CHALMERS

Automotive Path Following using Model Predictive Control

Master's Thesis in Systems, Control and Mechatronics

IDA PETERSSON
JOHANNA RISÖ

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2014

REPORT NO. EX018/2014

Automotive Path Following using Model Predictive Control

IDA PETERSSON
JOHANNA RISÖ

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2014

Automotive Path Following using Model Predictive Control
IDA PETERSSON
JOHANNA RISÖ

Technical report no. EX018/2014
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone +46 (0)31-772 1000

Cover: Volvo V40 test vehicle at Hällered Proving Ground. Photo: Johanna Risö.

Chalmers Reproservice
Göteborg, Sweden 2014

Automotive Path Following using Model Predictive Control
IDA PETERSSON and JOHANNA RISÖ
Department of Signals and Systems
Chalmers University of Technology

Abstract

The field of autonomous driving is rapidly expanding and extensive research is conducted in this field. Many of the autonomous functions may be simplified to the desire to follow a path, among them steer assist, self-parking and collision avoidance. The general idea is that an autonomous vehicle would be able to do the maneuver faster, safer and more accurate than a human driver.

Model Predictive Control (MPC) utilizes a model of the system and therefore has detailed knowledge of the system's future behavior. Based on this knowledge, an optimization problem can be solved to obtain an optimal input corresponding to a desired behavior. MPC can handle multiple variables and natively offers possibilities for performance trade-offs and actuator limitations. It can operate closer to constraints than conventional controllers. Since path following is a complex problem with multiple variables MPC is considered an attractive approach.

Throughout this thesis an MPC framework for automotive path following is developed, which is suitable for both simulation and vehicle. The project considered two applications with different dynamics used; a parking maneuver requiring high precision at low speed and an evasive maneuver with high dynamics at high speed.

The framework consists of functions for pretreating the path and packaging it for the MPC controller, and the MPC controller itself. To validate the framework the stability of the discretization and the accuracy of the internal models are evaluated. The MPC framework is compared to a conventional controller (P-controller) using certain performance criteria such as positioning and comfort.

It is concluded that it is possible to develop an MPC framework suitable for both low and high speed applications with the internal model as the only difference. With the methodology used in this project the MPC performance is clearly more satisfactory for a low speed case. The P-controller has better precision in high speed simulations, however, the MPC manages trade-offs for varying speeds better. There is a lack of coverage between the speeds 10 and 40 km/h. The MPC framework was successfully implemented in a Volvo V40.

Keywords: Path following, MPC, Vehicle tests, Automotive

Acknowledgments

We would like to thank our supervisors Mats Jonasson and Mikael Thor who have contributed with many advices and much enthusiasm throughout this project. We've really appreciated your support and commitment to our thesis.

We would also like to thank Bo Egardt for being our examiner.

Contents

1 Introduction

The field of autonomous driving is rapidly expanding and today almost all major car manufacturers are conducting research in this field. *Autonomous* means that the vehicle acts independently, hence without communication with other vehicles or infrastructure (Parent 2013). According to the U.S. Department of Transportation's National Highway Traffic Safety Administration (NHTSA), there are five levels of autonomy for a vehicle. At the lowest level, 0, the driver has sole control of brake, steering, throttle and motive power. At the highest level, 4, the vehicle performs safety-critical driving functions and monitors roadway conditions for an entire trip, hence, destination or navigation input is required (NHTSA 2013).

A commonly known autonomous system is the Cruise Control (CC) which maintains the vehicle at a certain set speed. Furthermore, there is also the more complex "Adaptive Cruise Control" (ACC) which not only maintains the set speed but also keeps a safe distance to the vehicle in front of the car. ACC is an example of "Combined Function Automation", level 2 (NHTSA 2013). Future generations of this system will also include steering assistance which will help the driver stay within its current lane (Volvo Cars 2013*a*). Another example of an existing system is the "Park Assist Pilot" which senses if the parking lot is large enough and steers while guiding the driver to control gears, acceleration and brakes (Volvo Cars 2013*b*).

Many of the autonomous functions may be simplified to the desire to follow a precalculated path, among them steer assist, self-parking and collision avoidance. The general idea is that an autonomous vehicle would be able to do the maneuver both faster and more accurate than a human driver.

Apart from the advantage of a response time much faster than a human's, an autonomous system also has access to controls that a human driver does not, such as applying brake torque to each wheel independently. The ability to accurately follow paths could lead to less accidents and more productive commute times, since the driver will be able to work during the commute. The precision of the parking maneuver could also lead to smaller parking spaces.

Many different control strategies can be applied to make a vehicle follow a path. A rather straightforward way would be to use a proportional controller on the deviation between the direction of the vehicle and the road. This, however, may have negative properties of either oscillations or corner cuttings.

A more sophisticated method is called Model Predictive Control (MPC) and is the one investigated in this project. MPC utilizes a model of the system and therefore has detailed knowledge of the system's future behavior. Based on this knowledge an optimal input can be obtained corresponding to a desired behavior. MPC natively offers possibilities to handle performance trade-offs, control problems with multiple variables and actuator limitations. It also operates closer to constraints than a conventional controller.

MPC has been used for this project to evaluate its possibilities for automotive path following both in simulation and vehicle. The project has been performed at Volvo Cars, Gothenburg, and the resulting MPC has been implemented and tested in a Volvo V40 provided by the company.

1.1 A Look in the Rear-View Mirror

The vision of vehicles that automatically follow a path has been around since at least the late 1930s. In 1939 General Motors (GM) presented their vision of the 1960s during their Futurama exhibit at the New York World's Fair. One part of the prediction was an automated highway system where the distance between the vehicles and their lateral positions would be maintained automatically. This was years before computers or even transistors were available.

However, by 1940 most major manufactures were producing products for the military due to World War II (Wetmore 2003) and such projects were postponed. During this time new electrical technologies and radar were developed. With these new technologies GM returned to the automated highway project in the 1950s in cooperation with Radio Corporation of America (RCA). They had by 1953 successfully built a miniature car, guided by wires on the floor. The prototype were used for experiments on how to steer and maintain proper following distance by using electronics (Wetmore 2003).

The first demonstration of a full-size passenger car driving without a driver's hands on the steering wheel occurred in 1958. A 1958 Chevrolet equipped with magnetic pickup coils followed an electric cable by sensing the alternating current (Wetmore 2003). At this time, almost all projects required changes of infrastructure.

The Eureka Prometheus project was initiated in 1987 and was a "program for a European traffic system with high efficiency and unprecedented safety" according to (EUREKA 1995). The project spanned over 96 months and had 11 participating countries, among them Sweden. A part of the project was autonomous driving and by 1994 test vehicles drove some 1000 km in Paris' metro area and the next year a test vehicle drove over 1678 km from Munich to Copenhagen and back. This was the largest project for autonomous driving ever and had an actual cost of over 749.0 M€(EUREKA 1995).

Defense Advanced Research Projects Agency (DARPA) founded the "DARPA Grand Challenge" in 2004. The objective for DARPA was to further the development of unmanned vehicles. The first year none of the competitors managed to follow the track within the 10 hour limit. In 2005, Stanford University participated with "Stanley". It finished the track without a scratch, in a winning time of 6 hours and 53 minutes giving a reward of $2 million (Russell 2006). Stanley used what was later called the "Stanley Method". This is a nonlinear feedback method utilizing a geometrical vehicle model (Snider 2009).

Today most of the major car manufacturers conduct research of autonomous driving. Maybe the most known autonomous car on the road is the one developed by Google which has been tested on public roads since 2010.

Volvo Cars has announced that their first autonomous steering feature, Adaptive Cruse Control with Steer Assist, will be introduced by the end of 2014. It will automatically follow the vehicle ahead in queues. Volvo Cars have also initiated a large scale project called "Drive Me", where 100 customers will join the project testing 100 self-driving cars on selected public roads in and around Gothenburg, Sweden (Volvo Cars 2013*c*). The first cars are expected on the roads by 2017 and "Drive Me" is the world's first large-scale autonomous driving pilot project.

1.2 Complete System for Vehicle Control

The path following controller is one of multiple subsystems needed to follow a path. The other related subsystems are used to collect information about the surroundings, generate the path and apply control signals. The function architecture describes which signals are assumed to be sent to the path following controller and which signals are required to come out. Figure 1.1 illustrates the structure. The bold red box is the path following controller, in this case the developed MPC controller.

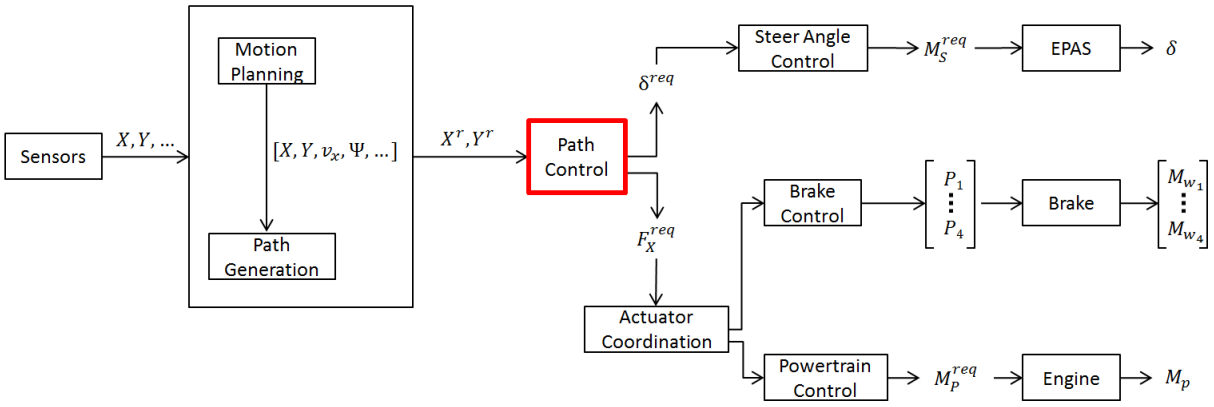


Figure 1.1. Function Architecture. The bold red box is the developed MPC controller.

Information about the actual states and surroundings are collected by the sensors. The motion planning and path generation uses this information to generate a path to follow, in a certain speed. The path is formulated in terms of global references for X and Y and is sent into the path following controller. Inside the path following controller, a wheel angle (δ) and longitudinal force (F_x) are determined and sent out as requests. This is in the project done with MPC. The δ^{req} is sent to the steer angle control while the F_x^{req} is sent to the actuator coordination. The steer angle control determines a requested steering torque (M_s^{req}) to send to the Electric Power Assisted Steering (EPAS) actuator. This sets the actual δ.

The actuator coordination determines if F_x^{req} is for brake torque or power torque. If braking is required the brake control determines how to divide the force between the wheels ($P_1, ..., P_4$) and the brake unit applies the wheel torques ($M_{w_1}, ..., M_{w_4}$). If a power torque is requested, the powertrain control determines a requested engine torque (M_p^{req}) and sends it to the engine, which applies actual power torque (M_p).

To summarize; the path following controller, henceforth referred to as the MPC controller, transforms a reference trajectory of a path to a requested steering angle and longitudinal force.

1.3 Problem Formulation

The aim for this project is to develop an MPC framework for automotive path following, which is suitable for both simulation and vehicle. Apart from the actual MPC controller the framework also consists of functions which reformulates data in order to fit the MPC controller. The framework will be evaluated both by simulation and in a vehicle at Volvo Cars. For the evaluation two path following applications with different dynamics will be used; a parking maneuver requiring high precision at low speed and an evasive maneuver with high dynamics at high speed. Finally, the performance of the MPC will be evaluated in a test vehicle.

Since MPC is a complex method extensive tuning is necessary. This will be carried out for certain performance measurements in order to make a comparison between the MPC controller and a P-controller. This comparison is done to conclude whether any performance can be gained from using MPC instead of traditional control methods. Although a P-controller is efficient when used on specific tasks, it is difficult to tune for a wider range. A P-controller may oscillate, overshoot or cut corners when operating close to constraints or when poorly tuned.

The problem formulation can be summarized in the following three research questions:

- Is it possible to develop an MPC framework suitable both for low and high speed applications?
- Does MPC have a better path following performance than a P-controller?
- Is it possible to implement the MPC framework to execute in a test vehicle?

1.4 Limitations and Assumptions

The scope consists of path following for a given path in a coordinate system. All parameters for the vehicle and its wheels are assumed to be known, which means that neither parameter nor state estimation will be part of the project.

The project consists of the following four parts:

- Development of MPC framework
- Tuning of the MPC controller
- Comparison of MPC controller and P-controller
- Implementation of MPC controller in vehicle

The vehicle powertrain, individual braking torques for the wheels and steering wheel angle are assumed to be possible to control. The model will be constructed such that only a horizontal surface is considered, making it possible to exclude the effects of slopes and banked motions.

To be able to assess the MPC performance, a reference regulator will be constructed for both car and simulation. The project is restricted to two applications, high precision maneuvers at low speed and highly dynamic maneuvers at high speed.

2 Notation

Capital Letters

A	front area of vehicle [m^2]
$\mathbf{A}$	state matrix
$\mathbf{B}$	input matrix
$\mathbf{C}$	output matrix
C	cornering stiffness coefficient [N/rad]
F	force [N]
H_p	prediction horizon
I_z	moment of inertia (around the vertical axis in CoG) [kgm^2]
K	rolling resistance coefficient
L	length between front and rear axle [m]
M_p	power torque [Nm]
M_s	steering torque [Nm]
M_w	wheel torque [Nm]
N	number of samples
P	brake torque [Nm]
P_c	comfort performance [cm/s^3]
P_d	maximum deviation performance [cm]
P_l	lateral deviation performance [cm]
P_p	positioning performance [cm]
$\mathbf{Q}$	weight matrix of state deviations
Q^d	diagonal values of weight matrix of state deviations
R	length between rear axle and center of rotation [m]
$\mathbf{R}$	weight matrix of input deviations
R^d	diagonal values of weight matrix of input deviations
T	sampling interval [s]
X	global longitudinal position [m]
Y	global lateral position [m]

Small Letters

a	acceleration [m/s^2]
d_p	preview distance [m]
g	gravitational acceleration [m/s^2]
i	index of prediction horizon
k	index of vector
l	distance from front or rear axle to CoG [m]
m	mass [kg]
$\mathbf{u}$	input vector
v	velocity [m/s]
$\mathbf{x}$	state vector
$\mathbf{y}$	output vector

Greek Letters

α	slip angle (of wheel) [rad]
β	slip angle (of body) [rad]
γ	angle between ARC and AVD [rad]
Δ	change of rate
δ	steering angle on front wheels [rad]
κ	streamline profile of a vehicle
ρ	density of air [kgm^3]
Ψ	yaw angle [rad]
ω	yaw rate [rad/s]

Subscripts

f	front
HS	for high speed
LS	for low speed
r	rear
x	longitudinal
y	lateral

Superscripts

n	nominal value
r	reference
req	requested

Diacritical marks

*	optimum
$\dot{}$	derivative
$\hat{}$	estimate
$\tilde{}$	deviation
$\bar{}$	Jacobian

Abbreviations

ACC	Adaptive Cruise Control
ARC	Along Road Centerline
AVD	Along Vehicle Direction
CC	Cruise Control
CoG	Center of Gravity
DARPA	Defense Advanced Research Projects Agency
EPAS	Electric Power Assisted Steering
GM	General Motors
HS1	Test track 1 for High Speed applications
HS2	Test track 2 for High Speed applications
KKT	Karush-Kuhn-Tucker
LP	Linear Programming
LS1	Test track 1 for Low Speed applications
LS2	Test track 2 for Low Speed applications
MPC	Model Predictive Control
NHTSA	Transportation's National Highway Traffic Safety Administration
QP	Quadratic Programming
RCA	Radio Corporation of America
SfB	S-function Builder

3 Model Predictive Control

Model Predictive Control (MPC) requires heavy calculations and has previously mainly been applied to processes with relatively low update rates, such as petrochemical industries. In recent years computers have become increasingly faster, making MPC available in other areas (Maciejowski 2002). It has the ability to handle control problems with multiple variables. It also handles actuator limitations well and can operate closer to constraints than conventional controllers, e.g. P-controllers (Maciejowski 2002). Since path following involves multiple variables and evasive maneuvers include operations close to the constraints, MPC is an attractive method.

MPC is a term for a wide range of methods which make explicit use of a model of a system to obtain a desired input signal ($\mathbf{u}$) (Camacho and Bordons 2004). The model is referred to as the *internal model* and is used to predict future behavior of the system. In this case the internal model is a vehicle model. Physical attributes such as positioning (X, Y) and velocity (v) are called states ($\mathbf{x}$).

The idea of MPC is to calculate an optimal input ($\mathbf{u}^*$) to the system for a predefined *prediction horizon*, leading to the best calculated future output ($\mathbf{y}$). This means that an optimization problem has to be solved online, that is, in real-time at every time step (Maciejowski 2002).

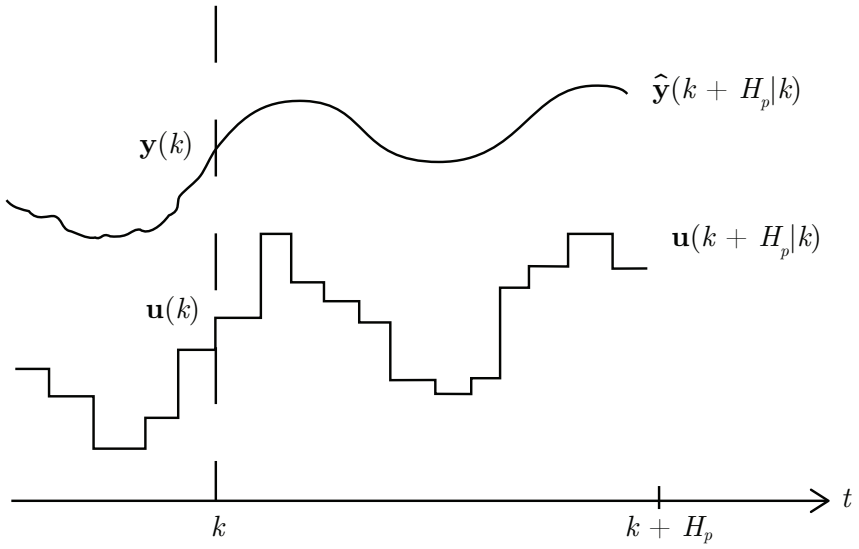


Figure 3.1. Scheme of a discrete MPC.

The number of predicted output samples is called the prediction horizon (H_p), visualized in Figure 3.1. By increasing H_p more of the dynamics of the internal model is taken into account. A long H_p is usually preferable but may not always contribute to a better estimation of the control signal depending of the dynamics of the system. A long H_p is, however, much more computationally demanding.

The future behavior of the process is computed for H_p in each time step which means that after $\mathbf{u}(k)$ has been applied, new measurements $\mathbf{y}(k+1)$ is obtained. This is used to optimize a new input trajectory, $\mathbf{u}^*(k+i|k+1)$ for $i = 0, 1, ..., H_p - 1$. In this way the length of the prediction will remain the same and H_p will move further into the future for each time step. This is referred to as a *receding horizon* (Maciejowski 2002).

Once the optimal input trajectory has been computed, usually only the first value is actually applied to the process. Because of the receding horizon the MPC is able to handle constraints on both $\mathbf{x}$ and $\mathbf{u}$ (Kwon and Han 2005). It is possible to use different lengths of the control horizon and the prediction horizon. However, for this project they have been kept the same.

3.1 Prediction

A linear system may be formulated on a discrete state space form. Assuming that $\mathbf{u}$ is known, that the model is time invariant and that there are no or small external disturbances it has the following appearance,

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \tag{3.1}$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) \tag{3.2}$$

where $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are the so called state, input and output matrices respectively. To predict the values of the following samples the notation $\hat{\mathbf{x}}(k+i|k)$ is used to represent the estimated value of $\mathbf{x}$ in sample $k+i$ using the information available at sample k:

$$\begin{aligned}
\hat{\mathbf{x}}(k+1|k) &= \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \\
\hat{\mathbf{x}}(k+2|k) &= \mathbf{A}\hat{\mathbf{x}}(k+1|k) + \mathbf{B}\mathbf{u}(k+1) \\
&= \mathbf{A}(\mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k)) + \mathbf{B}\mathbf{u}(k+1) \\
&= \mathbf{A}^2\mathbf{x}(k) + \mathbf{A}\mathbf{B}\mathbf{u}(k) + \mathbf{B}\mathbf{u}(k+1) \\
&\vdots \\
\hat{\mathbf{x}}(k+H_p|k) &= \mathbf{A}^{H_p}\mathbf{x}(k) + \mathbf{A}^{H_p-1}\mathbf{B}\mathbf{u}(k) + \mathbf{A}^{H_p-2}\mathbf{B}\mathbf{u}(k+1) \\
&\quad + \ldots + \mathbf{B}\mathbf{u}(k+H_p-1)
\end{aligned} \tag{3.3}$$

The same assumptions can be done to predict future outputs ($\hat{\mathbf{y}}$) using eq 3.2:

$$\begin{aligned}
\hat{\mathbf{y}}(k) &= \mathbf{C}\mathbf{x}(k) \\
\hat{\mathbf{y}}(k+1) &= \mathbf{C}\hat{\mathbf{x}}(k+1|k) \\
&\vdots \\
\hat{\mathbf{y}}(k+H_p) &= \mathbf{C}\hat{\mathbf{x}}(k+H_p|k)
\end{aligned} \tag{3.4}$$

Using eq 3.3 in eq 3.4 the following linear equation can be formulated for $\hat{\mathbf{y}}$:

$$\begin{aligned}
\hat{\mathbf{y}}(k) &= \mathbf{C}\mathbf{x}(k) \\
\hat{\mathbf{y}}(k+1) &= \mathbf{C}(\mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \\
&\vdots \\
\hat{\mathbf{y}}(k+H_p|k) &= \mathbf{C}\mathbf{A}^{H_p-1}\mathbf{x}(k) + \mathbf{C}\mathbf{A}^{H_p-2}\mathbf{B}\mathbf{u}(k) + \mathbf{C}\mathbf{A}^{H_p-3}\mathbf{B}\mathbf{u}(k+1) \\
&\quad + \ldots + \mathbf{C}\mathbf{B}\mathbf{u}(k+H_p-2)
\end{aligned} \tag{3.5}$$

3.2 Optimization

MPC uses an internal model to optimize future **u** to the system. The optimization is done by minimizing a so called cost function. A cost function is a mathematical function which associates a certain cost to a certain action. In this case, the deviation between the desired behavior of the system and the actual behavior of the system is minimized resulting in a **u**. Simultaneously the system has to satisfy constraints. This type of optimization problem is generally difficult to solve. However, if the problem is convex a local minimum is always a global minimum meaning the search can be terminated when a minimum is found. A convex optimization problem has the form

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \quad & g_j(x) \leq 0,\ j = 1, \ldots, l \\ & Ax = b \end{aligned}$$

where $f(x)$ is the cost function and the constraints consist of $g_j(x)$ and Ax =b. These must all be convex functions. A convex function is defined as $\forall x_1, x_2 \in \mathbb{X}, \forall \theta \in [0,1]$: $f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta) f(x_2)$. Here x defines the optimization variable which should not be confused with the notation for states, **x**.

There are two common types of convex optimization problems; linear programming (LP) and quadratic programming (QP). The difference between these lie in the formulation of the cost function which is linear respectively quadratic. There are ways of formulating the cost function of an MPC problem in both ways. However, by casting the MPC problem to a QP-problem it can be solved by numerically robust solvers (Kühne *et al.* 2004). A QP-problem is formulated as:

$$\begin{aligned} \min_{x} \quad & x^T Q x + p^T x \\ \text{subject to} \quad & G(x) \leq h \\ & Ax = b \end{aligned}$$

The cost function for an MPC problem can therefore be formulated as:

$$\min \sum_{k=1}^{T} \mathbf{x}(k)^T \mathbf{Q} \mathbf{x}(k) + \mathbf{u}(k-1)^T \mathbf{R} \mathbf{u}(k-1) \tag{3.6}$$

where **Q** and **R** are so called weight matrices, that is, diagonal matrices with weights to penalize the corresponding **x** and **u** respectively.

There are multiple methods to solve QP-problems and solvers are often specialized and therefore use different methods. One such solver is MATLAB's `quadprog` which uses many different methods. Another solver is generated by CVXGEN which uses so called Karush-Kuhn-Tucker conditions to find an optimal solution (Mattingley and Boyd 2011). CVXGEN is described further in section 4.3.2.

3.3 Internal Models for MPC - Vehicle Models

For this project vehicle models are used as internal models for the MPC. A vehicle moving at low speed does not behave similarly to a vehicle moving at high speed. Therefore, two different vehicle models are used to describe the two cases' differing dynamics. When modeling a vehicle such as a car, it is initially preferable to only consider one front wheel and one rear wheel, that is, modeling the vehicle like a bicycle.

A vehicle's position is measured in two different coordinate systems: a global frame and a local vehicle frame. In order to translate a position from the vehicle frame to the global frame the following relation is used:

$$\begin{aligned} \dot{X} &= v_x \cos(\Psi) - v_y \sin(\Psi) \\ \dot{Y} &= v_x \sin(\Psi) + v_y \cos(\Psi) \end{aligned} \tag{3.7}$$

Here, X and Y represents the global position while v_x and v_y defines the longitudinal and lateral velocities of the vehicle in the local frame. Ψ is the so called yaw angle which is the directional angle of the vehicle. This transformation is visualized in Figure 3.2.

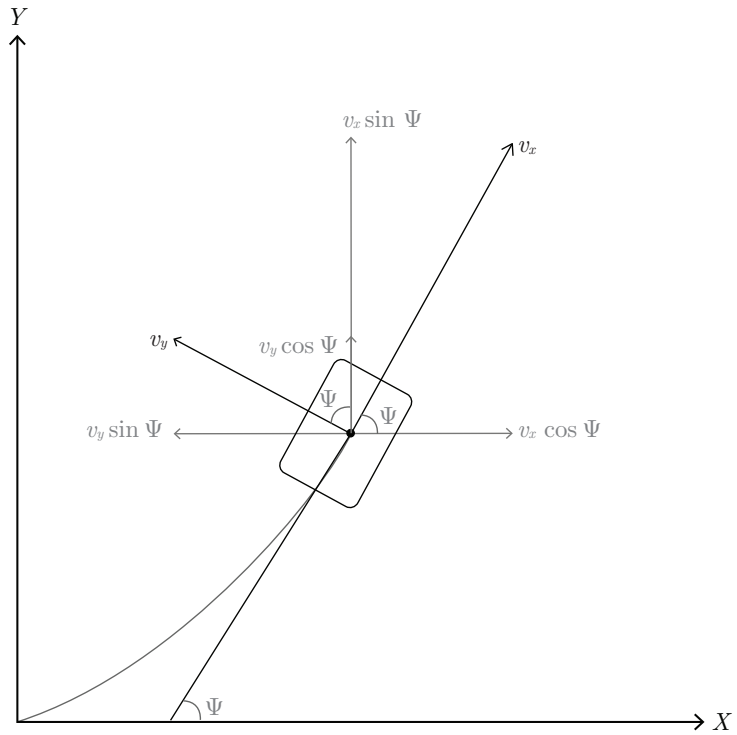


Figure 3.2. Change of position from the vehicle frame to the global frame.

3.3.1 Vehicle Model for High Speed

To achieve a model which represents a vehicle moving at high speed, dynamics can be introduced into the bicycle model. This is done by modeling the vehicle's movement in both the longitudinal and the lateral directions. The rotational motion of the vehicle is also included.

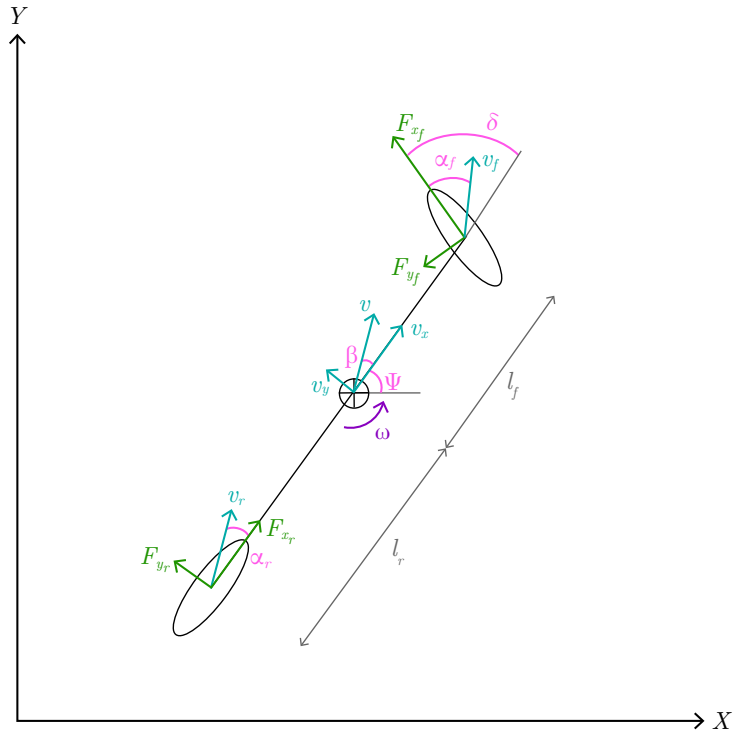


Figure 3.3. Dynamics of a high speed model which is used as an internal model for the MPC.

In Figure 3.3 it can be seen that there are both a longitudinal and a lateral force on the front wheel and on the rear wheel. The total longitudinal force (F_x) of the vehicle is for simplicity divided equally between the two wheels. The vehicle model permits different lateral forces (F_y) on the two wheels to be generated. The time derivative of the velocity at the center of gravity (CoG) of the vehicle is generated from three components; the longitudinal force on the rear wheel (F_{x_r}), the longitudinal component of the forces on the front wheel (F_{x_f}) and a contribution due to the yaw rate ($\omega = \dot{\Psi}$). By defining m as the mass of the vehicle and δ as the angle of the front wheel this adds up to the following relationship:

$$m\dot{v}_x = \frac{F_x}{2} + \frac{F_x}{2}\cos(\delta) - F_{y_f}\sin(\delta) + m\omega v_y \tag{3.8}$$

The lateral acceleration of the vehicle similarly consists of the lateral force on the rear wheel (F_{y_r}), the lateral components of the forces on the front wheel (F_{y_f}) and some contribution from ω which gives the following relationship:

$$m\dot{v}_y = F_{y_r} + F_{y_f}\cos(\delta) + \frac{F_x\sin(\delta)}{2} - m\omega v_x \tag{3.9}$$

Both F_{x_f} and F_{y_f} contribute to the yaw acceleration ($\dot{\omega}$) like levers. F_{x_r} does not contribute to any rotational movement. However, F_{y_r} will counteract on the motion from the front wheel forces and has to be considered. This leads to the following relationship

$$I_z\dot{\omega} = \frac{l_f F_x \sin(\delta)}{2} + l_f F_{y_f}\cos\delta - l_r F_{y_r} \tag{3.10}$$

where l_f and l_r are the distances between the CoG and the front respectively rear axles and I_z is the moment of inertia around a vertical axis in CoG. For simplicity the lateral forces are modeled as

$$\begin{aligned} F_{y_f} &= C_f\alpha_f \\ F_{y_r} &= C_r\alpha_r \end{aligned} \tag{3.11}$$

$$\begin{aligned} \alpha_f &= \delta - \arctan\left(\frac{v_y + l_f\omega}{v_x}\right) \approx \left(\delta - \frac{v_y + l_f\omega}{v_x}\right) \\ \alpha_r &= \arctan\left(-\frac{v_y - l_r\omega}{v_x}\right) \approx -\left(\frac{v_y - l_r\omega}{v_x}\right) \end{aligned} \tag{3.12}$$

where, C_f, C_r and α_f, α_r are the cornering stiffness coefficients respectively the slip angles for the front and rear axles. In eq 3.12 small angle approximations have been used since δ is assumed to be small at high speed and $v_y \ll v_x$. Based on eq 3.8, 3.9 and 3.10 together with eq 3.12 and 3.7 the following vehicle model for high speed (model$_{HS}$) is formed:

$$\begin{aligned} \dot{X} &= v_x\cos(\Psi) - v_y\sin(\Psi) \\ \dot{Y} &= v_x\sin(\Psi) + v_y\cos(\Psi) \\ \dot{\Psi} &= \omega \\ \dot{v}_x &= \frac{F_x(\cos(\delta)+1)}{2m} - \frac{C_f\left(\delta - \frac{v_y + l_f\omega}{v_x}\right)\sin(\delta)}{m} + \omega v_y \\ \dot{v}_y &= \frac{F_x\sin(\delta)}{2m} + \frac{C_f\left(\delta - \frac{v_y + l_f\omega}{v_x}\right)\cos(\delta)}{m} + \frac{C_r\left(-\frac{v_y - l_r\omega}{v_x}\right)}{m} - \omega v_x \\ \dot{\omega} &= \frac{F_x l_f\sin(\delta)}{2I_z} + \frac{C_f l_f\left(\delta - \frac{v_y + l_f\omega}{v_x}\right)\cos(\delta)}{I_z} - \frac{C_r l_r\left(-\frac{v_y - l_r\omega}{v_x}\right)}{I_z} \end{aligned} \tag{3.13}$$

3.3.2 Vehicle Model for Low Speed

In general, a vehicle moving at low speed experiences less forces than a vehicle moving at high speed. For example, v_y is negligible and the transformation between the vehicle frame and the global frame is approximated according to:

$$\begin{aligned} \dot{X} &= v_x\cos(\Psi) \\ \dot{Y} &= v_x\sin(\Psi) \end{aligned} \tag{3.14}$$

From Figure 3.4, a relationship between δ, the radius of the instantaneous radius of curvature (R) and the distance between the front and rear axles of the vehicle (L), can be determined from basic trigonometry, $\tan(\delta) = L/R$. At low speed the distance R is a result of the relationship between the yaw rate ($\dot{\Psi}$) and v_x, $R = v_x/\dot{\Psi}$. Combining the two relationships results in the following equation (Snider 2009):

$$\dot{\Psi} = \frac{v_x}{L}\tan(\delta) \tag{3.15}$$

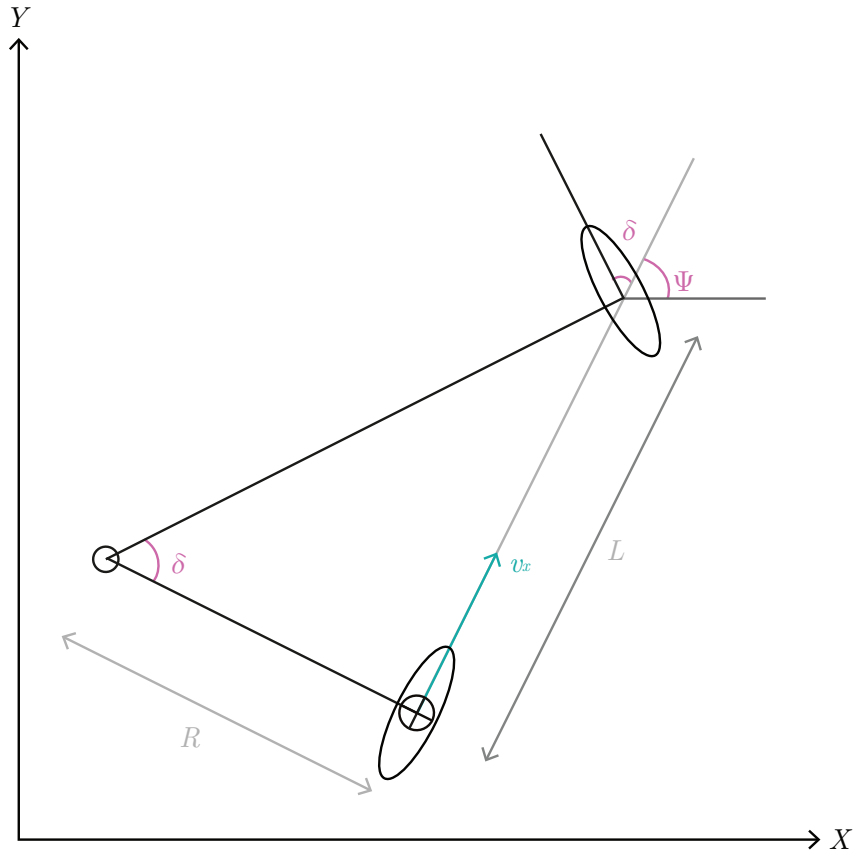


Figure 3.4. Dynamics of a low speed model which is used as internal model for the MPC.

The last part of the low speed model is the force actually driving the vehicle forward. This is modeled by Newton's second law of motion, $F_x = m\dot{v}_x$, which reformulated and together with the previously described equations form the final system for the vehicle model for low speed (model_{LS}):

$$\begin{aligned}
\dot{X} &= v_x\cos(\Psi) \\
\dot{Y} &= v_x\sin(\Psi) \\
\dot{\Psi} &= \frac{v_x\tan(\delta)}{L} \\
\dot{v}_x &= \frac{F_x}{m}
\end{aligned} \tag{3.16}$$

3.4 Adapting a Nonlinear Model for MPC

Both the model$_{HS}$ and the model$_{LS}$ are nonlinear. However, since linear constraints are guaranteed to be convex it is convenient to use a linear internal model. This is not to be confused with using a linear cost function. Even though there are multiple well developed ways of solving an MPC problem with a nonlinear model, computational demands are much higher compared to a linear case (Kühne *et al.* 2004).

Obtaining a linear model can be done by linearizing the nonlinear model successively (Kühne *et al.* 2004) around a nominal model. The nominal model consists of "references", or *nominal values*, for $\mathbf{x}$ and $\mathbf{u}$ at any given time sample. These nominal values for the state and the input vectors are denoted as $\mathbf{x}^n$ and $\mathbf{u}^n$, which according to (Kühne *et al.* 2004) behaves as:

$$\dot{\mathbf{x}}^n = f(\mathbf{x}^n, \mathbf{u}^n) \tag{3.17}$$

It is possible to linearize the nonlinear model around the nominal model by using Taylor expansions, which is why nominal values for all $\mathbf{x}$ and $\mathbf{u}$ are needed.

$$\dot{\mathbf{x}} = f(\mathbf{x}^n, \mathbf{u}^n) + \left.\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\mathbf{x}^n \\ \mathbf{u}=\mathbf{u}^n}} (\mathbf{x} - \mathbf{x}^n) + \left.\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}\right|_{\substack{\mathbf{x}=\mathbf{x}^n \\ \mathbf{u}=\mathbf{u}^n}} (\mathbf{u} - \mathbf{u}^n) \tag{3.18}$$

Subtracting 3.17 from 3.18 results in a deviation model according to

$$\dot{\tilde{\mathbf{x}}} = \left.\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\mathbf{x}^n \\ \mathbf{u}=\mathbf{u}^n}} \tilde{\mathbf{x}} + \left.\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}\right|_{\substack{\mathbf{x}=\mathbf{x}^n \\ \mathbf{u}=\mathbf{u}^n}} \tilde{\mathbf{u}} \tag{3.19}$$

where the state deviation is defined as $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x}^n$. Defining $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ as

$$\begin{aligned} \bar{\mathbf{A}} &= \left.\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\right|_{\substack{x=\mathbf{x}^n \\ u=\mathbf{u}^n}} \\ \bar{\mathbf{B}} &= \left.\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}\right|_{\substack{x=\mathbf{x}^n \\ u=\mathbf{u}^n}} \end{aligned} \tag{3.20}$$

makes it possible to formulate linear equations for predicted future deviations of the states according to:

$$\begin{aligned} \hat{\tilde{\mathbf{x}}}(k+1|k) &= \bar{\mathbf{A}}(k)\tilde{\mathbf{x}}(k) + \bar{\mathbf{B}}(k)\mathbf{u}(k) \\ \hat{\tilde{\mathbf{x}}}(k+2|k) &= \bar{\mathbf{A}}(k+1|k)\hat{\tilde{\mathbf{x}}}(k+1|k) + \bar{\mathbf{B}}(k+1|k)\tilde{\mathbf{u}}(k+1) \\ &= \bar{\mathbf{A}}(k+1|k)(\bar{\mathbf{A}}(k)\tilde{\mathbf{x}}(k) + \bar{\mathbf{B}}(k)\mathbf{u}(k)) + \bar{\mathbf{B}}(k+1|k)\tilde{\mathbf{u}}(k+1) \\ &= \bar{\mathbf{A}}(k+1|k)\bar{\mathbf{A}}(k)\tilde{\mathbf{x}}(k) + \bar{\mathbf{A}}(k+1|k)\bar{\mathbf{B}}(k)\mathbf{u}(k) + \bar{\mathbf{B}}(k+1|k)\tilde{\mathbf{u}}(k+1) \\ &\vdots \\ \hat{\tilde{\mathbf{x}}}(k+H_p|k) &= \prod_{i=0}^{H_p-1} \bar{\mathbf{A}}(k+i|k)\tilde{\mathbf{x}}(k) + \prod_{i=1}^{H_p-1} \bar{\mathbf{A}}(k+i|k)\bar{\mathbf{B}}(k)\mathbf{u}(k) \\ &\quad + \prod_{i=2}^{H_p-1} \bar{\mathbf{A}}(k+i|k)\bar{\mathbf{B}}(k+1|k)\tilde{\mathbf{u}}(k+1|k) \\ &\quad + \ldots + \bar{\mathbf{B}}(k+H_p-1|k)\tilde{\mathbf{u}}(k+H_p-1) \end{aligned} \tag{3.21}$$

Equation 3.21 is a set of linear equations representing the behavior or the nonlinear model. The QP-problem can now be formulated:

$$\begin{aligned} &\arg\min_{\tilde{u}} \sum_{k=1}^{T} \tilde{\mathbf{x}}(k)^T \mathbf{Q} \tilde{\mathbf{x}}(k) + \tilde{\mathbf{u}}(k-1)^T \mathbf{R} \tilde{\mathbf{u}}(k-1) \\ &\text{subject to } \tilde{\mathbf{x}}(k+1) = \bar{\mathbf{A}}(k)\tilde{\mathbf{x}}(k) + \bar{\mathbf{B}}(k)\tilde{\mathbf{u}}(k) \qquad \forall\, k = 0 \ldots H_p \end{aligned} \tag{3.22}$$

A simplified overview of the linearized system is shown by Figure 3.5.

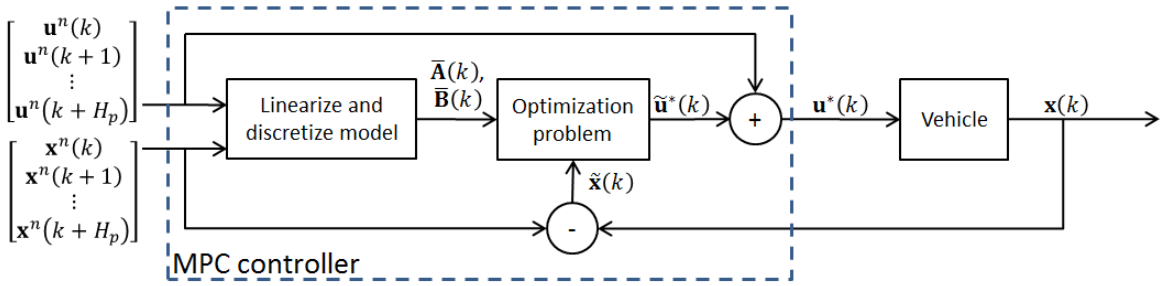


Figure 3.5. Simplified concept of an MPC controller including the linearized model and the optimization problem.

3.4.1 Linearization and Discretization of Vehicle Models

When linearizing the model$_{LS}$, $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ becomes:

$$\bar{\mathbf{A}} = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\bigg|_{\substack{x=\mathbf{x}^n \\ u=\mathbf{u}^n}} = \begin{bmatrix} 0 & 0 & -v_x^n \sin(\Psi^n) & \cos(\Psi^n) \\ 0 & 0 & v_x^n \cos(\Psi^n) & \sin(\Psi^n) \\ 0 & 0 & 0 & \frac{\tan(\delta^n)}{L} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.23}$$

$$\bar{\mathbf{B}} = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}\bigg|_{\substack{x=\mathbf{x}^n \\ u=\mathbf{u}^n}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{v_x^n(1+\tan^2(\delta^n))}{L} & 0 \\ 0 & \frac{1}{m} \end{bmatrix} \tag{3.24}$$

The discretization of model$_{LS}$ then becomes:

$$\bar{\mathbf{A}} = \begin{bmatrix} 1 & 0 & -\Delta t v_x^n(k)\sin(\Psi^n(k)) & \Delta t\cos(\Psi^n(k)) \\ 0 & 1 & \Delta t v_x^n(k)\cos(\Psi^n(k)) & \Delta t\sin(\Psi^n(k)) \\ 0 & 0 & 1 & \Delta t\frac{\tan(\delta^n(k))}{L} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.25}$$

$$\bar{\mathbf{B}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \Delta t\frac{v_x^n(k)(1+\tan^2(\delta^n(k)))}{L} & 0 \\ 0 & \Delta t\frac{1}{m} \end{bmatrix} \tag{3.26}$$

High Speed Model

The high speed model is linearized and discretized similarly as the low speed model. However, due to their sizes, these matrices are presented in appendix A.

3.4.2 Nominal Values of States and Inputs

The linearization requires nominal values for all $\mathbf{x}$ and $\mathbf{u}$. Both vehicle models are to follow a specified path ($\mathbf{X}^r$, $\mathbf{Y}^r$). All other $\mathbf{x}$ and $\mathbf{u}$ have to be estimated for the vehicle models. For the model$_{LS}$ this means that Ψ, v_x, δ and F_x needs to be estimated. For the model$_{HS}$ the two additional states v_y and ω also needs to be estimated. Ψ, v_x and δ are extracted from the model$_{LS}$ but used to calculate the nominal values for both models. Because of the different dynamics of the two vehicle models F_x is estimated in two ways.

All estimates are based on a discrete version of the vehicle model, hence it had to be discretized. This was done by the Euler Forward method, where a continuous system $y'(t) = f(t, y(t))$ is transformed into the discrete system $y_{k+1} = y_k + \Delta t f(t_k, y_k)$ where Δt is the step size.

Yaw Angle

By using the two first equations in the model$_{LS}$ (eq 3.16) Ψ can be extracted accordingly:

$$\begin{aligned} \dot{X} &= v_x\cos(\Psi) \\ \dot{Y} &= v_x\sin(\Psi) \\ v_x &= \frac{\dot{X}}{\cos(\Psi)} \\ \dot{Y} &= \frac{\dot{X}}{\cos(\Psi)}\sin(\Psi) = \dot{X}\tan(\Psi) \\ \frac{\dot{Y}}{\dot{X}} &= \tan(\Psi) \\ \Psi &= \arctan\left(\frac{\dot{Y}}{\dot{X}}\right) \end{aligned} \tag{3.27}$$

Assuming that $\dot{X}$ and $\dot{Y}$ have the following discrete equivalences

$$\dot{X} \approx \frac{X(k) - X(k-1)}{\Delta t} \tag{3.28}$$

$$\dot{Y} \approx \frac{Y(k) - Y(k-1)}{\Delta t} \tag{3.29}$$

the discretized version of Ψ becomes:

$$\Psi(k) = \arctan\left(\frac{Y(k) - Y(k-1)}{X(k) - X(k-1)}\right) \tag{3.30}$$

Longitudinal Velocity

v_x is the resulting velocity of the global longitudinal and lateral velocities, that is:

$$\sqrt{\dot{X}^2 + \dot{Y}^2} = \sqrt{(v_x \cos(\Psi))^2 + (v_x \sin(\Psi))^2} = v_x \tag{3.31}$$

In the discrete case this becomes:

$$v_x(k) = \frac{\sqrt{(X(k) - X(k-1))^2 + (Y(k) - Y(k-1))^2}}{\Delta t} \tag{3.32}$$

Lateral Velocity

In an optimal case there will be no offset on the lateral position and hence no, or negligible v_y. In the high speed case there will be lateral acceleration and hence v_y to consider. Deviations will appear when linearizing around v_y^n which will increase the cost function in the optimization problem. Different options have been evaluated briefly in the project but $v_y = 0$ was concluded to be the best nominal value in the high speed case as well.

Yaw Rate

ω is simply the rate of change of the yaw angle, hence:

$$\omega(k) = \frac{\Psi(k) - \Psi(k-1)}{\Delta t} \tag{3.33}$$

Front Wheel Angle

Based on the third equation of the model$_{LS}$, eq 3.16, δ can be extracted as:

$$\delta = \arctan\left(\frac{L\omega}{\dot{v}_x}\right) \tag{3.34}$$

which in the discrete case becomes:

$$\delta(k) = \arctan\left(\frac{L(\Psi(k) - \Psi(k-1))}{(v_x(k) - v_x(k-1))}\right) \tag{3.35}$$

Longitudinal Force - Low Speed

F_x is based on the fourth equation of the model$_{LS}$, eq 3.16:

$$F_x = m\dot{v}_x \tag{3.36}$$

which discretized becomes:

$$F_x(k) = \frac{m(v_x(k) - v_x(k-1))}{\Delta t} \tag{3.37}$$

Longitudinal Force - High Speed

A vehicle moving at high velocities is affected by many different forces, e.g. air drag force and rolling resistance which added leads to the following relationship

$$F_x = A\rho\kappa v_x^2 + Kmg + m\dot{v}_x \tag{3.38}$$

where A is the area of the vehicle front, ρ is the density of air, κ is the streamline profile of the vehicle, K is the rolling resistance coefficient and g is the gravitational force. In discretized form this becomes:

$$F_x(k) = A\rho\kappa v_x(k)^2 + Kmg + \frac{m(v_x(k) - v_x(k-1))}{\Delta t} \tag{3.39}$$

The nominal values needed for the linearization can therefore be summarized to:

$$\mathbf{x}_{LS}^n = \begin{bmatrix} X^r \\ Y^r \\ \Psi^n \\ v_x^n \end{bmatrix} = \begin{bmatrix} X^r \\ Y^r \\ \arctan\left(\frac{Y^r(k)-Y^r(k-1)}{X^r(k)-X^r(k-1)}\right) \\ \frac{\sqrt{(X^r(k)-X^r(k-1))^2+(Y^r(k)-Y^r(k-1))^2}}{\Delta t} \end{bmatrix} \tag{3.40}$$

$$\mathbf{u}_{LS}^n = \begin{bmatrix} \delta^n \\ F_x^n \end{bmatrix} = \begin{bmatrix} \arctan\left(\frac{L(\Psi^n(k)-\Psi^n(k-1))}{(v_x^n(k)-v_x^n(k-1))}\right) \\ \frac{m(v_x^n(k)-v_x^n(k-1))}{\Delta t} \end{bmatrix} \tag{3.41}$$

$$\mathbf{x}_{HS}^n = \begin{bmatrix} X^r \\ Y^r \\ \Psi^n \\ v_x^n \\ v_y^n \\ \omega^n \end{bmatrix} = \begin{bmatrix} X^r \\ Y^r \\ \arctan\left(\frac{Y^r(k)-Y^r(k-1)}{X^r(k)-X^r(k-1)}\right) \\ \frac{\sqrt{(X^r(k)-X^r(k-1))^2+(Y^r(k)-Y^r(k-1))^2}}{\Delta t} \\ 0 \\ \frac{\Psi^n(k)-\Psi^n(k-1)}{\Delta t} \end{bmatrix} \tag{3.42}$$

$$\mathbf{u}_{HS}^n = \begin{bmatrix} \delta^n \\ F_x^n \end{bmatrix} = \begin{bmatrix} \arctan\left(\frac{L(\Psi^n(k)-\Psi^n(k-1))}{(v_x^n(k)-v_x^n(k-1))}\right) \\ A\rho\kappa v_x^n(k)^2 + Kmg + \frac{m(v_x^n(k)-v_x^n(k-1))}{\Delta t} \end{bmatrix} \tag{3.43}$$

3.4.3 Weights on State and Input Deviations

For the two vehicle models their corresponding weight matrices can now be formulated. Notice that the weights $\mathbf{Q}$ and $\mathbf{R}$ are diagonal matrices with weights to penalize $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{u}}$. For the low speed case this leads to the following matrix:

$$\mathbf{Q}_{LS} = \begin{bmatrix} Q_{\tilde{X}} & 0 & 0 & 0 \\ 0 & Q_{\tilde{Y}} & 0 & 0 \\ 0 & 0 & Q_{\tilde{\Psi}} & 0 \\ 0 & 0 & 0 & Q_{\tilde{v}_x} \end{bmatrix} \tag{3.44}$$

Since only the diagonal elements affect the optimization they are summarized in a diagonal weight matrix sequence (Q^d):

$$Q^d_{LS} = \{Q_{\tilde{X}}, Q_{\tilde{Y}}, Q_{\tilde{\Psi}}, Q_{\tilde{v}_x}\} \tag{3.45}$$

For the high speed case $\mathbf{Q}$ and Q^d becomes:

$$\mathbf{Q}_{HS} = \begin{bmatrix} Q_{\tilde{X}} & 0 & 0 & 0 & 0 & 0 \\ 0 & Q_{\tilde{Y}} & 0 & 0 & 0 & 0 \\ 0 & 0 & Q_{\tilde{\Psi}} & 0 & 0 & 0 \\ 0 & 0 & 0 & Q_{\tilde{v}_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & Q_{\tilde{v}_y} & 0 \\ 0 & 0 & 0 & 0 & 0 & Q_{\tilde{\omega}} \end{bmatrix} \tag{3.46}$$

$$Q^d_{HS} = \{Q_{\tilde{X}}, Q_{\tilde{Y}}, Q_{\tilde{\Psi}}, Q_{\tilde{v}_x}, Q_{\tilde{v}_y}, Q_{\tilde{\omega}}\} \tag{3.47}$$

Since the inputs of model$_{LS}$ and model$_{HS}$ are the same, $\mathbf{R}$ and R^d will be used for both applications:

$$\mathbf{R} = \begin{bmatrix} R_{\tilde{\delta}} & 0 \\ 0 & R_{\tilde{F}_x} \end{bmatrix} \tag{3.48}$$

$$R^d = \{R_{\tilde{\delta}}, R_{\tilde{F}_x}\} \tag{3.49}$$

4 Development of MPC Framework

The framework consists of functions for resampling the path, calculating nominal values and packaging them for the MPC controller. It also includes the MPC controller itself. This section describes the performance criteria, the test tracks used, development of the framework as well as implementation in the two environments where the framework could be used; simulation and vehicle.

4.1 Performance

Four different performance criteria were considered. The lateral deviation from the reference (P_l), the deviation in positioning (P_p), the comfort (P_c) and the maximum deviation (P_d). P_l was formulated as:

$$P_l = \frac{100}{N} \sum_{k=1}^{N} \min_j \Big\{ \sqrt{(X(k) - X^r(j))^2 + (Y(k) - Y^r(j))^2} \Big\} \tag{4.1}$$

where N is the number of samples in the simulation. For P_l it is clear that if the vehicle stays in its initial position, it gets a perfect score.

The positioning criterion was determined based on the ability to follow the path, both in time and space. The distance from the desired position were summarized in each time step according to:

$$P_p = \frac{100}{N} \sum_{k=1}^{N} \sqrt{(X(k) - X^r(k))^2 + (Y(k) - Y^r(k))^2} \tag{4.2}$$

The comfort criterion was determined as the derivative of the acceleration, called jerk, both in longitudinal and lateral direction:

$$P_c = \frac{100}{N} \sum_{k=1}^{N} \sqrt{\dot{a}_x^2 + \dot{a}_y^2} \tag{4.3}$$

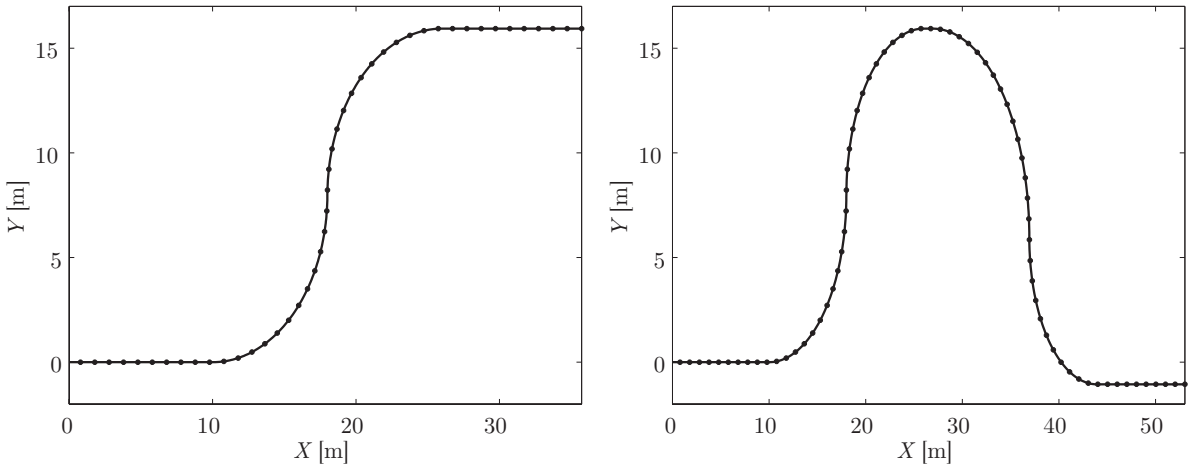
All above performance criteria measures the average deviation. However, P_d measures the maximum deviation:

$$P_d = \max_k \min_j \Big\{ 100 \sqrt{(X(k) - X^r(j))^2 + (Y(k) - Y^r(j))^2} \Big\} \tag{4.4}$$

4.2 Paths

The evaluation of the MPC controllers was performed on four different test tracks whereof two were designed for the low speed case and two for the high speed case. There are two possibilities concerning generation of test tracks; constructing a test track or recording a track from simulation or vehicle. For this project all test tracks have been constructed.

The low speed test tracks are shown in Figure 4.1a and 4.1b respectively. The test track shown in Figure 4.1a portrays a realistic maneuver for low speed situations and consists of a 10 m straight segment, two curves with 8 m radii and a final 10 m straight segment. This test track is henceforth referred to as LS1 and is primarily used for tuning of the deviation weights. The second test track for low speed maneuvering (LS2) is constructed to be more challenging than LS1 and is used for comparison between the MPC_{LS} controller and the P-controller_{LS}.



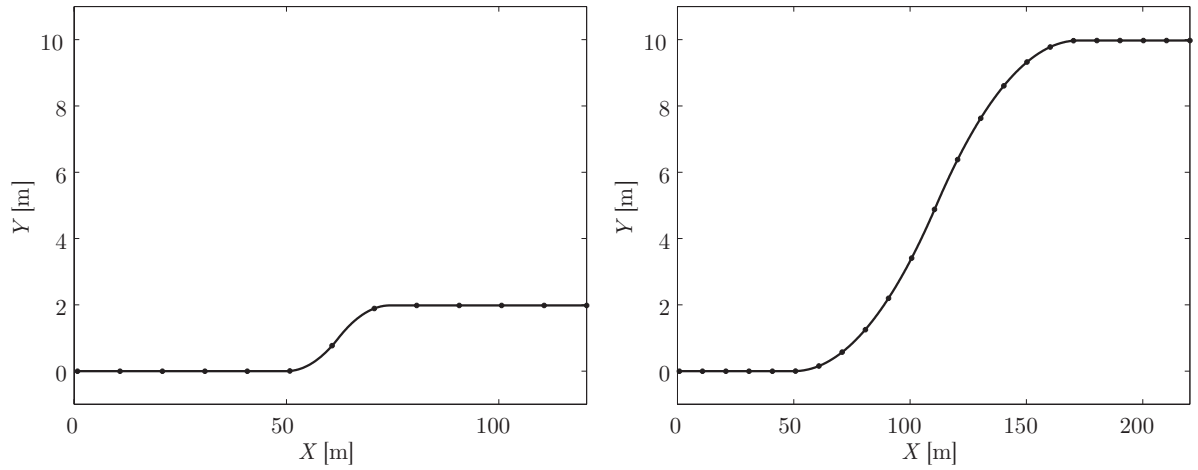
(a) Test track, LS1, with a 10 m straight segment, two curves with 8 m radii and a final 10 m straight segment.

(b) Test track, LS2, with a 10 m straight segment, two curves with 8 m radii, one with 7 m and, one with 10 m and a final 10 m straight segment.

Figure 4.1. Low speed test tracks.

Figure 4.2a and 4.2b visualizes the test tracks used for high speed maneuvers, henceforth called HS1 and HS2 respectively. HS1 has long straight segments, 50 m each. The radii were set using the formula for lateral acceleration, $a_y = v_x^2/R$. It was decided that a lateral acceleration of about 5 m/s^2 would be enough to excite the system. A higher lateral acceleration would imply that the linear tire model used would not be sufficient. Since the base speed for high speed simulations were set to 70 km/h the radii were set to $(\frac{70}{3.6})^2/5 \approx 76$ m. The angle of the curves were chosen to make a 2 m lateral distance between the first straight segment and the second. This could either represent a vehicle or a moose blocking the road, creating an evasive maneuver.

HS2 is a less demanding path with a lateral acceleration of 1 m/s^2 which results in radii of approximately 378 m. For this path the lateral distance between the straight segments were set to 10 m.



(a) Test track, HS1, with a 50 m straight segment, two curves with 76 m radii, a final 50 m straight segment and a lane change of 2 m.

(b) Test track, HS2, with a 50 m straight segment, two curves with 378 m radii, a final 50 m straight segment and a lane change of 10 m.

Figure 4.2. High speed test tracks.

4.3 Software

For this project four different softwares were used. MATLAB/Simulink was used for the main parts of the framework. The resampling of the path, calculating nominal values and packaging them for the MPC controller was done in MATLAB. The MPC controller itself and generation of code for the vehicle was done in Simulink. CVXGEN solved the optimization problem inside the MPC controller. CarMaker was used in cooperation with Simulink for simulations and finally dSpace was used for real-time testing in the vehicle.

4.3.1 MATLAB/Simulink

MATLAB has the ability to code generate standalone C and C++ code from a Simulink model. This was used for implementation of the MPC controller in the test vehicle. The code generator supports only a subset of the available features in MATLAB which limited the options during the development of the framework. Fortunately only the parts that has to be online, in this case the MPC controller, has to be code generated, hence the full features of MATLAB could be used for the rest of the framework.

4.3.2 CVXGEN

Due to the limitations of the code generator in MATLAB, the built in solver `quadprog` could not be used. Instead CVXGEN was chosen. CVXGEN is a software that generates C code which compiles into a solver for convex optimization problems. It generates library-free code suitable for real-time applications. The generated code is almost branch free and so has highly predictable run-time behavior (Mattingley 2013). This dictated the conditions for the rest of the framework. Here the solver generated by CVXGEN is referred to as CVX-solver.

The CVX-solver has the possibility of changing various settings, among them the tolerance and maximum number of iterations. The tolerance setting means that the problem will not be declared converged until the duality gap is known to be bound by the tolerance. This is set to 10^{-6} as default and the max iterations are set to 25 after which the solver will exit even if the tolerance is not reached.

Formulation of Optimization Problem

For CVXGEN to be able to understand the problem it has to be formulated in a certain high level language. Constants that are sent into the solver are specified as `parameters`. These can be scalars or vectors. It is also possible to specify different values of a parameter for each sample in H_p. Values that are determined by the solver are called `variables`. In addition to the parameters and variables the function that should be minimized and the constraints to be satisfied has to be formulated. This is done using the keywords `minimize` and `subject to`. Equation 3.22 was formulated in CVXGEN code as:

```
minimize
sum[k=1..Hp](quad(xtilde[k], Q) + quad(utilde[k-1], R))
subject to
xtilde[k+1] == A[k]*xtilde[k] + B[k]*utilde[t], k=0..Hp
```

The notation `k=0..Hp` was used instead of declaring $H_p + 1$ individual equations. The same notation was used for parameters with different values for different samples in H_p. The complete CVXGEN code can be found in appendix B.

4.3.3 CarMaker

When testing the MPC controller it is important to consider the selection of simulation model. If the simulation model is selected to be the same as the internal model the simulation model will perform ideal. Hence it is desired to have a different and more accurate simulation model to evaluate the performance of the MPC controller.

CarMaker is a full feature high fidelity simulation program for vehicles. It has the capability of close integration with Simulink. By generating a custom Simulink model from CarMaker it is possible to customize the input and register signals directly from Simulink. The signals move through multiple blocks in Simulink in each simulation step. By breaking up the connections between these blocks and connecting custom signals the vehicle may be directly controlled through Simulink. The structure can be seen in Figure 4.3.

It is always desired to have a simulation model that represents the real system as closely as possible. However, this might not be the case for the internal model. A too complex internal model has many states and complex relations between the states. This might lead to unnecessary calculations, both to formulate the problem and for the solver in real time. A sufficient internal model is therefore desired.

To be able to test if the model is sufficient a high fidelity model is a very useful tool. CarMaker has most of the dynamics of a real vehicle. By using CarMaker it is possible to see if the internal model is sufficient at different speeds. Because CarMaker is a high fidelity simulation program, weights tuned to work in simulation will probably work rather well in the real system. A screenshot from CarMaker is shown in Figure 4.4.

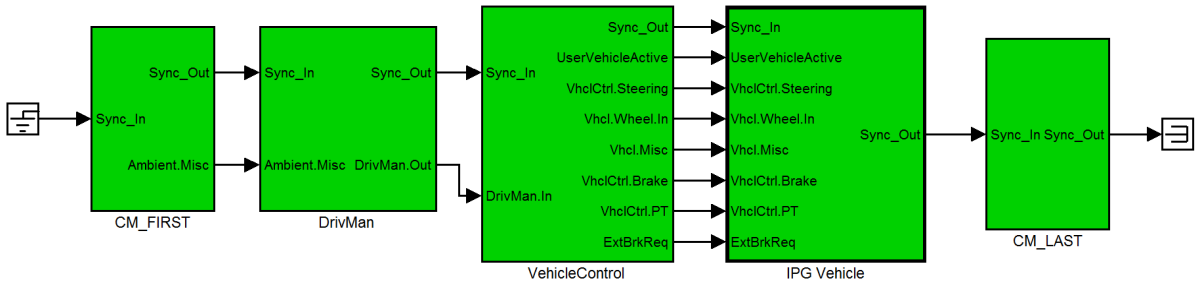


Figure 4.3. CarMaker generated Simulink structure.

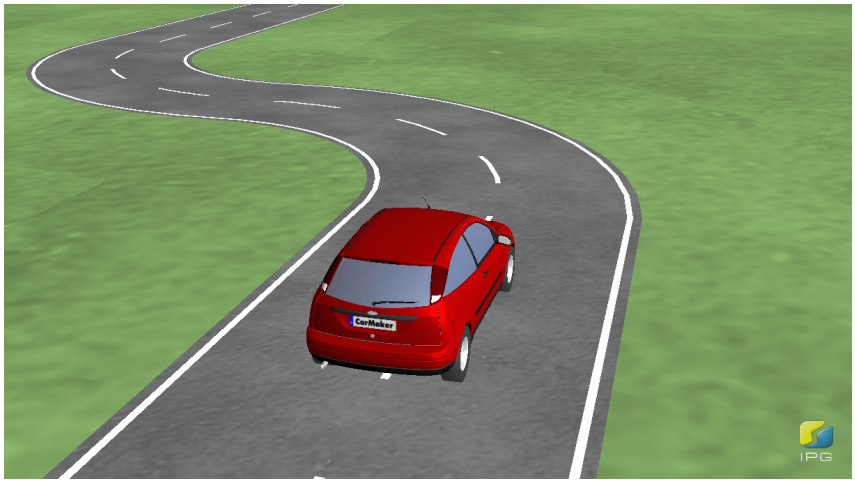


Figure 4.4. The simulation vehicle on test track LS1 following the center of the road.

4.4 Development of MPC Controller

Two MPC controllers, differing in internal models were designed. The MPC controller with model_{LS} as the internal model is referred to as the MPC_{LS} controller and similarly is the MPC controller with model_{HS} as internal model referred to as the MPC_{HS} controller. When designing an MPC controller there are five properties which should be considered: choice of internal model, constraints, length of H_p, tuning of weights and *sampling interval*. The sampling interval (T) of the MPC controller is the number of seconds between new outputs. The MPC controller was developed in such a way that these parameters were possible to change.

4.4.1 Constraints

When developing the MPC controller it was desirable to add constraints on $\mathbf{u}$ and the change rate of $\mathbf{u}$ ($\Delta\mathbf{u}$). Generally it would also be possible to add constraints on $\mathbf{x}$ and $\Delta\mathbf{x}$ but this was not implemented since no constraints were desired for those parameters in this project. The maximum value of $\mathbf{u}$ is referred to as $\mathbf{u}_{max}$ and the minimum as $\mathbf{u}_{min}$. Since the linearized model is defined for $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{u}}$ the constraints have to be put on those. This can be formulated as:

$$\tilde{\mathbf{u}}_{min} \leq \tilde{\mathbf{u}} \leq \tilde{\mathbf{u}}_{max} \tag{4.5}$$

To instead put the constraint on $\mathbf{u}$, $\mathbf{u}^n$ can be subtracted as:

$$\mathbf{u}_{min}(k) - \mathbf{u}^n(k) \leq \tilde{\mathbf{u}}(k) \leq \mathbf{u}_{max} - \mathbf{u}^n(k) \tag{4.6}$$

To constrain $\Delta\mathbf{u}$ the difference between the current and the previous values were constrained throughout H_p. The maximum and minimum value of the change rate is referred to as $\Delta\mathbf{u}_{min}$ and $\Delta\mathbf{u}_{max}$.

$$\Delta\mathbf{u}_{min} \leq (\tilde{\mathbf{u}}(k+i-1) - \tilde{\mathbf{u}}(k+i)) + (\mathbf{u}^n(k+i-1) - \mathbf{u}^n(k+i)) \leq \Delta\mathbf{u}_{max} \tag{4.7}$$

Therefore $\mathbf{u}^n$, $\mathbf{u}_{min}$, $\mathbf{u}_{max}$, $\Delta\mathbf{u}_{min}$ and $\Delta\mathbf{u}_{max}$ have to be set as parameters in CVXGEN.

4.4.2 Prediction Horizon

The size of H_p was largely constrained by the calculation time and the limitations of CVXGEN. As mentioned before every additional sample in H_p leads to more input parameters, equality constraints and inequality constraints.

The optimization problem for the MPC_{LS} controller with constraints and an H_p of 30 samples resulted in 3536 non-zero KKT matrix entries which is about average (Mattingley 2013). The generation of the solver works best for optimization problems up to around 4000 entries, after this the generation becomes very slow and might fail (Mattingley 2013). If the MPC controller runs with a T of 0.1 s, 30 samples corresponds to an H_p of 3 s. The MPC_{HS} controller, having six states instead of four, using the same constraints and H_p resulted in an optimization problem with 5832 entries. It was still possible to generate code for this number of entries, however, a larger H_p could not be used. An H_p of 0.5, 1, 2 and 3 s were tested for both the MPC_{LS} and MPC_{HS} controllers.

4.4.3 MPC-block

The CVX-solver was implemented in Simulink as a block by using the *S-function Builder*, here abbreviated to SfB. S-function stands for system-function and is a way to describe a Simulink block using computer language such as C. The SfB also serves as a wrapper for the S-function and offers a GUI for editing the C code, including libraries and external source code. It also offers the possibility to customize inputs/outputs of the block. The necessary files generated by CVXGEN were included together with the math library needed for the model equations. The included files and libraries are shown in Figure 4.5.

The MPC_{LS} controller implements model_{LS} in C code while the MPC_{HS} controller implements model_{HS}. The C code for the MPC_{LS} controller can be found in appendix C. The blocks are shown in Figure 4.6. For the CVX-solver to work within the MPC controller all parameters have to be defined to either values from the input ports or values calculated with C code inside the block. The values of the output ports were set to the values obtained from the CVX-solver or values used for debugging purposes. The MPC_{LS} and MPC_{HS} controllers are identical, except for the additional inputs in the MPC_{HS} controller and the equations used to calculate the system matrices.

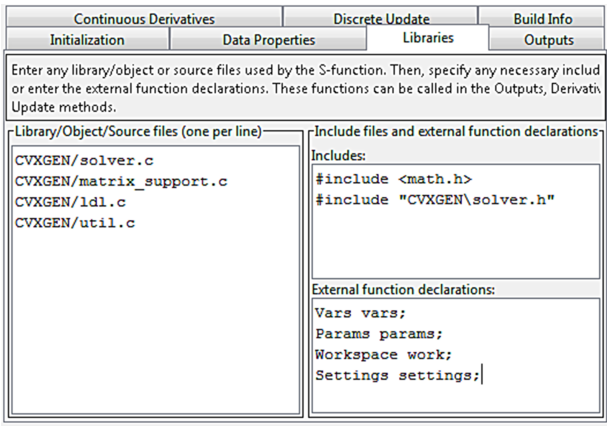
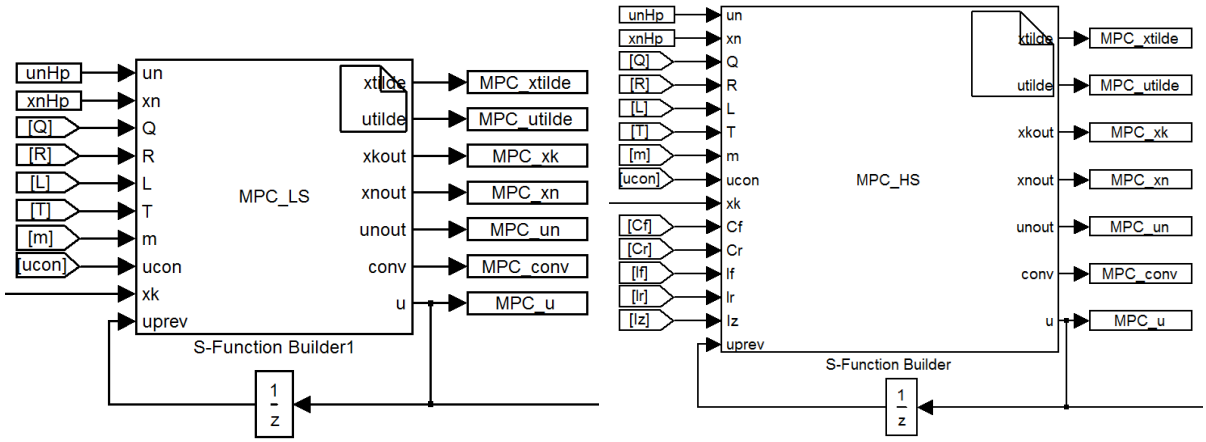


Figure 4.5. The CVX-solver implementation in S-Function Builder.



(a) MPC$_{LS}$ controller. **(b)** MPC$_{HS}$ controller.

Figure 4.6. MPC$_{LS}$ controller and MPC$_{HS}$ controller implemented as blocks in Simulink.

The MPC controllers requires $\mathbf{x}^n$ and $\mathbf{u}^n$ in each time step. These were, for each time step, formatted as two matrices starting with the current values and containing the values for H_p.

$$\mathbf{xnHp}(k) = \begin{bmatrix}\mathbf{x}^n(k) & \mathbf{x}^n(k+1) & \cdots & \mathbf{x}^n(k+H_p+1)\end{bmatrix} \tag{4.8}$$

$$\mathbf{unHp}(k) = \begin{bmatrix}\mathbf{u}^n(k) & \mathbf{u}^n(k+1) & \cdots & \mathbf{u}^n(k+H_p+1)\end{bmatrix} \tag{4.9}$$

The k indicates that the matrices are defined in time. Hence the **xnHp** and **unHp** matrices have three dimensions: number of states/inputs, H_p and simulated samples (M). Because of H_p the values occur in multiple of the in time succeeding matrices. The input parameters for the MPC controllers are summarized in appendix D.

Input/output Ports

The SfB requires that the dimensions of the input and output ports are specified. Unfortunately these may not be specified with variables nor constants. If, for example, H_p is changed the dimensions of the input ports have to be manually changed in the MPC-block to fit the new dimensions of the input.

4.4.4 Inputs to the MPC-block

The constraints, weights and vehicle parameters were set as constant inputs to the MPC controller. **xnHp** and **unHp** are time dependent representations of the path and the nominal values. To formulate these the path was pretreated with a suiting time relation for the MPC controller.

Three alternatives were implemented for following the path. In the first one the vehicle follows the path at the *same speed* as the path was originally constructed or recorded for. For the second alternative the vehicle follows the path in a *constant speed* and for the third alternative the vehicle follows the path with a *speed profile*.

Resampling for Same Speed

In this case the vehicle was desired to follow the path in the same speed as it was constructed for. This could include changes of speed at certain locations. To resample the sampling interval of the path to T, a new time vector was first created. This was done by taking the last value of the original time vector (t_{end}) and using that value as the last value of the new time vector. Equally distributed time samples from 0 to t_{end} were generated with the interval of T. The X-, and Y-coordinates were then interpolated from the old time vector to the new one with MATLAB's `interpol1`.

Resampling for Constant Speed

In this case a constant speed was desired for the whole path and T could differ from the sampling interval of the construction. To resample the coordinates to a constant speed the coordinates were resampled such that they became equidistant. This was done by determining the total distance (d_{end}) of the path and deciding a fixed distance (resolution) between the samples (d_{end}). A vector with distances was created from 0 to d_{end} with intervals of d_{res}. By using d_{end} and the desired speed, a new t_{end} could be calculated. By using the vector with distances and t_{end} a new vector with T as sampling interval could be obtained.

Resampling for Speed Profile

In this case a speed profile consisting of an acceleration phase, a constant speed phase and a deceleration phase was desired while following the path. By dividing the speed profile into segments the method for constant speed could be used. Figure 4.7 illustrates the different stages of the resampling for a speed profile. The top figure shows the original path which should be followed. The middle figure shows the equidistant phase with a d_{res} of 0.01 m. The bottom figure shows the final reference.

The speed profile that has been applied consists of an initial acceleration of 5 seconds to a constant speed of 3 m/s. The samples are therefore closer together at the beginning of the reference.

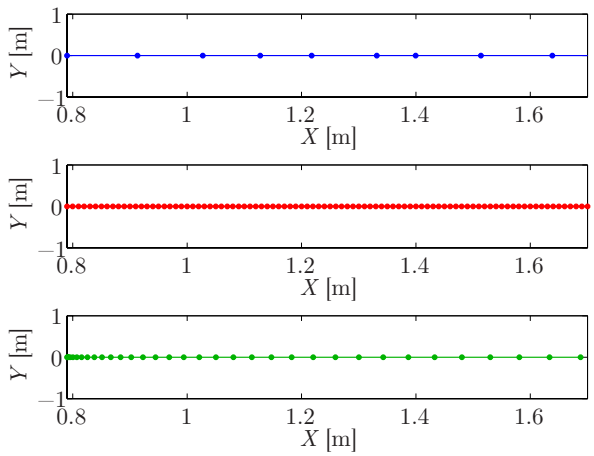


Figure 4.7. Visualization of the different stages of resampling with a speed profile. Top: original path. Middle: equidistant path. Bottom: final reference.

4.5 Simulation implementation - CarMaker

The MPC controller was connected to CarMaker as shown in Figure 4.8. This was done by modifications to the VehicleControl-block shown in Figure 4.3. The input to the MPC controller was sent to Simulink by using ”From Workspace”-blocks. All inputs to Simulink were defined with a time vector making it possible for CarMaker to decide the step length and obtain the right values at the right time.

The values of the vehicle's states were obtained in each time step by CarMaker sensor-blocks (bottom left corner in Figure 4.8) and sent as the current state to the MPC controller. δ was sent from the MPC controller into the IPGVehicle-block as shown in Figure 4.8. Notice that, for model$_{LS}$ the origin was placed at the rear axle while for model$_{HS}$ the origin was placed at the CoG. To compare CarMaker to the models the position sensor was placed at the corresponding place as shown in Figure 4.9a and 4.9b. F_x was transformed into a total wheel torque by multiplication of the wheel radius. This was distributed equally on the four wheels and sent to CM-blocks that applied the torques directly on the wheels.

Paths were put into CarMaker using ASCII encoded files containing tabulated data points generated by MATLAB scripts. The resolution of the road were 0.1 m for low speed simulations and 0.2 m for high speed simulations to fit CarMaker's digitized road feature. These resolutions were also used for the references in the MPC controllers, but resampled in accordance to section 4.4.4.

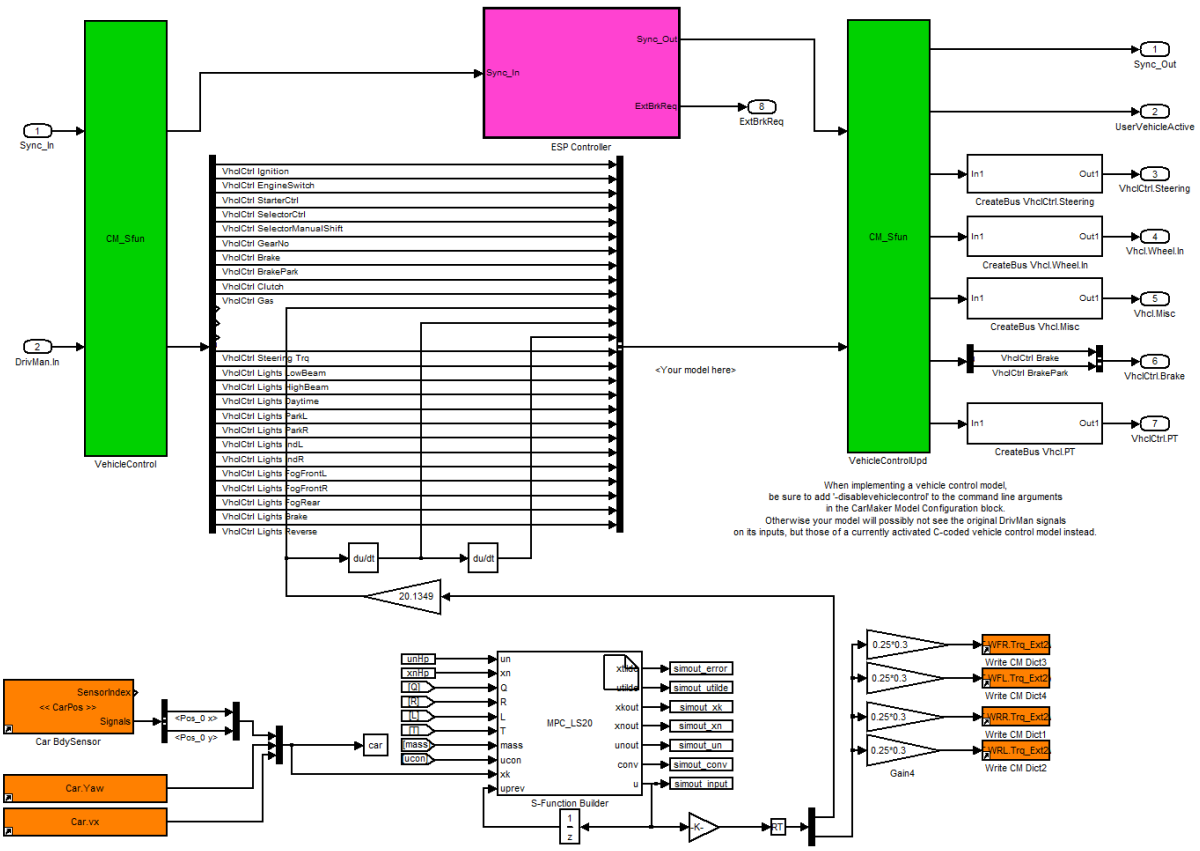


Figure 4.8. Implementation of the MPC_{LS} controller in simulation by modification to the VehicleControl-block.

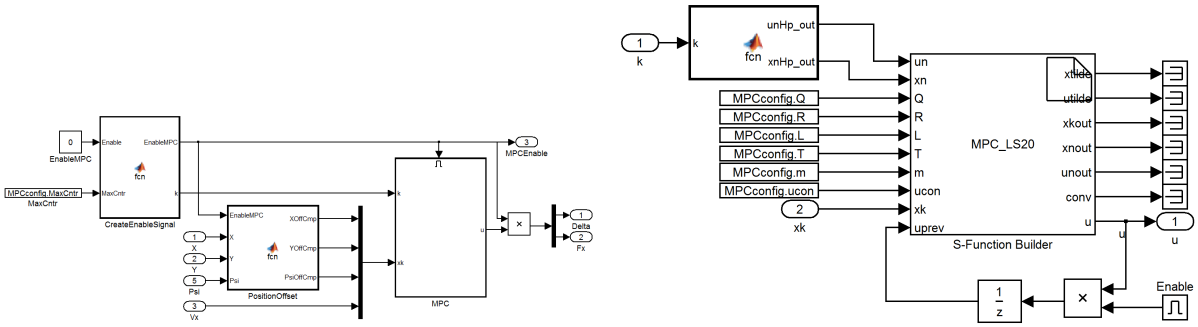
(a) Origin for model_{LS}, placed at the rear axle. **(b)** Origin for model_{HS}, placed at the CoG.

Figure 4.9. Position of the origin for the different internal models.

4.6 Vehicle Implementation

A dSpace MicroAutoBox was used to implement the MPC controller in the test vehicle. MicroAutoBox is a real-time system used for fast function prototyping. It is used in applications such as chassi control and powertrain but the same type of hardware is also used in for example aerospace applications (dSpace 2014).

Simulink together with the dSpace software generates code for the MicroAutoBox. The MPC-block was put in a predeveloped Simulink environment, seen in Figure 4.10. "From Workspace"-blocks were not suitable to use in this environment, instead an "Embedded Matlab Fucntion"-block was used to obtain **xnHp** and **unHp** for the current time sample. It started counting as the MPC controller were turned on and counted the number of samples passed since the start. In addition an "Enable MPC" feature was added as a constant that could be changed in real-time. This was done so the vehicle could be placed in a suitable area before starting the MPC controller.



(a) Simulink model for implementation in vehicle. The MPC_{LS} controller subsystem is shown in Figure 4.10b.

(b) The MPC_{LS} controller subsystem.

Figure 4.10. Simulink model for implementation of MPC_{LS} controller in vehicle.

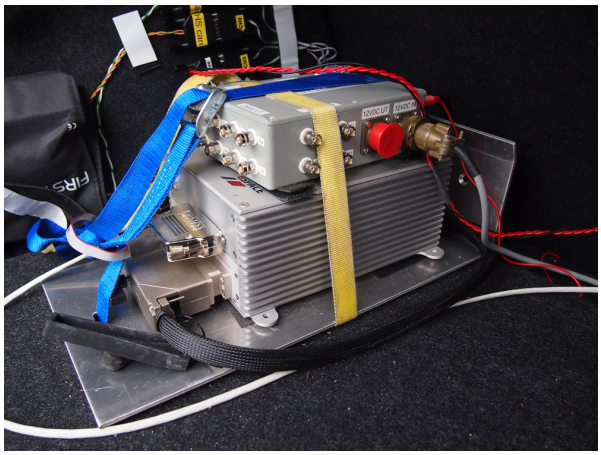
A laptop was connected to the MicroAutoBox and used to upload applications. Figure 4.11a and 4.11b shows the front seat where the laptop can be respectively is connected.

The MicroAutoBox was placed in the trunk as is shown in Figure 4.11c and 4.11d. The MicroAutoBox runs on an IBM PowerPC. To measure the current state of the test vehicle RT3000 were used. It is an advanced, precision Inertial and GPS Navigation system. This had in the performed test a resolution of about 0.3 m. The RT3000 is shown in Figure 4.11e and connected to the vehicle in Figure 4.11f.

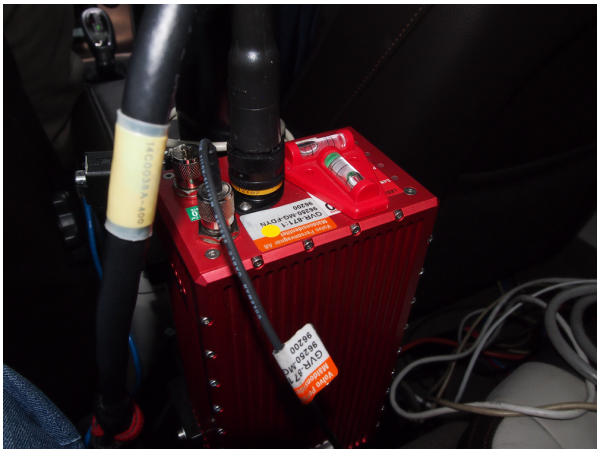
(a) Front seat of the test vehicle. Showing a laptop table.

(b) A connected laptop.

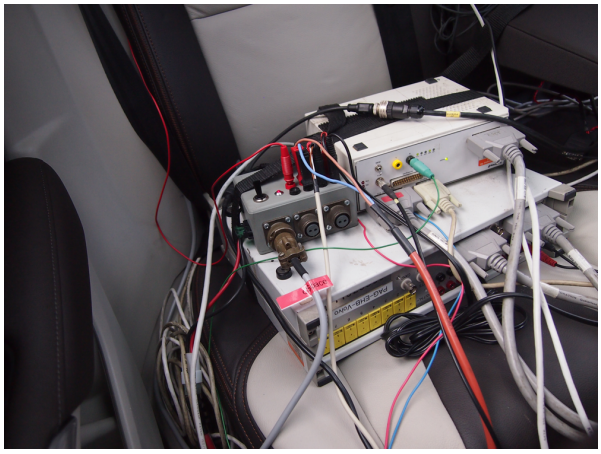
(c) Trunk of the test vehicle with the MicroAutoBox.



(d) The MicroAutoBox.



(e) A RT3000 family Inertial Navigation System with GPS receivers.



(f) Back seat of the test vehicle with wires and connection.

Figure 4.11. Hardware for implementation in vehicle.

5 Validation of MPC Framework

To validate the framework the stability of the discretization and the accuracy of the internal models have been evaluated. Finally the whole MPC framework was validated by multiple simulations with different weights. If the MPC controller makes the vehicle follow the path and the weights have the expected effect the framework would be considered to work as intended.

5.1 Stability Analysis for Discretization of Internal Models

For a discrete system to be stable the eigenvalues of the system should be enclosed by the unit cycle (Glad and Ljung 2000). Therefore the eigenvalues of model$_{LS}$ and model$_{HS}$ were calculated and plotted for different speeds and Δt. This is illustrated by Figure 5.1 where the top figure shows the eigenvalues at different speeds for model$_{LS}$ and model$_{HS}$ with a Δt of 0.1 s. In the lower figure Δt is 0.01 s.

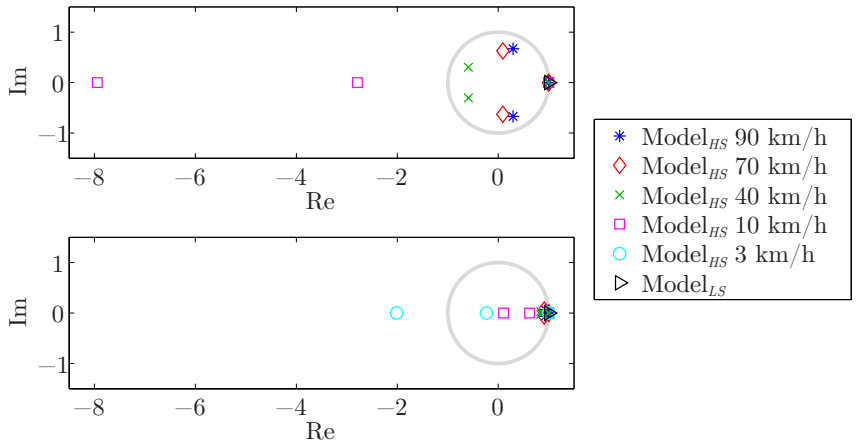


Figure 5.1. Eigenvalues for model$_{LS}$ and model$_{HS}$ and the corresponding stability region for discrete systems. Top figure: sampling interval 0.1. Bottom figure: sampling interval 0.01.

Figure 5.1 shows that for a Δt of 0.1 s model$_{HS}$ is unstable at speeds lower than $v_x =$ 10 km/h but stable for all speeds faster than 40 km/h. For a Δt of 0.01 model$_{HS}$ is stable when $v_x = 10$ km/h but is still unstable when $v_x = 3$ km/h. Model$_{LS}$ is always stable since all of the eigenvalues has the value 1. Note that these conclusions only are guaranteed for the Euler Forward discretization. Also note that Δt corresponds to T when the models are put into the MPC controller.

5.2 Validation of Internal Models using Open Loop Comparison

To test if the internal models agreed with the simulation model they were compared in open loop. $T = 0.1$ were used for all tests. For this comparison δ was varied as shown in Figure 5.2 while F_x was kept constant. The comparison was made at 4 different initial speeds; 3, 10, 70 and 90 km/h.

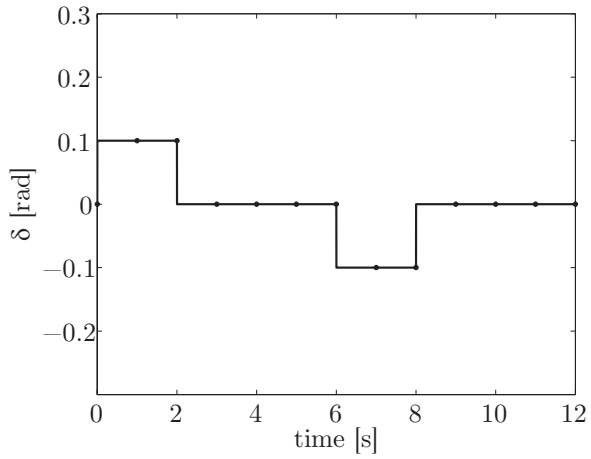
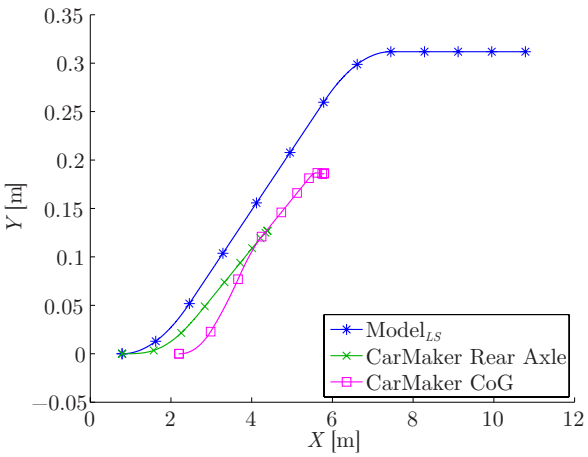


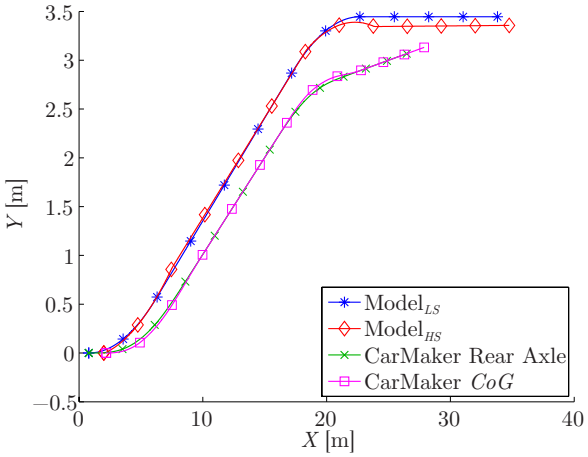
Figure 5.2. Wheel angle request for open loop comparison.

Figure 5.3a, 5.3b, 5.4a and 5.4b shows positions for the different cases. Since there was no force applied, the vehicle in CarMaker slows down quickly at low speeds, this is most clear in Figure 5.3a. Neither model$_{LS}$ nor model$_{HS}$ have models for mechanisms such as air force drag and friction losses. This is, however, featured in a high fidelity model such as CarMaker. This was the reason why the initial v_x was changed and not F_x. Even a small increase of F_x above zero would make model$_{LS}$ continue to accelerate forever which is simply not realistic.

At a speed around 10 km/h the models behave the same which is shown by Figure 5.3b. For high speeds the model$_{HS}$ behaves much better than model$_{LS}$ which is shown in Figure 5.4a and 5.4b. The results of model$_{HS}$ is not presented for 3 km/h since the discretization was determined to be unstable in this region.

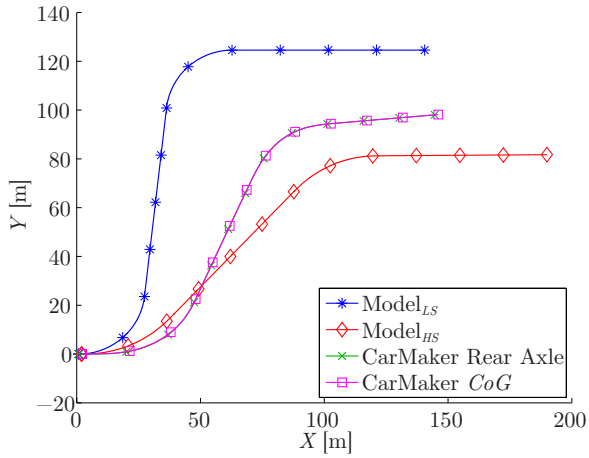


(a) Initial speed of 3 km/h.

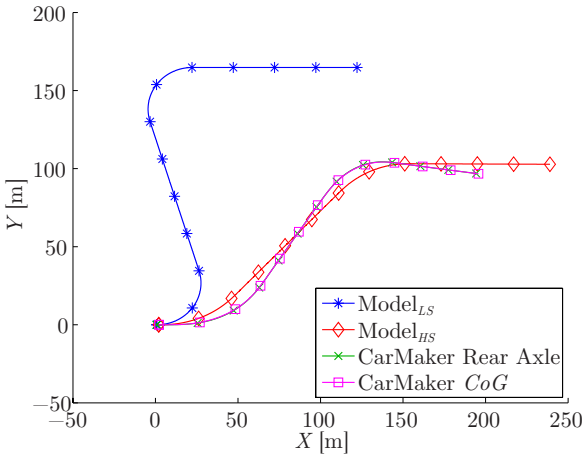


(b) Initial speed of 10 km/h.

Figure 5.3. Open loop comparison between the internal models and the simulation model for 3 and 10 km/h.



(a) Initial speed of 70 km/h.



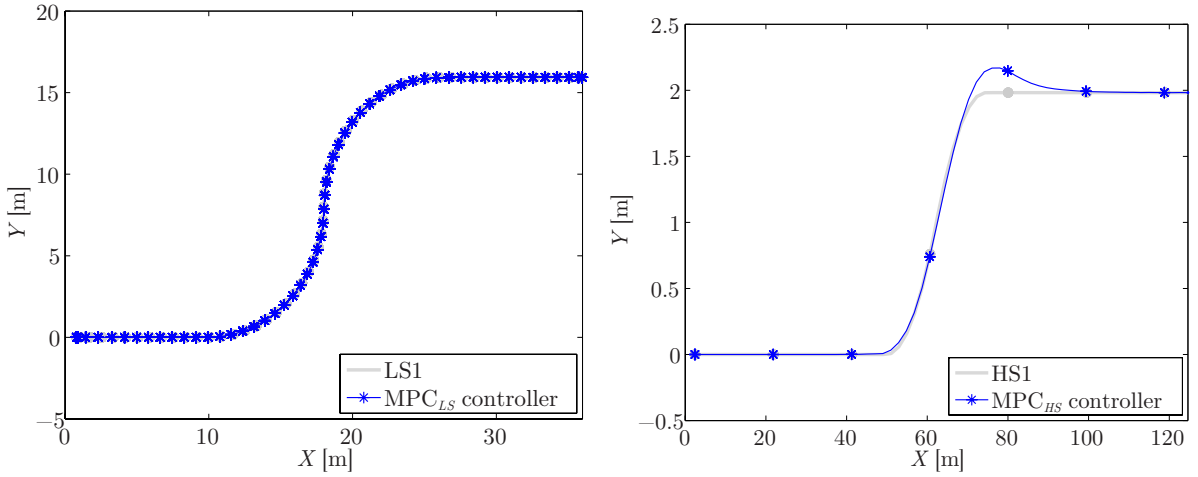
(b) Initial speed of 90 km/h.

Figure 5.4. Open loop comparison between the internal models and the simulation model for 70 and 90 km/h. Notice that the CarMaker Rear Axle line and the CarMaker CoG line are very similar for these speeds.

5.3 Validation of Framework using Simulations

First, the MPC framework was tested by simulating the vehicle on test track LS1 and HS1 with $T = 0.1$. Figure 5.5a shows how the MPC$_{LS}$ controller follows test track LS1 in 3 km/h. Figure 5.5b shows how the MPC$_{HS}$ controller follows test track HS1 in 70 km/h. The paths were considered to be followed satisfactory. Secondly, different weights were tested. This was done both to increase the understanding of the MPC controller and to verify that it behaves acceptably, hence validating the framework. If a deviation is weighted more, the MPC controller was expected to prioritize to minimize that deviation. Multiple simulations were run with a wide range of weights.

The impact of different weights, is more evident in the beginning of the simulation. After a short time with different inputs the comparison of the systems becomes complicated. The impact of the weights will be clearly visible only when the errors of the other states do not take over.



(a) MPC$_{LS}$ controller follows the test track LS1 in 3 km/h.

(b) MPC$_{HS}$ controller follows the test track HS1 in 70 km/h.

Figure 5.5. Path following by the MPC controllers on test track LS1 and HS2. The markers are placed 1 s apart.

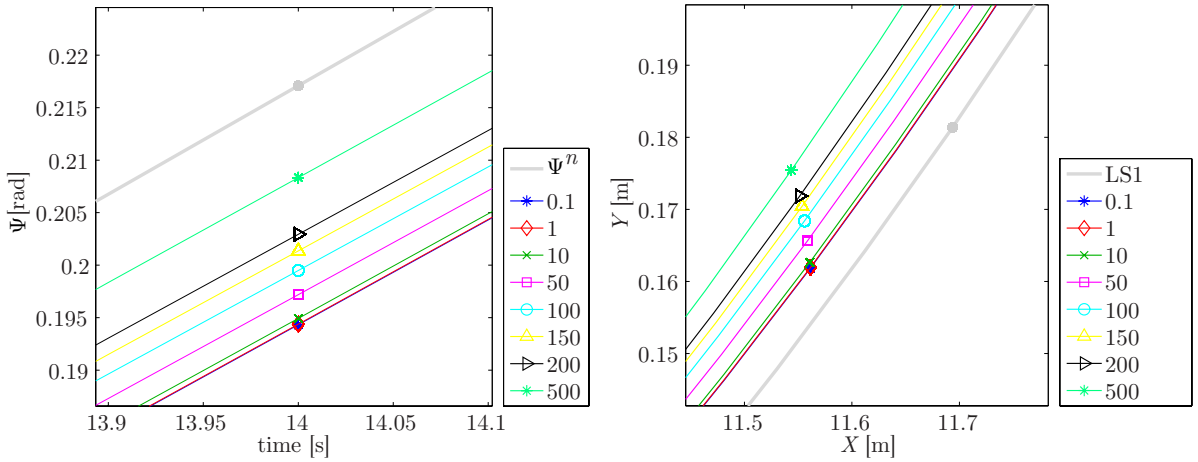
Weights on $\tilde{\Psi}$ for the MPC$_{LS}$ controller

For this test $Q_{\tilde{v}_x}$ was set to 10 while $Q_{\tilde{\Psi}}$ was varied between 0.1 and 500 as shown in Table 5.1. The exact values of $Q_{\tilde{\Psi}}$ were {0.1, 1, 10, 50, 100, 150, 200, 500} but will henceforth be referred to as the sequence {0.1,...,500}.

Table 5.1. Parameters used to evaluate weights on $\tilde{\Psi}$.

Parameter	Values
H_p	2 s
Q^d	{100, 100, {0.1,...,500}, 10}
R^d	{1, 10^{-5}}

The resulting behavior of Ψ and the positioning is shown in Figure 5.6. Figure 5.6a shows Ψ zoomed in at the first turn of test track LS1. The line with the highest $Q_{\tilde{\Psi}}$ is the closest to the nominal values. The other lines are in descending order which corresponds well to the expectation.



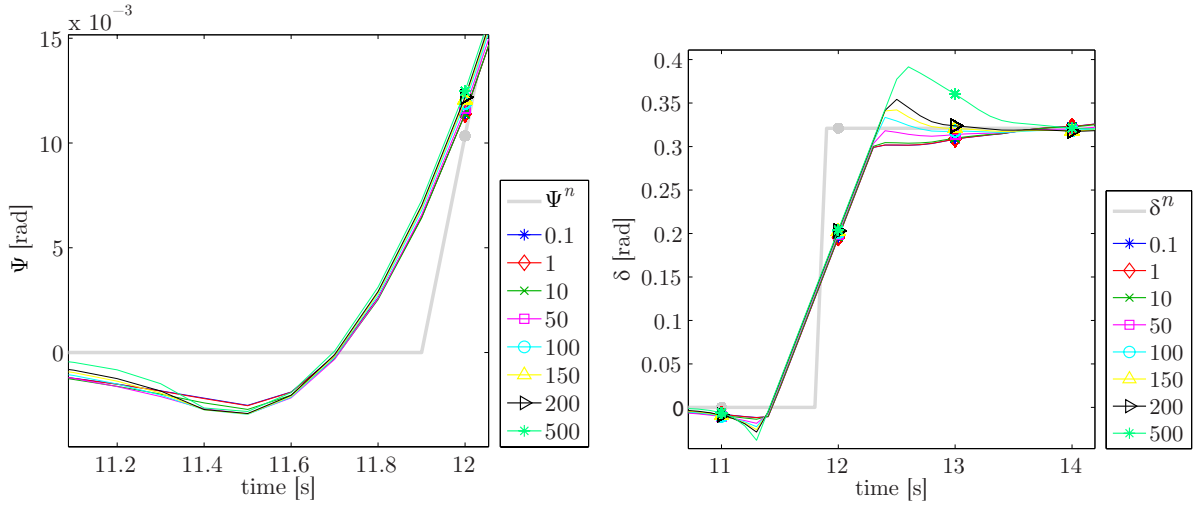
(a) Behavior of Ψ at the first turn, shows that a higher $Q_{\tilde{\Psi}}$ is closer to Ψ^n.

(b) Position in the first turn. A low $Q_{\tilde{\Psi}}$ results in a better lateral position and a higher $Q_{\tilde{\Psi}}$ more corner cutting.

Figure 5.6. Results of different $Q_{\tilde{\Psi}}$. Each line corresponds to a different $Q_{\tilde{\Psi}}$.

In all the simulations the vehicle was behind in time when entering the first curve. This was mostly due to the fact that some weight was put on $\tilde{F}_x$, which has zero as a nominal value at constant speed. If the vehicle is behind in time, a high $Q_{\tilde{\Psi}}$ forces the vehicle to turn too early and cut the corner. This is illustrated by Figure 5.6b which is a zoomed view of the first curve at approximately the same time as in Figure 5.6a. Although a lower $Q_{\tilde{\Psi}}$ gives a better lateral position the corner cutting phenomenon helps the vehicle keep up with the reference in time. This can be seen in Figure 5.6b from looking at the distance between respective marker and the gray dot.

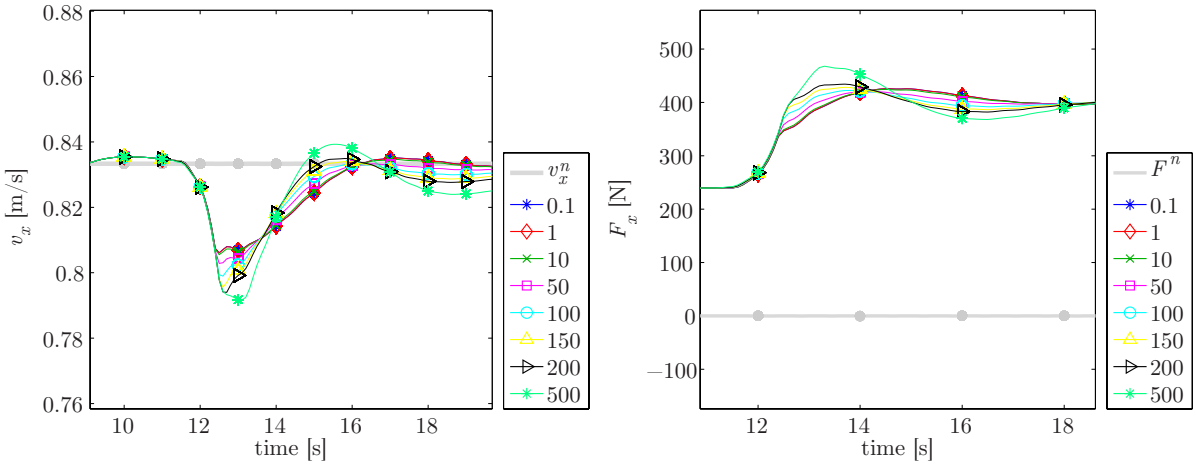
A higher $Q_{\tilde{\Psi}}$ also makes the vehicle more reluctant to make minor adjustments of the lateral position, thus making it more stable. Ψ^n makes a rapid change when the first turn begins, see Figure 6.1a. A too high $Q_{\tilde{\Psi}}$ will make the controller try to make the vehicle turn rapidly as well, see Figure 5.7a. This shows that the line with the highest $Q_{\tilde{\Psi}}$ will make the sharpest turn. To achieve this, the control signal has a larger amplitude and makes rapid changes which can be seen in Figure 5.7b. However, this impacts the comfort performance criterion negatively. In addition, the vehicle loses speed when it turns, which is visible in Figure 5.8a. The longer or sharper the turn, the more speed is lost. When this happens the MPC_{LS} controller responds with an increase in F_x, shown in Figure 5.8.



(a) Behavior of Ψ. A higher $Q_{\tilde{\Psi}}$ is closer to Ψ^n.

(b) Behavior of δ to give the behavior in Figure 5.7a.

Figure 5.7. Behavior of Ψ and δ for $Q_{\tilde{v}_x} = 10$ and varying $Q_{\tilde{\Psi}}$. Each line corresponds to a different $Q_{\tilde{\Psi}}$.



(a) Behavior of v_x. A higher $Q_{\tilde{\Psi}}$ results in more loss in speed.

(b) Behavior of F_x. A higher $Q_{\tilde{\Psi}}$ results in more applied F_x due to loss in speed.

Figure 5.8. Behavior of v_x and F_x for $Q_{\tilde{v}_x} = 10$ and varying $Q_{\tilde{\Psi}}$.

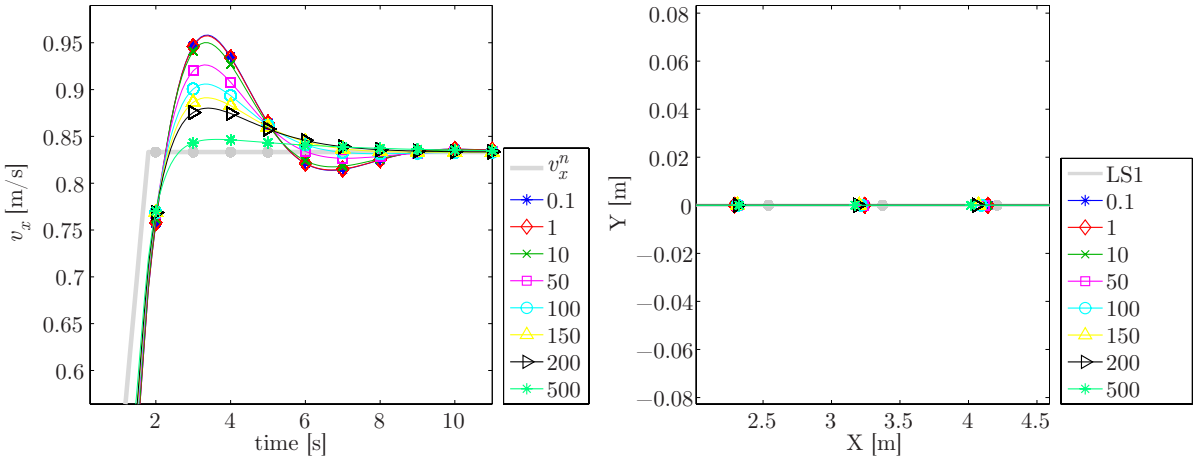
Weights on $\tilde{v}_x$ for the MPC$_{LS}$ controller

Here $Q_{\tilde{\Psi}}$ were set to 100 while the $Q_{\tilde{v}_x}$ were varied between 0.1 and 500 as shown in Table 5.2.

Table 5.2. Parameters used to evaluate weights on $\tilde{v}_x$.

Parameter	Values
H_p	2 s
Q^d	{100, 100, 100, {0.1,...,500}}
R^d	{1, 10^{-5}}

Figure 5.9a shows that during the acceleration phase a higher $Q_{\tilde{v}_x}$ results in a speed closer to the desired speed. This however, results in a suffering position where a high $Q_{\tilde{v}_x}$ means it ends up falling behind the reference, see Figure 5.9b. Hence, if the controller is not allowed to accelerate the vehicle will never catch up to the desired position.



(a) Behavior of v_x during the acceleration phase. More weight results in a speed closer to the nominal value.

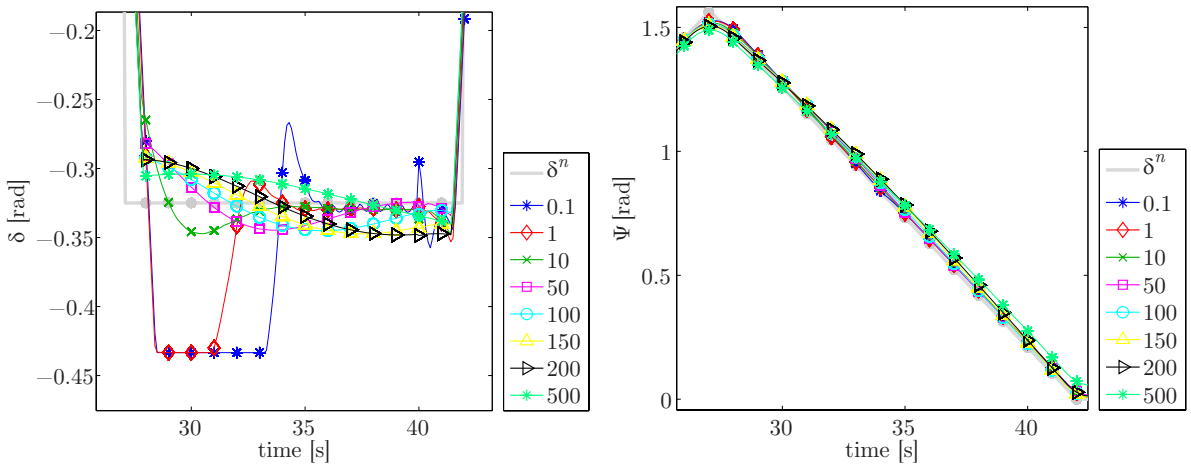
(b) Position during the acceleration phase. More weight forces the vehicle to fall behind.

Figure 5.9. Results of different $Q_{\tilde{v}_x}$. Each line correspond to a different $Q_{\tilde{v}_x}$.

Weights on $\tilde{\delta}$ for the MPC$_{LS}$ controller

By keeping $R_{\tilde{F}_x}$ at 10^{-5} while varying $R_{\tilde{\delta}}$ between 0.1 and 500, δ and Ψ were examined. Figure 5.10a shows that δ turns rapidly to follow its nominal value for a large $R_{\tilde{\delta}}$. For low $R_{\tilde{\delta}}$ oscillations appear on δ and it reaches a constraint at sample 29, seen in Figure 5.10a. The smoothest settling of δ corresponds to a $R_{\tilde{\delta}}$ of 10.

Although Figure 5.10a shows a varying set of scenarios for δ, this does not seem to affect Ψ nearly as much, see Figure 5.10b. Ψ lies close to the nominal value for all values of $R_{\tilde{\delta}}$. It is however, noticeable in Figure 5.10b that the Ψ corresponding to the largest value of $R_{\tilde{\delta}}$ lies the furthers from the nominal value.



(a) Behavior of δ. Low $R_{\tilde{\delta}}$ causes δ to hit a minimum constraint.

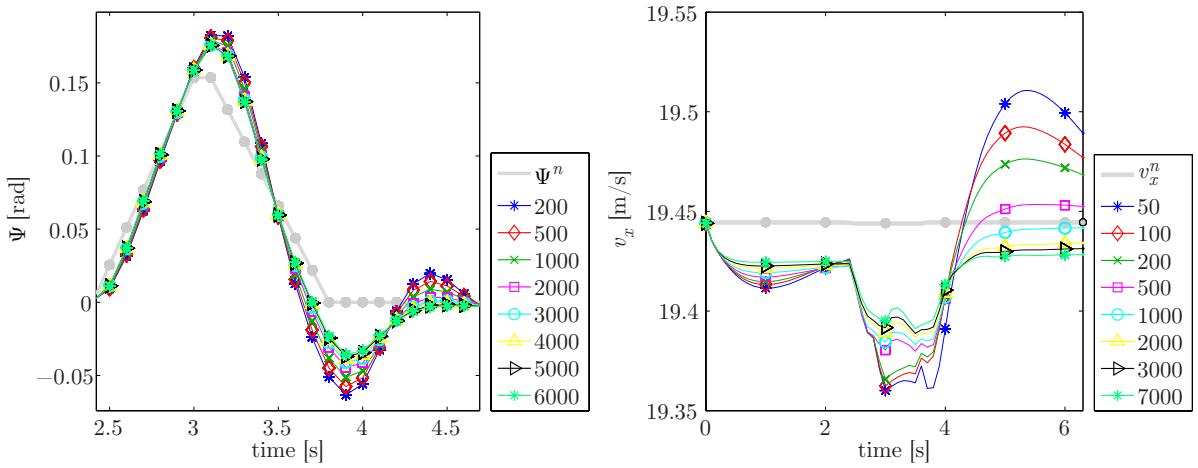
(b) Behavior of Ψ. High $R_{\tilde{\delta}}$ causes Ψ to move further from the nominal value.

Figure 5.10. Results of different weights on $R_{\tilde{\delta}}$. Each line corresponds to a different $R_{\tilde{\delta}}$.

It was assumed that increasing $R_{\tilde{\delta}}$ would increase both P_p and P_c while a high $R_{\tilde{F}_x}$ would decrease it. This was due to that δ^n were considered to fit the simulation model rather well while F_x^n were considered to fit poorly. Due to the small range of $R_{\tilde{F}_x}$ and the fact that the effect has been mentioned in connection to the other weights it has not been investigated further.

Weights for the MPC$_{HS}$ controller

For the MPC$_{HS}$ controller $Q_{\tilde{\Psi}}$ and $Q_{\tilde{v}_x}$ were varied to verify the framework with model$_{HS}$. Figure 5.11 shows that the most heavily weighted is closest to the reference as expected. Due to the fact that weighing each deviation has the expected effect, the MPC framework was considered validated.



(a) Varying values of $Q_{\tilde{\Psi}}$. The highest weight results in a behavior closest to the nominal values.

(b) Varying values of $Q_{\tilde{v_x}}$. The highest weight results in a behavior closest to the nominal values.

Figure 5.11. Varying values of $Q_{\tilde{\Psi}}$ and $Q_{\tilde{v}_x}$ to validate MPC$_{HS}$ controller on test track HS1.

6 Tuning of Weight Matrices for MPC

To tune H_p and weights, multiple tests were simulated on test track LS1 and HS1. The aim was to find the set of weights both for $\mathbf{x}$ and $\mathbf{u}$ that resulted in the lowest P_p. If multiple set of weights resulted in the same P_p their P_c would be prioritized.

6.1 Test Setup

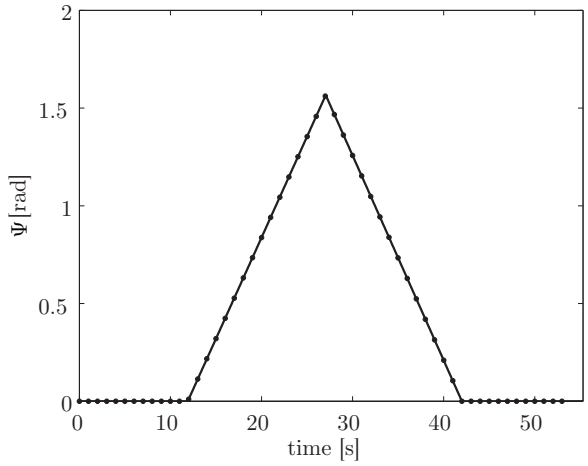
The MPC$_{LS}$ was tested on test track LS1 while the MPC$_{HS}$ was tested on test track HS1. The MPC$_{LS}$ controller was supposed to drive according to a speed profile consisting of three parts. The vehicle was initially at rest and accelerated to a desired constant speed to finally decelerate to a complete halt at the end of the track. The speed profile consists of an acceleration phase of 2 s, a phase of constant speed of 3 m/s and a deceleration phase of 2 s, which is shown in Figure 6.1b. The speed profile was designed to fit the test track LS1, such that the vehicle was to stop at the end of the track. The nominal values Ψ^n, v_x^n, δ^n and F_x^n for model$_{LS}$ are shown in Figure 6.1a, 6.2a and 6.2b respectively. The MPC$_{HS}$ controller was tested in a constant speed of 70 km/h. This is summarized in Table 6.1 and 6.2.

Table 6.1. Parameters used to tune the MPC$_{LS}$ controller.

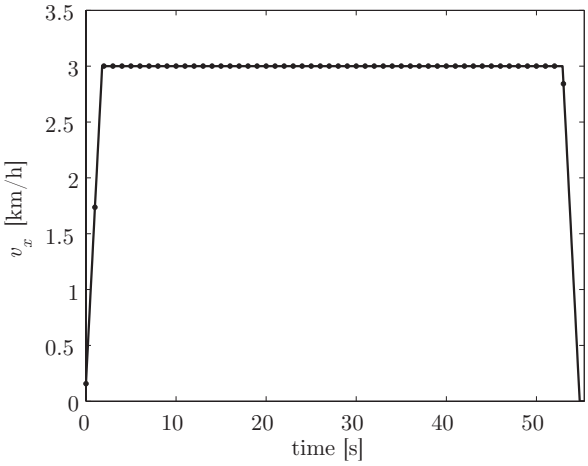
Parameter	Values
Test track	LS1
Acceleration time	2 s
Constant speed	3 km/h
Deceleration time	2 s

Table 6.2. Parameters used to tune the MPC$_{HS}$ controller.

Parameter	Values
Test track	HS1
Constant speed	70 km/h

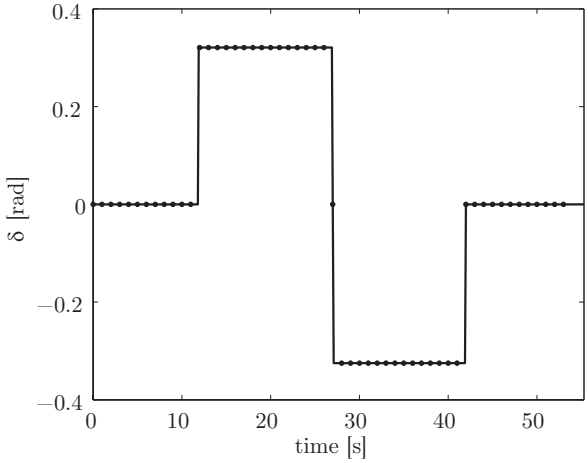


(a) Yaw angle.

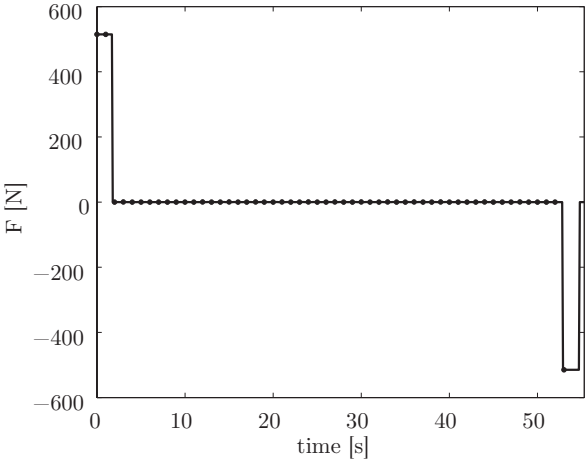


(b) Speed profile.

Figure 6.1. Nominal values for the states corresponding to LS1 and the speed profile.



(a) Front wheel angle.



(b) Longitudinal force.

Figure 6.2. Nominal values for the inputs corresponding to LS1 and the speed profile.

6.2 Tuning of Prediction Horizon

Two set of weights for $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{u}}$ were chosen. One to illustrate the impact of H_p as clearly as possible and one to follow the test track with the best performance possible. Since some of the tests did not converge, the simulations were also run with a loosened constraint on $\Delta\delta$, which is shown in Table 6.3.

Table 6.3. Parameters used to evaluate the H_p.

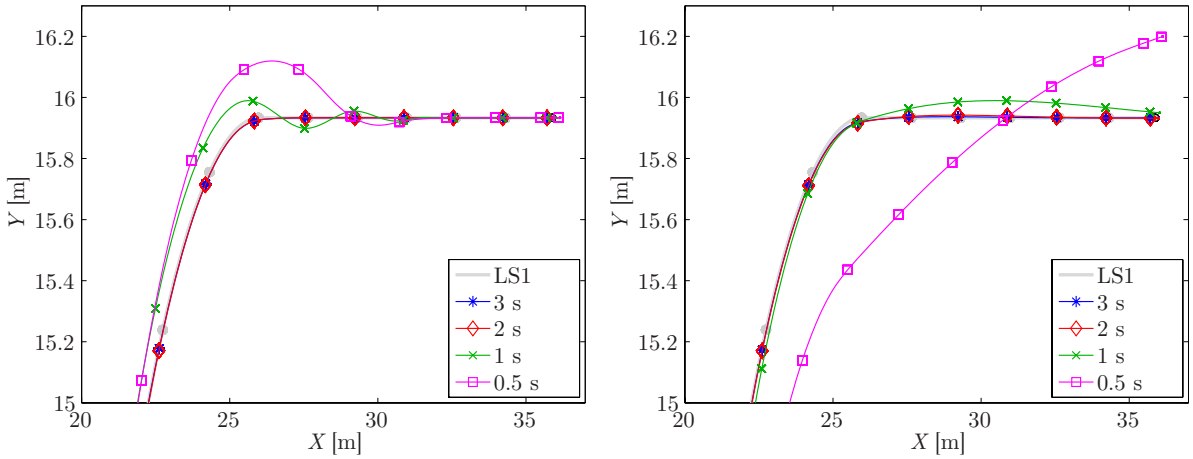
Parameter		Values
	H_p	$\{0.5, 1, 2, 3\}$ s
test 1.x	Q^d	$\{100, 100, 10, 10\}$
	R^d	$\{0.01, 10^{-5}\}$
test 2.x	Q^d	$\{100, 100, 100, 10\}$
	R^d	$\{50, 10^{-5}\}$
test x.1	$\Delta\delta$	$0.35T$
test x.2	$\Delta\delta$	$0.35T$*4

Test 1.1 describes a test with the first set of weights, $Q_d = \{100, 100, 10, 10\}$ and $R_d = \{0.01, 10^{-5}\}$ with a $\Delta\delta$ of $0.35T$. The four different H_p which were tested were: 0.5, 1, 2 and 3 s. This corresponds to 5, 10, 20 and 30 samples with a T of 0.1 s. Table 6.4 shows the results for the four tests.

Table 6.4. P_p and P_c for different H_p and weights. Non-converged results are represented with a strike-through.

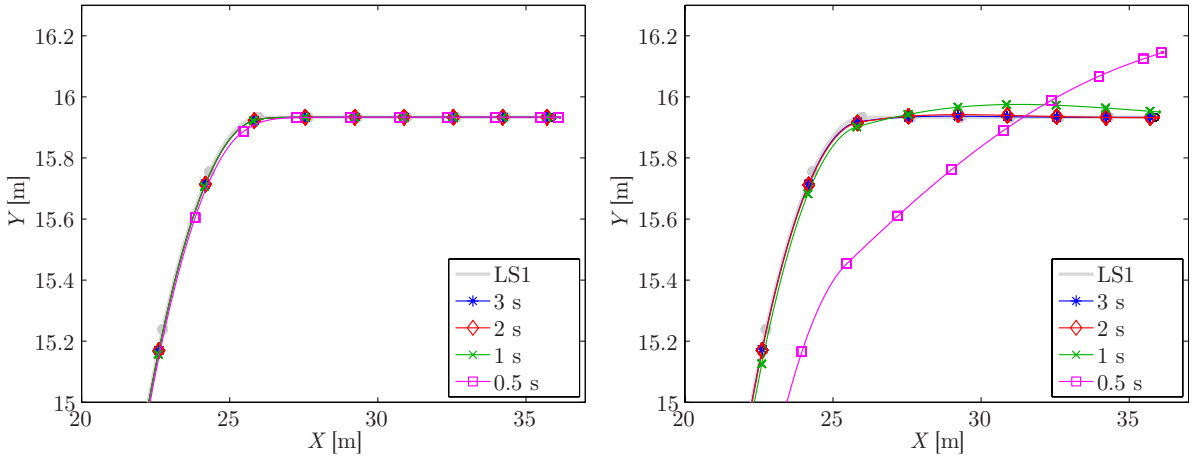
	H_p			
Test	5	10	20	30
1.1	~~43.13 | 3.83~~	~~16.11 | 7.00~~	12.33 | 10.52	12.28 | 8.56
2.1	~~47.57 | 2.26~~	~~15.25 | 2.38~~	12.14 | 2.17	11.98 | 2.15
1.2	42.54 | 7.61	14.81 | 20.13	12.44 | 31.55	12.27 | 32.28
2.2	47.29 | 2.84	14.95 | 3.07	12.31 | 3.07	12.14 | 3.08

Both a H_p of 0.5 s and 1 s have a few samples for which the problem does not converge for the normal constraints. Unfortunately there is no way of knowing whether the problem is infeasible or if it diverges due to the design of the CVX-solver. The results which do not converge are presented with a strike-through in all tables. Behavior for the two different weight setups are shown in Figure 6.3. Figure 6.4 shows the results with loosened constraints.



(a) Test 1.1. **(b)** Test 2.1

Figure 6.3. The position of the vehicle. The curves corresponds to different H_p. A higher H_p follow the path better.



(a) Test 1.2. **(b)** Test 2.2.

Figure 6.4. The position of the vehicle. The curves corresponds to different H_p. A higher H_p follow the path better. The constraint of $\Delta\delta$ is loosened by a factor four.

The difference between a H_p of 2 s and 3 s were deemed negligible for all tests while a shorter H_p loses performance and/or fails to converge for all constraints. A shorter H_p is more sensitive to both weights and constraints. Both H_p of 0.5 s and 1 s lack the information needed to avoid potentially infeasible situations caused by constraints. With a longer H_p the controller manages to plan ahead and follow all states more closely.

The reason why the constraints have less impact in test 2 is because of the higher $Q_{\tilde{\Psi}}$. Since it follows Ψ already, the constraint is not reached and does barely impact the trajectory. An H_p of 2 s was chosen for the MPC$_{LS}$ controller.

Due to the fact that an H_p of both 0.5 and 1 s were deemed too short in low speed, the assumption was that it would be too short in high speed as well. An H_p of 3 s instead of 2 s would, for the MPC$_{HS}$ controller significantly increase the size of the optimization problem. The optimization problem for 3 s is about 150 % larger than the optimization problem for 2 s. The tuning of the weights of the MPC$_{HS}$ controller was done with a H_p of 2 s. After tuning the weights, the choice of the H_p was validated to confirm that no major increase in performance would be gained from a longer H_p.

6.3 Tuning of Weights on State and Input Deviations

Tuning of the weights were done by multiple simulations. The values of P_p and P_c were used as criteria. Values for both these performances were summarized in tables and plots to obtain the weights resulting in the lowest values.

Weights on State Deviations for the MPC$_{LS}$ controller

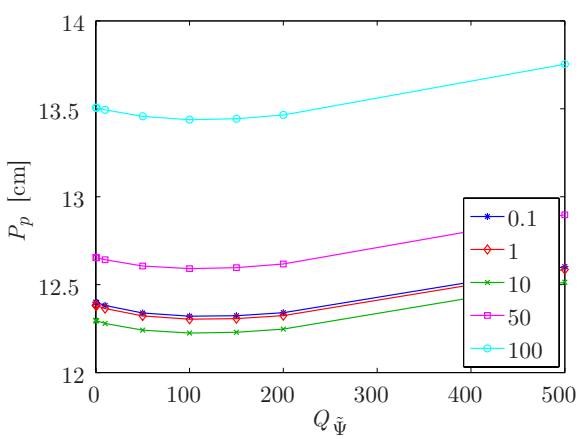
To tune the weights on the state deviations, weights on $\tilde{\mathbf{u}}$ were set to arbitrary values and the H_p that had previously been determined was used. Since it is the relationship between the weights that matter, $Q_{\tilde{X}}$ and $Q_{\tilde{Y}}$ were set to 100 while the $Q_{\tilde{\Psi}}$ and $Q_{\tilde{v}_x}$ were varied. The parameters are found in Table 6.5. The results are shown in Table 6.6 and by Figure 6.5. The first value of each column represent P_p while the second represent P_c.

Table 6.5. Parameters used to evaluate weights for state deviations.

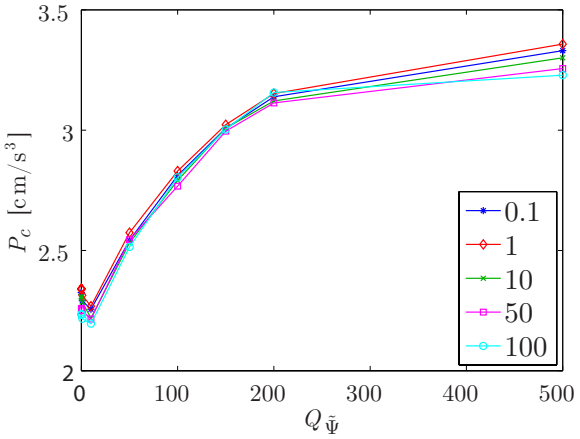
Parameter	Values
H_p	2 s
Q^d	{100, 100, {0.1,...,500}, {0.1,...,500}}
R^d	$\{1, 10^{-5}\}$

Table 6.6. P_p and P_c for different weights on $\tilde{\Psi}$ and $\tilde{v}_x$.

	$Q_{\tilde{v}_x}$				
$Q_{\tilde{\Psi}}$	0.1	1	10	50	100
0.1	12.40 | 2.32	12.38 | 2.34	12.30 | 2.31	12.65 | 2.26	13.51 | 2.23
1.0	12.40 | 2.29	12.38 | 2.31	12.29 | 2.29	12.65 | 2.25	13.50 | 2.22
10.0	12.38 | 2.26	12.36 | 2.27	12.28 | 2.22	12.64 | 2.21	13.49 | 2.20
50.0	12.34 | 2.54	12.32 | 2.57	12.24 | 2.53	12.61 | 2.54	13.46 | 2.51
100.0	12.32 | 2.81	12.30 | 2.83	12.22 | 2.79	12.59 | 2.77	13.44 | 2.80
150.0	12.32 | 3.01	12.31 | 3.02	12.23 | 3.01	12.60 | 3.00	13.44 | 3.00
200.0	12.34 | 3.14	12.32 | 3.15	12.25 | 3.12	12.62 | 3.11	13.46 | 3.16
500.0	12.60 | 3.33	12.59 | 3.36	12.51 | 3.30	12.90 | 3.26	13.75 | 3.23



(a) Visualization of P_p from Table 6.6. Each line corresponds to a different $Q_{\tilde{v}_x}$.



(b) Visualization of P_c from Table 6.6. Each line corresponds to a different $Q_{\tilde{v}_x}$.

Figure 6.5. Test references for the states corresponding to LS1 and the speed profile.

Table 6.6 and Figure 6.5a visualizes the effect on P_p. A minimum for $Q_{\tilde{\Psi}}$ was found at 100 and for $Q_{\tilde{v}_x}$ at 10. Figure 6.5b and Table 6.6 visualizes the effect on P_c. The minimum for $Q_{\tilde{\Psi}}$ was located at 0.1 and for $Q_{\tilde{v}_x}$ at 100. Hence, there is a trade-off between positioning and comfort. From Figure 6.5a it can be concluded that the positioning is an average of 12 cm off for the best values of $Q_{\tilde{\Psi}}$ and $Q_{\tilde{v}_x}$. The best values of $Q_{\tilde{\Psi}}$ and $Q_{\tilde{v}_x}$ in Figure 6.5b means the offset is only about 2 cm/s^3.

Weights on Input Deviations for the MPC$_{LS}$ controller

The weights on the state deviations were set to Q^d={100,100,100,10} while varying $\tilde{\delta}$ and $\tilde{F}_x$. The parameters are shown in Table 6.7 and the results are summarized in Table 6.8.

Table 6.7. Parameters used to evaluate the weights on the input deviations.

Parameter	Values
H_p	2 s
Q^d	{100, 100, 100, 10}
R^d	{{0.1,...,500}, {10^{-3},...,10^{-5}}}

Table 6.8. P_p and P_c for different weights on $\tilde{\delta}$ and $\tilde{F}_x$. Non-converged results are represented with a strike-through.

	$R_{\tilde{F}_x}$		
$R_{\tilde{\delta}}$	10^{-3}	10^{-4}	10^{-5}
0.10	209.95 | 6.96	37.10 | 5.72	12.24 | 5.32
1.00	209.84 | 5.27	37.05 | 3.05	12.22 | 2.79
10.00	210.37 | 4.17	36.95 | 2.24	12.18 | 2.29
50.00	214.89 | 2.31	36.82 | 2.09	12.14 | 2.17
100.00	~~218.05 | 2.18~~	36.95 | 1.96	12.18 | 2.10
150.00	~~215.13 | 2.17~~	37.21 | 1.94	12.29 | 2.05
200.00	~~210.35 | 2.14~~	37.77 | 1.96	12.50 | 2.02
500.00	223.51 | 1.89	43.46 | 1.80	15.33 | 1.86

The solver does not converge for $R_{\tilde{\delta}} = \{100, 150, 200\}$ and $R_{\tilde{F}_x} = 10^{-3}$. Neither does the solver converge when F_x becomes too small e.g. $R_{\tilde{F}_x} = 10^{-6}$. The final weights were decided to Q_d={100,100,100,10}, R_d={50,10^{-5}}.

Weights on State Deviations for the MPC$_{HS}$ controller

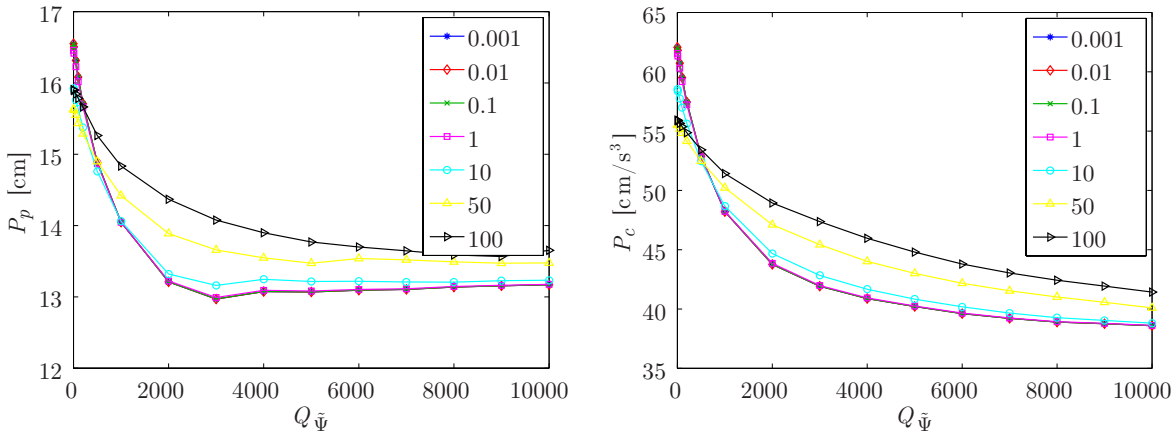
The tests for the MPC$_{HS}$ controller were performed in the same manner as those for the MPC$_{LS}$ controller but Q_d were tuned in two steps. First, $Q_{\tilde{\Psi}}$ and $Q_{\tilde{\omega}}$ were varied while the weights on both speeds were kept constant. $Q_{\tilde{v}_x}$ and $Q_{\tilde{v}_y}$ were set to 10 during the first phase. Secondly, the best values of $Q_{\tilde{\Psi}}$ and $Q_{\tilde{\omega}}$ were used while $Q_{\tilde{v}_x}$ and $Q_{\tilde{v}_y}$ were varied. $Q_{\tilde{X}}$ and $Q_{\tilde{Y}}$ were set to 100 throughout all tests. Table 6.9 shows the parameters used for this test. The results are found in Table 6.10 and Figure 6.6.

Table 6.9. Parameters used for tuning of weights on $\tilde{\Psi}$ and $\tilde{\omega}$.

Parameter	Values
H_p	2 s
Q^d	{100, 100, {0.001,...,1000}, 10, 10, {0.001,...,1000}}
R^d	{1, 10^{-5}}

Table 6.10. P_p and P_c for different weights on $\tilde{\Psi}$ and $\tilde{\omega}$.

	$Q_{\tilde{\omega}}$									
$Q_{\tilde{\Psi}}$	0.001		0.01		0.1		1		10	
200	15.71	57.45	15.71	57.45	15.70	57.43	15.66	57.22	15.37	55.63
500	14.89	52.88	14.89	52.88	14.89	52.87	14.87	52.82	14.76	52.45
1000	14.05	48.21	14.05	48.21	14.05	48.21	14.05	48.24	14.06	48.67
2000	13.21	43.76	13.21	43.76	13.21	43.77	13.22	43.85	13.32	44.67
3000	12.97	41.92	12.97	41.92	12.97	41.93	12.99	41.98	13.16	42.83
4000	13.07	40.88	13.07	40.88	13.08	40.88	13.09	40.94	13.25	41.67
5000	13.07	40.21	13.07	40.21	13.07	40.21	13.08	40.25	13.22	40.83
6000	13.09	39.61	13.09	39.61	13.09	39.62	13.10	39.66	13.22	40.19



(a) Visualization of P_p.
Each line corresponds to a different $Q_{\tilde{\omega}}$.

(b) Visualization of P_c.
Each line corresponds to a different $Q_{\tilde{\omega}}$.

Figure 6.6. P_p and P_c for different weights on $\tilde{\Psi}$ and $\tilde{\omega}$.

There is a trade-off between P_p and P_c, a larger $Q_{\tilde{\Psi}}$ generally increases the comfort but not always the positioning. The best value of $Q_{\tilde{\Psi}}$ was found at 3000 and at 0.01 for $Q_{\tilde{\omega}}$. Since $Q_{\tilde{\omega}}$ have the same value for 0.001 and 0.01 the one closest to $Q_{\tilde{X}}$ and $Q_{\tilde{Y}}$ were chosen to decrease the span of the weights. The best values of $Q_{\tilde{\Psi}}$ and $Q_{\tilde{\omega}}$ were used while $Q_{\tilde{v}_x}$ and $Q_{\tilde{v}_y}$ were varied as shown in Table 6.11. The result can be found in Figure 6.7.

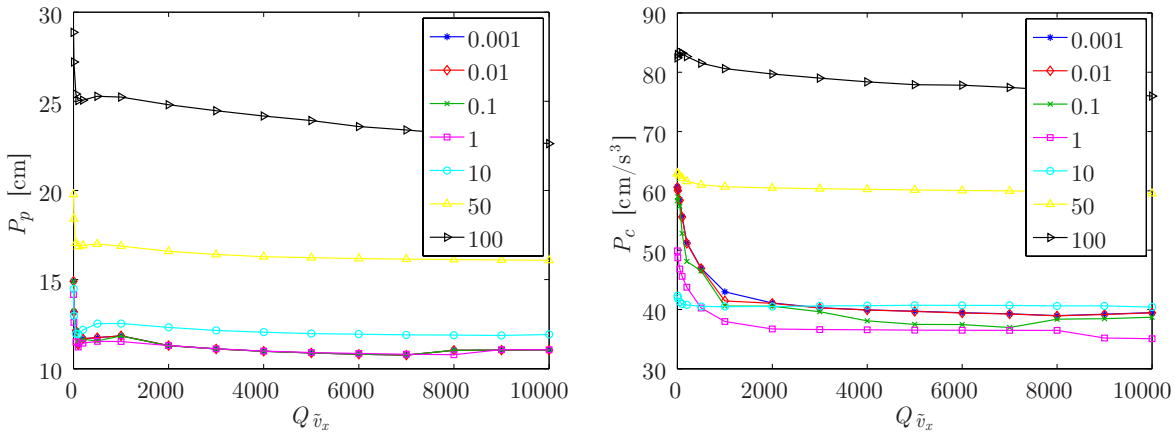
Table 6.11. Parameters used for tuning of weights on $\tilde{v}_x$ and $\tilde{v}_y$.

Parameter	Values
H_p	2 s
Q^d	{100, 100, 3000, {0.001,..,10000}, {0.001,...,10000},0.01}
R^d	$\{1, 10^{-5}\}$

Table 6.12 and Figure 6.7 shows the results of the second phase of tuning the weights for state deviations. It can be seen in Figure 6.7 that low values are found both close to 50 and around 7000, these are included in Table 6.12.

Table 6.12. P_p and P_c for different weights on $\tilde{v}_x$ and $\tilde{v}_y$.

	$Q_{\tilde{v}_y}$									
$Q_{\tilde{v}_x}$	0.001		0.01		0.1		1		10	
10	13.18	60.13	13.18	60.04	13.20	58.99	12.60	48.70	12.97	41.92
50	12.13	58.50	12.13	58.37	12.15	57.41	11.50	46.81	11.96	41.25
100	11.35	55.68	11.35	55.57	11.50	52.81	11.21	45.57	11.95	40.89
200	11.68	51.26	11.68	51.14	11.64	48.09	11.43	43.74	12.18	40.77
6000	10.82	39.44	10.81	39.40	10.81	37.45	10.85	36.48	11.94	40.68
7000	10.76	39.26	**10.76**	39.23	10.77	36.95	10.81	36.47	11.90	40.68
8000	11.03	38.95	11.03	38.91	11.03	38.36	10.78	36.46	11.89	40.61
9000	11.04	39.20	11.04	39.16	11.04	38.44	11.07	**35.18**	11.87	40.60



(a) Visualization of P_p from Table 6.12. Each line corresponds to a different $Q_{\tilde{v}_y}$.

(b) Visualization of P_c from Table 6.12. Each line corresponds to a different $Q_{\tilde{v}_y}$.

Figure 6.7. P_p and P_c for different weights on $\tilde{v}_x$ and $\tilde{v}_y$.

Table 6.12 shows that $Q_{\tilde{v}_x}$ should be set to 7000 and $Q_{\tilde{v}_y}$ to 0.01 to correspond to the lowest P_p. The determination of weights should be an iterative process were these weights are used to test the other weights again. Testing more combinations of such as $Q_{\tilde{\Psi}}$ and $Q_{\tilde{v}_x}$ should also be done iteratively. This has been excluded in the project due to time constraints. The weight determination were considered good enough to compare the MPC controllers with the P-controllers. The final weights of the MPC$_{HS}$ controller were decided to $Q^d = \{100, 100, 3000, 7000, 0.001, 0.01\}$.

Weights on Input Deviations for the MPC$_{HS}$ controller

To determine weights the parameters previously decided were used as shown in Table 6.13. The results are found in Table 6.14.

Table 6.13. Resulting parameters for the model$_{HS}$.

Parameter	Values
H_p	2 s
Q^d	{100, 100, 3000, 7000, 0.01, 0.01}
R^d	{{0.1,...,1000}, {10^{-3},...,10^{-5}}}

Table 6.14. P_p and P_c for different weights on $\tilde{\delta}$ and $\tilde{F}_x$.

	$R_{\tilde{F}_x}$			
$R_{\tilde{\delta}}$	10^{-3}	10^{-4}	10^{-5}	10^{-6}
0.01	77.50 | 123.18	19.10 | 61.29	10.76 | 39.27	11.01 | 39.13
0.1	77.50 | 123.16	19.10 | 61.29	10.76 | 39.27	11.01 | 39.13
1	77.49 | 122.97	19.10 | 61.29	10.76 | 39.23	11.01 | 39.13
10	77.41 | 121.63	19.08 | 59.24	10.77 | 36.88	10.99 | 39.08
50	77.14 | 117.83	19.04 | 58.39	10.77 | 36.30	10.95 | 37.24
100	76.79 | 114.77	18.95 | 57.02	10.80 | 35.95	10.90 | 36.80
150	76.44 | 113.17	18.81 | 54.95	10.82 | 35.70	10.87 | 36.31
200	77.27 | 111.37	18.67 | 52.70	10.84 | 35.59	10.85 | 36.30

Figure 6.8 shows the effect on δ with different $R_{\tilde{\delta}}$, which implies that the MPC$_{HS}$ controller take the weighting into consideration and tries to follow the nominal values. The oscillating behavior when $R_{\tilde{\delta}}$ is very low is visible in Figure 6.8 and in Table 6.14 where the comfort makes a jump for $Q_{\tilde{\delta}}$ between 1 and 10. Since the comfort is a secondary priority for evasive maneuvers the weights with the lowest P_p were still chosen. The weights were finally decided to $Q^d = \{100, 100, 3000, 7000, 0.001, 0.01\}$, R^d=$\{1, 10^{-5}\}$.

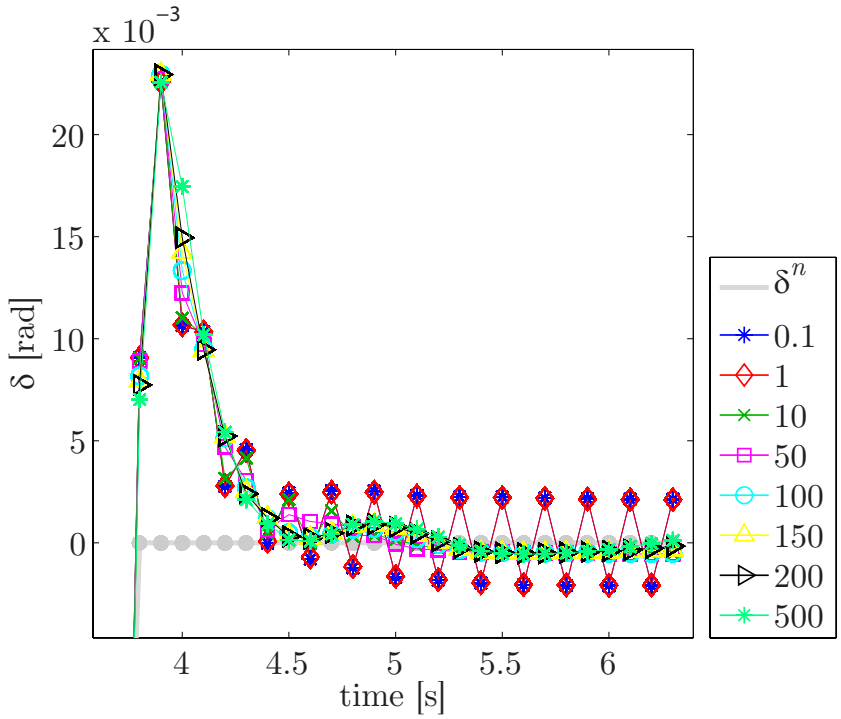


Figure 6.8. Zoomed in position of test 1.

Prediction Horizon Validation for the MPC$_{HS}$ controller

A short validation was done on the decided H_p of 2 s to verify that an increased length would not drastically improve the performance. The four different H_p which were tested were: 0.5, 1, 2 and 3 s. For a sampling time of 0.1 s the tests correspond to a H_p of 5, 10, 20 and 30 samples. The results are shown in Table 6.15.

Table 6.15. Results of different prediction horizons.

H_p			
5	10	20	30
12.92 | 48.32	11.59 | 38.89	10.76 | 39.23	10.35 | 38.70

As expected, a 3 s H_p did result in better values but the increase was not deemed to motivate the increase in calculations and 2 s was considered a valid choice. The vehicle follows the path as in Figure 5.5b for the final weights.

6.4 Resulting tuning for the MPC Controllers

Table 6.16. Resulting parameters for the model$_{LS}$.

Parameter	Values
H_p	2 s
Q^d	{100, 100, 100, 10}
R^d	{50, 10^{-5}}

Table 6.17. Resulting parameters for the model$_{HS}$.

Parameter	Values
H_p	2 s
Q^d	{100, 100, 3000, 7000, 0.01, 0.01}
R^d	{1, 10^{-5}}

7 Comparison Between MPC controllers and P-controllers in Simulation

Setup and results from the comparison of the MPC controllers and the P-controllers are presented in this chapter. The comparison is separated for the MPC_{LS} controller and the MPC_{HS} controller. In the setup, weights for the MPC controllers and different test tracks are presented. For the low speed comparison, the performance criteria P_l, P_c and P_d were used for evaluation. The high speed comparison was evaluated by P_l, P_c and P_d but also by P_p in relation to P_l.

7.1 Setup

The comparison of the MPC_{LS} controller and the P-controller_{LS} was performed on LS2. The MPC_{HS} controller and the P-controller_{HS} were tested on the track they were tuned for, HS1, but also on HS2.

The MPC controllers used an H_p of 2 s and the final tunings presented in Table 6.16 and 6.17. The P-controllers were tuned for the same paths as the corresponding MPC controllers. This evaluation can be found in appendix E, however, the preview distances (d_p) for the P-controllers are summarized in Table 7.1.

Table 7.1. Summary of preview distances for the speeds 3 and 70 km/h.

Speed [km/h]	d_p [dm]	Test track
3	49	LS
70	80	HS

To evaluate the robustness of the MPC_{HS} controller in comparison to the P-controller_{HS}, they were tested on HS1 in 110 and 130 km/h. The same test was also evaluated for a H_p of 3 s. The tuning and d_p, however, were unchanged from the tuning case of 70 km/h.

7.2 Results of Comparison

The MPC_{LS} controller performs better than the P-controller_{LS} in every aspect according to Table 7.2. The performance is also shown in Figure 7.1 where it is clear that the MPC_{LS} controller follows the path more closely.

Table 7.2. Performance criteria for the MPC_{LS} controller and the P-controller_{LS} at 3 km/h on LS2.

		P_l [cm]	P_c [cm/s^3]	P_d [cm]
3 km/h	P-controller_{LS}	30.47	3.01	80.32
	MPC_{LS} controller	4.54	2.31	33.30

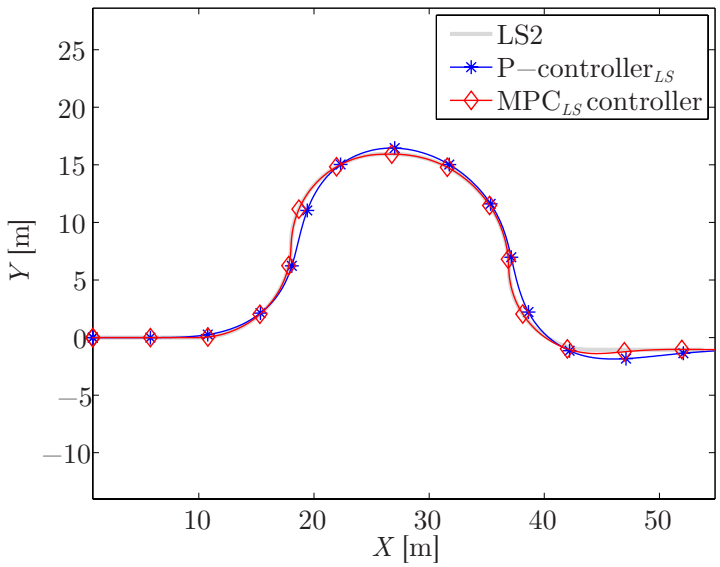


Figure 7.1. The MPC$_{LS}$ controller and the P-controller$_{LS}$ at 3 km/h on LS2.

Table 7.3 and 7.4 shows that the P-controller$_{HS}$ performs better in every aspect shown compared to the MPC$_{HS}$ controller independent of speed. However, the comparison is more complicated in high speed than low speed. When the vehicle turns and speed is lost, the P-controller$_{HS}$ will try to increase the speed to reach the desired speed again. The MPC$_{HS}$ controller on the other hand will in addition try to increase the speed further to take back lost time. This might, however, have a negative effect on the lateral position.

Table 7.3. Performance criteria, P_l, P_c and P_d for the MPC$_{HS}$ controller and the P-controller$_{HS}$ at 40, 70 and 100 km/h on HS1.

		P_l [cm]	P_c [cm/s^3]	P_d [cm]
40 km/h	P-controller$_{HS}$	4.75	4.73	17.44
	MPC$_{HS}$ controller	7.75	10.58	13.53
70 km/h	P-controller$_{HS}$	3.41	34.51	11.08
	MPC$_{HS}$ controller	10.76	39.23	23.11
100 km/h	P-controller$_{HS}$	14.56	104.70	64.61
	MPC$_{HS}$ controller	58.84	139.39	211.51

Table 7.4. Performance criteria, P_l, P_c and P_d for the MPC$_{HS}$ controller and the P-controller$_{HS}$ at 40, 70 and 100 km/h on HS2.

		P_l [cm]	P_c [cm/s^3]	P_d [cm]
40 km/h	P-controller$_{HS}$	3.62	0.79	6.30
	MPC$_{HS}$ controller	12.16	1.36	21.68
70 km/h	P-controller$_{HS}$	2.61	4.63	5.01
	MPC$_{HS}$ controller	12.57	8.18	24.49
100 km/h	P-controller$_{HS}$	4.42	18.68	8.80
	MPC$_{HS}$ controller	16.34	104.62	28.84

Table 7.5 presents values for P_l and P_p for the P-controller$_{HS}$ and the MPC$_{HS}$ controller. The table shows that for most cases the MPC$_{HS}$ controller loses almost none of its lateral positioning. The P-controller$_{HS}$ is constructed without any time variance which is also clear in the table. The MPC$_{HS}$ controller is tuned after P_p while the P-controller$_{HS}$ is tuned after P_l. The MPC$_{HS}$ controller and the P-controller$_{HS}$ are shown driving on HS1 in 70 km/h in Figure 7.2.

Table 7.5. Performance criteria, P_l and P_p for the MPC$_{HS}$ controller and the P-controller$_{HS}$ at 40, 70 and 100 km/h on HS1.

		P_l [cm]	P_p [cm]
40 km/h	P-controller$_{HS}$	4.75	5540.99
	MPC$_{HS}$ controller	7.75	7.75
70 km/h	P-controller$_{HS}$	3.41	5746.65
	MPC$_{HS}$ controller	10.76	10.76
100 km/h	P-controller$_{HS}$	14.56	6001.88
	MPC$_{HS}$ controller	58.84	134.46

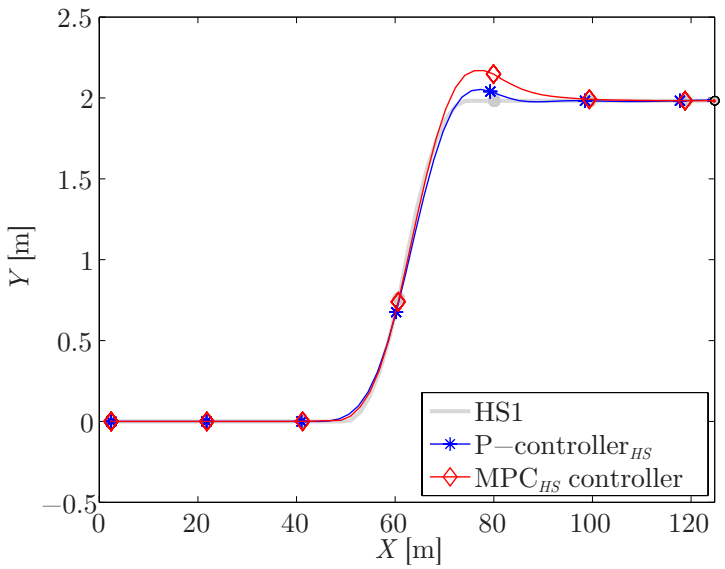


Figure 7.2. P-controller$_{HS}$ and MPC$_{HS}$ controller in 70 km/h.

The robustness results are visualized in Figure 7.3 and 7.4 for 110 km/h and 130 km/h respectively. For 110 km/h the P-controller$_{HS}$ and the MPC$_{HS}$ controller with a H_p of 2 s fail to stay on track. Hence both the MPC$_{HS}$ controller and the P-controller$_{HS}$ have some robustness issues. However, the MPC$_{HS}$ controller with a H_p of 2 s did not converge at all points.

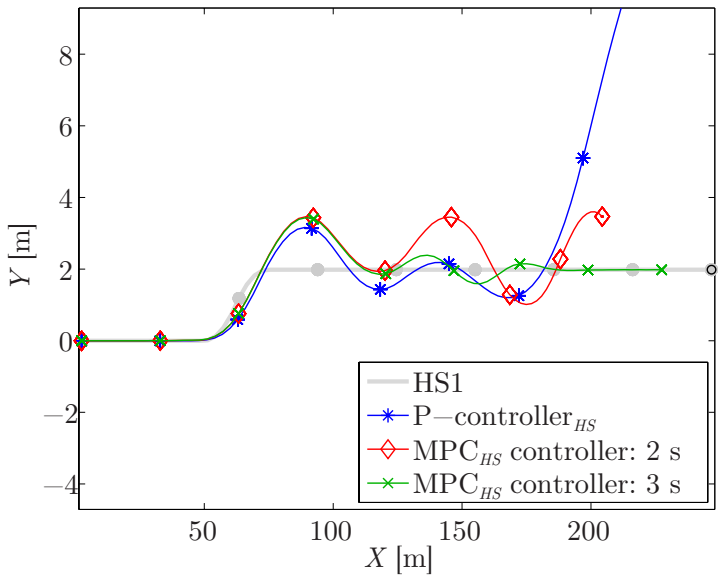


Figure 7.3. P-controller$_{HS}$ and MPC$_{HS}$ controller in 110 km/h.

For 130 km/h both MPC controllers converged for all values.

Figure 7.4 shows that the MPC$_{HS}$ controllers manages to stay on track for both H_p while the P-controller$_{HS}$ does not. When the MPC$_{HS}$ controller with a H_p of 3 s is about to settle it slows down and eases into the path while the MPC$_{HS}$ controller with a H_p of 2 s gets a small overshoot before it manages to reach the path.

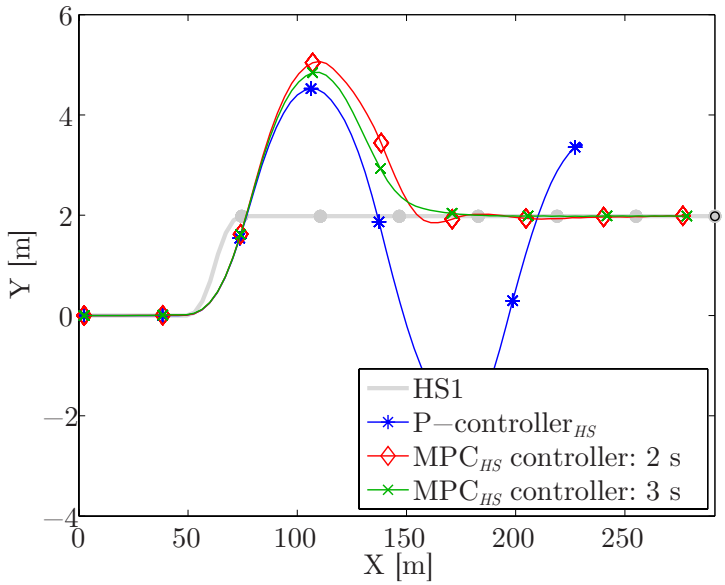


Figure 7.4. P-controller$_{HS}$ and MPC$_{HS}$ controller in 130 km/h.

Figure 7.5 is a screenshot of the MPC$_{HS}$ controller and P-controller$_{HS}$ at 130 km/h with $H_p = 3$ s at the end of HS1. The shaded vehicle represents the P-controller$_{HS}$ and the other vehicle corresponds to the MPC$_{HS}$ controller. The P-controller$_{HS}$ is behind in time and slipping.

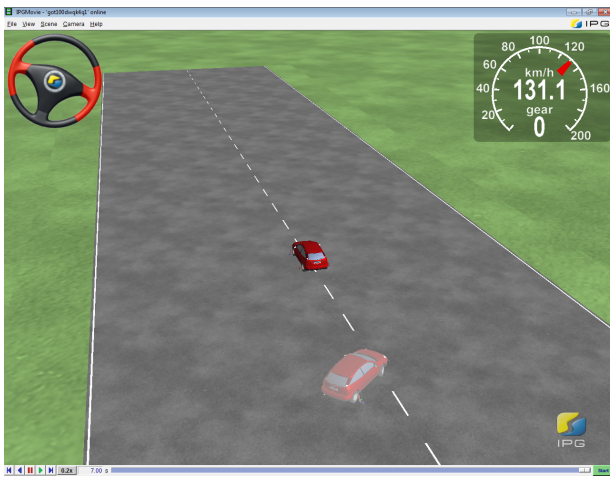


Figure 7.5. Simulation in CarMaker with P-controller$_{HS}$ and MPC$_{HS}$ controller in 130 km/h.

8 Tests in Vehicle

The test vehicle was set to follow the path from test track LS1 for low speed tests of 3 km/h. The tests were done at Volvo Cars Torslanda and at Hällered Proving Ground. Unfortunately due to technical difficulties and time constraints no test could be done with the developed P-controller. A few tests were done in speeds of 70 km/h but no viable data were collected.

8.1 Setup of Tests

The test in 3 km/h was done on large flat surfaces. The MPC_{LS} controller was implemented and data collected using dSpace control desk. The RT3000 was used to collect information about the current state. Before the test was started the vehicle was positioned with an angle as close as possible to 0 rad. The test vehicle has an automatic gearbox and hence drove forward as soon as the brake was released. The MPC controller was activated at the same time as the brake was released. When the MPC controller was activated an offset for X and Y was applied to always start from the beginning of the path. All tests were performed with a sampling interval of 0.1 s.

8.2 Results

The results for the five best tests are shown in Table 8.1 and Figure 8.1. Although the vehicle does not follow the path perfectly it clearly follows the path. The states and inputs for the test with the lowest P_p are shown, together with target values, in Figure 8.2. Figure 8.2a and 8.2b are measured signals compared to the desired path. Figure 8.2c and 8.2d are measured signals compared to the nominal values. The input signals are shown in Figure 8.2e and 8.2f. The MPC controller requests a signal, which is not necessarily the same as the signal sent to the actuator, e.g. F_x. Due to the automatic gear box, the vehicle will drive forward in a speed of around 6 km/h. For the controller to get a speed of 3 km/h, the requested force has to be negative.

Even though no data was collected the MPC_{HS} controller executed in the test vehicle. The settings for the tests can be found in appendix F.

Table 8.1. Performance criteria, P_l, P_p, P_c and P_d for the MPC_{HS} controller and the P-controller_{HS} at 40, 70 and 100 km/h on HS2.

	P_l [cm]	P_p [cm]	P_c [cm/s^3]	P_d [cm]
Vehicle test 1	31.64	50.28	2.56	87.59
Vehicle test 2	31.42	46.07	3.78	80.25
Vehicle test 3	39.98	58.19	2.64	109.70
Vehicle test 4	37.11	54.10	2.68	104.82
Vehicle test 5	24.02	44.47	2.62	86.05

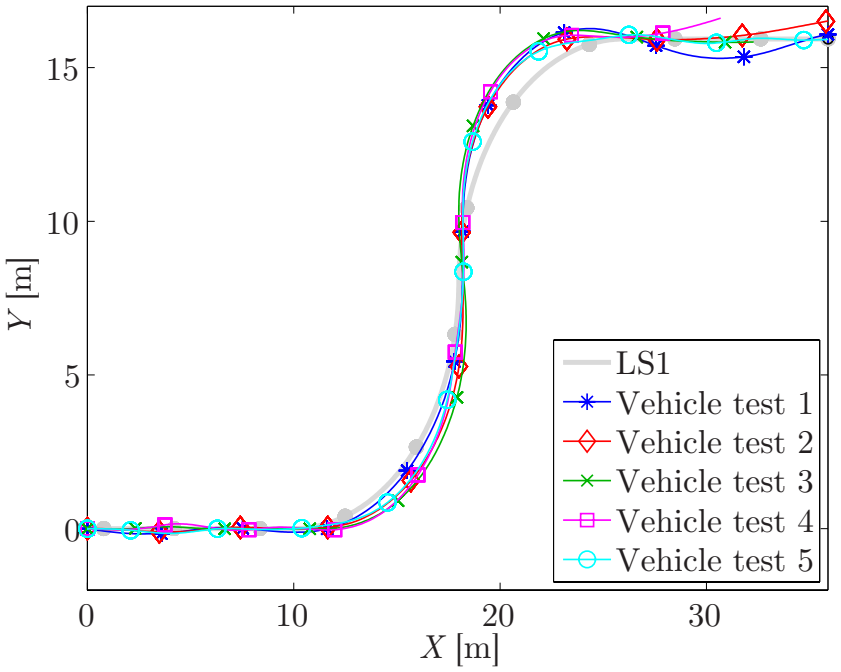
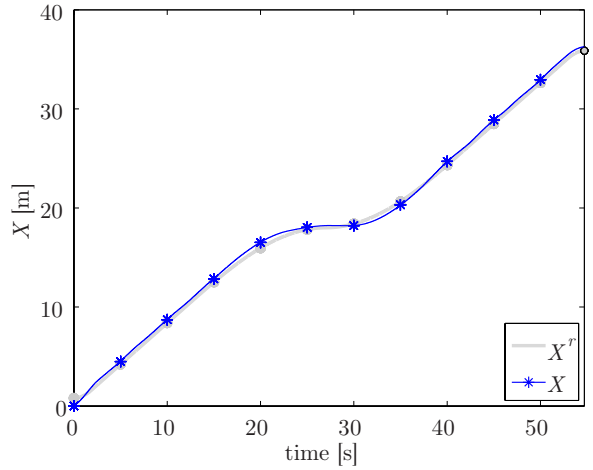
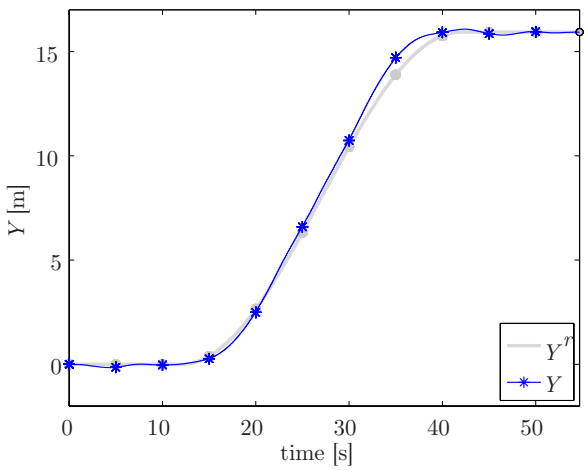


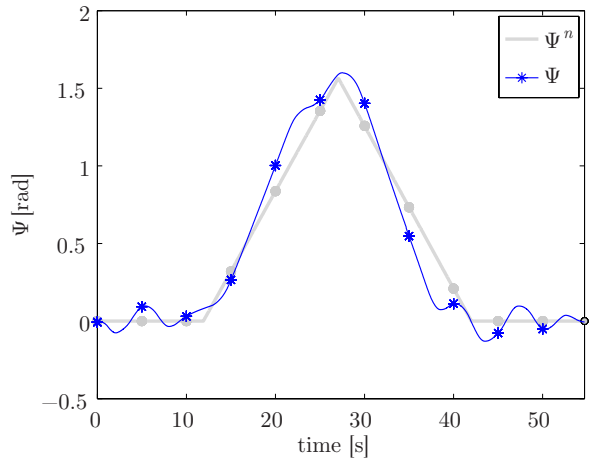
Figure 8.1. Five vehicle tests following test track LS1.



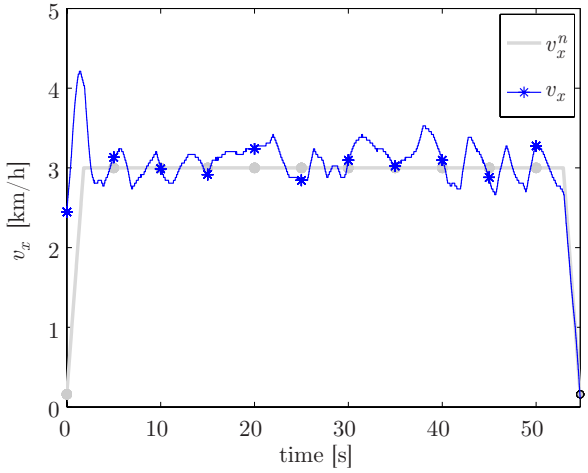
(a) Reference and global X.



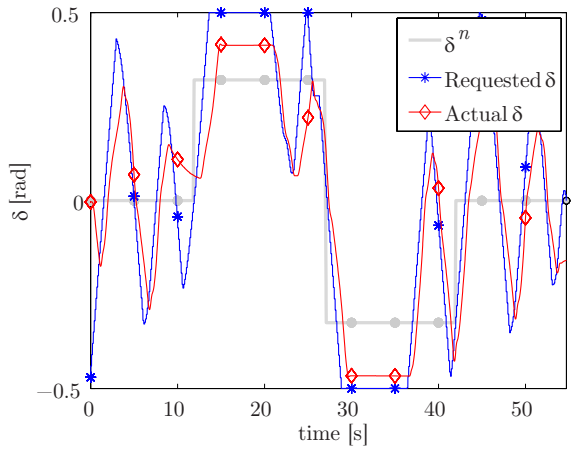
(b) Reference and global Y.



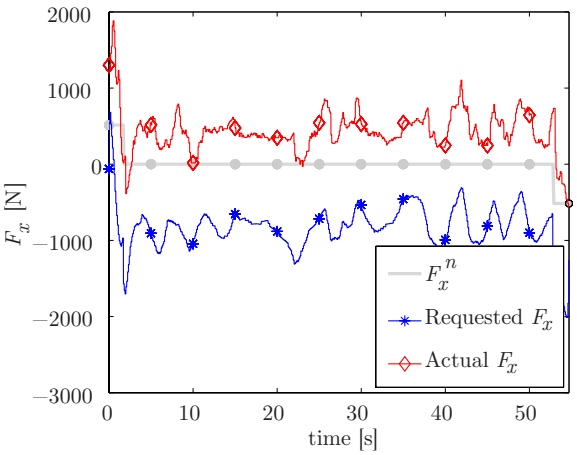
(c) Nominal value and Ψ.



(d) Nominal value and v_x.



(e) Nominal value, requested δ by MPC$_{LS}$ controller and actual δ applied.



(f) Nominal value, requested F_x by MPC$_{LS}$ controller and actual F_x applied.

Figure 8.2. Stated and inputs for the best performing vehicle test. Test track LS1 in 3 km/h.

9 Discussion

The MPC framework developed during this project has a large potential and is functional for both low and high speed applications as the results demonstrate. For low speed simulations the MPC_{LS} controller is considerably better than the P-controller_{LS}. The MPC_{LS} controller predicts the behavior in such a way that it is able to follow curves with varying radii very well. The simulations show that MPC has advantages applicable for path following. The tests in the vehicle also confirm that the hardware is sufficient to handle the calculations in real-time which is a great outcome. The fact that it executes in the vehicle is considered to be more worth than the actual performance of the vehicle, since some parameters were unknown and both measurement noise and state estimations were excluded. It is worth noticing though that the step between executing in a test vehicle and a real vehicle is quite large.

The MPC framework developed lacks coverage between the speeds 10 and 40 km/h which needs further investigation. This speed segment could probably be covered by the MPC_{HS} controller with another discretization method. This would require a shorter sampling interval as concluded from the stability analysis. Another discretization method than Euler Forward, such as the trapezoid method, Tustin's method or the Runge-Kutta method would probably increase the region of stability for the model_{HS}. However, many of the more promising discretization methods are more computationally demanding.

It is possible to change the sampling interval of the MPC. A shorter sampling interval means that the samples in H_p covers a smaller time span. Since CVXGEN constrains the number of samples in H_p, this limits the potential of the framework.

Different constraints were set for $\Delta\delta$ for the low and the high speed models. This is because high speed control demands excessive maneuvers. It was discovered that the choice of constraints had a major impact on the performance. With too strict constraints some of the simulations with the MPC controllers do not converge, whether this was due to infeasibility or divergence is unknown. This was only seen for shorter H_p. For the longer H_p, however, it seemed like the controller had time to plan and avoid these situations.

It would be possible to calculate a recovery path if the region where the linearization is valid is exceeded. This means that the system temporarily would follow a new path back to the original reference path. This has however not been investigated in this project.

9.1 Tuning

The weights have been determined by testing the effects of a limited range of different weights and choosing those with best performance. Since there are no guarantees that the found minimum is global, there are no way of knowing if the weights are optimal. When the weights on the state deviations had been determined, these were used to determine the weights on the input deviations. It would have been preferable to have an iterative process, where the new $\mathbf{R}$ would be used to determine a new $\mathbf{Q}$. This would be highly preferable during the tuning of the MPC_{HS} controller, where $\mathbf{Q}$ was first determined in two steps and then used to determine R. This has not been considered a problem for the project, since the weights were considered sufficient to be used during the comparison.

The weights on the state and input deviations are of great importance. For the low speed case, a low weight on Ψ leads to a better lateral position. Some weight on Ψ leads to additional corner cuttings. If the vehicle is behind the reference, some corner cutting will save time and hence result in a lower value on P_p. Too much corner cutting on the other hand leads to a higher value of P_p. The weights are therefore case specific and although one could prioritize performances and offsets; these will have to be balanced between each other.

As shown in section 6.3 there is a clear trade-off between the two performance criteria positioning and comfort. During a low speed maneuver such as parking the positioning is critical. However, it would be uncomfortable for the driver if the steering wheel would judder during a parking maneuvering. For an evasive maneuver to avoid a collision of some sort at high speeds the comfort is of very little importance. The positioning is also not critical to the same extent as in the low speed example. Hence, the trade-off between the performance criteria depends on the speed. In both cases the performance criteria are more critical at low speeds.

For this project the same weights have been used for the whole H_p. This could be expanded by weighing the states differently throughout H_p, so called time-varying weights. It might for example be preferable to weigh the samples near the end of H_p more. This would make the MPC controller prioritize the future more than the present. The effect has not been investigated further.

9.2 P-controller

A P-controller is sensitive to the resolution of its reference. Oscillations will appear when the points are sparse, however, the bandwidth and the processing speed will be slow when the points are close. Oscillations may also occur for a short preview distance while a long preview distance will make the vehicle cut corners. The counterpart for the MPC controllers would be H_p, which also affects the performance when very short or long. The longer H_p, the better the performance of an MPC controller. However, the length of H_p means it becomes much more computationally demanding to solve the optimization problem. Hence, it is important to find a balance both regarding d_p for the P-controller and H_p for the MPC controllers.

9.3 Vehicle Tests

The vehicle tests were limited due to technical difficulties. Much data could be collected for the low speed case but no viable data for the high speed. The fact that it executes in both low and high speeds is considered a successful result. The tests were focused on trying different weights and settings and evaluating the impact on the position. Two days were spent in the vehicle during the project, which is considered enough to conclude that it is possible to control the vehicle using MPC. It was not enough, however, to conclude any possible performance improvements.

10 Conclusions

Based on the discussion in section 9 it can be concluded that it is possible to develop an MPC Framework suitable for both low and high speed applications with the internal model as the only difference. With the methodology used in this project the MPC Framework is clearly more satisfactory in a low speed case. However, the high speed simulations suggest that the MPC_{HS} controller offers improved robustness compared to the P-controller_{HS}. For instance, the MPC_{HS} controller tuned for 70 km/h can handle e.g. 130 km/h while the P-controller_{HS} begins to oscillate and leaves the path.

Four different performance criteria were used in the project. Three of these criteria were lateral deviation, maximum lateral deviation and comfort, which were straightforward to compare between the MPC controllers and the P-controllers. The last performance criterion was used to evaluate the ability to follow the path both in space and time. That is, it measures the distance from the desired position at a given time. This performance criterion is more complex to compare between the MPC controllers and the P-controllers since they handle speed control differently.

The MPC_{LS} controller has better values regarding all performances compared to the P-controller_{LS}. For example, the maximum lateral deviation from the path was 80.32 cm for the P-controller_{LS} and 33.30 cm for the MPC_{LS} controller, that is, an improvement of 47 cm. The mean deviation was 30.47 cm for the P-controller_{LS} and 4.54 cm for the MPC_{LS} controller which is an improvement of 26 cm. The comparison is not as obvious for the high speed case. The P-controller has better values for lateral deviation, maximum lateral deviation and comfort but not for the desired position at a given time.

The MPC framework was successfully implemented in a Volvo V40. Data was collected for the low speed case and although no data was collected and analyzed for the high speed case the MPC_{HS} controller was confirmed to execute with a sampling time of 0.1 s.

11 Future Work

There is a large possibility for extending the models with more control signals. For example, individual braking torques on the wheels will probably improve the performance of the MPC_{HS} controller. This is possible to implement in the developed structure by adding output ports in the SfB, changing the equations inside and generating new code with CVXGEN.

The references represent the vehicle's desired position. Since the models are linearized around the references, each sample has its own region where the linearization is valid. Outside this region the linearization is no longer valid and unpredicted behaviour will occur. This region might cause problem for certain states. For example, if the lateral velocity in the MPC_{HS} controller has a nominal value separated from zero it would, according to the MPC controller, be desired to follow it. If the lateral velocity is high enough to cause slip this would mean that the MPC controller would also try to make the vehicle slip to the same amount. However, setting the nominal value of the lateral velocity to zero will make the linearization inaccurate. For this project a few different nominal values for this state have been tried. The problem of trying to follow the undesired nominal values seems larger than those of staying within the region of attraction. The best performance was found setting the nominal values for the lateral velocity to zero. This should be investigated further but has not been done in this project.

It was initially assumed that the vehicle would only move on flat surfaces, to restrict the complexity of the models. If banked motion and slopes are included the viability of the models will increase. This will be a simple addition to the system as a future improvement.

The references for X and Y were constructed mathematically. If the references would have been recorded, measurement noise and disturbances would have been included in the reference. This would result in the MPC controller trying to follow a noisy track and making adjustments to do so. It would also affect the nominal values since they are determined from the references. To avoid noisy nominal values some pretreatment of the references would probably be necessary. An option could be to pass the references, and the generated nominal values, through some sort of filter to remove outliers and disturbances. Since the full reference is assumed to be known, a zero-phase distortion filter such as `filtfilt` in MATLAB could be used. Another option could be to approximate the recorded path with a function and use the function to estimate nominal values. This could for example be done with MATLAB's `spline`.

Bibliography

Camacho, E. F. and Bordons, C. (2004). *Model predictive control*. 2nd ed. edn. Springer. London.

dSpace (2014). MicroAutoBox II. *dspace.com*. http://www.dspace.com/en/pub/home/products/hw/micautob.cfm (2014-04-09).

EUREKA (1995). 45 PROMETHEUS. *Eurekanetwork.org*. http://www.eurekanetwork.org/project/-/id/45 (2014-04-09).

Glad, T. and Ljung, L. (2000). *Control theory: multivariable and nonlinear methods*. Taylor & Francis. London.

Kühne, F., Lages, F. W. and Gomes da Silva Jr., J. M. (2004). Model Predictive Control of a Mobile Robot Using Linearization . Report 3-938153-30-X.

Kwon, W. H. and Han, S. (2005). *Receding horizon control: model predictive control for state models*. Springer. Berlin ; London.

Maciejowski, J. M. (2002). *Predictive control : with constraints*. Prentice Hall. Harlow.

Mattingley, J. (2013). CVXGEN: Code Generation for Convex Optimization. *http://cvxgen.com/*. http://cvxgen.com/docs/index.html (2014-04-09).

Mattingley, J. and Boyd, S. (2011). CVXGEN: a code generator for embedded convex optimization. Report. Springer Science+Business Media.

NHTSA (2013). U.S. Department of Transportation Releases Policy on Automated Vehicle Development. *nhtsa.gov*. http://www.nhtsa.gov/About+NHTSA/Press+Releases/U.S.+Department+of+Transportation+Releases+Policy+on+Automated+Vehicle+Development (2014-04-09).

Parent, M. (2013). Automated vehicles: autonomous or connected?

Russell, S. (2006). DARPA Grand Challenge Winner: Stanley the Robot! *popularmechanics.com*. http://www.popularmechanics.com/technology/engineering/robots/2169012 (2014-04-09).

Snider, J. M. (2009). *Automatic Steering Methods for Autonomous Automobile Path Tracking*. Carnegie Mellon University. Pittsburg, Pensylvania.

Volvo Cars (2013*a*). Adaptive Cruise Control with Steer Assist. *volvocars.com*. https://www.media.volvocars.com/global/en-gb/media/videos/49628/adaptive-cruise-control-with-steer-assist (2014-04-09).

Volvo Cars (2013*b*). Park Assist Pilot. *volvocars.com*. http://accessories.volvocars.com/AccessoriesWeb/Accessories.mvc/en-GB/RU/V40(13-)/2013/D4/Automatic/L.H.D/ShowDocument/VCC-463313 (2014-04-09).

Volvo Cars (2013*c*). Volvo Car Group initiates world unique Swedish pilot project with self-driving cars on public roads. *volvocars.com*. https://www.media.volvocars.com/global/en-gb/media/pressreleases/136182/volvo-car-group-initiates-world-unique-swedish-pilot-project-with-self-driving-cars-on-public-roads (2014-04-09).

Wetmore, J. M. (2003). Driving the Dream. Report.

Appendices

A High Speed Model

For the high speed model $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ becomes

$$\bar{\mathbf{A}} = \left.\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\mathbf{x}_r \\ \mathbf{u}=\mathbf{u}_r}} = \begin{bmatrix} 0 & 0 & \bar{A}_{13} & \bar{A}_{14} & \bar{A}_{15} & 0 \\ 0 & 0 & \bar{A}_{23} & \bar{A}_{24} & \bar{A}_{25} & 0 \\ 0 & 0 & 0 & 0 & 0 & \bar{A}_{36} \\ 0 & 0 & 0 & \bar{A}_{44} & \bar{A}_{45} & \bar{A}_{46} \\ 0 & 0 & 0 & \bar{A}_{54} & \bar{A}_{55} & \bar{A}_{56} \\ 0 & 0 & 0 & \bar{A}_{64} & \bar{A}_{65} & \bar{A}_{66} \end{bmatrix} \tag{A.1}$$

$$\bar{\mathbf{B}} = \left.\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}\right|_{\substack{\mathbf{x}=\mathbf{x}_r \\ \mathbf{u}=\mathbf{u}_r}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \bar{B}_{41} & \bar{B}_{42} \\ \bar{B}_{51} & \bar{B}_{52} \\ \bar{B}_{61} & \bar{B}_{62} \end{bmatrix} \tag{A.2}$$

where:

$$
\begin{aligned}
\bar{A}_{13} &= -(v_{x_n}\sin(\Psi_n) + v_{y_n}\cos(\Psi_n)) \\
\bar{A}_{14} &= \cos(\Psi_n) \\
\bar{A}_{15} &= -\sin(\Psi_n) \\
\bar{A}_{23} &= v_{x_n}\cos(\Psi_n) - v_{x_n}\sin(\Psi_n) \\
\bar{A}_{24} &= \sin(\Psi_n) \\
\bar{A}_{25} &= \cos(\Psi_n) \\
\bar{A}_{36} &= 1 \\
\bar{A}_{44} &= -\frac{C_f(v_{y_n} + l_f\omega_n)\sin(\delta_n)}{mv_{x_n}^2} \\
\bar{A}_{45} &= \frac{C_f\sin(\delta_n)}{mv_{x_n}} + \omega_n \\
\bar{A}_{46} &= \frac{C_f l_f\sin(\delta_n)}{mv_{x_n}} + v_{y_n} \\
\bar{A}_{54} &= \frac{C_f(v_{y_n} + l_f\omega_n)\cos(\delta_n) + Cr(v_{y_n} - l_r\omega_n)}{mv_{x_n}^2} - \omega_n \\
\bar{A}_{55} &= \frac{C_f\cos(\delta_n) + C_r}{mv_{x_n}} \\
\bar{A}_{56} &= \frac{C_r l_r - C_f l_f\cos(\delta_n)}{mv_{x_n}} - v_{x_n} \\
\bar{A}_{64} &= \frac{C_f l_f(v_{y_n} + l_f\omega_n)\cos(\delta_n) + C_r l_r(l_r\omega_n - v_{y_n})}{I_z v_{x_n}^2} \\
\bar{A}_{65} &= \frac{C_r l_r - C_f l_f\cos(\delta_n)}{I_z v_{x_n}} \\
\bar{A}_{66} &= -\frac{C_f l_f^2\cos(\delta_n) + C_r l_r^2}{I_z v_{x_n}}
\end{aligned}
\tag{A.3}
$$

$$
\begin{aligned}
\bar{B}_{41} &= -\frac{F_{x_n}\sin(\delta_n)}{2m} + \frac{C_f(\sin(\delta_n) + \delta_n\cos(\delta_n))}{m} - \frac{C_f(v_{y_n} + l_f\omega_n)\cos(\delta_n)}{mv_{x_n}} \\
\bar{B}_{42} &= \frac{\cos(\delta_n) + 1}{2m} \\
\bar{B}_{51} &= \frac{F_{x_n}\cos(\delta_n)}{2m} + \frac{C_f(\cos(\delta_n) - \delta_n\sin(\delta_n))}{m} + \frac{C_f(v_{y_n} + l_f\omega_n)\sin(\delta_n)}{mv_{x_n}} \\
\bar{B}_{52} &= \frac{\sin(\delta_n)}{2m} \\
\bar{B}_{61} &= \frac{F_{x_n} l_f\cos(\delta_n)}{2I_z} + \frac{C_f l_f(\cos(\delta_n) - \delta_n\sin(\delta_n))}{I_z} + \frac{C_f l_f(v_{y_n} + l_f\omega_n)\sin(\delta_n)}{I_z v_{x_n}} \\
\bar{B}_{62} &= \frac{l_f\sin(\delta_n)}{2I_z}
\end{aligned}
\tag{A.4}
$$

The discretization of this then becomes

$$\bar{\mathbf{A}} = \left.\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\mathbf{x}_r \\ \mathbf{u}=\mathbf{u}_r}} = \begin{bmatrix} 1 & 0 & T\bar{A}_{13} & T\bar{A}_{14} & T\bar{A}_{15} & 0 \\ 0 & 1 & T\bar{A}_{23} & T\bar{A}_{24} & T\bar{A}_{25} & 0 \\ 0 & 0 & 1 & 0 & 0 & T\bar{A}_{36} \\ 0 & 0 & 0 & 1+T\bar{A}_{44} & T\bar{A}_{45} & T\bar{A}_{46} \\ 0 & 0 & 0 & T\bar{A}_{54} & 1+T\bar{A}_{55} & T\bar{A}_{56} \\ 0 & 0 & 0 & T\bar{A}_{64} & T\bar{A}_{65} & 1+T\bar{A}_{66} \end{bmatrix} \tag{A.5}$$

$$\bar{\mathbf{B}} = \left.\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}\right|_{\substack{\mathbf{x}=\mathbf{x}_r \\ \mathbf{u}=\mathbf{u}_r}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ T\bar{B}_{41} & T\bar{B}_{42} \\ T\bar{B}_{51} & T\bar{B}_{52} \\ T\bar{B}_{61} & T\bar{B}_{62} \end{bmatrix} \tag{A.6}$$

where:

$$\begin{aligned}
\bar{A}_{13} &= -(v_{x_n}(k)\sin(\Psi_n(k)) + v_{y_n}(k)\cos(\Psi_n(k))) \\
\bar{A}_{14} &= \cos(\Psi_n(k)) \\
\bar{A}_{15} &= -\sin(\Psi_n(k)) \\
\bar{A}_{23} &= v_{x_n}(k)\cos(\Psi_n(k)) - v_{x_n}(k)\sin(\Psi_n(k)) \\
\bar{A}_{24} &= \sin(\Psi_n(k)) \\
\bar{A}_{25} &= \cos(\Psi_n(k)) \\
\bar{A}_{36} &= 1 \\
\bar{A}_{44} &= -\frac{C_f(v_{y_n}(k) + l_f\omega_n(k))\sin(\delta_n(k))}{mv_{x_n}^2(k)} \\
\bar{A}_{45} &= \frac{C_f\sin(\delta_n(k))}{mv_{x_n}(k)} + \omega_n(k) \\
\bar{A}_{46} &= \frac{C_f l_f\sin(\delta_n(k))}{mv_{x_n}(k)} + v_{y_n}(k) \\
\bar{A}_{54} &= \frac{C_f(v_{y_n}(k) + l_f\omega_n(k))\cos(\delta_n(k)) + Cr(v_{y_n}(k) - l_r\omega_n(k))}{mv_{x_n}^2(k)} - \omega_n(k) \\
\bar{A}_{55} &= \frac{C_f\cos(\delta_n(k)) + C_r}{mv_{x_n}(k)} \\
\bar{A}_{56} &= \frac{C_r l_r - C_f l_f\cos(\delta_n(k))}{mv_{x_n}(k)} - v_{x_n}(k) \\
\bar{A}_{64} &= \frac{C_f l_f(v_{y_n}(k) + l_f\omega_n(k))\cos(\delta_n(k)) + C_r l_r(l_r\omega_n(k) - v_{y_n}(k))}{I_z v_{x_n}^2(k)} \\
\bar{A}_{65} &= \frac{C_r l_r - C_f l_f\cos(\delta_n)}{I_z v_{x_n}} \\
\bar{A}_{66} &= -\frac{C_f l_f^2\cos(\delta_n) + C_r l_r^2}{I_z v_{x_n}}
\end{aligned} \tag{A.7}$$

$$
\begin{aligned}
\bar{B}_{41} = \quad & -\frac{F_{x_n}(k)\sin(\delta_n(k))}{2m} + \frac{C_f(\sin(\delta_n(k)) + \delta_n(k)\cos(\delta_n(k)))}{m} \\
& - \frac{C_f(v_{y_n}(k) + l_f\omega_n(k))\cos(\delta_n(k))}{mv_{x_n}(k)} \\
\bar{B}_{42} = \quad & \frac{\cos(\delta_n(k)) + 1}{2m} \\
\bar{B}_{51} = \quad & \frac{F_{x_n}(k)\cos(\delta_n(k))}{2m} + \frac{C_f(\cos(\delta_n(k)) - \delta_n(k)\sin(\delta_n(k)))}{m} \\
& + \frac{C_f(v_{y_n}(k) + l_f\omega_n(k))\sin(\delta_n(k))}{mv_{x_n}(k)} \\
\bar{B}_{52} = \quad & \frac{\sin(\delta_n(k))}{2m} \\
\bar{B}_{61} = \quad & \frac{F_{x_n}(k)l_f\cos(\delta_n(k))}{2I_z} + \frac{C_f l_f(\cos(\delta_n(k)) - \delta_n(k)\sin(\delta_n(k)))}{I_z} \\
& + \frac{C_f l_f(v_{y_n}(k) + l_f\omega_n(k))\sin(\delta_n(k))}{I_z v_{x_n}(k)} \\
\bar{B}_{62} = \quad & \frac{l_f\sin(\delta_n(k))}{2I_z}
\end{aligned}
\tag{A.8}
$$

B CVXGEN code

```
dimensions
  l = 2 # inputs.
  n = 4 # states.
  m = 4 # outputs.
  Hp = 20 # horizon.
end

parameters
  A[k] (n,n), k=0..Hp # dynamics matrix.
  B[k] (n,l), k=0..Hp # transfer matrix.
  Q (n,n) psd # state cost.
  R (l,l) psd # input cost.
  xtilde[0] (n) # initial state.
  umin[k] (l), k=0..Hp #constraints on control signal [delta_min,F_min]
  umax[k] (l), k=0..Hp #constraints on control signal [delta_max,F_max]
  dun[k] (l), k=0..Hp-1 #nominal values of the change of control signal
  uprev (l) #Previous value of the control signal
  unk (l) #Current nominal value for the control signal
  dumax (l)  #constraints on the change rate of the control signal
  dumin (l) #constraints on the change rate of the control signal
end

variables
  xtilde[k] (n), k=1..Hp+1 # state.
  utilde[k] (l), k=0..Hp # input.
end

minimize
  sum[k=1..Hp](quad(xtilde[k], Q) + quad(utilde[k-1], R))
subject to
  xtilde[k+1] == A[k]*xtilde[k] + B[k]*utilde[k], k=0..Hp
  umin[k]<=utilde[k]<=umax[k], k=0..Hp
  dumin <= utilde[k+1] - utilde[k] + dun[k] <= dumax, k=0..Hp-1
  dumin <= utilde[0]+unk-uprev <= dumax;
end
```

C C code

In this section is the C code from the MPC_{LS} controller presented. All variables beginning with `params` are parameters sent into the CVX-solver. Variables obtained from the CVX-solvers are called `vars`. The CVX-solver is called by `solve()`.

```
int i;
int n = 4;
int l = 2;
int Hp = 20;

set_defaults();  // Set basic algorithm parameters.
settings.verbose = 0;
setup_indexing();

params.dumax[0] = ucon[2];
params.dumax[1] = ucon[5];
params.dumin[0] = -ucon[2];
params.dumin[1] = -ucon[5];
params.unk[0] = un[0];
params.unk[1] = un[1];
params.uprev[0] = uprev[0];
params.uprev[1] = uprev[1];

for (i=0;i<n*n;i++)
    params.Q[i] = Q[i];

for (i=0;i<l*l;i++){
    params.R[i] = R[i];
    unout[i] = un[i];}

for (i=0;i<n;i++){
    params.xtilde[0][i]=xk[i]-xn[i];
    xtilde[i]=xk[i]-xn[i];
    xnout[i] = xn[i];
    xkout[i] = xk[i];}

for (i=0;i<Hp;i++){
    params.umax[i][0] = ucon[0]-un[0+l*i];
    params.umin[i][0] = ucon[1]-un[0+l*i];
    params.umax[i][1] = ucon[3]-un[1+l*i];
    params.umin[i][1] = ucon[4]-un[1+l*i];}

for (i=0;i<=Hp;i++){
    //A col 1
    params.A[i][0] = 1;
    params.A[i][1] = 0;
    params.A[i][2] = 0;
    params.A[i][3] = 0;

    //A col 2
    params.A[i][4] = 0;
    params.A[i][5] = 1;
```

```
    params.A[i][6] = 0;
    params.A[i][7] = 0;

    //A col 3
    params.A[i][8] = -T[0] * xn[3+n*i] * sin(xn[2+n*i]);
    params.A[i][9] = T[0] * xn[3+n*i] * cos(xn[2+n*i]);
    params.A[i][10] = 1;
    params.A[i][11] = 0;

    //A col 4
    params.A[i][12] = T[0] * cos(xn[2+n*i]);
    params.A[i][13] = T[0] * sin(xn[2+n*i]);
    params.A[i][14] = T[0]/L[0] * tan(un[0+l*i]);
    params.A[i][15] = 1;

    //B col 1
    params.B[i][0] = 0;
    params.B[i][1] = 0;
    params.B[i][2] = T[0] * (xn[3+n*i]/L[0]) * (1+pow(tan(un[0+l*i]),2));
    params.B[i][3] = 0;

    //B col 2
    params.B[i][4] = 0;
    params.B[i][5] = 0;
    params.B[i][6] = 0;
    params.B[i][7] = T[0]/mass[0];}

for (i=0;i<Hp;i++){
    params.dun[i][0] = un[0+l*(i+1)]-un[0+l*i];
    params.dun[i][1] = un[1+l*(i+1)]-un[1+l*i];
}
solve();

u[0]=vars.utilde[0][0]+un[0];
u[1]=vars.utilde[0][1]+un[1];
utilde[0] = vars.utilde[0][0];
utilde[1] = vars.utilde[0][1];
conv[0] = work.converged;
```

D Input Parameters

Table D.1. Parameters used in low and high speed.

Parameter	Description	Dimension LS | HS	Value LS | HS
xrHp	Matrix with current and future state references and nominal values for states	4xH_pxM |6xH_pxM	|
urHp	Matrix with current and future nominal values for inputs	2xH_pxM |2xH_pxM	|
N	Simulation steps	1 |1	|
T	MPC sample interval [s]	1 |1	0.1 |0.1
Q	Matrix with weights for error states	4x4 |6x6	|
R	Matrix with weights for error inputs	2x2 |2x2	|
m	Mass of vehicle [kg]	1 |1	1174 |1174
L	Length of vehicle [m]	1 |-	2.4270 |-
l_f	Length of vehicle from CoG to front axle [m]	- |1	- |1.066
l_r	Length of vehicle from CoG to rear axle [m]	- |1	- |1.614
C_f	Cornering stiffness front	- |1	- |6.48*10^4
C_r	Cornering stiffness rear	- |1	- |8.83*10^4
I_z	Moment of inertia [kgm^2]	- |1	- |1360
g	Gravitational acceleration [m/s^2]	- |1	- |9.82
$ucon$	Constraints on input values	1x6 |1x6	|
	- Max δ [rad]	|	0.43 |0.43
	- Min δ [rad]	|	-0.43 |-0.43
	- Max $\Delta\delta$ [rad/sample]	|	0.35T |0.52T
	- Max F_x [N]	|	6000 |6000
	- Min F_x [N]	|	-6000 |-6000
	- Max ΔF_x [N/sample]	|	6000T |6000T
Origin	Reference position of the car from the rear end [m]	1 |1	0.79 |2.404

E P-controller with Preview Distance

A simpler controller for automotive path following is to only control the deviation of the yaw angle with a proportional controller (P-controller).

There are two points of importance for this type of controller. The first one is placed on a certain preview distance (d_p) from an origin, along the vehicle direction (AVD). Since the point around which the car rotates is not the same for low and high speed the origin can be placed at different locations; the rear axle or the CoG. The second point is placed as close to the AVD point as possible along the road centerline (ARC). These are indicated in Figure E.1.

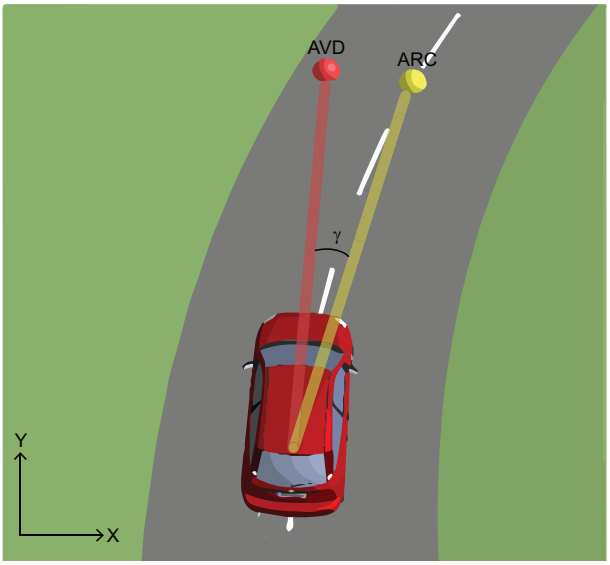


Figure E.1. Concept of a P-controller used for path following.

The path can be stored as coordinates with start at the origin. The current vehicle position is here referred to as X and Y and the reference coordinates as $X_r(k)$ and $Y_r(k)$ where k is the sample in time from the start.

By assuming that Ψ is known, the AVD point is defined as $(d_p \cos(\Psi), d_p \sin(\Psi))$. The ARC point can be obtained by finding the time index k which minimizes the difference between the AVD point and the reference coordinates, k^*:

$$k^* = \arg\min_k \left\{ \sqrt{(X_{AVD} - \mathbf{X}_r(k))^2 + (Y_{AVD} - \mathbf{Y}_r(k))^2} \right\} \tag{E.1}$$

Hence the ARC point may be written as $(X_r(k^*),\ Y_r(k^*))$. The angle of the front wheels are controlled to a proportional gain of the angle between the AVD and the ARC points, γ, shown in Figure E.1.

The main problem with this approach is to decide d_p. A too short d_p will cause the vehicle to oscillate while a too long preview distance will make the vehicle cut corners or even leave the road. The d_p is both dependent of speed and radius of the curve.

E.1 Tuning the P-controller

The P-controllers were tuned on the same test tracks as the MPC controllers developed in the project. The P-controller$_{LS}$ were tuned on test track LS1 and P-controller$_{HS}$ on HS1. The tuning were done in constant speed.

E.1.1 Test Track

The resolution of the resampled track had to be considered for the P-controller. If the points are too sparse the P-controller will be more prone to oscillate. However, with too many points there will be a negative effect both on the processing speed and the bandwidth. This is because all points has to be searched at every time step. Before sending the references to the P-controller they were resampled to be equidistant. For all tests the distance between the points of the resampled track was set to 0.1 m.

E.1.2 Speed Control

Because CarMaker's internal speed control could not drive satisfactory in very low speeds a proportional controller was developed for this purpose. The developed speed controller was used in all simulated speeds for consistency. To eliminate the residual steady-state error in the speed, an integration part was added to the controller. The gain on the proportional part was set to 2000 and the gain on the integration part was set to 100. Since the initial speed was set to the same as the desired speed these gains had minor impact for the simulations.

E.1.3 Method of Tuning Preview Distance

The d_p was tuned by trial and error. After each test the P_l was compared to find a minimum. The impact of different d_p are illustrated by Figure E.2. The aim was to find fitting preview distances to be used when comparing the P-controllers with the MPC controllers. Hence, d_p was tuned both for 3 km/h and 70 km/h. From the figure a d_p of somewhere between 4 and 6 m can be concluded appropriate for a speed of 3 km/h.

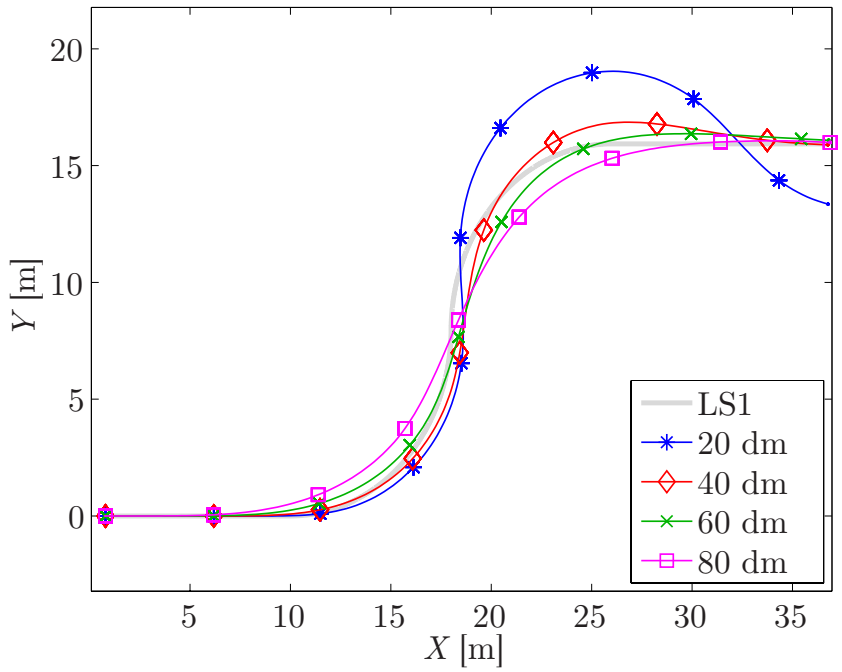


Figure E.2. P-controller with different preview distances run in 10 km/h.

Preview Distance for Low Speed Simulations

The parameters for the tuning are found in Table E.1. The results are shown in Table E.2. A d_p of 4.9 meter has the lowest P_l, however, the lowest P_c is achieved with a longer d_p. As lateral positioning was desired, a d_p of 4.9 m was chosen for the P-controller$_{LS}$.

Table E.1. Parameters used to evaluate d_p for the LS P-controller.

Parameter	Values
Test track	LS1
Constant speed	3 km/h
Gain on γ	1
Origin	Rear axle

Table E.2. P_l and P_c for different d_p in 3 km/h.

	d_p [dm]				
v_x [km/h]	47	48	49	50	51
3	26.75 | 2.89	26.57 | 2.80	**26.54** | 2.81	26.60 | 2.76	26.65 | **2.70**

Preview Distance for High Speed Simulations

The results from the high speed tuning using parameters from Table E.3 are found in Table E.4. A longer d_p provides more comfort which is expected. The d_p was chosen to 8 m for the P-controller$_{HS}$.

Table E.3. Parameters used to evaluate d_p for the P-controller$_{HS}$.

Parameter	Values
Test track	HS1
Constant speed	70 km/h
Gain γ	1
Origin	CoG

Table E.4. P_l and P_c for different d_p in 70 km/h.

	d_p [dm]				
v_x [km/h]	77	78	79	80	81
70.00	3.42 | 35.69	3.42 | 35.10	3.41 | 34.51	3.42 | 33.76	3.42 | 33.10

E.2 Resulting Preview Distances

The resulting d_p for both controllers are found in Table E.5. These were used for the comparison with the MPC controllers.

Table E.5. Preview distances for the speeds 3 and 70 km/h.

Speed [km/h]	d_p [dm]	Test track
3	49	LS1
70	80	HS1

F Parameters for Vehicle tests

Table F.1. Parameters used for vehicle tests in low speed.

	Qd	R_d	δ_{max}	δ_{min}	$\Delta\delta$
Vehicle test 1	{100,100,100,10}	$\{1,10^{-5}\}$	0.43	0.43	0.015
Vehicle test 2	{100,100,100,10}	$\{1,10^{-5}\}$	0.43	0.43	0.015
Vehicle test 3	{100,100,100,10}	$\{1,10^{-5}\}$	0.43	0.43	0.03
Vehicle test 4	{100,100,100,10}	$\{1,10^{-5}\}$	0.5	0.5	0.04
Vehicle test 5	{100,100,100,10}	$\{1,10^{-5}\}$	0.5	0.5	0.03