to solve the full MPC problem</td>
</tr>
<tr>
<td>( u_{\text{dev}} )</td>
<td>Input deviations</td>
<td>How much does ( u^* ) deviate from ( u_{\text{ref}} )</td>
</tr>
<tr>
<td>( z_{\text{max}} )</td>
<td>Max ( \dot{z} ) deviation</td>
<td>The maximum deviation from the path</td>
</tr>
<tr>
<td>( \Delta \dot{z} )</td>
<td>Average ( \dot{z} ) deviation</td>
<td>Measures how well the vehicle is able to match the waypoints.</td>
</tr>
</tbody>
</table>
Figure 6.3.1: The corner used for the simulation together with its boundaries and the center path.

The $T_{\text{sol}}$ indicator measures how long the computer takes to solve the MPC problem. The tasks included in $T_{\text{sol}}$ are linearizing the system for each waypoint, calculating optimal speed, acceleration and steering angle, applying the longitudinal controller and/or solve the full MPC. In [6, 7, 8, 10] the authors have used a sampling time of $T_s = 0.05$ seconds, while [14] has used $T_s = 0.01$ seconds. The papers with longer sampling time usually have more complex dynamics than the results here, and hence $T_s = 0.01$ seconds will be used as a benchmark. For the MPC to be practical, $T_{\text{sol}} \leq T_s$ or else one may end up with suboptimal solutions [10].

The waypoint based indicators are used to show how well the controllers follow the reference path. The indicator $\tilde{z}_{\text{max}}$ presents the largest deviations from the path that the vehicle has done during the simulation. A $\tilde{z}_{\text{max}} \leq 0.5$ meters is deemed acceptable. The second indicator $\Delta\tilde{z}$ is a measurement of average waypoint deviation, and measured as average centimeter deviation per waypoint.

The last indicator $u_{\text{dev}}$ measures the average deviation in control input from control reference. Although not being a critical performance indicator, it shows how much the controller has to adjust the control inputs from the references in order to perform path following.
6.5 MPC Solver Evaluation

To solve the quadratic programming and MPC problems constructed in this thesis, a numerical solver tool is necessary to use. Three different solvers are tested: MATLAB and its function `quadprog`, CVXGEN QP solver and CVXGEN MPC solver. The CVXGEN solvers are created by professor Stephen Boyd and Ph.D candidate Jacob Mattingley. The QP solver is explained in their paper [24] and it can be used for multiple types of convex optimization including quadratic programming and MPC. The details for the MPC specific solver can be found in [25].

The first two solvers are quadratic programming solvers which require the user to rewrite the MPC problem into a QP problem. The third solver is a MPC solver and does not require the user to rewrite the MPC into QP which simplifies parts of the implementation. For the QP solvers, the MPC has to be written on the form given in equation (5.6.11). The aim is to optimize a sequence of control inputs \( u \). At each sampling time the system is linearised around \( z_{ref} \) and the matrices \( \bar{A} \) and \( \bar{B} \) are constructed and then fed to the solver which outputs a \( H_p \) long sequence of optimal control inputs. The first input \( u_0 \) is given to the simulation of the system which updates the vehicle position, and the future state predictions from the MPC are plotted. Table 6.5.1 collects all the information about different MATLAB simulations performed on the kinematic system. Performance numbers as well as simulation parameters such as sampling time and speed are noted.

The built-in QP solver in MATLAB is easy to use but is a relatively slow QP solver. The QP solver from CVXGEN is documented to be faster than MATLAB’s `quadprog` but requires the user to compile the solver source code each time the MPC is changed. The CVXGEN MPC tool is a solver for MPC problems which does not require having to manually rewrite the MPC into a QP. The major upside to this is that one can easily add, remove or modify constraints without having to rewrite the matrices \( D \) and \( d \) as it was done in Section 5.6.5.

The results from the solver testing are shown in Table 6.5.1 which concludes that the CVXGEN solvers are faster than MATLAB’s solver. The CVXGEN MPC solver is only slightly slower than its QP solver, but the more simple implementation has made the CVXGEN MPC solver the chosen solver for this thesis.

Table 6.5.1: Simulations using kinematic MPC on a kinematic vehicle model and cruise controller at \( v = 10 \text{ m/s} \)

<table>
<thead>
<tr>
<th>Solver</th>
<th>( \Delta \hat{z} ) [cm]</th>
<th>( \hat{z}_{max} ) [cm]</th>
<th>( T_{sol} ) [ms]</th>
<th>( T_s ) [ms]</th>
<th>( u_{dev} ) [°]</th>
</tr>
</thead>
<tbody>
<tr>
<td>QuadProg</td>
<td>0</td>
<td>0</td>
<td>10.5</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>CVXGEN QP</td>
<td>0</td>
<td>0</td>
<td>8.0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>CVXGEN MPC</td>
<td>0</td>
<td>0</td>
<td>8.8</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>
6.6 Tuning the Controllers

The tuning of the controllers have taken inspiration from [6][7][10], among many, but it is also much trial-and-error. An important note on tuning is the significance of the terminal cost matrix $P$. Given the situation that our vehicle starts outside of the path and it is now trying to get back on it. With an MPC where $P = Q$, every state in the horizon is equally important and the solution that brings the vehicle closest to the waypoints will be sought after. To improve the performance of the MPC, the matrix $P$ can be tuned differently. The elements of $P$ matrix can be greater than those of $Q$, meaning that the last waypoint to follow is more important than the previous ones. This is especially useful in situations when the vehicle has been deviating from the path already. The terminal cost will force the vehicle to steer in a way that the last waypoint is reached and the proceeding waypoints do not have to be strictly followed. In Figure 6.6.1, the vehicle is starting 0.5 meters south of the path and is trying to re-enter the path. The x and y axes are scaled differently to easier see the effect. The dashed line shows how the vehicle travels when $P = Q$ and the dotted dashed line uses $P > Q$. The latter tuning strategy reduces the oscillations and converges more quickly towards the path compared to the other tuning.

Figure 6.6.1: Comparison of using terminal cost versus not using it.
CHAPTER 7

SIMULATION RESULTS

7.1 DECOUPLED CONTROL

7.1.1 KINEMATIC DECOUPLED MPC ON KINEMATIC MODEL

The results in Section 6.5 show that a kinematic MPC with constant speed controlling a kinematic vehicle model leads to near perfect trajectory following, this section will demonstrate the principles of decoupled lateral and longitudinal control. From henceforth the CVXGEN MPC solver will be used. The longitudinal controller will be a PI-controller which is described in Section 5.3. The longitudinal controller will have control input saturations in order to use slightly more realistic dynamics in terms of breaking and acceleration. The acceleration limit, deceleration limit and controller gain is set to $a_{\text{max}} = 1 \text{ m/s}^2$, $a_{\text{min}} = -5 \text{ m/s}^2$ and $K_P = 3, K_I = 1$.

The simulation is performed using the more aggressive speed profile, while the new speed profile is tested separately in Section 7.4. The vehicle model used has four states and one control input, and the tuning parameters for lateral controller is shown in Table 7.1.1.

<table>
<thead>
<tr>
<th>$Q$</th>
<th>$R$</th>
<th>$P$</th>
<th>$\delta_{\text{max}}, \delta_{\text{min}}$</th>
<th>$\Delta\delta_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[100, 100, 1, 0]</td>
<td>10</td>
<td>[500, 500, 5, 0]</td>
<td>$\pm 50^\circ$</td>
<td>$40^\circ / s$</td>
</tr>
</tbody>
</table>

As expected with similar models in the MPC and on the simulated vehicle, one achieve great path following in all three cases. The reason that the deviations arise is that the limits on the longitudinal controller causes the speed of the vehicle to not be optimal in the corner. If one investigates the plots in Figure 7.1.1 it is possible to see that the longitudinal controller cannot react to the reference speed change without breaking the acceleration limit and is thus always lagging behind the reference. The steering angle input follows the reference pretty accurately apart from a few control
reference jumps.

![Graph](image1.png)

Figure 7.1.1: Vehicle speed and speed reference (left) and steering angle control input and input reference (right).

An important thing to notice in this simulation is that the solver time is well below the stipulated sampling time of $T_s = 0.01$ seconds. Hence, using such a small horizon of $H_p = 10$ will not fully utilize the capacity of the processor of the computer. This means that there is room to increase the complexity of the MPC which can be done by either adding constraints to the system or simply increase the horizon.

The control strategy was re-simulated with increased horizon and compared to its short horizon equivalent. The $H_p = 30$ horizon proved to have slightly too long solving time and a third tuning with $H_p = 25$ was also tested. The result and comparison is shown in Table 7.1.2.

<table>
<thead>
<tr>
<th>$H_p$</th>
<th>$\Delta z$ [cm]</th>
<th>$z_{\text{max}}$ [cm]</th>
<th>$T_{\text{sol}}$ [ms]</th>
<th>$u_{\text{dev}}$ [$^\circ$]</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.7</td>
<td>10.3</td>
<td>2.4</td>
<td>0.2</td>
</tr>
<tr>
<td>25</td>
<td>0.3</td>
<td>1.1</td>
<td>9.4</td>
<td>0.2</td>
</tr>
<tr>
<td>30</td>
<td>0.3</td>
<td>1.2</td>
<td>11.2</td>
<td>0.2</td>
</tr>
</tbody>
</table>

The increased horizon reduces the average waypoint deviation by 50% and reduces the maximum deviation from $10\text{cm}$ to roughly $1\text{ cm}$. Since $H_p = 25$ has a sufficient solver time it will be used as standard for the kinematic modelled MPC.
7.1.2 **Kinematic Decoupled MPC on Dynamic Model**

After showing how using identical models in simulations and in the MPC yields near perfect tracking with both constant speed and decoupled longitudinal controller, one would now like to investigate how the kinematic MPC controllers handle a system governed by a more complex dynamical bicycle model. The following simulations are also important in the sense that the kinematic model in (4.1.1) is a simplified version of the dynamical bicycle model in (4.2.1). An example of a simplification done is the assumption that in the kinematic equations there exists no lateral speed \( v_y \), and what we want to find out here is if the kinematic MPC can control the dynamical system. The two different models are more equal if \( v_y \) is kept low and other dynamics are within some small bounded region.

The controllers used in this simulation are the same as in Section [7.1.1] and the simulation is performed on the bicycle model presented in (4.2.1). The performance numbers of the three simulations are summarized in Table 7.1.3.

Table 7.1.3: Performance indicators for kinematic decoupled control with \( H_p = 25 \)

<table>
<thead>
<tr>
<th>( \Delta z ) [( cm )]</th>
<th>( z_{\text{max}} ) [( cm )]</th>
<th>( T_{\text{sol}} ) [( ms )]</th>
<th>( u_{\text{dev}} ) [( ^\circ )]</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4</td>
<td>35.2</td>
<td>9.6</td>
<td>0.5</td>
</tr>
</tbody>
</table>

The reference input values \( u_s \) are collected from a kinematic vehicle model and forcing the MPC to follow these references simply displays that the reference values are not optimal for the simulated vehicle. The control input deviations are larger for this setup compared to the ones in Table [7.1.2] which is expected as the MPC has to work harder in order to follow the path.

7.1.3 **Dynamic Lateral MPC on Dynamic Model**

The next MPC to investigate is the one based on the dynamic model presented in (4.2.1). Firstly, the lateral controller will be constructed and tuned for constant speed and a decoupled and coupled version will be tested. The six state vehicle model requires extra tuning parameters for the MPC as well as extended matrices \( Q, R, P \) in the cost function. Similar to the kinematic based MPC controllers, the bicycle model based MPC will track position and orientation states and the importance of these states are the same in both kinematic and dynamic cases. The major difference is that the dynamic model distinguishes the velocity from a simple \( v \) in the kinematic model to component speeds \( v_x \) and \( v_y \) in the bicycle model which are tracked separately. The change is vehicle orientation \( \dot{\phi} = v_\phi \) is now a state the MPC can work with. However, for our path following these states are not tracked but they are constrained.

When tuning the lateral MPC a couple of assumptions regarding the evolution of the states are made. The speed \( v_x \) is tracked and deviations from calculated reference
speeds are penalized. The lateral speed $v_y$ of the vehicle is necessary in order to turn the vehicle along a path, however the evolution of $v_y$ is usually not of interest and is therefore not tracked or given any reference speeds \cite{12}. The rotational velocity $v_\phi$ has similar properties to $v_y$ and will be handled in similar fashion. The tuning for the lateral controller is summarized in Table\ref{table:7.1.4}.

Table 7.1.4: Tuning and limits of dynamic lateral controller

<table>
<thead>
<tr>
<th>Q</th>
<th>R</th>
<th>P</th>
<th>$\delta_{max}, \delta_{min}$</th>
<th>$a_{max}, a_{min}$ [m/s$^2$]</th>
<th>$\Delta \delta_{max}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[100, 100, 10, 1]</td>
<td>10</td>
<td>[500, 500, 50, 5]</td>
<td>$\pm 45^\circ$</td>
<td>1, -5</td>
<td>40$^\circ$/s</td>
</tr>
</tbody>
</table>

This means in practice that the kinematic vehicle can turn at any speed without slipping sideways, and an MPC based on a kinematic model will predict that the vehicle can make any turn. This is not the case for a dynamic vehicle where lateral accelerations are a large factor when it comes to vehicle stability.

### 7.1.4 Dynamic Decoupled MPC on Dynamic Model

Using the lateral controller from Table\ref{table:7.1.4} and the longitudinal PI-controller from Section\ref{section:5.3} the corner simulation is performed and the result is presented in Table\ref{table:7.1.5}.
Table 7.1.5: Performance indicators for decoupled dynamic controller with $H_p = 25$.

<table>
<thead>
<tr>
<th>$\Delta \tilde{z}$ [cm]</th>
<th>$\tilde{z}_{\text{max}}$ [cm]</th>
<th>$T_{\text{sol}}$ [ms]</th>
<th>$u_{\text{dev}}$ [$^\circ$]</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.0</td>
<td>25.6</td>
<td>9.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

7.2 COUPLED CONTROL

7.2.1 KINEMATIC COUPLED MPC ON DYNAMIC MODEL

After tuning the lateral MPC controller for constant speed and decoupled longitudinal control, the coupled approach mentioned in Section 5.5 is tested using a lateral MPC based on the kinematic vehicle model. Using the four state kinematic model presented in Section 4.1, different controller tunings are tested and the chosen ones are presented in Table 7.2.1.

Table 7.2.1: Tuning and limits of coupled controller

<table>
<thead>
<tr>
<th>Q</th>
<th>R</th>
<th>P</th>
<th>$\delta_{\text{max}}, \delta_{\text{min}}$</th>
<th>$a_{\text{max}}, a_{\text{min}}$ [m/s$^2$]</th>
<th>$\Delta \delta_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[300,300,30,3]</td>
<td>[20,10]</td>
<td>[500,500,50,5]</td>
<td>$\pm 45^\circ$</td>
<td>1, $-5$</td>
<td>$40^7$/s</td>
</tr>
</tbody>
</table>

Table 7.2.2: Performance indicators for coupled control with $H_p = 25$

<table>
<thead>
<tr>
<th>$\Delta \tilde{z}$ [cm]</th>
<th>$\tilde{z}_{\text{max}}$ [cm]</th>
<th>$T_{\text{sol}}$ [ms]</th>
<th>$u_{\text{dev}}$ [$^\circ$]</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.5</td>
<td>34.1</td>
<td>9.9</td>
<td>0.6</td>
</tr>
</tbody>
</table>

The performance indicators for the coupled controller shown in Table 7.2.2 show that the tracking is similar to the decoupled kinematic MPC. In Figure 7.2.1 one can see the difference in speed tracking for the decoupled control setup’s PI-controller and the coupled MPC controller.

7.2.2 DYNAMIC COUPLED MPC ON DYNAMIC MODEL

Finally, we simulate the coupled longitudinal and lateral control strategy based on the dynamic bicycle model. Using the same lateral controller as in Section 7.1.4 and the coupling strategy presented in Section 5.5, the controller tuning is summarized in Table 7.2.3.

Table 7.2.3: Tuning and limits of dynamic coupled lateral controller

<table>
<thead>
<tr>
<th>Q</th>
<th>R</th>
<th>P</th>
<th>$\delta_{\text{max}}, \delta_{\text{min}}$</th>
<th>$a_{\text{max}}, a_{\text{min}}$ [m/s$^2$]</th>
<th>$\Delta \delta_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[100,100,10,1,1,1]</td>
<td>[10,10]</td>
<td>[500,500,50,5,5,5]</td>
<td>$\pm 45^\circ$</td>
<td>1, $-5$</td>
<td>$40^7$/s</td>
</tr>
</tbody>
</table>
The simulations were performed on the test corner and the result is summarized in Table 7.2.4:

Table 7.2.4: Performance indicators for coupled dynamic controller with $H_p = 25$.

<table>
<thead>
<tr>
<th>$\Delta \tilde{z}$ [cm]</th>
<th>$\tilde{z}_{\text{max}}$ [cm]</th>
<th>$T_{\text{sol}}$ [ms]</th>
<th>$\theta_{\text{dev}}$ [$^\circ$]</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.4</td>
<td>29.7</td>
<td>10.7</td>
<td>0.8</td>
</tr>
</tbody>
</table>

7.3 Lateral Stability Test

The simulations have confirmed that model matching between MPC and the simulated vehicle lead to better path following. However, having discrepancy between the MPC and the plant is still viable as shown in Section 7.1.2. If one wants optimal tracking performance one should have as small of a difference between the MPC and the plant as possible, and to test this claim a small simulation comparison will be performed. For the comparison, one lateral MPC based on the kinematic model and one lateral MPC based on the dynamic model will be used with a cruise controller keeping constant speed through the previously used test corner. The test will have a dynamic vehicle model enter and drive through the corner to see how well
they complete the corner and for what speeds the path following stops performing.

The results of the simulations are shown in table 7.3.1, and an important note is that for this test the agility of the controllers is tested and hence some comfort aspects such as tyre slip and lateral forces/speeds will most likely be larger than the constraints set in previous simulations. For this purpose, the controllers have been tuned individually for each speed in order to optimize the path following without penalizing e.g. jerking in steering angles or tyre slip.

### Table 7.3.1: Stability test during constant cornering speed

<table>
<thead>
<tr>
<th>Controller</th>
<th>( v ) [m/s]</th>
<th>( \Delta z ) [cm]</th>
<th>( z_{\text{max}} ) [cm]</th>
<th>( T_{\text{sol}} ) [ms]</th>
<th>( u_{\text{dev}} ) [°]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kinematic</td>
<td>10</td>
<td>11.8</td>
<td>52.3</td>
<td>9.8</td>
<td>0.4</td>
</tr>
<tr>
<td>Dynamic</td>
<td>10</td>
<td>14.3</td>
<td>48.9</td>
<td>9.9</td>
<td>0.4</td>
</tr>
<tr>
<td>Kinematic</td>
<td>11</td>
<td>12.6</td>
<td>56.3</td>
<td>9.9</td>
<td>0.6</td>
</tr>
<tr>
<td>Dynamic</td>
<td>11</td>
<td>15.7</td>
<td>55.1</td>
<td>9.8</td>
<td>0.6</td>
</tr>
<tr>
<td>Kinematic</td>
<td>12</td>
<td>13.3</td>
<td>59.9</td>
<td>9.8</td>
<td>0.9</td>
</tr>
<tr>
<td>Dynamic</td>
<td>12</td>
<td>17.1</td>
<td>61.9</td>
<td>9.9</td>
<td>0.9</td>
</tr>
<tr>
<td>Kinematic</td>
<td>13</td>
<td>16.7</td>
<td>63.3</td>
<td>9.9</td>
<td>2.52</td>
</tr>
<tr>
<td>Dynamic</td>
<td>13</td>
<td>19.1</td>
<td>69.2</td>
<td>9.9</td>
<td>1.5</td>
</tr>
<tr>
<td>Kinematic</td>
<td>14</td>
<td>38.3</td>
<td>98.2</td>
<td>9.9</td>
<td>3.8</td>
</tr>
<tr>
<td>Dynamic</td>
<td>14</td>
<td>20.8</td>
<td>76.8</td>
<td>9.8</td>
<td>2.1</td>
</tr>
<tr>
<td>Kinematic</td>
<td>15</td>
<td>Fails</td>
<td>Fails</td>
<td>Fails</td>
<td>Fails</td>
</tr>
<tr>
<td>Dynamic</td>
<td>15</td>
<td>22.3</td>
<td>85.6</td>
<td>9.9</td>
<td>3.0</td>
</tr>
<tr>
<td>Kinematic</td>
<td>16</td>
<td>Fails</td>
<td>Fails</td>
<td>Fails</td>
<td>Fails</td>
</tr>
<tr>
<td>Dynamic</td>
<td>16</td>
<td>Fails</td>
<td>Fails</td>
<td>Fails</td>
<td>Fails</td>
</tr>
</tbody>
</table>

Important to note about the \( z_{\text{max}} \) values is that they are much larger than in the simulations with longitudinal controllers that can vary the speed of the vehicle. The path we simulate on has an optimal speed of \( \approx 7 \) m/s which means everything above that speed will have trouble maintaining the perfect path throughout the corner. What the stability test aims to show is how well the different controllers can recuperate from large path deviations due to excessive speed. The controllers that are given the flag "Fails" did not successfully re-enter the path after the corner.

### 7.4 Speed Profile Evaluation

In Section 5.7 a new method for creating optimal speeds for a specific path was presented. The idea is to improve performance by trying to imitate the behaviour of a human driver when she approaches a corner. Instead of always driving at the maximum allowed speed, the driver would instead use a piece of the straight, or near straight, path segment to decelerate before entering the corner. The corner would then be completed using a constant speed and the acceleration out of the corner would be performed in a gentle way.
In the simulations of Chapter 7, the optimal speed has always been the maximum allowed speed for the corner. A comparison between the two different speed profiles is shown in Figure 7.4.1 and in Table 7.4.1 simulations using the best controllers of each strategy are summarized where the two different profiles are compared.

Figure 7.4.1: The old $v_{\text{max}}$ profile compared to the new profile.

Table 7.4.1: Controller performance for two different speed profiles

<table>
<thead>
<tr>
<th>Controller</th>
<th>Profile</th>
<th>$\Delta z$ [cm]</th>
<th>$\tilde{z}_{\text{max}}$ [cm]</th>
<th>$T_{\text{sol}}$ [ms]</th>
<th>$\nu_{\text{dev}}$ [$^\circ$]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kin. decoupled</td>
<td>$v_{\text{max}}$</td>
<td>12.4</td>
<td>35.2</td>
<td>9.6</td>
<td>0.5</td>
</tr>
<tr>
<td>Kin. decoupled</td>
<td>new</td>
<td>10.5</td>
<td>33.3</td>
<td>8.9</td>
<td>0.5</td>
</tr>
<tr>
<td>Kin. coupled</td>
<td>$v_{\text{max}}$</td>
<td>11.5</td>
<td>34.1</td>
<td>9.9</td>
<td>0.6</td>
</tr>
<tr>
<td>Kin. coupled</td>
<td>new</td>
<td>10.2</td>
<td>32.5</td>
<td>9.8</td>
<td>0.6</td>
</tr>
<tr>
<td>Dyn. decoupled</td>
<td>$v_{\text{max}}$</td>
<td>9.0</td>
<td>25.6</td>
<td>9.4</td>
<td>0.4</td>
</tr>
<tr>
<td>Dyn. decoupled</td>
<td>new</td>
<td>8.4</td>
<td>24.5</td>
<td>9.6</td>
<td>0.6</td>
</tr>
<tr>
<td>Dyn. coupled</td>
<td>$v_{\text{max}}$</td>
<td>11.4</td>
<td>29.7</td>
<td>10.7</td>
<td>0.8</td>
</tr>
<tr>
<td>Dyn. coupled</td>
<td>new</td>
<td>8.4</td>
<td>26.8</td>
<td>10.8</td>
<td>0.5</td>
</tr>
</tbody>
</table>
CHAPTER 8

DISCUSSION

8.1 MODEL COMPLEXITY IN MPC

A key point of the thesis is to investigate how model complexity effects the path following performance of a vehicle. It was made clear that using the same simple model in both the MPC and the plant would yield very accurate path following. When upping the complexity of the vehicle model both used in the MPC and the plant it showed that a more accurate MPC model will yield better path following than a less complex model would.

According to the simulations it was possible to use a less complex MPC model to control a dynamic vehicle model in fair conditions. This was especially evident in simulations where a longitudinal controller of some sort was used and the speed inside corners was slowed down. In Section 4 it was mentioned that the kinematic vehicle and the dynamic vehicle model are in fact similar with additions of some vehicle dynamics in the bicycle model. If the dynamic terms such as $v_\phi$ and $v_y$ are kept low, the bicycle model resembles the kinematic model even more. Therefore, using a longitudinal controller to keep down the velocities will open up for the use of a less complex vehicle model to control a complex vehicle. In the constant speed test presented Table 7.3.1 it was again shown that in more extreme situations the kinematic based MPC is not good enough and the need for a more complex model is apparent. Even tough the kinematic MPC managed to complete the turn at 14 m/s, the performance indicators from both 13 and 14 m/s show that $u_{dev}$ is considerably larger than those of the dynamic MPC. This means that the kinematic MPC needs to work a lot harder in terms of steering angles in order to maintain the path following.

8.2 LONGITUDINAL CONTROL

To control the speed of the vehicle a longitudinal controller was added and in the decoupled strategy this was a PI-controller while in the coupled strategy the acceleration of the vehicle was handled by the extended MPC. In terms of speed tracking
and control inputs they were behaving similarly as was shown in Figure 7.2.1. In these specific cases the PI-controller was slower when it came to speed tracking, however this can be fixed by designing a PID-controller with optimal tuning. However in this thesis the stability and path following was more important than the trajectory following and speed reference matching. The major advantage of having a longitudinal controller compared to travelling at constant speed is that one gets a more stable vehicle in the corners due to adapting speed.

8.3 Coupled vs. Decoupled Lateral and Longitudinal Control

Another major goal of the thesis was to design and implement a coupled lateral and longitudinal control strategy and compare its performance against a more industry standard decoupled approach. To perform the coupling an additional control input variable was added to the MPC in form of a vehicle acceleration and an online speed reference calculator was added in order to feed the MPC with reference values for speed and acceleration.

The first approach has an MPC based on the kinematic vehicle model and comparing the best performing controller in the coupled and decoupled simulation respectively one finds that their performance is very similar. The coupled strategy has an average deviation of 1 cm less than the decoupled while it also had 0.1° larger average input deviation. This means that the lateral controller of the coupled strategy had to work harder in order to keep the vehicle on the path. In terms of tuning and implementation, the coupled strategy is slightly more complicated to implement than the decoupled one.

For the coupled strategy with an MPC based on the dynamic vehicle model the implementation was a bit more cumbersome. Going from a lateral MPC based on the kinematic model to the dynamic model means adding three states. In the case of coupling one also adds a term in the equation for $v_x$ as it already exists dynamics for it in the model. If one studies Table 7.1.3 and 7.2.4 it shows that the coupled strategy is performing worse than the decoupled, despite having the same tuning for the lateral parts. Adding a speed tracking together with acceleration control makes the optimization problem much more complex and one can see how the average solver time is now almost 11 ms which is approximately 2 ms longer than the decoupled strategy solver needs. In practice this means that the coupled control strategy violates the solver constraint of 10 ms and in order to take it below 10 ms again one would need to adjust the horizon of the MPC in order to reduce the number of iteration the solver runs. One can also improve the hardware by acquiring a faster PC or improve the software and use a faster solver. The question then becomes if the coupled control strategy is much better than the decoupled that one can afford to lower the horizon without affecting its performance too much. In this case, this does not seem to be a good idea as in the simulations where we let the coupled control solver
run for longer, it still cannot outperform the decoupled strategy.

Much of this has to do with tuning of the longitudinal part of the MPC as it affects the lateral controller in a significant way, unlike in the coupled kinematic MPC where the longitudinal and lateral part were only affecting each other very slightly. When tuning and deciding the elements of the cost matrices $Q$, $R$ and $P$ one has to think of how large the elements are in respect to each other. Especially important is the relation between cost elements and the state or control input they are penalizing. If one looks at the penalty of deviations in $x$ position, the corresponding element of $Q$ has been around 100 which means a 1 cm deviation in $x$ position will contribute $100 \times 0.01 = 1$ to the total cost function. If we look at $v_x$ it has been penalized with approximately 1 which causes a 1 m/s deviation to contribute $1 \times 1 = 1$ to the cost function. The idea is then that deviations in $x$ position are more important to correct for the MPC rather than speed deviations. What has been complicated in the dynamic coupled strategy is the relationship between elements of $Q$ and $R$ where a heavy tuning on $Q$ would lead to poor control inputs and bad path following while heavy tuning on $R$ would yield better control inputs but allowed for larger deviations in states, and especially in position and orientation. The delicacy of the tuning has proven to be tricky to handle and has lead the dynamic coupled strategy to be worse performing than its decoupled counterpart.

### 8.4 Use of a More Conservative Speed Profile

After running all controller simulations it became apparent how speed affects the stability of the lateral controller and general path following performance. The first constructed speed reference generator was based on feeding the longitudinal controller with the optimal speed of the current position and let it adjust the vehicle speed accordingly. This lead to stability issues as sometimes the needed acceleration for tracking the speed was outside the allowed bounds for the controller. In worst case scenario when the longitudinal controller could not adapt the speed of the vehicle quick enough it would cause the whole vehicle to be unstable. When using the coupled strategy one could use the fact that the horizon of the MPC could be used to give the longitudinal controller a “heads up” about upcoming speed changes. However, with a horizon of 25 and a sampling time of 0.01 seconds this meant that the controller knew the speed approximately 2.5 meters ahead which in scenarios is not early enough. To bypass this problem a more conservative speed profile is introduced and the simulations in Table 7.4.1 show how the new profile improves the overall path following performance, and sometimes very drastically. A hidden benefit of the conservative speed profile is that the corners are completed with constant speed, which in the case of the coupled controllers meant that the MPC was able to optimize only the steering angle in the corners and only the velocity on the straights. By using this speed profile, one avoids the problem of constructing $R$ and trying to balance the relation between the two elements. In cases where the elements were unbalanced, the MPC would sometimes return suboptimal values for
one of the control inputs. In practice, this lead to that one could have good steering or good acceleration, but not both at the same time.

8.5 Tuning Strategies

Tuning the MPC based on the kinematic model is rather straight-forward in terms of increasing and decreasing weights. If a reference state was not followed well enough, one could increase the weight on said state and see an immediate improvement. The MPC based dynamic model was slightly more complicated as it had more dynamics and increasing one weight in the cost matrices could affect another state reference as well. For the dynamic MPC, the relation between the weights was equally important as the size of individual weights. An interesting finding was that the path following would perform worse when the elements of $Q$ and $P$ were too large. In one simulation on a dynamic lateral MPC, two new tunings were tested: controller 2 and controller 3. The controllers had the same $R$ matrices, but the elements of $P$ and $Q$ in controller 2 was three times larger than those of controller 3. In Figure 8.5.1 the path taken and the steering angle inputs from controller 2 and 3 is shown. From the data one can see how the controller 2 deviates much from reference steering angle and swivels along the path. Compared to controller 3 where the elements of $P$ and $Q$ is smaller, the steering angle deviations and the path deviations are reduced as shown in figure 8.5.1a and the steering angle follows the reference input closer as seen in Figure 8.5.1b. The results show that simply increasing the weights will not necessarily improve performance.

![Figure 8.5.1](image)

(a) Comparison of path taken.  
(b) Difference in control input.

Figure 8.5.1: Comparison between controller 2 and controller 3

8.6 Comments on the Complete System

For this autonomous vehicle system we introduced four different control strategies: coupled and decoupled control based on kinematic and dynamic vehicle models.
The decoupled strategies with PI-controller for longitudinal control were the easiest to implement where the MPC problem only focused on lateral dynamics. The kinematic MPC was able to control a dynamic vehicle model but the dynamic MPC performed better than the kinematic. The coupled control strategy performed very well, especially with the more conservative speed profile. However, tuning and setting up the coupled control is significantly more cumbersome compared to the decoupled strategy. In Figure 8.6.1 the four control strategies have been evaluated in a "triangle-form". The black dot represents how the control strategies is relative to the three corners of the triangle. The corners are: easy setup, which means how easy it is to tune and use the strategy, Time, which describes how fast the control strategy can be solved by an MPC solver, and performance, which accounts for path and control deviations. Note that all four control strategies did perform good path following, and the performance indicator for one strategy is relative to the others. The kinematic strategies are closer to time and easy setup while the dynamic strategies are closer to performance but farther away from easy setup. The coupled control strategies are as well more difficult to tune and implement than their decoupled counterpart.

Figure 8.6.1: Evaluation of the four control strategies for an autonomous vehicle.
CHAPTER 9

CONCLUSION

This work has demonstrated the effectiveness of an MPC based path and trajectory following control for autonomous vehicles. Stable and well performing MPC lateral controllers have been developed and tested and served as the base for developing the whole autonomous system. The lateral MPC controllers have performed well in the tasks of path and trajectory following and due to the great customization ability of the controller. The used tuning strategies have given stable controllers which also have proven to be robust enough to allow for model discrepancies. This work has seen that MPC controllers based on simplified models can still work on more complex vehicle models which has allowed for fast solver times as well as easy implementation. Using more advanced vehicle models as base for the MPC does yield better performance in terms of stability and tracking, but in many circumstances the basic MPC is performing well enough that the extra solver time and added complexity of more advanced MPC might in some cases not be needed.

The developed coupled control strategy for lateral and longitudinal control has performed slightly better than the decoupled strategy. The advantage of having one coupled control system with one complex MPC is the small performance improvement as well as overall system simplicity in only having one controller for the vehicle. The set-up and tuning of the coupled strategy is however more complex than the decoupled strategy and the risk of suboptimal control inputs due to incorrect tuning has proven to be a drawback.

In terms of creating a complete system for autonomous vehicles there has been issues of performance. The functions for finding reference inputs in both speed, acceleration and steering angle are all basic and static functions. The performance of the complete vehicle control system is held back by its weakest link. If the speed reference function for instance fails and outputs poor reference values, there is no guarantee that any of the control strategies would keep the vehicle on the road. The focus of this paper has been on developing and tuning the lateral and longitudinal controllers, but experience from the implementation and simulations have shown that equal amount of attention should be given to the actual path and the informa-
tion the vehicle can extract from it. The improved speed profile is a result of this insight and the increased complexity of the new speed profiler showed great results in terms of performance and vehicle stability.

The most complex, non-control related task is the waypointing and path finding. In this thesis there was only one available path but the vehicle had to work out which waypoints to follow. The solution for this was to work with a discretization of the path and then run an algorithm to solve for the appropriate waypoints. The major drawback of this approach was the size of the discretized path. The path in which the simulations were made consisted of thousands of available waypoints and running the algorithm in its current state was slower than the MPC solver. In the simulations the waypointing was excluded from all time measurements with the reasoning that a well-written and optimized waypoint finding algorithm would not be as resource heavy as the current version. The topics of path finding and waypointing were too great to be researched properly in this thesis and focus was hence laid on lateral and longitudinal control.
CHAPTER 10

FUTURE WORK

IMPLEMENTATION

The next step for testing the work of this thesis is to implement the controllers on either downscaled model vehicles or a full size one. This will include optimizing other parts of the control such as reference generating, path finding and waypointing to make it work in real time.

TUNING

The main tuning strategy for all controllers have been trial-and-error which was very time consuming for the more complex MPC controllers. Finding a method for tuning and optimizing the MPC is also a field one can investigate further.

WAYPOINTING AND PATH FINDING

Finally, a large topic which was not covered in depth was path finding and waypointing. Developing smart and efficient algorithms that work well with the full autonomous system would be interesting as some results have pointed to that the quality of the information fed to the controllers have a drastic effect on performance. Creating a path finder that can replan a route online and perhaps avoid obstacles would improve the overall performance of the suggested control strategies.
BIBLIOGRAPHY


