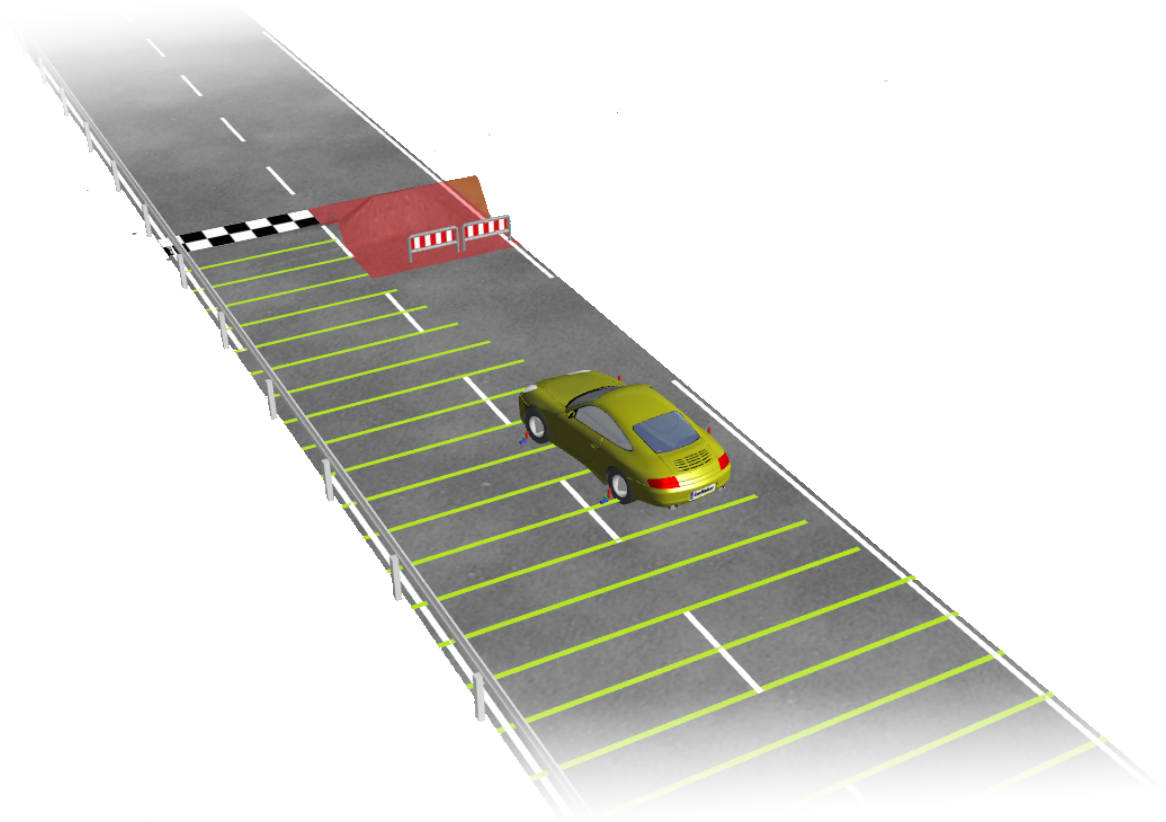
CHALMERS
UNIVERSITY OF TECHNOLOGY



Obstacle Avoidance with Safety Guarantees

Feasibility of MPC-based Steering Algorithms

Master's thesis in Signals and Systems

Stephan Heinrich

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015

Master's thesis 2015:11

Obstacle Avoidance with Safety Guarantees

Feasibility of MPC-based Steering Algorithms

Stephan Heinrich

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Signals and Systems
Mechatronics Research Group
Chalmers University of Technology
Gothenburg, Sweden 2015

Obstacle Avoidance with Safety Guarantees
Feasibility of MPC-based Steering Algorithms
Stephan Heinrich

Examiner:
Paolo Falcone, Department of Signals and Systems
Mathias Lidberg, Department of Applied Mechanics

Master's Thesis 2015:11
Department of Signals and Systems
Mechatronics Research Group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Simulation vehicle performing a lane change maneuver with projections of safe states on the road

Typeset in LaTeX
Gothenburg, Sweden 2015

Obstacle Avoidance with Safety Guarantees
Feasibility of MPC-based Steering Algorithms
Stephan Heinrich
Department of Signals and Systems
Department of Applied Mechanics
Chalmers University of Technology

Abstract

Enabled by the increasing computational power available in embedded microcontrollers, model predictive control (MPC) for automotive applications has been extensively studied in the past ten years and has attracted industry, among others, for autonomous driving applications.
Implementing an MPC control algorithm to determine an optimal steering action in a lane change maneuver involves iteratively solving a Finite Time Constrained Optimal Control Problem (FTCOCP). Hence, in a situation where the environment is restricted by lane boundaries and obstacles, the FTCOCP might be unfeasible because collision avoidance constraints represent strict boundaries that cannot be violated if passenger safety shall be guaranteed. Intuitively, in such problems infeasibility is more likely to occur as the vehicle velocity increases. Furthermore, because of the problem structure, classical results from MPC theory for guaranteeing feasibility cannot be efficiently applied.
This thesis investigates methods to predict an upper bound on the vehicle's velocity leading to feasibility guarantees for an MPC-based lane change algorithm. By exploiting invariant set theory and reachability analysis tools, we focus on the problem of (offline) characterizing the set of states for which a solution to the lane change MPC problem exists for a given speed.
An MPC controller is implemented in a vehicle and test data from lane change maneuvers ranging from 50-100 km/h is collected at the AstaZero proving grounds. The applicability of the proposed method is evaluated by comparison with experimental data.

Keywords: model predictive control, vehicle, feasibility, convex optimization, obstacle avoidance, advanced driver assistance systems

Acknowledgements

During the past year working on this thesis I had the pleasure to work with a number of amazing people that dedicated their time and effort to support me in this process. First of all I would like to thank my examiners Paolo Falcone and Mathias Lidberg for making this thesis possible and supporting me throughout this period. I don't think I could have asked for a better team of advisers in this field. Your support in both vehicle dynamics as well as control theory let me consider and challenge ideas and concepts that I was not even aware of at the time when I started this project. Working along side you really shaped my approach work, to keep challenging existing thoughts and think outside of the box.
Next I would like to thank Chris Gerdes and the DDL at Stanford. Joining their lab during the summer was an incredible time. I would not have wanted to miss the late nights coding and the sizzling hot track days together with Team Marty for anything. The DDL is full of exceptional people and being part of that for some time was a very humbling experience.
I would also like to thank my friends and colleagues in the Mechatronics group as well as the Vehicle Dynamics group at Chalmers. Working together with you has been an exceptionally welcoming and supportive experience.
I would also like to thank my parents and my brother for supporting me through all this time. Cheering me up when things got challenging, and always having my back while I was far away from home let me get through the ups and downs of this process and makes me feel confident for future challenges to come.
This work would not have been possible without the help and support of all these exceptional people.

Stephan Heinrich, Gothenburg, 11/2015

Contents

List of Figures

List of Tables

1
Introduction

With car manufacturers and tech companies running an arms race towards making the autonomous vehicle a reality, the field of vehicle motion control attempts to solve the sub-problem of using steering, braking and throttle to control the motion of the vehicle within its environment.

Figure 1.1: Vehicle approaching slower traffic on a multilane road

This thesis is focused on vehicle motion control in autonomous lane change scenarios as shown in figure 1.1. As a vehicle approaches slower traffic on a multilane road, the control algorithm steering the car needs to find a way to avoid the obstacle in its path. Model predictive control (MPC) has proven to be an efficient approach for these types of control problems [8]. The MPC controller studied in this thesis finds the sequence of optimal steering commands to avoid the slower vehicle and change the lane as the solution of a finite time optimal control problem.

$$\begin{aligned} &\text{minimize:} && \text{a function of steering} \\ &\text{subject to:} && \text{vehicle dynamics} \\ & && \text{collision avoidance} \\ & && \text{lane boundaries} \end{aligned} \tag{1.1}$$

By using models of the vehicle dynamics and the environment, the optimization algorithm predicts an optimal path together with the necessary steering actions to drive it. It is intuitively clear for every driver that the existence of such a path to avoid a slower vehicle ahead is very much dependent on the vehicle's own speed, the speed of the preceding vehicle as well as the dimensions of the road.
For the situation depicted in figure 1.1 the controller may easily find a solution when the car is driving 20 km/h faster than the truck but it may be impossible to calculate a path to avoid the truck if it is standing still and the vehicle is travelling 130 km/h.

With model predictive control becoming more and more popular for vehicle motion control, and extensive body of work on this subject has been developed in recent years.

1.1 Literature Review

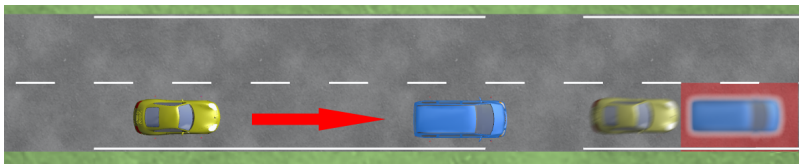
In the mid 2000s Borelli, Falcone et al. [4], [11], [12] applied online model predictive control to implement steering algorithms for autonomous road vehicles. Using a Linear Time Varying (LTV) approach, they linearized the vehicle dynamics around the previous optimization results and the current velocity of the vehicle. They were able to implement real time capable controllers for path tracking with promising results. In their work they assumed a known trajectory and they succeeded in tracking predefined paths.
To account for a changing environment, such as unexpected obstacles, Falcone et al. [13] introduced a hierarchical approach that allowed online replanning of the trajectories which act as a reference for the low level path following controller. This general structure is widely adopted in literature with multiple different approaches on how this top level trajectory planning problem can be solved efficiently. In their work Falcone et al. used a vehicle representation as a point mass to represent the situation as a convex quadratic programming (QP) problem that can be solved efficiently.
Funke [16] et al. used a parametrizable sequence of clothoids to plan efficient trajectories for obstacle avoidance.
Both approaches use the path curvature and the available road friction to determine an upper bound on the velocity for which the path is assumed to be feasible. In a similar approach, Gray et al. [20] used a path planning algorithm based on a set of parametrizable, predefined motion primitives that can be combined to achieve efficient trajectory calculations.
Maintaining the approach of hierarchically separated path planning and path following, Gao et al. [18] used a general purpose nonlinear solver (NPSOL) to approach the path planning problem with fewer simplifications. By solving the general nonlinear problem they were able to determine lateral and longitudinal velocity of the desired path in one step. At the same time they still implemented an efficient QP-based path following algorithm as a low level steering controller. While this approach is very general, the solution of the path planning algorithm does not provide convergence guarantees and is limited to relatively slow controller sampling rates of 200 ms in their experiments. Such a slow path replanning frequency may not be sufficient to control vehicles travelling at highway speeds. The vehicle will only start reacting to a sudden appearance of an obstacle after it has been detected and incorporated into the solution of the path planner. At typical highway speeds of 130 km/h this means that the vehicle will travel for more than 7 m before the path planning algorithm has calculated a new solution and the steering controller can start to react to the new situation.
To avoid the use of computationally expensive general purpose nonlinear solvers, Carvalho et al. [7] used an approximation based on sequential quadratic programming (SQP) to repetitively solve and relinearize the problem in one time step while

using an efficient QP-solver algorithm. While the computation time of this approach can be manually bounded by limiting the number of solver iterations, the approach cannot guarantee convergence either.
In an alternative approach Hsu and Gerdes [24] as well as following work by Beal, Erlien and Funke [1], [2], [9], [10], [17] demonstrated a steering controller that combines vehicle stability boundaries and environmental boundaries in one hierarchy level to be able to avoid obstacles without a separate path replanning step. By introducing environmental constraints in the low level steering controller, it becomes capable of reacting directly to unexpected obstacles that suddenly appear. They showed experimental results at controller update rates up to 100 Hz. To achieve this frequency they linearize the problem around the current vehicle velocity and utilize efficient problem specific solvers for convex optimization in combination with powerful computing hardware.
While there has been extensive work and great results from different model predictive control approaches for vehicle motion control, the overall trade-offs between generality and accuracy of the solution on the one side and computation time on the other side still remain a challenge and the optimal approach to this problem is yet an open field of research.
Trajectory planning algorithms that utilize parametrized maneuvers or nonlinear optimization can yield pre-calculated trajectories and desired velocity profiles but typically suffer a trade-off between computational complexity and a need for simplifications that neglect some aspects of the vehicle dynamics.
Approaches that utilize convex QP-solvers have shown to be effective in line following, vehicle stabilization and obstacle avoidance but the problem statements are linearized and parametrized by velocity. While this is not problematic for simple path tracking controllers, where the current vehicle speed provides a good approximation over a relatively short prediction horizon, this property becomes more critical for controllers that consider longer horizons. These extended horizons are typically needed if road boundaries and obstacles are to be considered. For problem formulations of that type the velocity may not be assumed as given and the choice of the desired velocity along the horizon will determine the existence of the solution of the optimization problem.

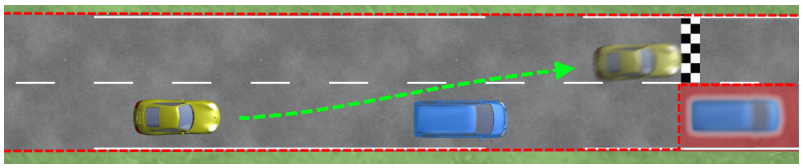
1.2 Problem Statement

For this thesis we consider a controller similar to [9] which considers both, vehicle handling limits as well as environmental boundaries in a linearized, convex problem formulation. For this type of controller the problem statement (1.1) becomes parametrized by velocity. We consider the application of this controller in a highway lane change maneuver as shown in figure 1.1 where the road is restricted by solid boundaries that have to be avoided at all costs.
To describe the environmental boundaries in this problem the absolute velocity of the ego vehicle and the differential velocity between the ego and the obstacle in the way can be used to estimate the location where the vehicles would collide. Assuming constant velocities we can locate where the potential collision would occur as shown in figure 1.2a. Figure 1.2b illustrates how such problem can be described by lane

boundaries and the location where the ego vehicle must have translated to the other lane can be described. Therefore this problem can be treated equally to an obstacle avoidance maneuver with a stationary object.



(a) Lane change situation and projected location of the collision



(b) Lane boundaries and target location

Figure 1.2: Spatial localization and boundaries of a lane change maneuver

Without trying to precalculate a path or attempting to solve the parametrized convex optimization problem of the MPC controller for multiple different velocities, we aim to determine an upper bound on the velocity such that we can guarantee that a solution to the problem exists and the vehicle can pass the obstacle. We attempt to classify the sets of vehicle states, located at discrete locations before the projected point of collision, from where, for discrete constant velocities, feasibility for the controller can be guaranteed and the the vehicle can safely pass the preceding vehicle.
This upper bound would allow us to classify whether the current vehicle speed is safe to attempt the lane change or if heavy braking must be considered as an alternative measure to ensure the passenger safety. Findings in [16] indicate that, for serious obstacle maneuvers where the distance to the obstacle and the vehicle speed becomes critical, full braking and consecutive steering is an adequate approach to an obstacle avoidance maneuver.
Figure 1.3 visualizes the idea behind this approach. By illustrating and examining states, for which the maneuver is feasible for different, distinct velocities we we can check if the current vehicle state is feasible. For a high speed, shown in blue, the vehicle's current state is not contained in the locations for which a feasible trajectory can be guaranteed. For lower speeds as indicated in yellow or red the feasibility for the maneuver may be guaranteed.

1.3 Structure of the Thesis

In the following three chapters the necessary methods and tools used in this work are introduced. An overview of the necessary basics in invariant set theory and reachability analysis is provided in chapter 2. In chapter 3 the vehicle models used to mathematically describe the motion of the vehicle for simulation and control

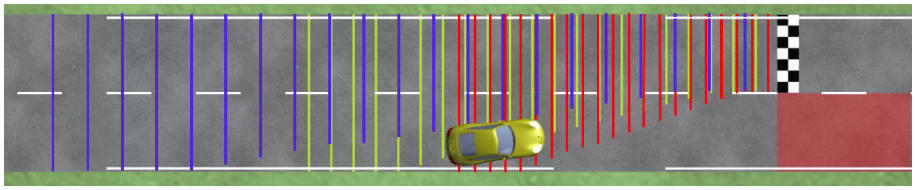


Figure 1.3: Discrete visualization of location from where feasibility of the maneuver can be guaranteed. Blue = high speed, green = medium speed, red = low speed.

design are presented. Chapter 4 introduces the fundamentals of model predictive control used in this thesis and explains some of the challenges and restrictions of using convex optimization in online MPC which completes the introductory topics. Chapter 5 provides the core study of the problem at hand. Different approaches are examined and the challenges involved are illustrated. An approach for a solution that allows the use of this concept is proposed.
To gather comparison data, chapter 6 describes the design and implementation of a model predictive controller that is capable of controlling the vehicle in a lane change maneuver.
Experimental testing procedures and data from real world autonomous lane change maneuvers are illustrated in chapter 7 for comparison with the theoretical predictions.
The closing chapters 8 and 9 provide a comparison between the experimental data and the theoretical calculations derived in chapter 5 and a discussion on the findings of the thesis.

2

Invariant Set Theory and Reachability Analysis

Invariant set theory and reachability analysis are a powerful tools to mathematically describe and evaluate possible trajectories of dynamic systems within their state-space. This chapter provides an introduction to the methods from these fields that are used to evaluate the feasibility of the highway lane change maneuver.
For a detailed background on the computational geometry involved, the reader is referred to G. Ziegler's *Lectures on Polytopes* [30]. A background on the applications of these general concepts in the field of constrained optimization and control is presented in *Constrained Optimal Control for Linear and Hybrid Systems* by F. Borrelli [5]. The following definitions provide a condensed excerpt from [5] and [30] of the most important concepts used in this work. For a further background a survey of applications of invariant set theory in control can be found in [3].

2.1 Convex Sets

A Set $\mathcal{S} \in \mathbb{R}^n$ is called convex if

$$\lambda z_1 + (1-\lambda) z_2 \in \mathcal{S} \quad \text{for all} \quad z_1, z_2 \in S, \ \lambda \in [0,1]. \tag{2.1}$$

This can be interpreted as that any line that connecting any two points in the set is also contained in the set. This concept is visualized in figure 2.1.

2.1.1 Hyperplanes & Halfspaces

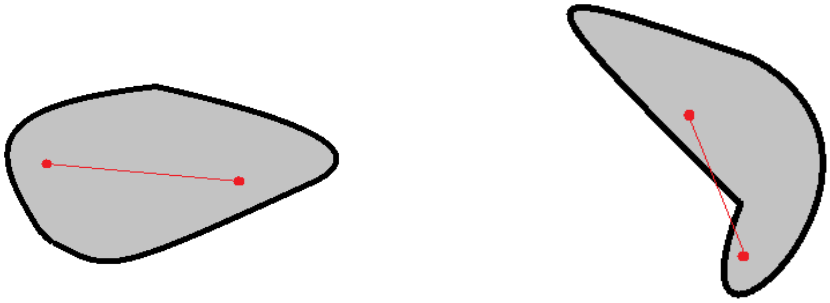
A Hyperplane is described by a vector $a \in \mathbb{R}^n, a \neq \emptyset$ and a scalar $b \in \mathbb{R}$ and is formed by

$$\mathcal{S} = \left\{ x \in \mathbb{R}^n \ \mid \ a^T x = b \right\}. \tag{2.2}$$

Geometrically a hyperplane can be interpreted as the set of points orthogonal to a normal vector with a fixed offset from the origin.
The set of points located on either side of a hyperplane can be described by a halfspace. A halfspace is defined by a linear inequality in the form

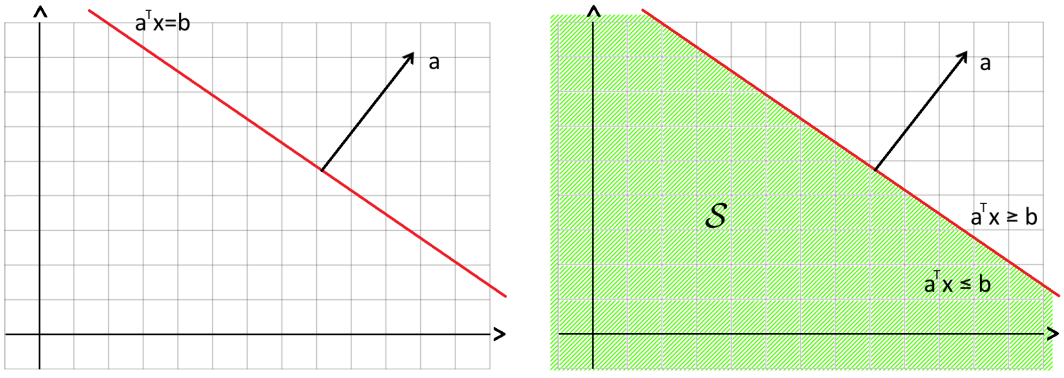
$$\mathcal{S} = \left\{ x \in \mathbb{R}^n \ \mid \ a^T x \leq b \right\}. \tag{2.3}$$

Both hyperplanes as well as halfspaces are convex sets. Figure 2.2 illustrates a hyperplane and the two halfspaces created by it in 2 dimensions.



(a) Convex set **(b)** Non-convex set

Figure 2.1: A convex and a non-convex set in $\mathbb{R}^2$



(a) Straight line illustrating a hyperplane in $\mathbb{R}^2$ **(b)** Halfspace in $\mathbb{R}^2$

Figure 2.2: $\mathbb{R}^2$ divided into two halfspaces by a hyperplane

2.1.2 Polyhedra and Polytopes

Polytopes and polyhedra are geometrical objects with flat sides. They are sets that can be described by the intersection of a finite number of halfspaces. This thesis follows the definitions from Ziegler [30].
A polytope is a bounded polyhedron which means that it does not contain any ray of the type $\{x + ty \mid t \geq 0\}$ for any $y \neq 0$. This means its outer bound is defined in all directions and it has a finite volume in $\mathbb{R}^n$.

2.2 Operations on Polytopes

This section introduces basic operations on Polytopes.

2.2.1 Convex Hull

The tightest polytope that encloses a set of points is called its convex hull. For a set of points $V = \{V^i\}_{i=1}^{N_V}$ with $V^i \in R^n$ it is defined as

$$\text{conv}(V) = \left\{ x \in \mathbb{R}^n \mid x = \sum_{i=1}^{N_V} \alpha^i V^i,\ 0 \leq \alpha^i \leq 1,\ \sum_{i=1}^{N_V} \alpha^i = 1 \right\}. \tag{2.4}$$

2.2.2 Vertex Enumeration

Vertex enumeration is the dual operation of the convex hull. If the hyperplanes bounding a polytope are known, vertex enumeration is the process used to calculate the extreme points of a polytope. Vertex enumeration is typically an expensive operation as the calculation is exponential in the the number of facets and common algorithms to solve the vertex enumeration problem are the double description method or reverse search [5].
Given a polytope $\mathcal{P}$ the set of its vertices V is calculated using the vertex enumeration operation

$$V = \{V_i\}_{i=1}^{N_V} = \text{vert}(\mathcal{P}). \tag{2.5}$$

2.2.3 Projection

By projecting a polytope $\mathcal{P} \in \mathbb{R}^{n+m}$,

$$\mathcal{P} = \left\{ [x'\ y']' \in \mathcal{R}^{n+m} \mid P^x x + P^y y \leq P^c \right\} \subset \mathcal{R}^{n+m} \tag{2.6}$$

onto a lower dimensional subspace $\mathbb{R}^n$, a new lower dimensional polytope

$$\text{proj}_x(\mathcal{P}) = \{x \in \mathcal{R}^n \mid \exists\, y \in \mathcal{R}^m \mid P^x x + P^y y \leq P^c\} \subset \mathcal{R}^n \tag{2.7}$$

is decribed.
Figure 2.3 illustrates the concept of a projection geometrically from 2 dimensions to one dimension. The projection includes all values in $\mathbb{R}^1$ for which the original

polytope $\mathcal{P}$ is defined in $\mathbb{R}^2$. This idea can be generalized to arbitrary dimensions. The projection describes the set of states in the remaining dimensions for which the set was defined in any location of the dimension that is being removed in the projection.

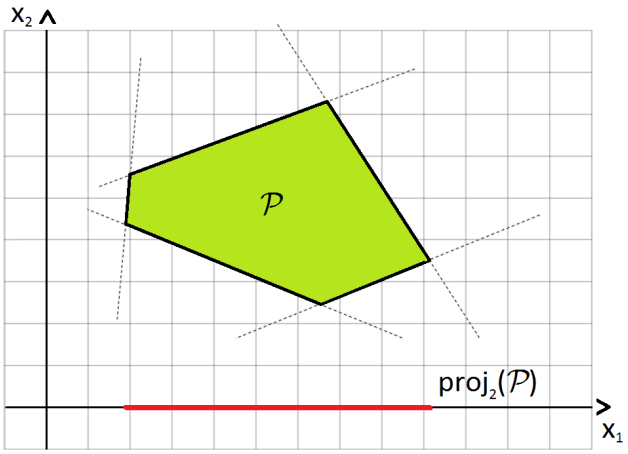


Figure 2.3: Projection of a polytope in $\mathbb{R}^2$ onto the x_1-subspace

2.2.4 Minkowsky sum

The Minkowsky sum is an operation on two polytopes $\mathcal{P}$ and $\mathcal{Q}$ that forms another polytope. The sum is the set that contains all possible combinations of sums of the points $p \in \mathcal{P}$ and $q \in \mathcal{Q}$. The concept is visualized in figure 2.4.

$$\mathcal{P} \oplus \mathcal{Q} := \{p + q \in \mathbb{R}^n \mid p \in \mathcal{P},\ q \in \mathcal{Q}\} \tag{2.8}$$

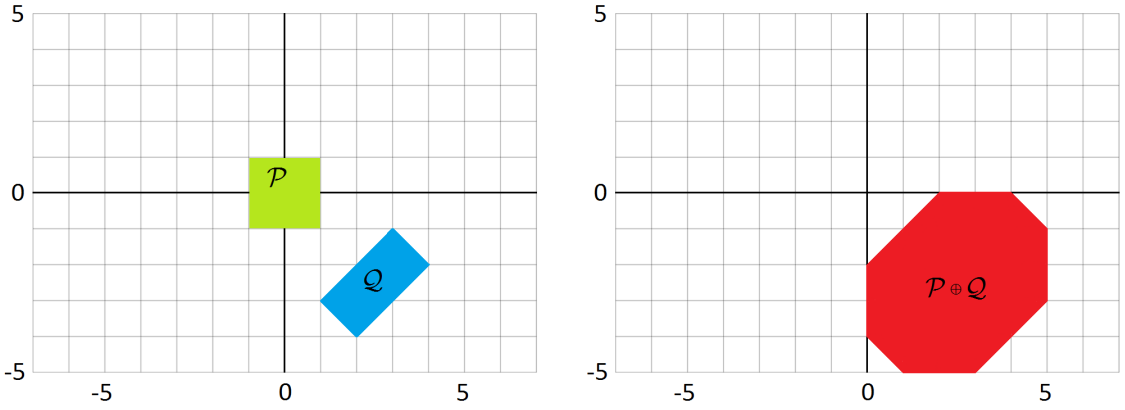


Figure 2.4: Minkowsky sum of two polytopes

2.3 Representations of Convex sets

2.3.1 $\mathcal{H}$-Representation

For the computational representation of sets, two forms of notations are used. The $\mathcal{H}$-representation of a polyhedron describes the intersection of a finite number of closed halfspaces in $\mathbb{R}^n$

$$P = \{x \in \mathbb{R}^n \mid Ax \leq b\}. \tag{2.9}$$

The set of inequalities $Ax \leq b$ describes a finite number of halfspaces bounding the polytope in $\mathbb{R}^n$.

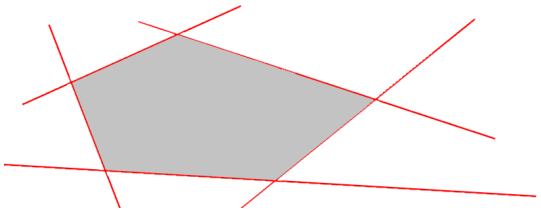


Figure 2.5: Convex set in $\mathcal{H}$-representation

2.3.2 $\mathcal{V}$-Representation

An alternative notation to describe a bounded polyhedron is the vertex representation. By describing the location its extremal points a polytope can be uniquely identified. A $\mathcal{V}$-polytope is generated by the convex hull of a finite set of points $V = \{V_i\}_{i=1}^{N_v}$, where the convex hull is the smallest convex set containing all the points.

$$P = \text{conv}(V) \tag{2.10}$$

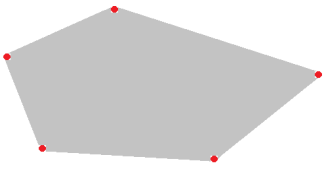


Figure 2.6: Convex set in $\mathcal{V}$-representation

2.3.3 Properties of $\mathcal{H}$- and $\mathcal{V}$-Representations

Any polytope in $\mathcal{H}$-representation can be represented by a $\mathcal{V}$-representation and vice versa. For a mathematical proof the reader is referred to [30]. The $\mathcal{H}$-representation of a polytope can be calculated from a $\mathcal{V}$-polytope using the convex hull operation. To move from a $\mathcal{H}$-representation to a $\mathcal{V}$-representation the vertex enumeration operation is used.

Both $\mathcal{H}$- and $\mathcal{V}$-polytopes can contain redundancies. A polytope in $\mathcal{H}$-representation in $\mathbb{R}^n$ bounded by m halfspaces is said to be in minimal representation if the removal of any row

$$\left[a_{(i,1)}, ..., a_{(i,n)}\right] x \leq b_i, \ i = 1, ..., m \tag{2.11}$$

would change the polytope.
Similarly a $\mathcal{V}$-polytope is said to be in minimal representation if the removal of any point V_j would change it. A minimal $\mathcal{V}$-polytope is described by the set of its vertices. A non-minimal representation can contain an arbitrary number of points within the interior of the polytope.

2.4 Invariant Sets

This section introduces a number of set definitions in the context of discrete time linear systems. Denote by $f()$ the state update function of a discrete time linear system with subject to external inputs $u(k)$.

$$x(k+1) = f(x(k), u(k)) = Ax(k) + Bu(k) \tag{2.12}$$

System states and inputs are subject to the constraints

$$x(k) \in \mathcal{X}, \quad u(k) \in \mathcal{U} \tag{2.13}$$

which are described by the convex polytopes

$$\begin{aligned} \mathcal{X} &\triangleq \{x \mid Hx \leq h\} \\ \mathcal{U} &\triangleq \{u \mid H_u u \leq h_u\}. \end{aligned} \tag{2.14}$$

The system is assumed to be n-dimensional in state and subject to m-inputs.

2.4.1 Reach()-Set

The one step reachable set for the system (2.12) describes the set of states that can be reached by the function $f()$ under some input $u \in \mathcal{U}$. It is defined for a set of initial states $\mathcal{S}$ and described by:

$$\text{Reach}(\mathcal{S}) \triangleq \{x \in \mathbb{R}^n \ \mid \ \exists\, x(0) \in \mathcal{S}, \ \exists\, u(0) \in \mathcal{U} \ s.t. \ x = f(x(0), u(0))\} \tag{2.15}$$

2.4.2 Pre()-Set

As the dual of the one step reachable set, Pre() sets describe the set of the system (2.12) that can evolve to a target set $\mathcal{S}$ in one step under some input $u \in \mathcal{U}$.

$$\text{Pre}(\mathcal{S}) \triangleq \{x \in \mathbb{R}^n \ \mid \ \exists u \ \in \mathcal{U} \ s.t. \ f(x, u) \subseteq \mathcal{S}\} \tag{2.16}$$

The algorithm presented in [5] to calculate the Pre() for a non-autonomous system is based on the description of the state and input constraints by the polytopes $\mathcal{X}$ and $\mathcal{U}$ in $\mathcal{H}$-representation.

$\text{Pre}(\mathcal{S})$ is calculated by a linear mapping of the state constraints through the system dynamics.

$$\begin{aligned}\text{Pre}(\mathcal{S}) &\triangleq \{x \in \mathbb{R}^n \mid \exists\, u \in \mathcal{U} \qquad\qquad s.t.\ f(x,u) \in \mathcal{S}\} \\ &\triangleq \left\{x \in \mathbb{R}^n \mid \exists\, u \in \mathbb{R}^m \quad s.t. \begin{bmatrix} HA & HB \\ 0 & H_u \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \leq \begin{bmatrix} h \\ h_u \end{bmatrix}\right\}\end{aligned} \tag{2.17}$$

This definition yields a resulting polytope in the $n+m$-dimensional state-input space. To describe $\text{Pre}(\mathcal{X})$ in the n-dimensional state space a projection operation as described in section 2.2.3 can be used.

2.4.3 Control Invariant Set

Generalizing this idea, we can describe a set of states $\mathcal{C}$ for which there exist an input $u \in \mathcal{U}$ such that the system can remain within $\mathcal{C} \subseteq \mathcal{X}$ for all future states.
The set $\mathcal{C} \subseteq \mathcal{X}$ is characterized as a *control invariant set* for the system (2.12) if and only if

$$\mathcal{C} \triangleq \{x_k \in \mathbb{R}^n \mid \exists\, u_k \in \mathcal{U},\ x_{k+1} \in \mathcal{C}\} \tag{2.18}$$

The set that contains all invariant sets for a system (2.12) subject to the constraints (2.13) is called *maximal control invariant set* $\mathcal{C}_\infty$.
The following algorithm provided in [5] shows the calculation for the maximal control invariant set:

Data: State Update Function $f(x,u)$, Set of State Constraints $\mathcal{X}$, Set of Input Constraints $\mathcal{U}$
Result: $\mathcal{C}_\infty$
$k = 0$;
$\Omega_0 = \mathcal{X}$;
$\Omega_{k+1} = \text{Pre}(\Omega_k + 1) \cap \Omega_k$;
while $\Omega_{k+1} \neq \Omega_k$ **do**
  $k = k + 1$;
  $\Omega_{k+1} = \text{Pre}(\Omega_k) \cap \Omega_k$;
end
$C_\infty = \Omega_{k+1}$;

Algorithm 1: Calculation of the Maximal Control Invariant Set

2.4.4 N-Step Controllable Set

For an arbitrary target set $\mathcal{O} \subseteq \mathcal{X}$ the set of states that can be driven to an arbitrary target set $\mathcal{O}$ in N-steps is called the *N-step controllable set* $\mathcal{K}_N(\mathcal{O})$.
It is defined by:

$$\mathcal{K}_N(\mathcal{O}) \triangleq \left\{x_0 \in \mathbb{R}^n \mid \exists \{u_k \in \mathcal{U}\}_0^{N-1}, \{x_k \in \mathcal{X}\}_0^{N-1}, x_N \in \mathcal{O}\right\} \tag{2.19}$$

This set is computed in a very similar way as the maximal control invariant set by iteratively calculating the Pre() and determining its intersection with the state

constraints. Algorithm 2 presets the procedure to calculate the N-step controllable Sets for a linear discrete time system.

Data: Target Set $\mathcal{O}$, Number of Iterations N, State Update Function $f = (A, B)$, State Constraints $\mathcal{X}$, System Input Constraints $\mathcal{U}$
Result: $\mathcal{K}_N(\mathcal{O})$
$\mathcal{K}_0(\mathcal{O}) = \mathcal{O}$;
for $k \in [0, 1, 2, ..., N]$ **do**
  $\mathcal{K}_{k+1}(\mathcal{O}) = \text{Pre}(\mathcal{K}_k) \cap \mathcal{X}$
end

Algorithm 2: Calculation of the N-Step Controllable Set

2.5 Example: A Simple Discrete Integrator

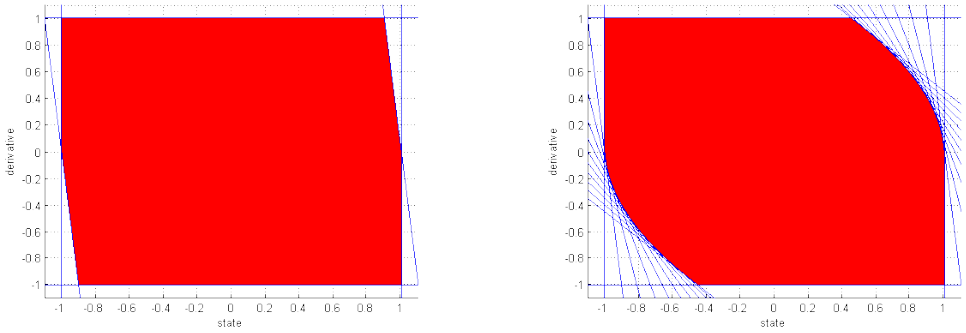
The meaning of these definitions can be illustrated by a small example in two dimensions. Consider a discrete time integrator system with a sampling time of $t_s = 0.1$ s defined by

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & t_s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ t_s \end{bmatrix} u(k) \tag{2.20}$$

subject to the state and input constraints

$$\begin{aligned} \mathcal{X} &\triangleq \left\{ x \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right\} \\ \mathcal{U} &\triangleq \left\{ u(k) \in \mathbb{R} \;\middle|\; \begin{bmatrix} 1 \\ -1 \end{bmatrix} u(k) \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}. \end{aligned} \tag{2.21}$$

Using algorithm 2 we can calculate the N-step controllable set for the target set $\mathcal{X}$. For the resulting set of states we can guarantee that the future trajectory can remain within the state constraints $\mathcal{X}$ for N-steps while using control action contained in $\mathcal{U}$. Iterating this procedure for a finite number of steps yields an outer approximation of the maximal control invariant set introduced in section 2.4.3.
The results of this calculation are shown in figure 2.7. The one-step maximal controllable set is only slightly reshaped compared with the original state constraints. Two additional hyperplanes were added to introduce tighter bounds on the polytope. After multiple iteration steps the shape of the set becomes skewed. The approximation becomes round shapes and the number of hyperplanes increases significantly.
The shape can be intuitively explained by examining the system equations (2.20). Consider the case when the derivative state $x_2 = 1$. Due to the input constraints for each discrete step x_2 can only decrease by 0.1. This means to avoid that the state x_1 to violates the constraints in the future we can calculate how much x_1 will change over the period when x_2 is driven from one to zero to avoid a further increase of x_1.



(a) 1-step controllable set **(b)** 20-step controllable set

Figure 2.7: Evolution of the n-step controllable set for the discrete integrator

Using this approach we can derive that

$$x_1 \leq 1 - 0.1 * (1 + 0.9 + 0.8 + ... + 0.1) = 0.45. \tag{2.22}$$

Therefore x_1 may not be larger than 0.45 if $x_2 = 1$.
Assume we would attempt to formulate a controller that ensures that the state and input constraints of the system are never violated. It is clear that there exists no solution for such a controller if the current state of the system was outside the maximum control invariant set. The existence of a feasible control law to solve the problem depends on the initial state of the system.

3

Vehicle System Modelling

In this chapter we introduce the dynamic models of the vehicle used in this work. We describe the motion in the vehicle reference frame in terms of longitudinal and lateral velocities u and v and the yaw rate r.
In this thesis we describe the vehicle's the position on a straight multilane road. The coordinate s describes the position along the road, e denotes the the lateral offset from the center of a desired lane and ψ measures the heading error relative to the road orientation as illustrated in figure 3.1.

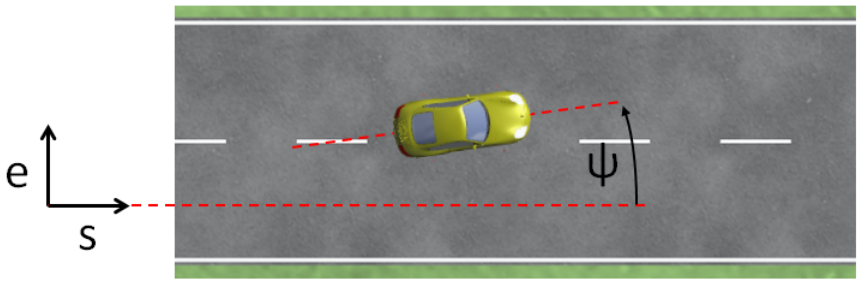


Figure 3.1: Road coordinate system

Assuming small angles for the vehicle heading angle ψ and $u \gg v$ we can describe the motion of the vehicle in this coordinate system by

$$\begin{aligned} \dot{\psi} &= r \\ \dot{e} &= \sin(\psi)\ u + \cos(\psi) v \approx \psi\ u + v \\ \dot{s} &= \cos(\psi) u - \sin(\psi) v \approx u - \psi\ v \approx u. \end{aligned} \tag{3.1}$$

For each position along the road s we define lane boundaries

$$e_{\min}(s) \leq e(s) \leq e_{\max}(s) \tag{3.2}$$

limiting the available lateral space on the road.
The vehicle is subject to longitudinal forces F_x as well as lateral forces F_y. The normal tire forces are denoted by F_z. The vehicle is characterized by its mass m and its moment of inertia I_z. The wheelbase is denoted by l and the distances from front and rear axle to the location of the center of gravity are denoted by a and b and the track width is d. The steering angle of the front wheels is δ and the tire slip angles are α.

3.1 Two Track Model

The two track model is a general model to describe the planar motion of a vehicle. It takes into account the forces acting on the vehicle on the four individual wheels. It is derived from first principles by calculating the force balances for lateral and longitudinal acceleration and a moment balance for the yaw rotation as shown in figure 3.2. The interaction with the road of each of the individual tires is modelled by a separate forces $\{F_{xi}\}_{i=1}^{4}$ and $\{F_{yi}\}_{i=1}^{4}$ oriented longitudinal and lateral to the tire's heading direction. The equations for the two track model are shown in (3.3)

$$\begin{aligned} m\left(\dot{u}-rv\right) =& \cos\delta\left(F_{x1}+F_{x2}\right)-\sin\delta\left(F_{y1}+F_{y2}\right)+F_{x3}+F_{x4} \\ m\left(\dot{v}+ru\right) =& \cos\delta\left(F_{y1}+F_{y2}\right)+\sin\delta\left(F_{x1}+F_{x2}\right)+F_{y3}+F_{y4} \\ I_z\dot{r} =& a\left(\cos\delta\left(F_{y1}+F_{y2}\right)+\sin\delta\left(F_{x1}+F_{x2}\right)\right)-b\left(F_{y3}+F_{y4}\right) \\ &+d\left(\sin\delta\left(F_{y1}-F_{y2}\right)+\cos\delta\left(F_{x2}-F_{x1}\right)-F_{x3}+F_{x4}\right). \end{aligned} \tag{3.3}$$

The equations (3.3) are highly nonlinear due to the trigonometric functions and the multiplicative coupling between the states. While this model is well suited for simulation applications, simpler models are better suited for control design.

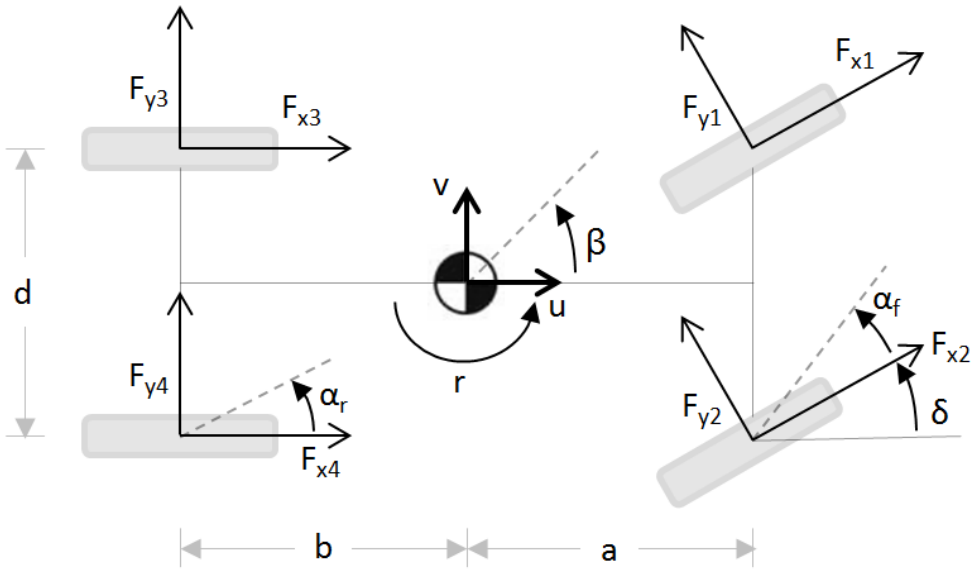


Figure 3.2: Two track vehicle bodel

3.2 Bicycle Model

To formulate the model predictive control problem as a convex optimization problem, for which efficient solver exist, a less complex and linearized version of the vehicle model is necessary. The equations in (3.3) can be simplified by lumping the left and right tire forces together as illustrated in figure 3.3.

$$\begin{aligned} F_{yf} =& F_{y1} + F_{y2} \\ F_{yr} =& F_{y3} + F_{y4} \end{aligned} \tag{3.4}$$

Furthermore we assume small angles for the steering angle

$$\begin{aligned} \sin\delta =& \delta \\ \cos\delta =& 1 \end{aligned}. \tag{3.5}$$

Combining equations (3.3) and using the assumptions (3.4) and (3.5) we can derive the equations of motion for the planar rigid one track model, commonly called the bicycle model

$$\begin{aligned} \dot{u} =& \frac{F_{xf} + F_{xr} - \delta F_{yf}}{m} + rv \\ \dot{v} =& \frac{F_{yf} + F_{yr} + \delta F_{xf}}{m} - ru \\ \dot{r} =& \frac{aF_{yf} - bF_{yr}}{I_z}. \end{aligned} \tag{3.6}$$

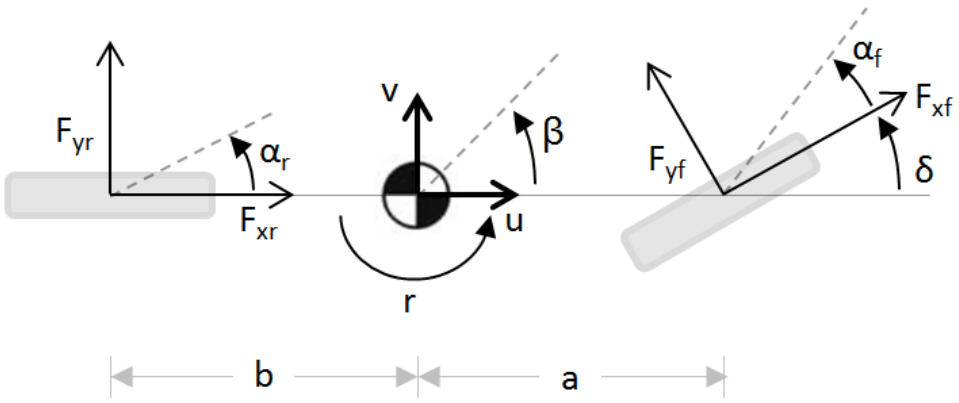


Figure 3.3: Bicycle model

We can describe the side slip angle β of the vehicle at its center of gravity and its small angle approximation by

$$\beta = \tan^{-1}\left(\frac{v}{u}\right) \approx \frac{v}{u}. \tag{3.7}$$

For the bicycle model in figure 3.3 we can denote the front and rear tire slip angles by

$$\begin{aligned} \alpha_f =& \arctan\left(\frac{v + ar}{u}\right) - \delta \approx \beta + \frac{a}{u}r - \delta \\ \alpha_r =& \arctan\left(\frac{v - br}{u}\right) \approx \beta - \frac{b}{u}r. \end{aligned} \tag{3.8}$$

3.3 Tire Models

Since the entire interaction of a vehicle with the road is based on the distribution of forces through four small tire contact patches, each approximately the size of a hand, the modelling of this interaction is crucial to the modeling of the vehicle motion as a whole.
Pacejka [28] describes several state-of-the-art tire models. In this thesis two different tire models are used. Due to the necessity of analytical calculation and the restriction to linear model formulations for control design, this thesis makes use of simple parameterizable tire models.

3.3.1 Linear Tire Model

The linear tire model is such a model that assumes a simple linear relationship between tire side slip angle and the generation of lateral tire force. This model is fairly accurate for small slip angles but the accuracy quickly decreases as the lateral force reaches approximately half its maximum value as shown in figure 3.4. The tire model is simply characterized by the so called cornering stiffness

$$F_y = -C_\alpha \alpha. \tag{3.9}$$

3.3.2 Brush Tire Model

The more complex brush tire model is an approximation of the tire forces under the assumption that the contact patch of the tire is subject to a parabolic vertical load distribution along the contact patch. Due to the slip angle of the tire the lateral deflection of the tire is larger towards the rear of the contact patch. Thus individual parts of the contact patch can be either in adhesive or sliding friction. The brush tire model used in this thesis is similar to the the one proposed by Fiala [15] and was presented by Fromm in [21] and used by Hindiyeh and Gerdes in [2] as well as in multiple following pieces of work e.g. by Erlien and Gerdes in [9].
The advantage of the brush tire model is that it gives a more accurate description of the vehicle tires behavior by modelling the saturation of the lateral tire force depending on the road friction coefficient. At the same time the brush tire model sill only requires two parameters which can easily be identified from experimental data.
The saturation of tire forces is modelled as shown in equation (3.10) and can be applied to individual tires as well as lumped front or rear axles

$$F_y = \begin{cases} -C_\alpha \tan\alpha + \frac{C_\alpha^2}{3\mu F_z} |\tan\alpha| \tan\alpha - \frac{C_\alpha^3}{27\mu^2 F_z^2} \tan\alpha^3 & , |\alpha| \leq \alpha_{sl} \\ -\mu F_z \text{sgn}\alpha & , |\alpha| > \alpha_{sl}. \end{cases} \tag{3.10}$$

The lateral force is saturated at its peak value for slip angles where the entire contact patch is sliding [23]. This value can be stated analytically and results in

$$\alpha_{sl} = \arctan \frac{3\mu F_z}{C_\alpha}. \tag{3.11}$$

While more advanced tire models can capture additional phenomena, such as reduced friction for high tire slip angles or the coupling of longitudinal and lateral tire forces, the models used in this thesis disregards these effects. Since the controller in this thesis attempts to control the vehicle only in the region up to the peak friction value, accounting for these effect is not necessary in this context.
An example tire curve from the brush tire model in comparison with the linear model and a more complex Magic Tire Formula is shown in figure 3.4. The Magic Tire Formula and the brush tire model have matching peak friction values. All three models are parametrized with the same cornering stiffness. The figure shows how, in addition to the saturation effect, the Magic Tire Formula also accounts for the decrease in lateral force past when the tire starts sliding.

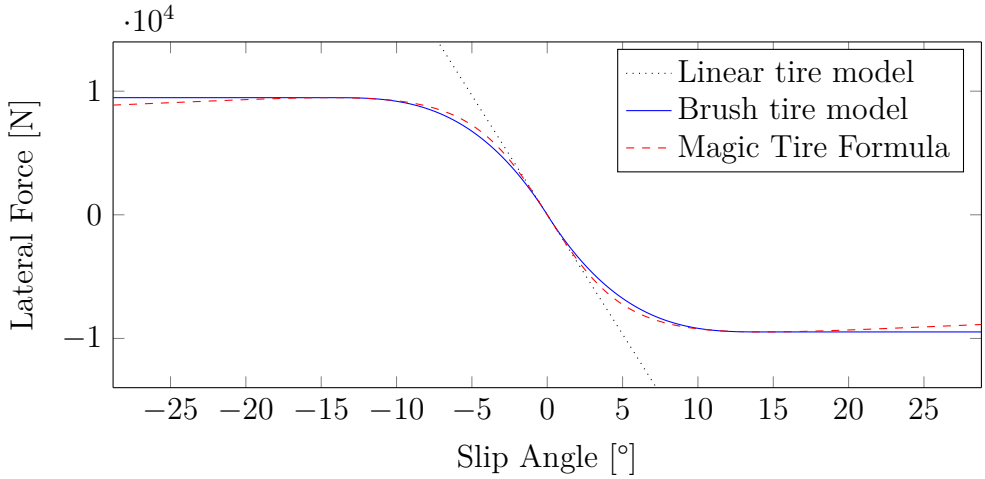


Figure 3.4: Comparison between tire models

3.4 Linear Bicycle model

During straight line vehicle motion we can assume

$$\begin{aligned} u &= \overline{u} \\ \delta &= 0 \\ v &= 0 \\ r &= 0. \end{aligned} \tag{3.12}$$

By linearizing the equations of motion of the bicycle model (3.6) for small deviation around the constant velocity $\overline{u}$ and conditions for straight line motion (3.12) we can derive the linear bicycle model

$$\begin{aligned} \dot{v} &= \frac{F_{yf} + F_{yr}}{m} - r\overline{u} \\ \dot{r} &= \frac{aF_{yf} - bF_{yr}}{I_z}. \end{aligned} \tag{3.13}$$

Using the approximation of the vehicle side slip from equation (3.7), (3.6) can be stated in terms of sidslip and yaw rate as

$$\begin{aligned}\dot{\beta} =& \frac{F_{yf} + F_{yr}}{m\overline{u}} - r \\ \dot{r} =& \frac{aF_{yf} - bF_{yr}}{I_z}.\end{aligned} \tag{3.14}$$

3.5 Force Input Model

By combining a linear rear tire model (3.9) and the approximation for the rear slip angle (3.8) we can state the rear tire force as a function of the vehicle states β and r as

$$F_{yr} = -C_{\alpha r}\alpha_r = -C_{\alpha r}\left(\beta - \frac{b}{u}r\right). \tag{3.15}$$

By replacing the rear tire force in the linear bicycle model (3.14) with equation (3.15), the lateral vehicle dynamics are derived as a function of the lateral force generated by front steering F_{yf} as

$$\begin{aligned}\dot{\beta} &= -\frac{C_{\alpha r}}{m\overline{u}}\beta + \left(\frac{bC_{\alpha r}}{m\overline{u}^2} - 1\right)r + \frac{1}{m\overline{u}}F_{yf} \\ \dot{r} &= \frac{bC_{\alpha r}}{I_{zz}}\beta - \frac{b^2C_{\alpha r}}{I_{zz}\overline{u}}r + \frac{a}{I_{zz}}F_{yf}.\end{aligned} \tag{3.16}$$

Since the effects from the front tire enter the dynamics of the bicycle model simply as a lateral force input, the non-linearity of the front tire can be captured outside the dynamic equations. This is not possible for the rear tire as the rear tire forces are directly coupled with the vehicle states.
In state space formulation the equations of motion of the linear force input model (3.16) can be stated as

$$\begin{bmatrix}\dot{\beta} \\ \dot{r}\end{bmatrix} = \begin{bmatrix}-\frac{C_{\alpha r}}{m\overline{u}} & \frac{bC_{\alpha r}}{m\overline{u}^2} - 1 \\ \frac{bC_{\alpha r}}{I_{zz}} & -\frac{b^2C_{\alpha r}}{I_{zz}\overline{u}}\end{bmatrix}\begin{bmatrix}\beta \\ r\end{bmatrix} + \begin{bmatrix}\frac{1}{m\overline{u}} \\ \frac{a}{I_{zz}}\end{bmatrix}F_{yf}. \tag{3.17}$$

3.6 Linear Time Varying Model

To obtain a better approximation of the nonlinear rear tire properties illustrated in figure 3.4 in dynamic maneuvers, the rear tire force can be linearized around the current rear slip angle instead of the origin as shown in model (3.17).
Using a first order Taylor expansion of the brush tire model 3.10, the lateral rear tire force F_{yr} at a specific operating point $\overline{\alpha}_r$ is described by a current lateral tire force $\overline{F}_{yr}$ and a local cornering stiffness $\widetilde{C}_{\alpha r}$.

$$F_{yr} = \overline{F}_{yr} + \widetilde{C}_{\alpha r}\left(\alpha_r - \overline{\alpha}_r\right) \tag{3.18}$$

By replacing the rear tire model (3.15) with (3.18) in the linear bicycle model (3.14) the model (3.17) can be modified to become

$$\begin{bmatrix}\dot{\beta}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}-\frac{\widetilde{C}_{\alpha r}}{m\overline{u}} & \frac{b\widetilde{C}_{\alpha r}}{m\overline{u}^2}-1\\ \frac{b\widetilde{C}_{\alpha r}}{I_{zz}} & -\frac{b^2\widetilde{C}_{\alpha r}}{I_{zz}\overline{u}}\end{bmatrix}\begin{bmatrix}\beta\\ r\end{bmatrix} + \begin{bmatrix}\frac{1}{m\overline{u}}\\ \frac{a}{I_{zz}}\end{bmatrix} F_{yf} + \begin{bmatrix}\frac{\left(\overline{F}_{yr}+\widetilde{C}_{\alpha r}\overline{\alpha}_r\right)}{m\overline{u}}\\ \frac{-b\left(\overline{F}_{yr}+\widetilde{C}_{\alpha r}\overline{\alpha}_r\right)}{I_{zz}}\end{bmatrix}. \tag{3.19}$$

Using this technique, the model is able to account for the saturation effects incorporated in the brush tire model. This approach can be used for control design when the current rear sideslip can be measured from high precision sensors to increase model accuracy over a short prediction horizon. Therefore the time varying model (3.19) yields better accuracy in dynamic maneuvers than the linear force input model in equation (3.17) and still fulfils the requirements imposed to be used in efficient optimization algorithms. The model was used by Beal and Gerdes [1] for model predictive control applications at the limits of handling. However it can only be used when a good estimate for the rear side slip angle is available. For the feasibility predictions derived in this work this model cannot be used since we don't have any knowledge about the future vehicle motion and therefore are forced to used the LTV model with the rear tire linearization around the origin as presented in equation (3.17)

3.7 Safe Handling Envelope

To ensure handling stability for the vehicle models 3.17 and 3.19 yaw rate and sideslip must be bounded. For excessive values of yaw rate and sideslip a vehicle can become unstable and spin out [1].
Hsu and Gerdes [24] showed an approach that uses model predictive control in combination with a safe handling envelope that imposes limits on yaw rate and side slip that can help a vehicle avoid approaching these situations before they occur.
As a limit on yaw rate they use the maximum possible steady state yaw rate r_{max} based on the vehicle tire friction μ. Using the linear bicycle model from equation (3.14) and setting $\dot{\beta} = 0$ and $\dot{r} = 0$ we can solve for the yaw rate limits

$$|r| \leq r_{max} = \begin{cases}\frac{F_{yr_{max}}(1+b/a)}{mu} & | \quad F_{yf_{max}} \geq \frac{b}{a}F_{yr_{max}}\\ \frac{F_{yr_{max}}(1+a/b)}{mu} & | \quad F_{yf_{max}} < \frac{b}{a}F_{yr_{max}}\end{cases}. \tag{3.20}$$

Using the description for the rear slip angle $\alpha_r = \beta - \frac{b}{u}r$, a natural limit for the side slip arises. By using equation (3.11) and limiting the sideslip to the value where the rear tire reaches its maximum force $\alpha_r \leq \alpha_{r_{sl}}$ the side slip bound becomes

$$\frac{b}{u}r - \alpha_{r_{sl}} \leq \beta \leq \frac{b}{u}r + \alpha_{r_{sl}}. \tag{3.21}$$

Both bounds on this safe handling envelope scale naturally with the longitudinal velocity u which makes them applicable over a large range of velocities. In their work Beal and Gerdes showed that for any point on the safe handling envelope a controller is capable of keeping the vehicle within these boundaries. These constraints describe

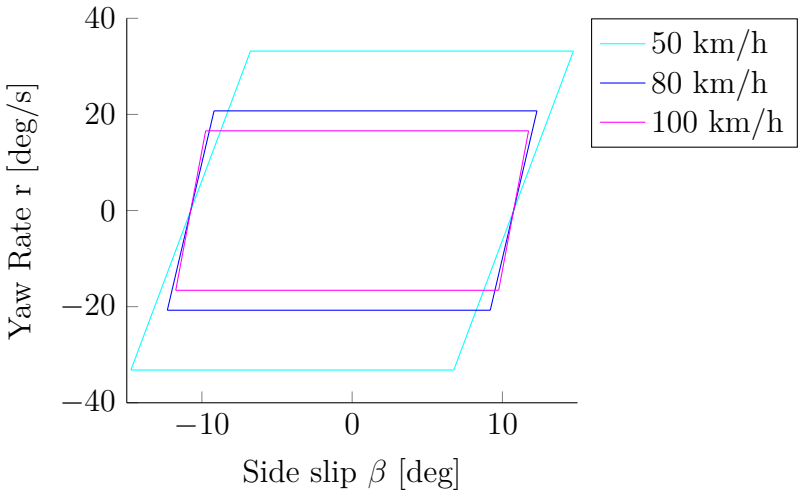


Figure 3.5: Scaling of the Safe Handling Envelope for different velocities

an invariant set for yaw rate and side slip that will be used in further control design and reachability studies in this work.

4

Model Predictive Control

This chapter introduces the basic concepts of model predictive control to illustrate some challenges for utilizing MPC as a real time control method for vehicle motion control. This chapter uses notations and derivation from [5].

Model predictive control is an optimal control approach typically aimed at solving control problems in the presence of constraints. These can be actuator constraints such as limited steering angles or steering rates in a car or constraints that describe requirements on the system state. This type of constraint can for example be a limit on a maximum allowed velocity in some direction or a formulation of road boundaries that shall not be crossed.

By minimizing or maximizing a cost function the optimal inputs to the system can be determined by solving an optimization problem. Depending on the size of the problem the solution of such optimization problem is a computationally expensive process.

The approach became popular in the process industry in the 1980s. Model predictive control provided the possibility to solve complex multi-variable control problems that were earlier difficult to handle. As these system typically involved very slow dynamics, it was sufficient to adjust input over the period of minutes and model predictive control schemes could be applied with the limited computing power available [26].

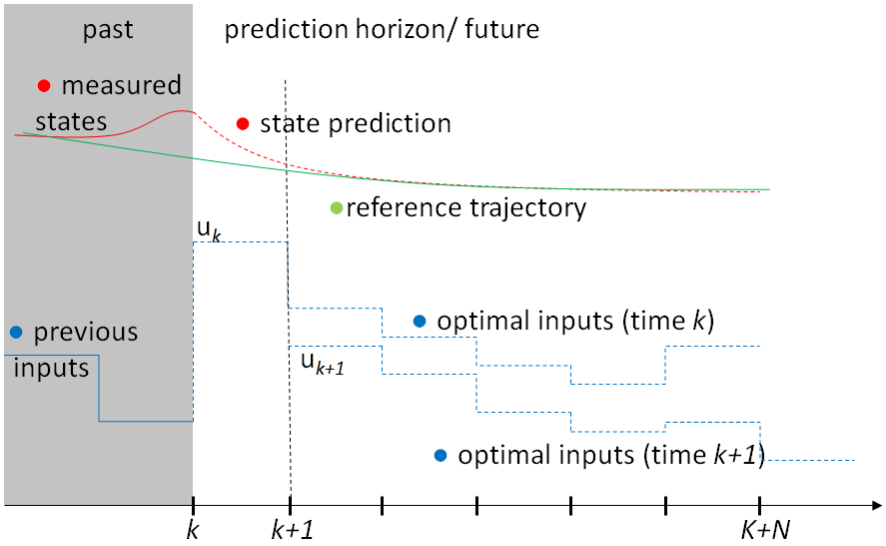


Figure 4.1: Receding horizon principle

Enabled by the increasing computing power available in embedded systems, model predictive control for vehicle motion became a vibrant field of research in the past 10 years [8].

4.1 Receding Horizon Principle

The general idea of receding horizon control consists of repetitively solving a finite time optimal control problem as illustrated in figure 4.1. The behavior of the system is predicted over a discrete time horizon using a mathematical model of the system. From the resulting optimal input sequence, the first element is applied as the next input to the system. After the sampling time t_s the optimal control problem is solved again based on the updated measurements and the new prediction horizon which is shifted one step with respect to the previous calculation [26].
A receding horizon controller can be formulated for any type of nonlinear, time-invariant system

$$x(k+1) = g\left(x(k), u(k)\right) \tag{4.1}$$

subject to a set of constraints

$$\begin{aligned} x &\in \mathcal{X} \\ u &\in \mathcal{U} \end{aligned} \tag{4.2}$$

in the form of a nonlinear programming problem [5].
Given a discrete time horizon length N and the vector of future inputs

$$U_{0\to N} \triangleq \left[u_0', ...u_{N-1}'\right] \tag{4.3}$$

we can define an objective function

$$J_{0\to N}\left(x_0, U_{0\to N}\right) \triangleq p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k) \tag{4.4}$$

that consists of stage costs $q\left(x_k, u_k\right)$ for each time step of the prediction horizon and a terminal cost $p(x_N)$.
We can formulate a general finite time optimal control problem for a current state $x(0)$ and the desired set of terminal set $\mathcal{X}_f$ as

$$\begin{aligned} J^*_{0\to N} = \min_{U_{0\to N}}: \quad & J_{0\to N}\left(x(0), U_{0\to N}\right) & \\ \text{subj. to:} \quad & x_{k+1} = g(x_k, u_k), & k = 0, ..., N-1 \\ & h\left(x(t), u(t)\right) \leq 0, & k = 0, ..., N-1 \\ & u_k \in \mathcal{U}, & k = 0, ..., N-1 \\ & x_k \in \mathcal{X}, & k = 0, ..., N-1 \\ & x_N \in \mathcal{X}_f & \\ & x_0 = x(0). & \end{aligned} \tag{4.5}$$

The application of this control approach in a system with hard real time constraints requires the solution of the optimization problem in finite time. This cannot be

guaranteed for an arbitrary nonlinear program as shown in (4.5). While there have been applications of general purpose nonlinear solver algorithms in automotive applications in recent years, they cannot provide convergence guarantees and have so far been limited to relatively large sampling times due to high computational complexity [18].
The majority of model predictive control approaches for vehicle motion control in the past years has made use of efficient solver algorithms for convex optimization problems [1], [2], [7], [9], [10].
These algorithms have the advantage of efficiently solving large scale optimization problems. On the downside they require the user to simplify the general problem (4.5) and approximate it such that the problem formulation can be represented by a linear system with a convex cost function and affine constraints.
To fulfil these requirements the nonlinear system in (4.1) must be linearized such that it can be represented in the form

$$x(t+1) = Ax(t) + Bu(t) \tag{4.6}$$

subject to the state and input constraints

$$\begin{aligned} \mathcal{X} &= \{x \mid Hx \leq h\} \\ \mathcal{U} &= \{u \mid H_u u \leq h_u\}. \end{aligned} \tag{4.7}$$

For the vehicle models introduced in chapter 3 we showed how the nonlinear model (3.6) can be linearized as shown in (3.17) to fit these requirements.
The general objective function (4.4) can replaced by a quadratic version

$$J_0\left(x_0, U_{0\to N}\right) \triangleq x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k \tag{4.8}$$

to ensure that the cost function is convex. The terms Q and R are positive semidefinite matrices that assign cost to individual states and inputs over the prediction horizon. The positive semidefinite matrix P assigns the terminal cost to the final state.
This allows to describe the optimal control problem as a quadratic program

$$\begin{aligned} J_0^* = \min_{U_{0\to N}}: \quad & J_0\left(x(0), U_{0\to N}\right) \\ \text{subj. to:} \quad & x_{k+1} = Ax_k + Bu_k, \quad k = 0, ..., N-1 \\ & H_k x_k \leq h_k, \qquad\qquad k = 1, ..., N \\ & H_{u_k} x_k \leq h_{u_k}, \qquad k = 0, ..., N-1 \\ & x_0 = x(0) \end{aligned} \tag{4.9}$$

for which efficient algorithms exist that can be used in real time constrained optimal control.

4.2 Modelling of Constraints

Solving a finite time optimal control problem (4.9) for a linear system (4.6) subject to state and input constraints (4.7) there may exist initial conditions $x_0 \in \mathcal{X}$ such

that there exist no solutions to the problem without violating either input or state constraints.
Considering the simple discrete integrator example introduced in section 2.5, it is easy to find an initial state x_0 such that there does not exist any feasible trajectory for a controller to keep the system states within $\mathcal{X}$ for a finite length horizon. For the initial state $x_0 = [1 \quad 1]'$ there exist no allowed control action $u \in \mathcal{U}$ such that the state x remains in $\mathcal{X}$ in the next time step.
If an MPC controller is faced with this situation the outcome resulting behavior is dependent on the actual implementation of the solver used. Since no valid result can be calculated it becomes unclear what control action should be taken. Considering a car heading for an obstacle at highway speed this certainly presents a situation that should be avoided.
A common way of avoiding infeasibility is to formulate state constraints as *soft constraints* [6]. Additional slack variables λ_i are introduced in the constraint inequalities that are penalized extensively in the cost function if they becomes non-zero. While input constraints in real systems are usually limited by physical constraints, state constraints often reflect desired behavior. Therefore is may be acceptable in many situations that it is acceptable that the state constraints can be slightly violated to allow the existence of a valid solution.
Using soft constraints the optimization problem (4.9) can be formulated as

$$\begin{aligned} J_0^* = \min_{U_{0\to N}}: \quad & J_0(x(0), U_{0\to N}) + l(\lambda) \\ \text{subj. to:} \quad & x_{k+1} = Ax_k + Bu_k, && k = 0, ..., N-1 \\ & H_k x_k \leq h_k + \lambda, && k = 1, ..., N \\ & H_{u_k} x_k \leq h_{u_k}, && k = 0, ..., N-1 \\ & x_0 = x(0) \\ & \lambda \geq 0 \end{aligned} \tag{4.10}$$

with the additional cost term

$$l(\lambda) = \begin{cases} 0 & , \lambda = 0 \\ \gg J_0(x(0), U_0) & , \lambda \neq 0. \end{cases} \tag{4.11}$$

If $l(\lambda_i)$ is much larger than the original cost function for all values $\lambda \neq 0$ then the optimal solution remains the same as long as there exists a feasible solution to the problem. If the problem becomes infeasible the solution will be the optimal trajectory to bring the system back into the desired region of the state space.
While in many control problems state constraints are often desired values rather than fixed physical limits this is a very useful strategy. A disadvantage of this strategy however is, the increased complexity of the problem due to the additional variables. Also the introduction of cost terms that are several orders of magnitude larger than the original cost function J, issues with numerical accuracy and convergence may arise and the system can become ill-conditioned.
In the lane change problem introduced in section 1.2 infeasibility can arise because the velocity sequence $\overline{u}_{1,...N-1}$ chosen to linearize the vehicle model 3.17 for the individual steps of the prediction horizon is too high. To avoid these cases of infeasibility

the velocity profile for linearization should be chosen such that infeasibility does not occur.

4.3 CVXgen

To solve convex optimization problem as illustrated in equation (4.9) CVXgen is used in this work. CVXgen is a code generator that can generate fast custom C-code for solving convex optimization problems [27]. It provides an online interface to specify the optimization problem and generates C-code that can be implemented into any embedded system that is capable of compiling C-code. The code is optimized for fast calculations and is almost branch free to allow consistent and predictable execution times.
The simple QP-problem in equation (4.9) can be described in CVXgen as the following.

```
minimize
  quad(x[N], P) + sum[k=1..N-1](quad(x[k], Q) + quad(u[k-1], R))
subject to
  x[k+1] == A*x[k] + B*u[k], k=0..N-1  # dynamics constraints.
  H[k]*x[k] <= h[k], k=1..N  # state constraints
  H_u[k]*u[k] <= h_u[k], k=0..N-1  # input constraints
  x[0] == x0
end
```

5

Feasibility by Construction

As introduced in chapter 1.2 we consider lane change maneuver on a multilane road. The moving obstacle is projected onto a fixed position on the road according to the differential velocity between the ego vehicle and the preceding vehicle and and treated as a static obstacle on the road at the location where the ego vehicle must be in the other lane to avoid a collision.
An MPC-controller (see section 4.1) attempts to solves a finite time optimal control problem (4.9) to calculate a sequence of optimal steering inputs and a trajectory such that the vehicle (**??**) can be steered to safely avoid the obstacle ahead of it.
The controller minimizes a cost function (4.8) that penalizes deviation from the desire lane as well as violent steering maneuvers to ensure smooth motion of the the vehicle if possible.
The optimal control action is subject to the dynamics of the vehicle model (**??**), environmental constraints due to road boundaries and obstacle location (see section **??**) as well as tire force limitations (3.10) incorporated into the handling limits as described in section 3.7.
The finite time optimal control problem is stated as

minimize:	Cost function (4.8)	(5.1a)
subject to:	Vehicle dynamics(3.19)	(5.1b)
	Safe handling envelope (3.20) , (3.21)	(5.1c)
	Environmental boundaries (3.2)	(5.1d)

Designing a steering control algorithm (5.1) where the path is restricted by solid obstacles and lane boundaries poses a control problem where infeasibility may occur. As shown in section 2.5 there may not exist a solution to the problem for some initial conditions. The existence of the solution is also dependent on the velocity used to parametrize the vehicle dynamics (3.19). The problem (5.1) is time varying and standard results to guarantee feasibility from literature do not apply.
Therefore this chapter aims at using tools introduced in chapter 2 to guarantee persistent feasibility for the problem (5.1). Given a certain distance to an obstacle, fixed lane boundaries and vehicle capabilities and the state of the vehicle, there exists an upper bound on the velocity for which feasibility of the problem (5.1) can be ensured.
In [14] Falcone et al. proposed an algorithm for model based threat assessment by using methods from reachability analysis and set invariance theory. By targeting

a set of safe states within their lane, they predicted threats for lane crossing and vehicle instability over a finite time horizon. Using a similar approach the feasibility of the constrained model optimal control problem (5.1) can be studied.
By calculating a control invariant set $\mathcal{C}$ (algorithm 1 in section 2.4.3) for the vehicle model (3.17) subject to the same constraints as in (5.1) we predict the vehicle states that ensure persistent feasibility and safe driving conditions for future states within one lane of the road.
If an obstacle is blocking one lane of the road we use this control invariant set within a single lane $\mathcal{C}$ as a target set located in the open lane as illustrated in figure 5.1.
Using algorithm 2 (section 2.4.4) n-step controllable sets $\mathcal{K}_{1,\dots}(\mathcal{C})$ are determined that guarantee that for all states contained in these sets there exists a sequence of inputs such that the state can be driven to the the target set $\mathcal{C}$ in the safe lane.
With the linear vehicle model (3.17) and road coordinates (3.1), the vehicle motion can be predicted in the state space $X = [\beta\ , r,\ \psi,\ e]'$ by the discrete, parametrized motion model for discrete steps in s in the form

$$X(k+1) = A(\overline{u})x(k) + B(\overline{u})F_{yf}. \tag{5.2}$$

Given the longitudinal velocity of the vehicle $\overline{u}$ and the distance to the obstacle Δ_s in s direction, the number of steps for the controller until the obstacle is reached is given by $N = \Delta_s/\overline{u}$.
With the motion model 5.2 being parametrized by the velocity $\overline{u}$ the control invariant set $\mathcal{C}(\overline{u})$ and the n-step controllable sets $\mathcal{K}_N(\mathcal{C}, \overline{u})$ both become functions of the velocity.
If the state x_0 of the vehicle is contained in the N-step controllable set $\mathcal{K}_N(\mathcal{C}, \overline{u})$ for a fixed velocity u, then persistent feasibility for the maneuver can be guaranteed for that velocity.
This concept is illustrated in figure 5.1. The checkered line depicts the location of the control invariant set $\mathcal{C}$ that guarantees persistent feasibility for future steps. The green lines illustrate locations of the n-step controllable sets $\mathcal{K}_{1,\dots}(\mathcal{C})$. While the image can only show projections of the sets onto the road, the sets are bounded in the four dimensional state space of the vehicle.

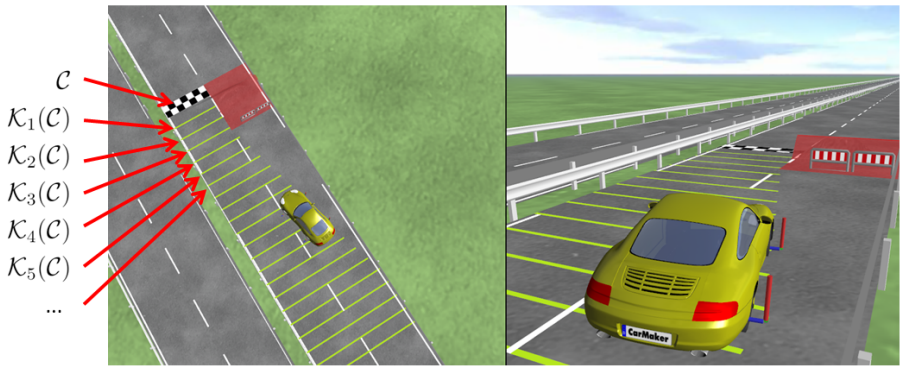


Figure 5.1: Projection of feasible states for a car approaching a static obstacle

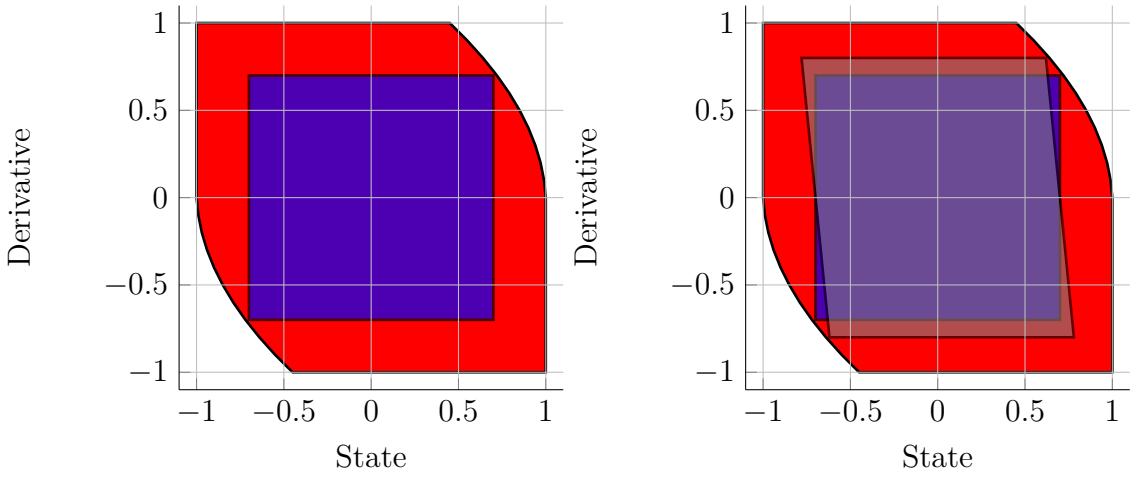
Since the vehicle model is parametrized by the velocity we aim to solve the calculation of $\mathcal{K}_N(\mathcal{C}, \overline{u})$ analytically in section 5.1 to obtain an upper bound on the

velocity that guarantees persistent feasibility. An analytical solution would allow to determine that velocity threshold as an online calculation in a vehicle.
Section 5.2 shows an alternative approach to the problem. By studying the problem offline, it is possible to perform these calculations numerically for a range of velocities. An algorithm implemented in a vehicle could then use these precalculated sets to online determine a velocity for which such a maneuver becomes feasible. The complexity of the online calculation can be lowered significantly at the cost of extensive offline calculations and significant data storage necessary.

5.1 Analytical Approach Using Hyperrectangle Constraints

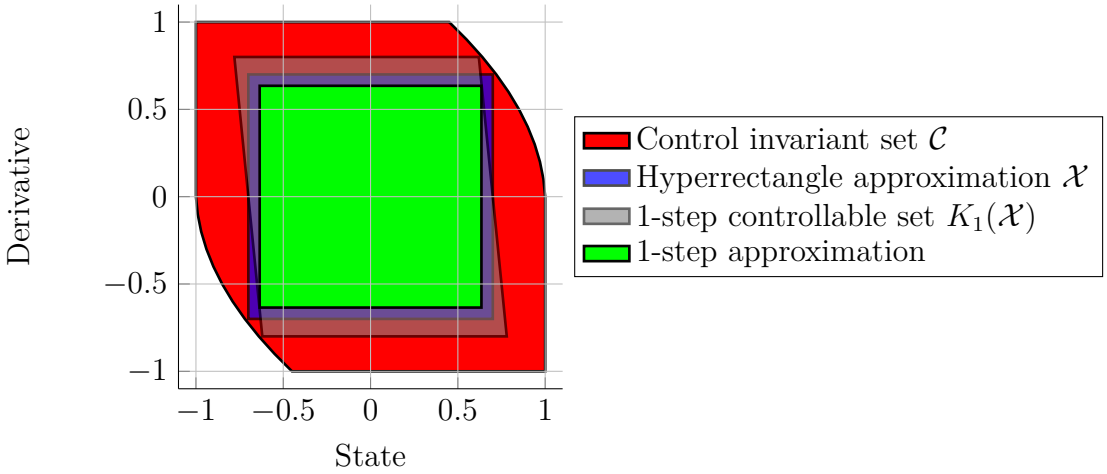
This section investigates the possibility to solve the problem of finding an upper bound on the velocity for the lane change maneuver in an analytical way. The computation to determine such an upper bound should be quick and efficient to allow implementation in embedded computers used in vehicles. At the very least, the calculation should be less computationally expensive than what is needed to repetitively attempt to solve the optimization problem for different velocities until an feasible solution is found.
As shown in section 2.5, one disadvantage of algorithm 2 to calculate N-step controllable sets is, that the complexity of each iteration grows as more hyperplanes are added to the state constraints. If an algorithm can be found that yields an efficient calculation of an inner approximation of the one-step controllable set $\mathcal{K}_1(\mathcal{X})$ and can be represented in a structure such that is has the same structure and the same number of hyperplanes as the target set $\mathcal{X}$, such algorithm may be used to efficiently calculate multiple iterations of algorithm 2 (section 2.4.4) to calculate n-step controllable sets.
Since the physical road boundaries are represented by simple limitations on the lateral position the simplest form to approximate the constraints in (5.1) is a bounded polytope in the shape of a hyperrectangle. A hyperrectangle is a box type geometric shape in a higher dimension $N > 3$ where each state is limited by a single upper and a lower bound.
A simplified illustration of the approach in two dimensions is shown in figure 5.2 for the discrete integrator introduced in section 2.5 in two dimensions. The invariant set $\mathcal{C}$, previously shown in figure 2.7, is approximated by an inner rectangle as shown in figure 5.2a. Then the one-step controllable set is calculated in 5.2b and approximated by another inner rectangle in 5.2c.
For the vehicle performing the lane change maneuver, we consider the discretized version of the linear force input model (3.17) with the road coordinate system (3.1)

$$\begin{bmatrix} v(k+1) \\ r(k+1) \\ \psi(k+1) \\ e(k+1) \end{bmatrix} = \begin{bmatrix} 1 - \frac{C_{\alpha r} t_s}{m\overline{u}} & t_s\left(\frac{bC_{\alpha r}}{mu} - \overline{u}\right) & 0 & 0 \\ \frac{bC_{\alpha r} t_s}{I_{zz}\overline{u}} & 1 - \frac{b^2 C_{\alpha r} t_s}{I_{zz}\overline{u}} & 0 & 0 \\ 0 & t_s & 1 & 0 \\ t_s & 0 & t_s\overline{u} & 1 \end{bmatrix} \begin{bmatrix} v(k) \\ r(k) \\ \psi(k) \\ e(k) \end{bmatrix} + \begin{bmatrix} \frac{t_s}{m} \\ \frac{at_s}{I_{zz}} \\ 0 \\ 0 \end{bmatrix} F_{\mathrm{yf}}(k). \qquad (5.3)$$



(a) Control invariant set $\mathcal{C}$ and hyperrectangle shaped approximation $\mathcal{X}$

(b) 1-step controllable set $K_1(\mathcal{X})$ of the hyperrectangle shaped inner approximation $\mathcal{X}$



(c) Inner approximation of 1-step controllable set $K_1(\mathcal{X})$

Figure 5.2: Hyperrectangle shaped inner approximation of the 1-step controllable set for the discrete integrator

We can formulate the state constraints shown in equation (5.4) as a target set $\mathcal{X}$. Considering a straight road we can assume the constraints on lateral velocity, yaw rate and heading to be symmetric.

$$\mathcal{X} \triangleq \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} v(k) \\ r(k) \\ \psi(k) \\ e(k) \end{bmatrix} \leq \begin{bmatrix} v_{\max} \\ v_{\max} \\ r_{\max} \\ r_{\max} \\ \psi_{\max} \\ \psi_{\max} \\ e^{\mathcal{X}}_{\max} \\ -e^{\mathcal{X}}_{\min} \end{bmatrix} \tag{5.4}$$

Using algorithm 2 we attempt to calculate $\mathcal{K}_1(\mathcal{X}, \overline{u})$ for the system (5.3) analytically. By performing the the matrix calculations presented in chapter 2.4.2 we can derive $\mathrm{Pre}(\mathcal{X})$. The result of this first step of algorithm 2 is shown in appendix A.2. In the second part of the algortithm to derive the 1-step controllable set, the projection (see section 2.2.3) of $\mathrm{Pre}(\mathcal{X})$ onto the $\mathbb{R}^4$-state space has to be calculated.
Using the Fourier-Motzkin algorithm provided in appendix A.1, the set can be projected analytically. The one-step controllable set is derived by intersecting the result of the projection with the set of state constraints at the location of the one-step controllable set. The result is shown in appendix A.3. Assigning an inner hyperrectangle into this set of constraints is generally possible without extended numerical calculations.
The results show that in principle it is possible to efficiently find an inner approximation of the set $\mathcal{K}_1(\mathcal{X})$ in the shape of a hyperrectangle set that was derived from hyperrectangle type constraints $\mathcal{X}$.
To examine the applicability of these results we can take a closer look at constraints three and four from the one-step controllable set shown in A.3

$$\begin{bmatrix} t_s & 0 & t_s u & 1 \\ t_s & 0 & t_s u & -1 \end{bmatrix} \begin{bmatrix} v(k-1) \\ r(k-1) \\ \psi(k-1) \\ e(k-1) \end{bmatrix} \leq \begin{bmatrix} e^{\mathcal{X}}_{\max} \\ -e^{\mathcal{X}}_{\min} \end{bmatrix}. \tag{5.5}$$

Rewriting this equation in terms of the lateral position of the one-step controllable set $e(t-1)$ these hyperplanes can be stated as

$$\begin{aligned} e(k-1) &\leq e^{\mathcal{X}}_{\max} - t_s\, v(k-1) - t_s\, \overline{u}\, \psi(k-1) \\ e(k-1) &\geq e^{\mathcal{X}}_{\min} + t_s\, v(k-1) + t_s\, \overline{u}\, \psi(k-1). \end{aligned} \tag{5.6}$$

These two equations simply represent the linear mapping of the boundaries on the lateral position e for one discrete time step of the motion model (5.3). Since the model does not have any direct input term on the lateral position, this mapping is only dependent on the lateral velocity and the lateral motion due to the heading direction.
Assuming $t_s > 0$ and $\overline{u} > 0$, which is true for regular driving situations, we can again assign a new hyperrectangle approximation into these constraints. With the

target set 5.4 being symmetric in all dimensions we can assume the same for the approximation of the one-step controllable set.
The boundaries on the approximation of the one-step controllable set are defined by

$$\begin{aligned} -v_{max}(k-1) \leq v(k-1) \leq v_{max}(k-1) \\ -r_{max}(k-1) \leq r(k-1) \leq r_{max}(k-1) \\ -\psi_{max}(k-1) \leq \psi(k-1) \leq \psi_{max}(k-1) \\ e_{min}(k-1) \leq e(k-1) \leq e_{max}(k-1). \end{aligned} \tag{5.7}$$

With $v_{max}(k-1) \geq 0, r_{max}(k-1) \geq 0, \psi_{max}(k-1) \geq 0, e_{min}(k-1) \leq 0$ and $e_{max}(k-1) \geq 0$ this set has the same geometrical structure as the target set (5.4). To fit this hyperrectangle into the polytope $\mathcal{K}_1(\mathcal{X})$, all inequalities have to be fulfilled for all values inside the hyperrectangle. Since $\mathcal{K}_1(\mathcal{X})$ is a convex set, it is sufficient to test if all the extremal values of the hyperrectangle fulfil the constraints. The combinations of these extremal values in all dimensions correspond with the vertices of the hyperrectangle. Inserting the extremal values for $e(k-1)$, $v(k-1)$ and $\psi(k-1)$ into equations 5.6 yield the following inequalities that must be true for the hyperrectangle to be a valid inner approximation.

$$e(k-1) \leq e^{\mathcal{X}}_{max} - t_s v_{max}(k-1) - t_s\, u\, \psi_{max}(k-1) \tag{5.8a}$$
$$e(k-1) \leq e^{\mathcal{X}}_{max} + t_s v_{max}(k-1) - t_s\, u\, \psi_{max}(k-1) \tag{5.8b}$$
$$e(k-1) \leq e^{\mathcal{X}}_{max} - t_s v_{max}(k-1) + t_s\, u\, \psi_{max}(k-1) \tag{5.8c}$$
$$e(k-1) \leq e^{\mathcal{X}}_{max} + t_s v_{max}(k-1) + t_s\, u\, \psi_{max}(k-1) \tag{5.8d}$$
$$e(k-1) \geq e^{\mathcal{X}}_{min} - t_s v_{max}(k-1) - t_s\, u\, \psi_{max}(k-1) \tag{5.8e}$$
$$e(k-1) \geq e^{\mathcal{X}}_{min} + t_s v_{max}(k-1) - t_s\, u\, \psi_{max}(k-1) \tag{5.8f}$$
$$e(k-1) \geq e^{\mathcal{X}}_{min} - t_s v_{max}(k-1) + t_s\, u\, \psi_{max}(k-1) \tag{5.8g}$$
$$e(k-1) \geq e^{\mathcal{X}}_{min} + t_s v_{max}(k-1) + t_s\, u\, \psi_{max}(k-1) \tag{5.8h}$$

With $e^{\mathcal{X}}_{\min} = -e^{\mathcal{X}}_{\max}$ and all other variables on the right hand side being positive it is clear that the inequalities (5.8a) and (5.8h) represent the tightest bounds on $e(k-1)$. Therefore we can neglect the other inequalities as they will hold true if these two are fulfilled. The relevant bounds on the lateral position of the one-step controllable set are described by

$$e^{\mathcal{X}}_{min} + t_s v_{max}(k-1) + t_s\, u\, \psi_{max}(k-1) \leq e(k-1) \leq e^{\mathcal{X}}_{max} - t_s v_{max}(k-1) - t_s\, u\, \psi_{max}(k-1). \tag{5.9}$$

For $v_{max}(k-1) > 0$ and $\psi_{max}(k-1) > 0$ we can conclude that $e_{max}(k-1) < e^{\mathcal{X}}_{max}$ and $e_{min}(k-1) > e^{\mathcal{X}}_{min}$. The lateral bounds of the hyperrectangle approximation of the one-step controllable set are shrinking in each iteration step if we attempt to assign a hyperrectangle with a nonzero range in the derivative states v and ψ.
Considering an obstacle maneuver in which the safe target set is to the side of the obstacle while a vehicle is heading straight towards the obstacle renders this approach impractical. For the lane change maneuver the target set of the N-step

controllable set calculation is located in the an open lane this set needs to expand for each iteration to eventually enclose the vehicle state.
This analytical result can also be confirmed in the simple case of a discrete integrator as shown in figure 5.2. From the figure it is clear that an approximation of the one-step controllable set in the shape of a rectangle is always smaller that the target set $\mathcal{X}$ if the target set has the shape of a rectangle.
Therefore a simple approximation of the n-step controllable set using hyperrectangle constraints cannot be used to efficiently calculate iterations of n-step controllable sets through a lane change maneuver.

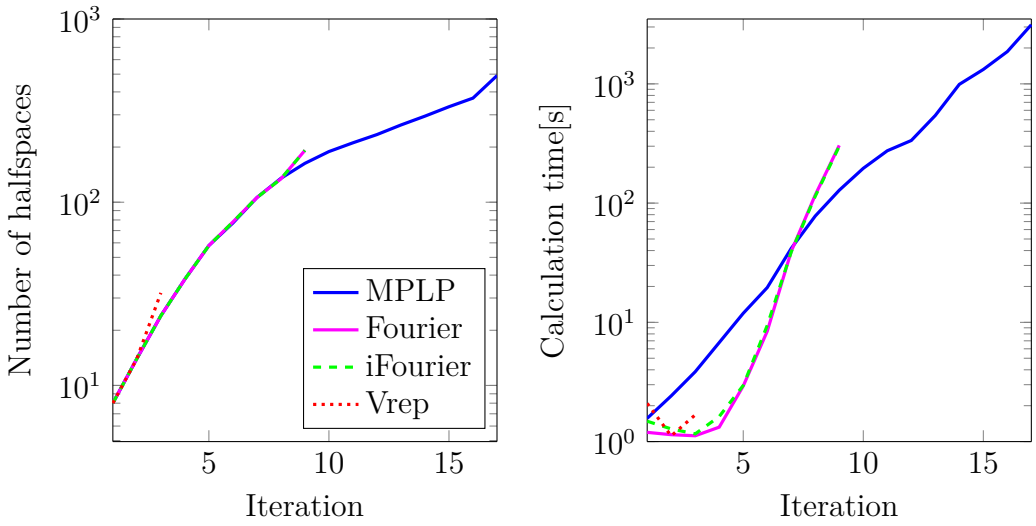
5.2 Numerical Approach

As an alternative approach to the analytical calculations presented in section 5.1 we use numerical calculation to determine an upper bound on the velocity that guarantees feasibility during a lane change maneuver. Using offline calculations we can pre-calculate trajectories of n-step controllable sets for a range of velocities as shown in figure 1.2.
If sufficient trajectories of n-step controllable sets for different velocities are stored, simple set membership tests can be used to test which velocity allows a safe avoidance of an obstacle if the vehicle is located at a certain distance from the obstacle.

5.2.1 Reachability Calculations with MPT-Toolbox

For numerical calculations the *Multi-Parametric Toolbox* (MPT) [22] in its current version 3.0.4 is used. The toolbox is developed by the Automatic Control Laboratory at the ETH Zürich in Switzerland and provides a Matlab interface to implement state of the art solver techniques for parametric optimization problems.
MPT-toobox version 3.0 provides an interface for algorithms to perform both elementary operations on polytopes as explained in section 2.2 as well a integrated algorithms for reachability analysis as described in section 2.4.
Examining the algorithms for the control invariant set 1 and for n-step controllable sets 2 in section 2.4, the main part of both operations is an iterative cycle of calculating the $Pre()$ of a set, projecting the resulting polytope in the state-input space back onto the original state space and calculating the intersection with the state constraints at that iteration step.
The $Pre()$ calculation is a simple matrix operation that can be quickly performed for large number of half spaces as shown in equation (2.17). The intersection with additional state constraints for a polytope in $\mathcal{H}$-representation is performed by appending the additional constraints to the set of halfspaces bounding the polytope.
To perform the projection of the $Pre()$ onto the original $[\beta, r, \psi, e]$ state space the MPT toolbox provides a number or different algorithms. This step is by far the most computationally expensive operation in this iterative calculation.
The most consistent results were obtained using the *MPLP*-algorithm. As an alternative the toolbox provides two algorithms based on Fourier elimination, with and without intermediate redundancy elimination, and one algorithm that computes the projection based on a vertex representation.

Attempting to calculate the invariant set results of the four dimensional vehicle model using the Fourier-elimination based algorithms lead to a Matlab crash after only ten iterations. The vertex based algorithm crashed with a Matlab error after only three iterations. While the exact cause of these errors could not be determined, it is evident that the implementations of these algorithms are not robust for four dimensional system with the complexity of the model 5.3. The *MPLP*-algorithm is the only algorithm provided in the toolbox that is capable of reliably performing higher dimensional projections with large numbers of halfspaces.
Figure 5.3 illustrates the performance of the different projection algorithms provided in the MPT-toolbox and how the complexity and the n-step controllable set increases if no means for simplification are taken.



(a) Complexity of the N-step controllable set

(b) Calculation time per iteration

Figure 5.3: Calculation of the N-step controllable set without approximation

As shown by the example of the discrete integrator in section 2.5, calculating n-step controllable sets of systems containing integrators yield in skewed shaped polytopes. Therefore the accuracy of this calculation is directly correlated with the number of halfspaces used in the approximation. When calculating such sets the number of halfspaces increases with each iteration step. While this does increases the accuracy of the representation it also makes the calculation of the next step more computationally expensive.
Figure 5.3 shows that this increase in calculation time is exponential almost exponential in the number of halfspaces bounding resulting polytopes. Starting from eight initial constraints the calculation results in more than 500 hyperplanes in less than 20 iterations. The calculation time increases from 1.5 seconds for the first iteration to five minutes. These calculations were performed with Matlab 2011a and the MPT toolbox 3.2 on a quad core Intel i7 CPU with 2.5 Ghz. From the results it is clear that this trend makes the calculation of long trajectories with regular computing hardware impossible without suitable approximations algorithms that bound the calculation time.

5.2.2 Inner Approximation of Polytopes

To allow the calculation of the n-step controllable set for longer distances as well as to achieve a result from control invariant set calculations an approximation algorithm is necessary. Since any subset of an n-step controllable set is also an n-step controllable set for the system with the desired target set and every subset of an invariant set is still an invariant set for the system for both calculations the approximation algorithm must ensure that an approximated polytope $\mathcal{P}_{approx} \subseteq \mathcal{P}$.

Goubault, Mullier et al. [19] state in their paper on inner approximated reachability analysis: "Computing a tight inner approximation of the range of a function over some set is notoriously difficult, way beyond obtaining outer approximations". While there is a large number of algorithms for outer approximation of polytopes in literature, there is rather limited material on inner approximation.

In this thesis a heuristic approach for an inner approximation of a convex polytope is developed that allows an efficient reduction in the number of half spaces in a polytope. The algorithm utilizes the assumption that the polytope considered is symmetric with respect to the origin. This allows the algorithm to correct for possible numerical biases in the other steps of the iterative calculations studied in this work. Due to numerical rounding effects during the projection is can happen that the symmetry of the set is lost.

The system constraints 5.1 provide symmetric boundaries on the system states β, r and ψ. For a vehicle driving in a single lane the bounda on the lateral position e are symmetric as well. Therefore all sets calculated using algorithm 1 will be symmetric as well.

For the sets $\mathcal{K}_{1,\dots}(\mathcal{X})$ calculated using algorithm 2 we can ensure the same property if we simply assume that the preceding traffic is located in both outer lanes on a 3 lane road. With this assumption we can define symmetric bounds on the lateral position $e_{min} \leq e \leq e_{max}$ relative to the symmetric control invariant set $\mathcal{C}$ as shown in figure 5.4. The lateral limits are defined for the center of gravity of the vehicle, therefore they are offset from the road boundaries by half the vehicle width and a safety margin.

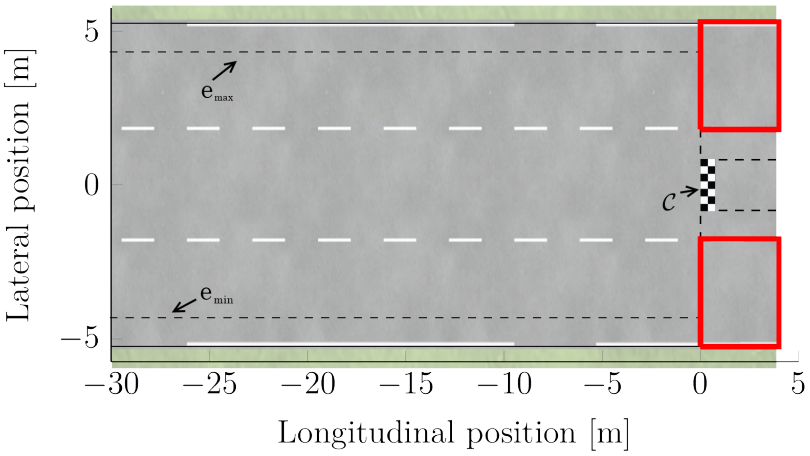
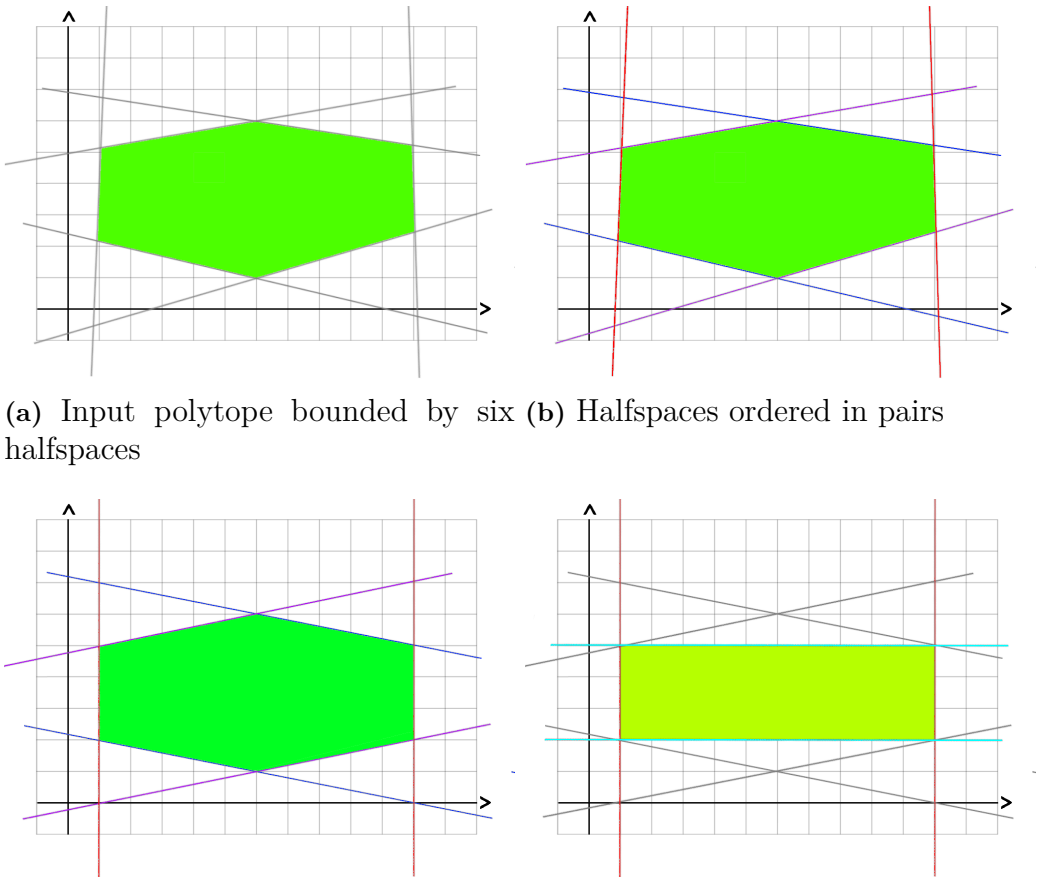


Figure 5.4: Symmetric road geometry for lane change predictions

The algorithm implemented is illustrated in figure 5.5 in two dimensions. The halfspaces of the polytope shown in 5.5a are not exactly symmetrical which is a common case since they are results of numerical calculations. By comparing the normal vectors of the half spaces, opposing facets can be identified and sorted in pairs, as illustrated in 5.5b.
By using averages of the normal vectors, the symmetry of the polytopes can be enforced as shown in the third step. The distance to the origin of the new hyperplanes is modified such that all vertices on the original facets are on or outside the modified facets. Illustration 5.5d shows the final inner approximation of the polytope. By combining facets that have small angle differences, the number of hyperplanes can be reduced. Averaging the normal vectors and assigning a new distance to the origin such that all previous vertices are on or outside the newly created hyperplane leads to a inner approximation. In the figure the blue and purple halfspaces are combined to reduce complexity. This procedure can be performed iteratively until the number of halfspaces is reduced to a bearable level. Using this approach the symmetry of the polytope can be enforced and the number of hyperplanes is reduced. This approach can be generalized for higher dimensions. The implementation of the algorithm is shown in appendix B.



(a) Input polytope bounded by six halfspaces

(b) Halfspaces ordered in pairs

(c) Numerical correction of pairs of halfspaces

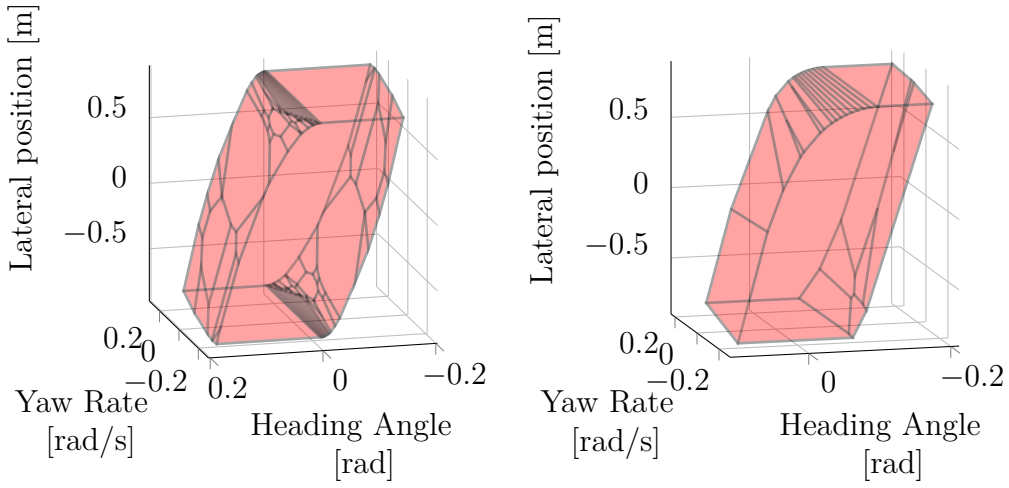
(d) Inner approximation with four halfspaces

Figure 5.5: Inner approximation algorithm for symmetric polytopes

While this algorithm has no claim for optimality, it does provide a reliable method for inner approximation such the sets calculated in algorithm 1 for the control invariant set, as well as algorithm 2 for the N-step controllable sets, can be calculated with bounded numbers of hyperplanes.

5.2.3 Numerical Control Invariant Sets

Using the inner approximation algorithm presented, calculations of invariant sets and n-step controllable sets can be performed. By iterating algorithm 1 (section 2.4.3) for the discretized linear force input model (3.17) and envelope constraints (3.20) , (3.21) a safe, control invariant set $\mathcal{C}(\overline{u})$ within one lane can be determined for a specific velocity $\overline{u}$.
Since the calculations are performed offline, it is possible to utilize these algorithm although they are very computationally expensive.
Figure 5.6 shows slices through the invariant set of the vehicle within one lane for the vehicle driving at 70 km/h. The polytope shown is an inner approximation with 160 halfspaces and 1018 vertices. The set is calculated in 100 iteration steps of algorithm 1 with an approximation combining halfspaces within two degrees in each step.
As the set is defined in four dimensions, it is not possible to visualize the shape of the set in a single comprehensive figure. Therefore slices and projections are used to gain an understanding of the nature of the shape of these representations.



(a) Slice at $\beta = 0$ rad **(b)** Slice at $\beta = 0.10$ rad

Figure 5.6: Slices of the control invariant set for 70 km/h

The shape of the set reveals some properties that are intuitively understandable. In the lateral position on the road it is bounded by the physical bounds described by the width of the road and the width of the vehicle. Along the lateral boundaries is is clear that a positive heading angle at the negative lateral road boundary is no problem because it brings the vehicle back towards the center of the track in the next step. In the same position negative heading is not allowed in the slice at zero

sideslip because the vehicle would violate the road boundary in the next step. For a sideslip of 0.1 rad some amount of negative heading is possible since the lateral the lateral motion due to the heading angle can be compensated by the sideslip.

5.2.4 Numerical N-step Controllable Sets

To calculate the N-step controllable sets $\mathcal{K}_N$ for the vehicle in lane change maneuver algorithm 2 (section 2.4.4) is used. Starting from the control invariant set $\mathcal{C}(70\text{km/h})$ shown in figure 5.6, the algorithm uses the safe handling envelope constraints (3.20) , (3.21) together with boundaries on the lateral position (3.2) that restrict the allowed region to the adjacent lanes to calculate n-step controllable sets. For each iteration step N, the N-step controllable set predicts the set of states for which it can be guaranteed that a feasible trajectory to the target set exists, located a distance $\Delta_s = N\ t_s\ u$ ahead of the obstacle.
Figure 5.7 shows a bird's eye view on the predictions for the lane change maneuver with 70 km/h. Projections in green limit the lateral positions on the road where there exist any states for which feasibility of the future trajectory can be guaranteed. The markers in red indicate slices of the sets at $\beta = 0$, $r = 0$ and $\psi = 0$. This corresponds to the locations from where the vehicle can start a lane change maneuver to pass the obstacle if is currently driving straight. The road is delimited by solid black lines and the dashed black lines delimit the region in which the center of the car is allowed to move.

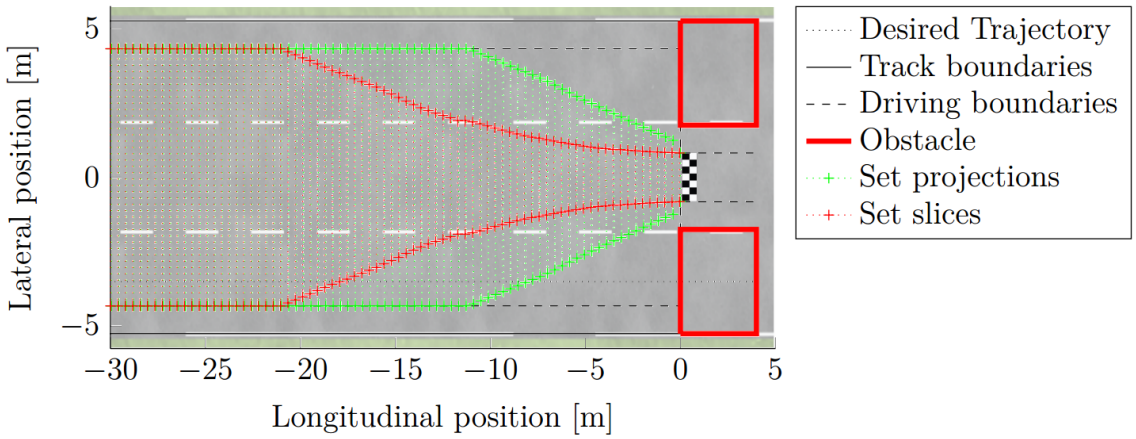
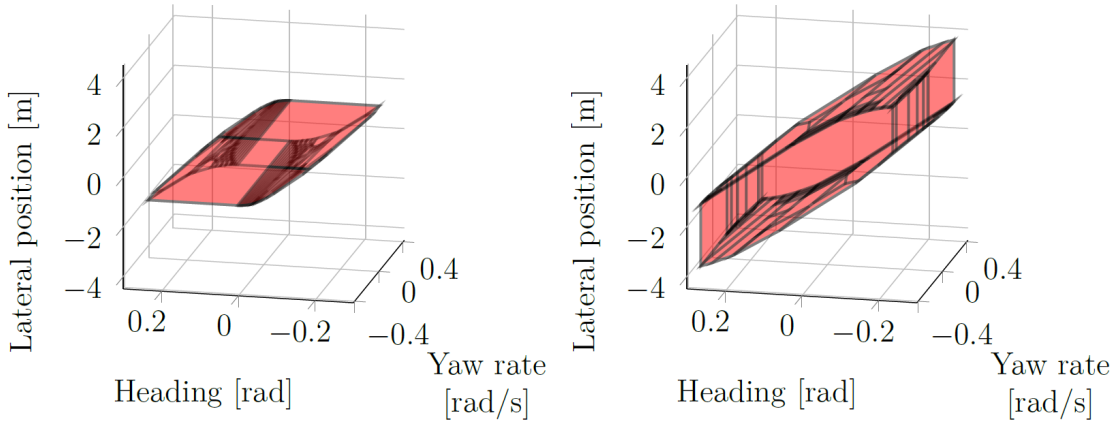


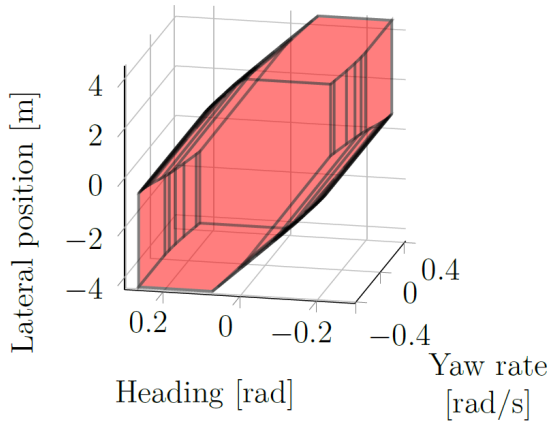
Figure 5.7: Projections onto the lateral position and slices at $\beta = 0$, $r = 0$ and $\psi = 0$ of n-step controllable sets for a lane change maneuver with 70 km/h

The evolution of the N-step controllable set is shown in figure 5.8. Slices at $\beta = 0$ are provided for different iteration steps. The different slices illustrate the expansion of the set over the backwards trajectory. Since significant lateral motion and therefore a significant expansion of the set in the lateral direction is mainly caused by driving at significant heading angles, we can see that the set initially gets more skewed. Large negative lateral positions become possible rather quickly for significant positive heading angles.

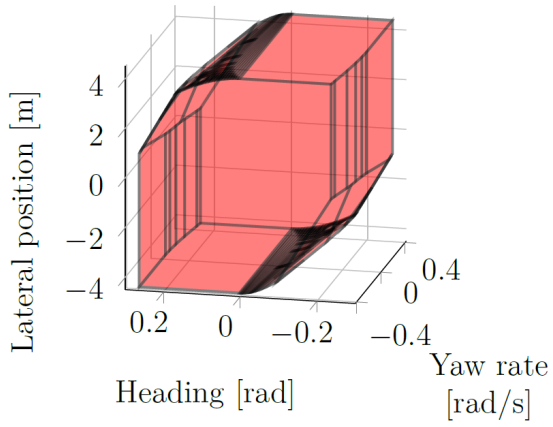


(a) $N = 0$

(b) $N = 25$



(c) $N = 50$



(d) $N = 75$

Figure 5.8: Slices at $\beta = 0$ of different N-step controllable set for the lane change maneuver at 70 km/h

When the set spans the entire lateral width of the road as depicted in figure 5.8c, it continues to expand further to also include states with smaller heading angles for negative lateral position and larger heading angles for positive lateral positions. Eventually the set expands to a size such that it contains the target set $\mathcal{C}$ laterally located at the center of each of the lanes

6
Controller Design

To gather experimental data to compare a real vehicle's capabilities with the predictions derived in the previous chapter, an MPC-controller based on similar assumptions (5.1) is implemented.
The general concept of the controller implementation follows the work from Erlien, Funke and Gerdes [9]. In their paper they formulate a shared vehicle control for a steer-by-wire vehicle that tracks a driver's steering intent while allowing interventions if the driver input would lead to unsafe situations. In [17] Funke applies the the concept to fully autonomous vehicles for path tracking with obstacle avoidance. In this work we assume the reference trajectory to be a straight line within a highway lane that the controller attempts to track. If an obstacle is detected in in that lane, the controller has to intervene and calculate an optimal steering trajectory to avoid the obstacle and return to the original lane.

6.1 Controller Formulation

The controller aims to determine an optimal steering input force F_{yf} for the vehicle model (3.19) such that the vehicle can safely navigate through the environment while trying to stay as close to the desired lateral position as possible and maintaining a smooth motion of the vehicle by minimizing the objective function

$$\min_{F_{yf}} \sum_{k=n_{near}+1}^{n_{near}+n_{far}} \sigma_{env} \left(S_{env}^{(k)}\right)^2 + \sum_{k=1}^{n_{near}+n_{far}} \sigma_{sh} |S_{sh}^{(k)}| \tag{6.1a}$$

$$+ \sum_{k=n_{near}+1}^{n_{near}+n_{far}} \sigma_e |e^{(k)}| + \sigma_\psi \left(\psi^{(k)}\right)^2 \tag{6.1b}$$

$$+ \sum_{k=0}^{n_{near}-1} \sigma_{u_{near}} \left(F_{yf}^{(k)}\right)^2 + \sum_{k=n_{near}}^{n_{near}+n_{far}-1} \sigma_{u_{far}} \left(F_{yf}^{(k)}\right)^2 \tag{6.1c}$$

$$+ \sum_{k=0}^{n_{near}-1} \sigma_{du_{near}} \left(F_{yf}^{(k+1)} - F_{yf}^{(k)}\right)^2 + \sum_{k=n_{near}}^{n_{near}+n_{far}-2} \sigma_{du_{far}} (F_{yf}^{(k+1)} - F_{yf}^{(k)})^2. \tag{6.1d}$$

These possibly conflicting goals are managed by assigning different weighted cost terms σ_x to the objectives formulated in equation (6.1).
Forcing the vehicle to stay inside the road boundaries and avoiding obstacles (3.2) is enforced by soft constraints with the slack variable S_{env}. Second highest priority is the avoidance of possibly dangerous driving situations by violating the vehicle's

handling capabilities. To ensure the vehicle stability the safe handling envelope (3.20), (3.21) is enforced with soft constraints using the slack variable S_{sh}. These two cost terms (6.1a) with $\sigma_{env} \gg \sigma_{sh}$ are enforcing safety for the maneuver. Both terms are significantly larger than all other cost terms in (6.1).
The lane tracking objective σ_e (6.1b) penalizes the lateral deviation from the desired path. To avoid severe steering maneuvers when the vehicle is required to travel in a different lane due to an obstacle, this objective is enforced by a linear term.
To influence the trajectory taken while the vehicle is changing the lane to avoid an obstacle, additional terms are introduced. A small cost σ_ψ on the vehicle heading ψ helps avoid oscillations and provides damping to the system. Additional cost terms on the input force (6.1c) as well as the rate of the input (6.1d) allow reducing the severity of the steering action when the vehicle has to navigate around an obstacle and return to its original path.

6.1.1 Motion Prediction

The MPC controller predicts the vehicle motion over the discrete time horizon using two different motion models (3.17) and (3.19)

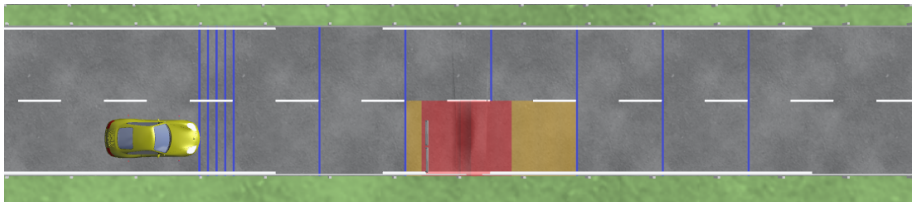
$$\begin{aligned} x^{(k+1)} =& A_d^{(k)}(x^{(0)}, t_{near}, V_x)x^{(k)} + B_d^{(k)}(x^{(0)}, t_{near}, V_x)F_{yf}^{(k)} \\ &+ d_d^{(k)}(x^{(0)}, t_{near}, V_x) \qquad k \in [0...n_{near}-1] \end{aligned} \tag{6.2a}$$

$$x^{(k+1)} = A_d^{(k)}(0, t_{corr}, V_x)x^{(k)} + B_d^{(k)}(0, t_{corr}, V_x)F_{yf}^{(k)} \qquad k = n_{near} \tag{6.2b}$$

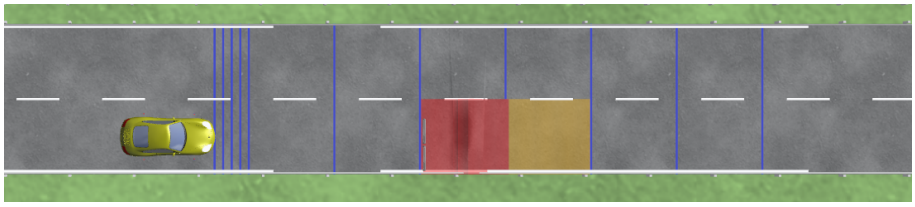
$$x^{(k+1)} = A_d^{(k)}(0, t_{far}, V_x)x^{(k)} + B_d^{(k)}(0, t_{far}, V_x)F_{yf}^{(k)} \qquad k \in [n_{near}+1...n_{far}-1]. \tag{6.2c}$$

The prediction horizon is split into a near horizon (6.2a) and a far horizon (6.2c) and a correction step (6.2b). A small discretization time is chosen for the near horizon and a significantly longer step size is chosen for the far horizon. For the near horizon $k = [0, ..., n_{near}]$ the controller uses the linear time varying vehicle model (3.19) which is a bicycle model linearized at the current velocity $\overline{u} = u_0$ and current rear tire slip angle $\overline{\alpha}_{r_0}$. This allows to account for the rear tire saturation effects. For the prediction horizon $k = [n_{near}+1, ..., n_{far}]$ controller uses the linear force input model (3.17) which is also linearized at the current vehicle velocity but assumes a rear tire linearization at $\overline{\alpha}_r = 0$. This is necessary because for the prediction horizon further ahead, an estimation of the future rear tire slip angle is becomes inaccurate since the future motion of the vehicle is yet unknown.
In their paper Erlien, Funke and Gerdes [9] use a 10-step horizon with a sampling time of 10 ms to capture the vehicle dynamics in the near future and a 30-step far horizon with a sampling time of 200 ms to cover a longer distance farther ahead of the vehicle for path planning and obstacle avoidance. The combination of the two horizons adds up to a total prediction time horizon of 6.1 s.

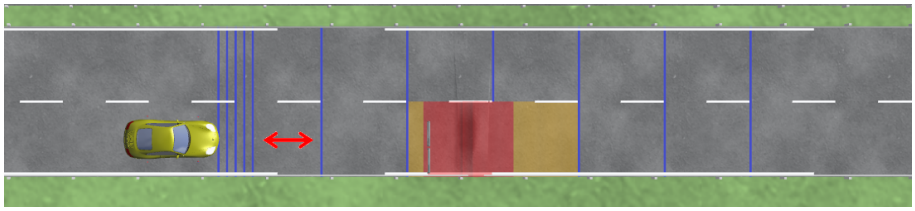
The motion model also implements a correction time step (6.2b) between the near and the far horizon with variable length as proposed in [10]. Since lateral boundaries can only be assigned at the discretization points, any obstacle detected in the prediction horizon has to be extended in the direction of travel to span between discretization points.
By shortening the first step time of the far horizon dynamically, the spatial location of all other time steps in the far horizon can remain spatially fixed while the car is cruising towards the obstacle. This allows the optimization to account for the fact that the ego vehicle is approaching the obstacle in each sampling interval as illustrated in figure 6.1. Without the correction step, the optimization would not account for the fact that it is getting closer until the obstacle would be captured by the next far horizon discretization point. Since this involves multiple sampling intervals of the controller it would lead to non-smooth control actions.



(a) Vehicle approaching an obstacle in the far horizon



(b) Location of discretization points a short time later without correction step



(c) Location of discretization points a short time later with correction step

Figure 6.1: Visualization of the influence of the correction step on obstacle represenation in the horizon

6.1.2 System Constraints

The system is subject to a number of constraints to ensure vehicle safety. The environmental envelope which models lane boundaries as well as rigid obstacles (3.2) is most critical to passenger safety. These boundaries are implemented as a

soft constraint (6.3a) and enforced over the far horizon as illustrated in figure 6.2.

$$H_{env}x^{(k)} \leq G_{env} + S_{env}^{(k)} \qquad k \in [n_{near}+1...n_{near}+n_{far}-1] \tag{6.3a}$$

$$H_{sh}x^{(k)} \leq G_{sh} + S_{sh}^{(k)} \qquad k \in [1...n_{near}+n_{far}] \tag{6.3b}$$

$$F_{yf}^{(k)} \leq F_{yf,max} \qquad k \in [0...n_{near}+n_{far}-1] \tag{6.3c}$$

$$|F_{yf}^{(k)} - F_{yf}^{(k-1)}| \leq \Delta F_{yf,max_{near}} \qquad k \in [0...n_{near}-1] \tag{6.3d}$$

The safe handling envelope introduced in (3.20) and (3.21) is enforced by constraint . To make sure that the optimization results can actually be achieved by the vehicle, the input to the system has to be bounded according to physical limits. Equation (6.3c) assigns a limit on the maximum lateral front force and the constraint (6.3d) assigns limits on the maximum rate of change in the input. The total limit of the lateral force is limited by the lateral force that the front tires can physically transmit. The rate of change is limited by the steering rate of the robot as well as the tire properties. The constraint for the rate of change of the input force is only enforced over the near horizon since it would not have any effect for sampling rates in the order of 200 ms in the far horizon. The steering actuator is fast enough to achieve any commanded force input which renders the constraint in the far horizon unnecessary.

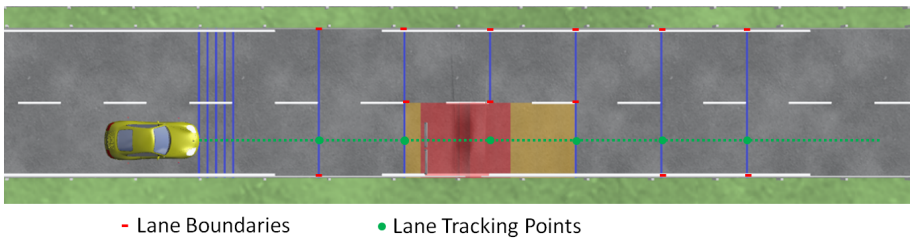


Figure 6.2: Representation of the environment boundaries as system constraints at discrete locations

6.2 Implementation

The controller specified in sections 6.1 to 6.1.2 is implemented using a custom solver generated by CVXgen as presented in chapter 4.3. The detailed formulation to generate the custom solver used in experimental testing is shown in appendix C.1. CVXgen provides an online interface to generate and download the C-code for the solver together with a Matlab-MEX interface for the embedded solver and an implementation of the specified problem statement using the general purpose CVX-solver that is not optimized for embedded execution. This enables the possibility to compare the custom solver with a more general solver to make sure it performs according to specification. To ensure that simulation results are consistent with the implementation in the vehicle, the custom solver C-code is also used for simulations in this work.

The C-code is integrated into a Simulink model using an S-Function Builder Block. This block offers a simple interface in which inputs and outputs to the S-function are specified and a wrapper function is used to feed the solver code with the numerical data of the optimization problem. The interface of the block is shown in figure 6.3.

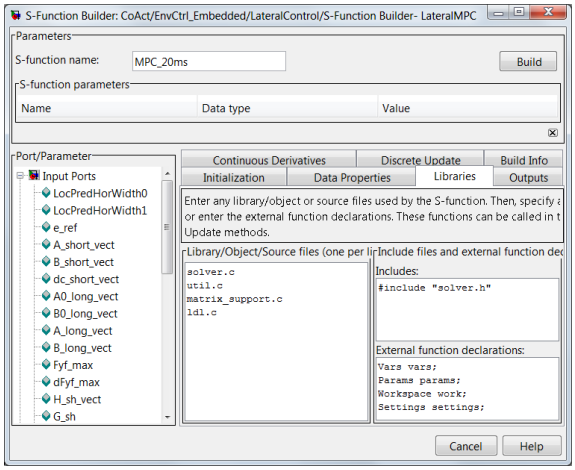


Figure 6.3: S-Function Builder block that implements the CVXgen code

6.2.1 Horizon length

As described in section 6.1.1 the controller uses a predicion horizon that is split up into a near horizon to account for vehicle dynamcis and a far horizon for path planning. In their work Erlien et al. [9] use a 10-step near horizon with a sampling time of 10 ms and a 30-step far horizon with a sampling time of 200 ms.
Using this configuration and exploiting the possibility of CVXgen to specify sparse matrices yields in a complexity of slightly over 4000 non-zero KKT-entries which is approximately the limit of the solver generator. The generated code with that horizon length performs well in simulation but implementation on the available real-time hardware do not allow the execution of the controller in real-time. On the embedded hardware used in the experimental setup described in chapter 7.2.1 the convergence time of the solver exceeds the desired sampling time of the controller.
To allow a real-time implementation the near horizon is modified to 5 steps with 20 ms time step and the far horizon is shortened to 20 steps as illustrated in table 6.1. Figure 6.4 shows simulation results for the double lane change comparing a controller with a 30-step far horizon and a 10ms sampling rate with the controller that uses the reduced horizon length at 20 ms sampling rate. The results indicate that the modifications do not change the controller's performance in a double lane change maneuver.
For low velocities a long look ahead distance is not necessary because the motion of the vehicle is not significantly limited by the dynamics. For velocities above 20 m/s a, four second look ahead more than sufficient to cover the entire trajectory of the double lane change before the vehicle has to start steering. Therefore the reduction in horizon length does not negatively affect the performance of the controller for this maneuver.

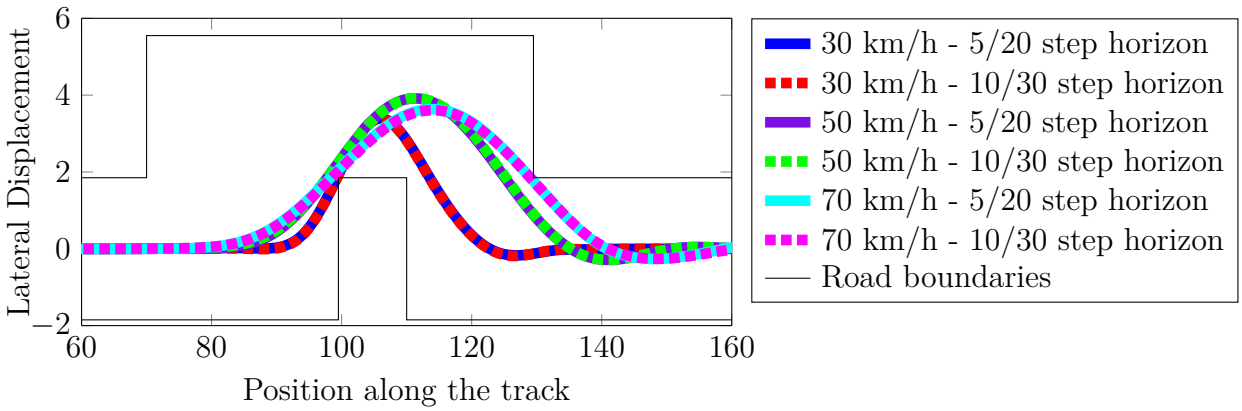


Figure 6.4: Simulation results of the double lane change with different prediction-horizon lengths

Near horizon length n_{near}	5 steps
Long horizon length n_{far}	20 steps
Near horizon time step t_{near}	20 ms
Far horizon time step t_{far}	200 ms
Controller sampling time t_s	20 ms

Table 6.1: Specification of the prediction horizon

6.2.2 Simulation environment

For closed loop simulation the controller was implemented in a Simulink model in connection with IPG CarMaker. Implementing the controller as a Simulink library allows the identical function to be used for simulation as well as for experimental testing in a real vehicle vehicle.
CarMaker is a simulation environment that provides a set of detailed vehicle models and a modular environment capable of simulating complex driving situations. It also provides a Simulink extension in which individual elements of the simulation can be replaced. It allows the implementation of custom vehicle parts such as tires or engines and lets the user modify control signals to the vehicle such as steering and throttle to evaluate vehicle motion control algorithms.

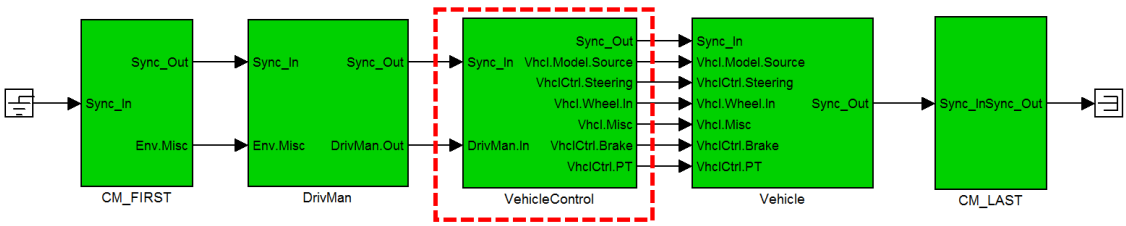


Figure 6.5: CarMaker Simulink Framework

The CarMaker Simulink extension provides access to the simulated vehicle motion data similar to what a high precision GPS system can provide in a real vehicle. For the purpose of simulating the steering controller the *VehicleControl* subsystem

shown in figure 6.5 is modified. By replacing the steering command from the driver simulation with a steering command calculated by the controller, the vehicle motion can be simulated in closed loop.

7
Experimental Testing

7.1 Test Scenario

The test scenario considered in this work is a double lane change maneuver for obstacle avoidance to evaluate controller performance and gather comparison data to evaluate the vehicles actual capabilities.
The longitudinal dimensions of the test track are derived from the ISO 3888-1 standardized double lane change maneuver [25]. Lateral dimensions of the track are modified to yield more vehicle independent results. While the ISO standard defines the track width based on the vehicle width and an additive margin, this thesis uses a fixed lane width of 3.5 m to represent a standard highway lane. An illustration of the track is shown in figure 7.1. Dimensions of the individual sections are provided in table 7.1.

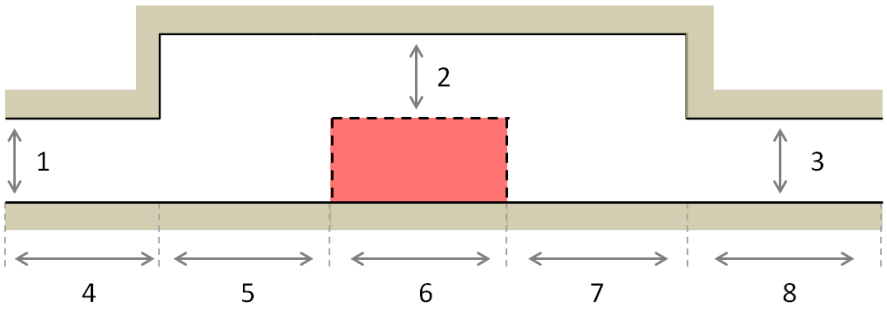


Figure 7.1: Double lane change track

Lane width 1, 2, 3	3.5 m
Section length 4	15 m
Section length 5	30 m
Section length 6	25 m
Section length 7	25 m
Section length 8	15 m

Table 7.1: Double lane-change track dimensions

Controller tests are performed at constant velocity in the range from 50 to 100 km/h. During the tests the obstacle, visualized in red, becomes recognized by the controller only after the vehicle comes closer than a specified distance to the obstacle. This

distance is adjusted according to the longitudinal velocity of the test vehicle and used to adjust the difficulty of the maneuver.
By varying the reference path in the starting lane, the difficulty of the double lane change maneuver is varied. Tests are performed with the reference trajectory along the center of the starting lane as well as with the reference path located at ±0.5 m lateral offset from the center of the starting lane.

7.2 Test Vehicle

Experimental tests were performed using a 2012 Volvo S60 sedan with an automatic, six-speed gearbox. The vehicle provides an open CAN-bus interface provided by the Volvo Cars Corporation that allows access to data from the vehicle's internal sensors as well provides the possibility to send braking and acceleration commands to the cruise control.

Figure 7.2: SR 60 Orbit steering robot mounted in the test vehicle

7.2.1 Testing Equipment

The controller is implemented on a dSpace MicroAutobox real-time embedded computer. This rapid control prototyping environment provides the possibility to generate real time capable code from directly from a Matlab/Simulink. The MicroAutobox is equipped with a Power PC 750 GL microprocessor clocked at 900 Mhz. The system provides 16 MB RAM and 16 MB of local flash memory.
To localize the vehicle on the track, an Oxford Technologies RT3002 precision Inertial and GPS Navigation system is used. Using differential corrections from a fixed base station, localization with an accuracy up to 0.01 m is possible.
To control the lateral motion of the vehicle, an Anthony Best Dynamics SR 60 Orbit steering robot is installed. As shown in figure 7.2, the steering robot is mounted behind the steering wheel to allow quick installation without removing the steering wheel and the car can still be driven manually with the robot installed.

The additional components are connected in an private CAN-bus network in the vehicle as shown in the network topology in figure 7.3. The RT3002 GPS System broadcasts the vehicle motion and position measurements in cyclic messages every 10 ms on the bus. Motion data is read by the MicroAutobox and every 20ms the next control action is calculated and sent to the SR60 steering robot. A laptop connected to the system is used to operate the components and to collect measurements.
The network topology allows for three different data collection sources. Measurements from the steering robot can be collected via a USB connection. Message data from the CAN-Bus can be collected using a Vector CAN-Case with CANalyzer measurement software and internal data from the controller can be collected via the Host-PC interface to the MicroAutobox.

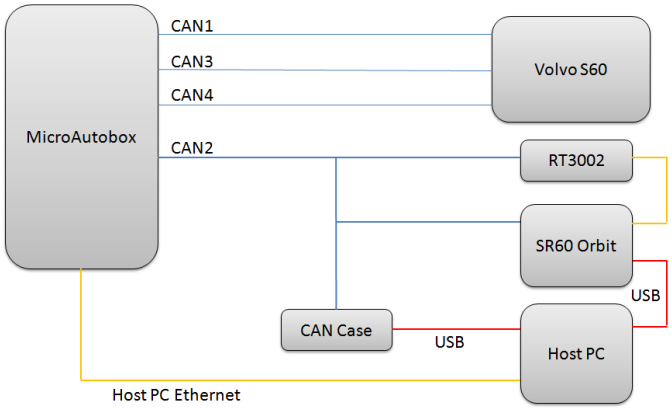


Figure 7.3: Network topology of the experimental setup

7.2.2 Vehicle Parameters

To make practical use of the vehicle models described in chapter 3 in a simulation environment or for control design the physical vehicle parameters have to be identified. While parameters like vehicle mass, wheelbase and weight distribution are readily available from vehicle documentation, tire parameters have to be estimated from measurement data to match their actual performance on the test surface. To fully parametrize the vehicle model (3.19) the effective cornering stiffnesses and the friction coefficients for both axles of the vehicle are determined.
These parameters are estimated from measurements collected from a steady state cornering test. Starting from a very low velocity the vehicle drives around a circle with a fixed radius while slowly increasing the velocity. By assuming steady state cornering condition with $\dot{r} = 0$ and $\dot{\beta} = 0$ the equations of motions from the linear bicycle model (3.6) can be used to estimate front and rear lateral tire forces. By slowly increasing the velocity while measuring slip angles the lateral tire curves can be approximated.
Results from the test are shown in figure 7.4 together with curve fits of the brush tire model (3.10).
The set of vehicle parameters used for experimental testing is shown in table 7.2.

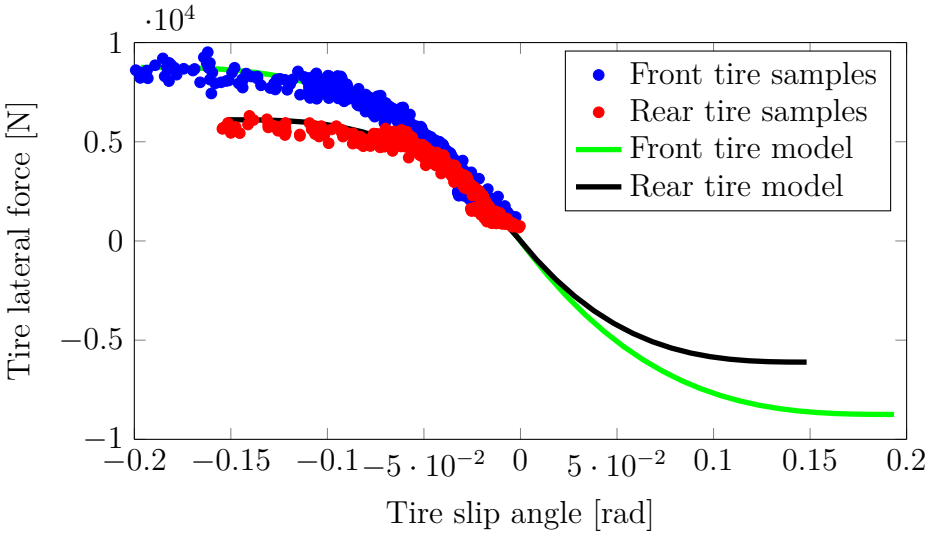


Figure 7.4: Brush tire approximation from measurement data

Mass (Equipment & Driver)	1823 kg
Wheelbase	2.77 m
Track width	1.865
Location CG to front/rear axle	1.104m/ 1.666m
Yaw moment of inertia	3500
Tire friction coefficient	0.88
Cornering stiffness front	110650 N/rad
Cornering stiffness rear	92393 N/rad

Table 7.2: Volvo S60 vehicle parameters

7.3 Results

Experimental tests were performed at the AstaZero proving grounds.

The controller described in section 6.1 was used to gather experimental data to determine the test vehicles capabilities in performing a double lane change maneuver and avoiding an static obstacle while maintaining vehicle stability as described in section 7.1. The controller parameters that were used during experimental testing are shown in appendix C.2.

Table 7.3 shows an overview of the results from the tests performed. It illustrates which maneuvers were performed without violating the safe handling envelope and which maneuvers were only possible with driving past these limits.

Test results for the velocities below 70 km/h lead to fast and violent steering maneuvers but they did not violate the save handling envelope during testing. The vehicle tends to understeer which makes it hard to avoid the obstacle but does not pose a threat to vehicle stability. For higher velocities the vehicle is much more likely to violate the stability boundaries.

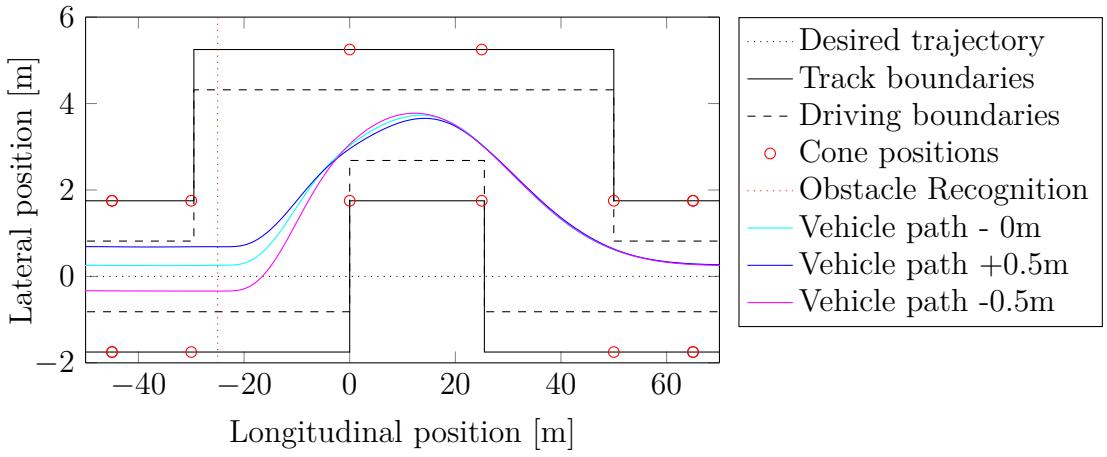
Figures 7.5 to 7.7 present experimental test results from the double lane change maneuver at 50, 70 and 100 km/h. The first plot for each velocity shows a bird's eye

view on the situation with the cones and outer lane boundaries physically delimiting the track. Dashed inner boundaries represent the area that the center of gravity of the car has to stay within. The dotted black line indicates the center line of the starting lane that is obstructed by the obstacle.
For each velocity three results are presented where the reference line is either on the center of the starting lane or parallel with a lateral offset of ±0.5 m.

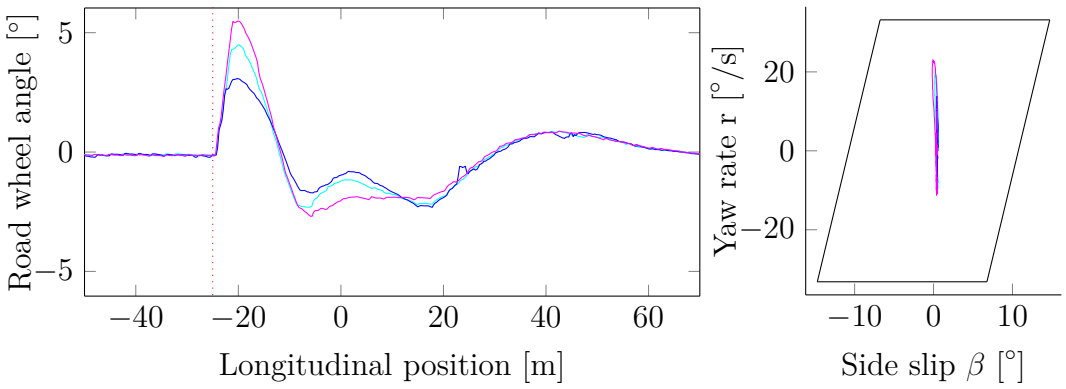
Velocity	Dist. of recognition	-0.5 m offset	0 m offset	0.5 m offset
50 km/h	25 m	no	no	no
50 km/h	30 m	no	no	no
60 km/h	30 m	no	no	no
70 km/h	30 m	no	close	yes
80 km/h	40 m	no	no	no
90 km/h	40 m	no	close	yes
100 km/h	45 m	close	yes	yes
100 km/h	50 m	no	close	yes

Table 7.3: Safe handling envelope violation during double lane change maneuvers

While results from experimental testing of a similar controller in [9] show a significant build-up of side slip when performing similar maneuvers on low friction ($\mu \approx 0.55$) surfaces when approaching the handling limits. This behavior was not observed during the testing for this work. On the high friction asphalt ($\mu \approx 0.88$) in this work the vehicle builds up significant yaw rate before building up side slip.
This observation is consistent with expectations that can be derived from the comparison between high- and low-friction tire curves presented in literature [28]. Tire curves measured on low friction surfaces tend to have lower cornering stiffness and a less pronounced peaking behavior as opposed to tires on high friction surfaces. Therefore the rear tire can build up more side slip while still remaining in the controllable area. With stiffer tires on a high friction surface the yaw rate may build up quicker without significant rear tire slip. Only when the rear tires passes the peak friction level, the side slip will increase significantly.
Overall it can be concluded that the controller is able to stabilize the vehicle and allows the operation of the vehicle close to its handling limits. As illustrated in figure 7.6c and **??**c the controller even manages to control the vehicle when the of the yaw rate limit of the safe handling envelope is slightly violated.
These results provide a good measure to evaluate the applicability of the feasibility predictions derived in the previous chapter.



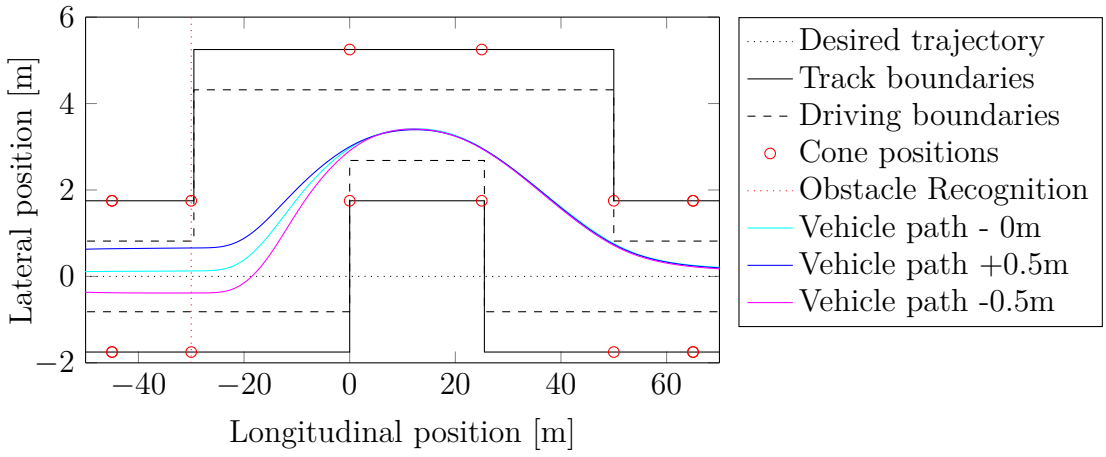
(a) Vehicle trajectory during double lane change



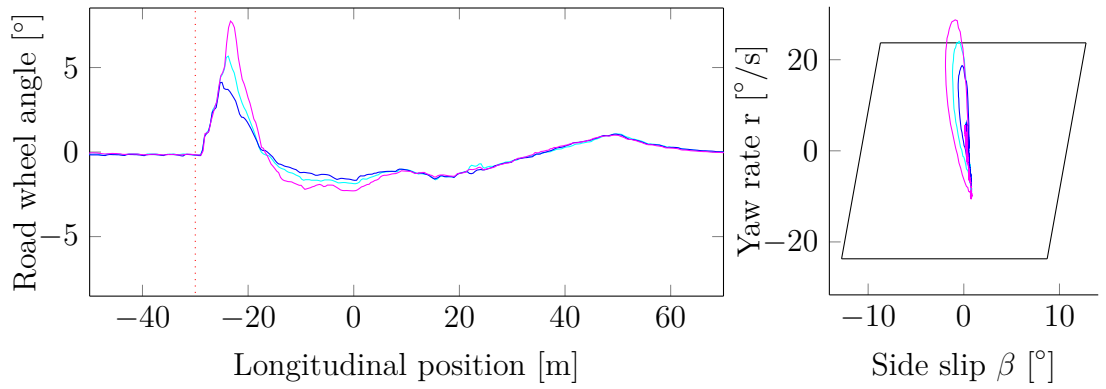
(b) Steering input

(c) Safe handling envelope

Figure 7.5: Experimental results from the double lane change maneuver with 50 km/h with 25 m pop-up obstacle



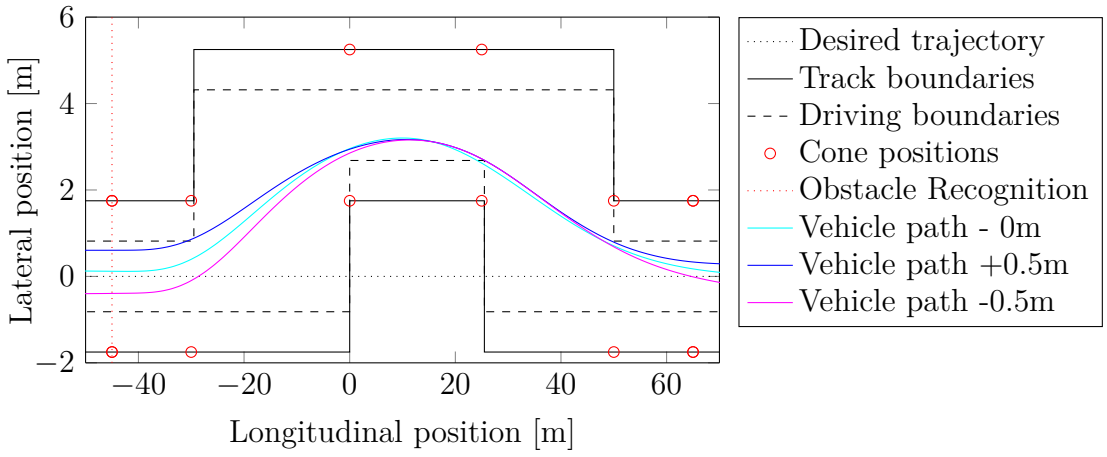
(a) Vehicle trajectory during double lane change



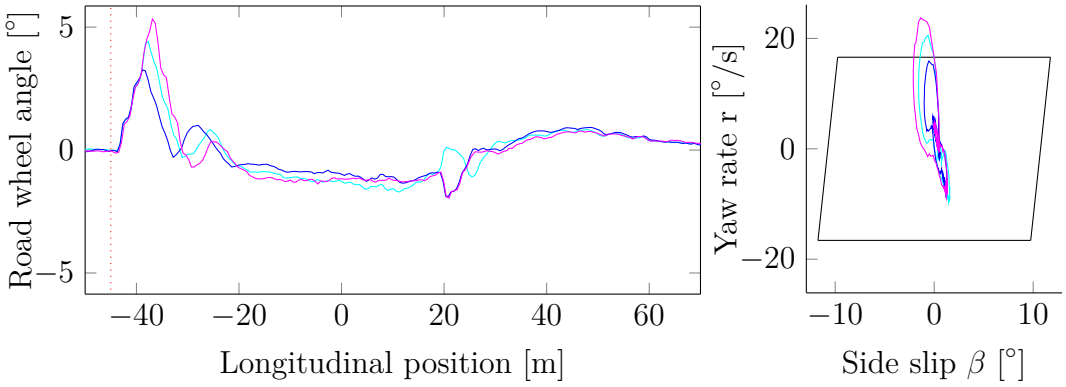
(b) Steering input

(c) Safe handling envelope

Figure 7.6: Experimental results from the double lane change maneuver with 70 km/h with 30 m pop-up obstacle



(a) Vehicle trajectory during double lane change



(b) Steering input

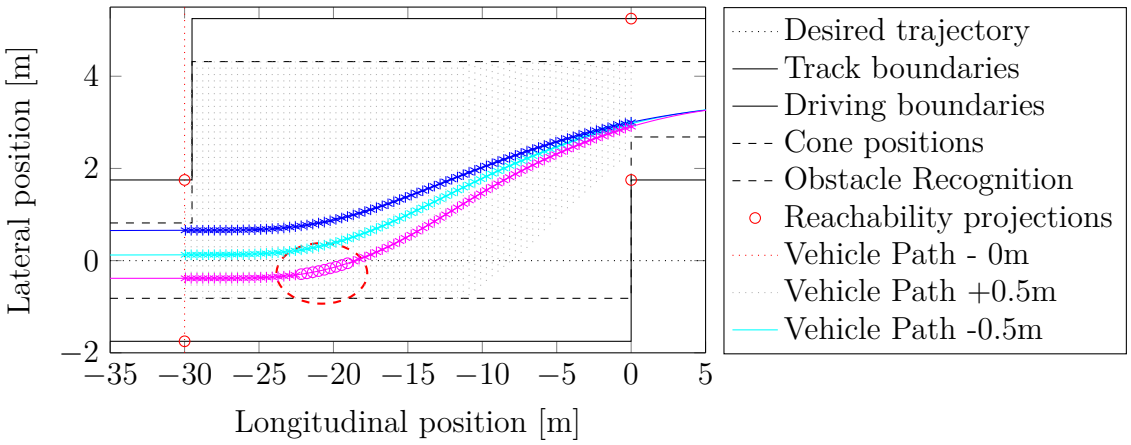
(c) Safe handling envelope

Figure 7.7: Experimental results from the double lane change maneuver with 100 km/h with 45 m pop-up obstacle

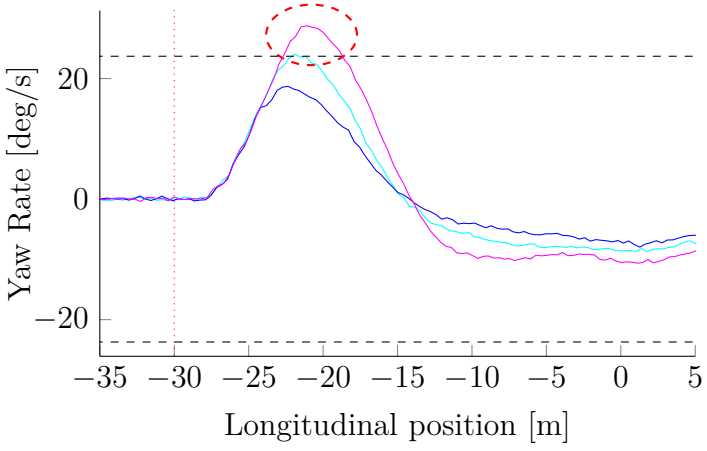
8

Discussion

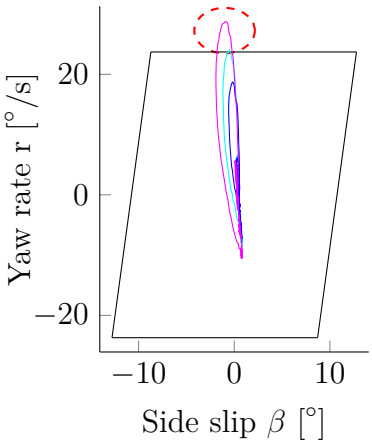
Using the data collected from experimental results shown in section 7.3 we can evaluate the applicability of the numerical offline calculations. Figure 8.1 shows the experimental results of vehicle performing the lane change maneuver at 70 km/h in comparison with the numerical predictions calculated in chapter 5.
Dotted gray lines indicate the spatial locations of projections of the offline precalculated N-step controllable sets on the lateral position. Stars on the trajectories indicate points that are contained in the predictions at a given location and circles along the trajectory indicate measurements which are not contained in the corresponding N-step controllable set.
At these instances the feasibility of the controller as well as the safety for the future trajectory in the lane change maneuver cannot be guaranteed.
The measurements which are not contained in their respective N-step controllable sets correspond to the time instances where the yaw rate in the experimental tests was already outside of the the safe handling envelope as indicated by the red circles in the figures 8.1a to 8.1c. Clearly from such point a feasible trajectory that satisfies the handling constraints is unlikely to exist as it would require the system to bring the vehicle states back into the safe handling envelope within a single time step.
For other points along the observed trajectories, even before the constraint violation in the magenta trajectory, the numerical offline calculations predict the existence of feasible solutions for the maneuver.
Therefore we can conclude that the predictions are rather overoptimistic with respect to what the controller can actually achieve. The N-step controllable set calculation from section 5.2.4 promises the existence of a feasible trajectory without constraint violation for all three initial locations but the controller in closed loop does not manage to keep the vehicle along such trajectory and within the safe handling envelope for the increased lateral offset in the lane change maneuver.
By considering the steering input data shown in figure 7.6 and the yaw rate data in figure 8.1b, it seems likely that with a smoother control action it would be possible to pass the obstacle while remaining within the handling envelope. If the controller would be able to maintain a more constant yaw rate it should be possible to pass the obstacle for all three starting positions without violating stability boundaries.
This lack of optimality of the observed control action can be attributed to a model mismatch between the controller prediction model and the real vehicle dynamics. Model mismatches between the prediction model used in the controller and the real vehicle are inevitable since every prediction model can only be an approximation of reality.
The bicycle model used in the controller is based on a number of simplifications (3.4)



(a) Spatial projections of feasibility predictions



(b) Yaw rate envelope violations



(c) Handling envelope

Figure 8.1: Comparison of experimental results with numerical feasibility predictions for lane change with 70 km/h

and (3.5). This mismatch may lead to situations where control actions that were predicted and calculated to keep the vehicle within the stability bounds actually lead to a violation of the these limits.
Modeling assumptions (3.4) and (3.5) as well as the linear rear tire model (3.9) used in the reachability calculations 5.2 also lead to mismatches between the feasibility calculation and actual capabilities of the vehicle.
Although these assumptions typically yield accurate results only up to half of the vehicle's handling limits [28], the experimental results from the lane change at the limits of handling still shows close resemblance to the numerical predictions.
While the experimental results showed that the controller implemented is able to stabilize the vehicle up and even slightly beyond its handling limits, this will typically not be necessary for the highway lane change maneuver considered in this work. Significantly reduced limits on yaw rate and side slip will be enforced for a fully automated vehicle driving on a highway. To ensure the comfort and the perception of safety of a passenger in such vehicle the motion will have to very smooth. Therefore for the application in a fully automated lane change we can expect a better match between experimental performance and feasibility prediction.
A further approach for improving the concept presented in this work resulting in better prediction accuracy could be attempted by explicitly accounting for model mismatches in the sense of disturbances. Both suboptimal control action in the vehicle as well as the errors introduced due to model mismatch between the prediction model for the offline calculations and the real vehicle's handling capabilities could be described by disturbances. If quantified properly they could be used to yield more conservative robust control invariant sets as well as robust N-step controllable sets which would improve prediction accuracy.

9
Conclusion

The work presented in this thesis provides an analysis of possibilities for using reachability analysis and invariant set theory to predict feasibility for a MPC controller performing an lane change maneuver.
To study the possibility of an online implementable algorithm the analytical solution of one iteration step of the N-step controllable set algorithm using hyperrectangles for inner approximation was determined.
We were able to prove that iteratively performing inner approximations of intermediate sets for control invariant set calculations or N-step controllable set calculations for systems containing integrators using such simple geometrical shapes leads to an inevitable collapse of the sets considered.
Although computationally very expensive, it was shown that with a heuristic approximation algorithm it is possible to numerically calculate the control invariant set as well as N-step controllable sets for a vehicle in a lane change maneuver.
To compare these offline precalculated predictions with experimental data an MPC controller was implemented that proved to be able to stabilize and control the vehicle during a lane change maneuver up and even slightly beyond the handling limits.
The predictions are slightly overestimating the ability of the controller to find a solution to pass an obstacle without violating vehicle stability boundaries.
We can attribute this result to modelling assumptions necessary to enable both the calculation of the feasibility prediction as well as the real-time controller implementation which loose accuracy in the high dynamic maneuvers performed in this work.
Future work can account for these inaccuracies by modelling them as disturbances which would lead to more conservative and precise results. Furthermore an application in a real autonomous vehicle will lead to an increased accuracy of the predictions since the modelling assumptions used are more accurate in typical driving situations compared to the severe lane change maneuver used for evaluation in this work.
From these results we can conclude that the feasibility of an MPC controller in a lane change maneuver can be evaluated using the techniques illustrated in this work. For an implementation in a vehicle it is possible to determine the upper bound for a feasible velocity of a maneuver by offline precalculating the maneuver for a range of velocities and using this data online to determine the feasibility of the maneuver for a given vehicle state in relation to an obstacle or a preceding car in its lane.

Bibliography

[1] Beal, C. E. (2011). Applications of Model Predictive Control to Vehicle Dynamics for Active Safety and Stability, Doctoral dissertation, Stanford University, USA

[2] Beal, C. E., & Gerdes, J. C. (2013). Model Predictive Control for Vehicle Stabilization at the Limits of Handling. Ieee Transactions on Control Systems Technology, 21(4), 1258–1269.

[3] Blanchini, F. (1999). Set invariance in control. Automatica, 35(11), 1747–1767.

[4] Borrelli, F., Falcone, P., Keviczky, T., Asgari, J., & Hrovat, D. (2005). MPC-based approach to active steering for autonomous vehicle systems. International Journal of Vehicle Autonomous Systems, 3(2/3/4), 265.

[5] F. Borrelli, A. Bemporad, M. Morari (2010) Constrained Optimal Control and Predictive Control for linear and hybrid systems, Berlin, Germany: Springer-Verlag, Sept. 2010

[6] Boyd, S., & Vandenberghe, L. (2010). Convex Optimization. Optimization Methods and Software (Vol. 25).

[7] Carvalho, A., Gao, Y., Gray, A., Tseng, H. E., & Borrelli, F. (2013). Predictive control of an autonomous ground vehicle using an iterative linearization approach. Intelligent Transportation Systems - (ITSC), 2013 16th International IEEE Conference on, (Itsc), 2335–2340.

[8] Del Re, L., Allgower, F., Glielmo, L., Guardiola, C. (2010). Automotive Model Predictive Control Models, Methods and Applications. Berlin Heidelberg: Springer-Verlag.

[9] Erlien, S. M., Funke, J., & Gerdes, J. C. (2014). Incorporating non-linear tire dynamics into a convex approach to shared steering control. Proceedings of the American Control Conference, 3468–3473.

[10] Erlien, S.M. (2015) Shared Vehicle Control using Safe Driving Envelopes for Obstacle Avoidance and Stability, Doctoral dissertation, Stanford University, USA

[11] P Falcone, F Borrelli, J Asgari, HE Tseng, D Hrovat (2011) Predictive active steering control for autonomous vehicle systems; Control Systems Technology, IEEE Transactions on 15 (3), 566-580.

[12] Falcone, P., Borrelli, F., Asgari, J., Tseng, H. E., & Hrovat, D. (2007). A model predictive control approach for combined braking and steering in autonomous vehicles. 2007 Mediterranean Conference on Control Automation, 1–6.

[13] Falcone, P., Borrelli, F., Tseng, H. E., Asgari, J., & Hrovat, D. (2008). A hierarchical Model Predictive Control framework for autonomous ground vehicles. 2008 American Control Conference, 3719–3724.

[14] Falcone, P., Ali, M., & Sjöberg, J. (2011). Predictive threat assessment via reachability analysis and set invariance theory. IEEE Transactions on Intelligent Transportation Systems, 12(4), 1352–1361.

[15] Fiala, E. (1954) Lateral forces on rolling pneumatic tires, Zeitschrift V.D.I., vol. 96/29, 973–979.

[16] Funke, J., & Gerdes, J. (2013). Simple Clothoid Paths for Autonomous Vehicle Lane Changes at the Limits of Handling. Asme 2013, 1–10.

[17] Funke, J., (2016) Collision Avoidance Up to the Handling Limits for Autonomous Vehicles, Doctoral dissertation, Stanford University, USA

[18] Gao, Y., Gray, A., Frasch, J. V, Lin, T., Tseng, E., Hedrick, J. K., & Borrelli, F. (2012). Spatial Predictive Control for Agile Semi-Autonomous Ground Vehicles. Proceedings of the 11th International Symposium on Advanced Vehicle Control, VD11(2), 1–6.

[19] Goubault, E., Mullier, O., Putot, S., & Kieffer, M. (2014). Inner Approximated Reachability Analysis. In Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control (pp. 163–172). New York, NY, USA: ACM.

[20] Gray, A., Gao, Y., Lin, T., Hedrick, J. K., Tseng, H. E., & Borrelli, F. (2012). Predictive control for agile semi-autonomous ground vehicles using motion primitives. American Control Conference (ACC), 2012, 4239–4244.

[21] Hadekel, R., (1952) The mechanical characteristics of pneumatic tyres. In S and T Memo TPA3/TIB.

[22] Herceg, M., Kvasnica, M., Jones, C., & Morari, M. (2013). Multi-Parametric Toolbox 3.0. In Proceedings of the European Control Conference (pp. 502–510).

[23] Hindiyeh, R. Y. (2013). Dynamics and Control of Drifting in Automobiles, (March 2013)., PHD Dissertation, Stanford University. Department of Mechanical Engineering

[24] Hsu, Y. H. J., & Gerdes, J. C. (2009). Envelope Control: Keeping the Vehicle Within its Handling Limits Using Front Steering. Proceedings of the 21st International Symposium on Dynamics of Vehicles on Roads and Tracks.

[25] ISO 3888-1:1999 Passenger cars — Test track for a severe lane-change manoeuvre — Part 1: Double lane-change, ISO, Geneva, Switzerland

[26] Maciejowski, J. M. (2002). Predictive Control with Constraints. Computers and Electronics in Agriculture (Vol. 63).

[27] Mattingley, J., & Boyd, S. (2012). CVXGEN: A code generator for embedded convex optimization. Optimization and Engineering, 13(1), 1–27.

[28] Pacejka, H.B (2005) Tire and Vehicle Dynamics. Society of Automotive Engineers, Warrendale, PA USA, 2nd edition

[29] Schrijver, A. (1998). Theory of Linear and Integer Programming. John Wiley & sons. pp. 155–156. ISBN 0-471-98232-6.

[30] Ziegler, G. M., (1995) Lectures on Polytopes, Springer New York, updated Seventh Printing of the First Edition

A

Analytical Projection

A.1 Fourier Motzkin Projection Algorithm

The Fourier Motzkin Elimination algorithm is a method to eliminate variables from a system of linear inequalities. Since the hyperplanes bounding the convex sets considered in this thesis are described by sets of linear inequations this algorithm can be used to project a polytope in $\mathbb{R}^N$ onto an $\mathbb{R}^{N-1}$ dimensional subspace. This explanation of the algorithm follows the presentation in [29].

Data: A system of inequalities $Ax \leq b$ with matrix $A \in \mathbb{R}^{m \times n}$, vector $b \in \mathbb{R}^m$, index $j \in [1..n]$ specifying the column to be eliminated
Result: System of inequalities $Dx \leq d$ with Matrix $D \in \mathbb{R}^{r \times n}$ such that $D_{i,j} = 0$ for all $i = 1, ..., r]$, Vector $d \in \mathbb{R}^r$

$Z \leftarrow \{i \in M \mid a_{i,j} = 0\}$
$N \leftarrow \{i \in M \mid a_{i,j} < 0\}$
$P \leftarrow \{i \in M \mid a_{i,j} > 0\}$
$R \leftarrow Z \cup (N \times P)$
$r \leftarrow |R|$
$p \leftarrow$ an index of the elements in R, $p : 1, ...r \rightarrow R$ **for** $i \in [1, ..., r]$ **do**
  if $p(i) \in Z$ **then**
    $D_i \leftarrow A_{p(i)}$
    $d_i \leftarrow b_{p(i)}$
  else if $p(i) = (s, t) \in N \times P$ **then**
    $D_i \leftarrow a_{tj} A_s - a_{sj} A_t$
    $d_i \leftarrow a_{tj} b_s - a_{sj} b_t$
  end
end

Algorithm 3: Calculation of the N-Step Controllable Set

The rows of the matrix A are sorted according to the value of the elements $a_{i,j}$ in the row j to be eliminated. Indices of Zero entries are stored in Z while indices of positive and negative entries are stored in P and N. The set R is used to represent the intersection of all indices of Z as well as tuples created by $N \times P$. Iterating through the elements of R the new set of inequalities is built. If an inequality had a zero entry for the variable to be eliminated it remains unchanged. For all the tuples in R the inequalities are combined in such way that the entry for the inequality to be eliminated becomes zero. The new set of inequalities contains only zero entries in the jth column and therefore becomes independent of that dimension.

A.2 Pre($\mathcal{X}$) of the AFI-Model using Hypercube Constraints

$$\begin{bmatrix} 1-\frac{C_{\alpha r}t_s}{mu(t)} & t_s\left(\frac{bC_{\alpha r}}{mu(t)}-u(t)\right) & 0 & 0 & \frac{t_s}{m} \\ \frac{C_{\alpha r}t_s}{mu(t)}-1 & \frac{t_s\left(u(t)^2-\frac{bC_{\alpha r}}{m}\right)}{u(t)} & 0 & 0 & -\frac{t_s}{m} \\ \frac{bC_{\alpha r}t_s}{I_z u(t)} & 1-\frac{b^2C_{\alpha r}t_s}{I_z u(t)} & 0 & 0 & \frac{at_s}{I_z} \\ -\frac{bC_{\alpha r}t_s}{I_z u(t)} & \frac{b^2C_{\alpha r}t_s}{I_z u(t)}-1 & 0 & 0 & -\frac{at_s}{I_z} \\ 0 & t_s & 1 & 0 & 0 \\ 0 & -t_s & -1 & 0 & 0 \\ t_s & 0 & t_s u(t) & 1 & 0 \\ -t_s & 0 & -t_s u(t) & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} v(t) \\ r(t) \\ \psi(t) \\ e(t) \\ F_{\text{yf}}(t) \end{bmatrix} \leq \begin{bmatrix} v_{\max} \\ v_{\max} \\ r_{\max} \\ r_{\max} \\ \psi_{\max} \\ -\psi_{\min} \\ e_{\max}(t+1) \\ -e_{\min}(t+1) \\ F_{\text{yf}_{\max}} \\ F_{\text{yf}_{\max}} \end{bmatrix} \tag{A.1}$$

A.3 One-Step Controllable Set for the AFI-Model with Hyperrectangle Constraints

$$
\begin{pmatrix}
0 & t_s & 1 & 0 \\
0 & -t_s & -1 & 0 \\
t_s & 0 & t_s u(t) & 1 \\
-t_s & 0 & -t_s u(t) & -1 \\
\frac{aC_{\alpha r}t_s^2}{mI_z u(t)} - \frac{at_s}{I_z} + \frac{bC_{\alpha r}t_s^2}{mI_z u(t)} & -\frac{abC_{\alpha r}t_s^2}{mI_z u(t)} + \frac{at_s^2 u(t)}{I_z} - \frac{b^2 C_{\alpha r}t_s^2}{mI_z u(t)} + \frac{t_s}{m} & 0 & 0 \\
\frac{C_{\alpha r}t_s}{mu(t)} - 1 & t_s u(t) - \frac{bC_{\alpha r}t_s}{mu(t)} & 0 & 0 \\
-\frac{aC_{\alpha r}t_s^2}{mI_z u(t)} + \frac{at_s}{I_z} - \frac{bC_{\alpha r}t_s^2}{mI_z u(t)} & \frac{abC_{\alpha r}t_s^2}{mI_z u(t)} - \frac{at_s^2 u(t)}{I_z} + \frac{b^2 C_{\alpha r}t_s^2}{mI_z u(t)} - \frac{t_s}{m} & 0 & 0 \\
-\frac{bC_{\alpha r}t_s}{I_z u(t)} & \frac{b^2 C_{\alpha r}t_s}{I_z u(t)} - 1 & 0 & 0 \\
1 - \frac{C_{\alpha r}t_s}{mu(t)} & \frac{bC_{\alpha r}t_s}{mu(t)} - t_s u(t) & 0 & 0 \\
\frac{bC_{\alpha r}t_s}{I_z u(t)} & 1 - \frac{b^2 C_{\alpha r}t_s}{I_z u(t)} & 0 & 0 \\
1 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & -1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & -1
\end{pmatrix}
\cdot
\begin{pmatrix}
v(t) \\ r(t) \\ \psi(t) \\ e(t)
\end{pmatrix}
\leq
\begin{pmatrix}
\psi_{\max} \\
\psi_{\max} \\
e_{\max}(t+1) \\
-e_{\min}(t+1) \\
\frac{at_s v_{\max}}{I_z} + \frac{t_s r_{\max}}{m} \\
\frac{F_{\text{yfmax}}t_s}{m} + v_{\max} \\
\frac{at_s v_{\max}}{I_z} + \frac{t_s r_{\max}}{m} \\
\frac{aF_{\text{yfmax}}t_s}{I_z} + r_{\max} \\
\frac{F_{\text{yfmax}}t_s}{m} + v_{\max} \\
\frac{aF_{\text{yfmax}}t_s}{I_z} + r_{\max} \\
v_{\max} \\
v_{\max} \\
r_{\max} \\
r_{\max} \\
\psi_{\max} \\
\psi_{\max} \\
e_{\max}(t) \\
-e_{\min}(t)
\end{pmatrix}
\tag{A.2}
$$

B

Inner Approximation of Symmetric Polytopes

```
function [ P_out ] = ForcePolytopeSymmetrySimplification_4D( P , ...
                                        nSymmetricConstraints, tolAngle)
% ForcePolytopeSymmetrySimplification_4D - Calculate symmetric inner
% approximation of a 4D-Polytope
%
% Inputs:
%    P - Input Polytope
%    nSymmetricConstraints - Number of desired remaining hyperplanes
%    tolAngle - Desired minimal angle between resulting hyperplanes
% Outputs:
%    P_out -  Output Polytope
% Example:
%    P_out = ForcePolytopeSymmetrySimplification_4D( P , 500, 2)
%
% Toolboxes required: MPT Toolbox 3.0.4
%                     Parallel Computing Toolbox
% Author: Stephan Heinrich
% email: stehei@student.chalmers.se
% Website:
% April 2015; Last revision: 25-October-2015
%------------- BEGIN CODE --------------
% Norm the input Polytope
Pnorm = normPolytope(P.minHRep());
H = Pnorm.H;
%% Calculate the facets for the input polytope
for i = 1:size(H,1)
    Facet(i) = Pnorm.getFacet(i);
end
%% Calculate the scaled difference between the hyperplane supporting vectors
OuterBox = Pnorm.outerApprox;
BoxScaling = OuterBox.H(1:4,5)';
MinErr = zeros(1,size(H,1));
MinIndex = zeros(1,size(H,1));
for i = 1:size(H,1)
    [MinErr(i),MinIndex(i)] = min(sum(...
        [abs(H(:,1)+H(i,1))/BoxScaling(1),...
        abs(H(:,2)+H(i,2))/BoxScaling(2),...
        abs(H(:,3)+H(i,3))/BoxScaling(3),...
        abs(H(:,4)+H(i,4))/BoxScaling(4) ],2));
end
```

```
%% Check if they are all pairwise ordered or if a new hyperplane
% must be added
SingleHyperplaneI = setdiff([1:size(H,1)], unique(MinIndex));
H_SingleHyperplanes = [];
h_SingleHyperplanes = [];
if ~isempty(SingleHyperplaneI)
    disp('Symmetry violated and needs to be fixed')
    H_SingleHyperplanes = [H(SingleHyperplaneI, 1:4) ;...
                           (-1)*H(SingleHyperplaneI, 1:4)];
    h_SingleHyperplanes = [H(SingleHyperplaneI, 5); H(SingleHyperplaneI, 5)];
end
%% Order the halfplanes in pairs
PairList = [];
for i = 1:size(H,1)
    if MinIndex(MinIndex(i)) == i
        PairList = [PairList; [sort([i, MinIndex(i)])]];
        MinIndex(i) = 0;
    end
end
%% Calculate the average slope for each pair. Sign is kept from the first
% entry of the pair
SlopeList = zeros(size(PairList,1),4);
for i = 1:size(PairList)
    SlopeList(i,:) =  [H(PairList(i,1), 1:4)+(-1).*H(PairList(i,2), 1:4)];
end
SlopeList = SlopeList./2;
%% Reevaluate the distance to the obstacle to ensure the corrected symmetric
% facets are inner approximations
dList = zeros(size(PairList,1),1);
VertexArrayPos= cell(1,size(PairList,1));
VertexArrayNeg= cell(1,size(PairList,1));
parfor i = 1:size(PairList,1)
    % Enumerate all vertices assiciated with the first halfspace of a pair
    dPos = 1;
    if ~Facet(PairList(i,1)).isEmptySet
        VertexArrayPos{i} = Facet(PairList(i,1)).V;
        try
            % Recalculate distance from the origin to make sure all
            % vertices are outside
            dPos = min(abs(SlopeList(i,:)* VertexArrayPos{i}'));
        catch
            disp('Corrupt facet detected... Caught error')
        end
    else
        disp('Corrupt facet detected..')
    end
    % Enumerate all vertices assiciated with the 2nd halfspace of a pair
    dNeg = 1;
    if ~Facet(PairList(i,2)).isEmptySet
        VertexArrayNeg{i} = Facet(PairList(i,2)).V;
        try
            % Recalculate distance from the origin to make sure all
            % vertices are outside
            dNeg = min(abs(SlopeList(i,:)* VertexArrayPos{i}'));
        catch
            disp('Corrupt facet detected... Caught error')
```

```
        end

    else
        disp('Corrupt facet detected..')
    end
    % Keep the smaller distance to the origin to remain symmetric
    dList(i) = min([dPos dNeg]);
end
%% Find Halfspaces with similar normal vectors and combine them to an inner
% approximation
% Initially determine angles between hyperplanes
AngleMatrix = zeros(size(SlopeList,1),size(SlopeList,1));
for i = 1:size(SlopeList,1)
    AngleMatrix(:,i) = SlopeList*SlopeList(i,:)'./ ...
    (sqrt(sum(SlopeList.*SlopeList,2))*sqrt(SlopeList(i,:)*SlopeList(i,:)'));
end
% Ignore diagonals
AngleMatrix(eye(size(AngleMatrix))== true) = 0;
% Determine the angle between hyperplanes
[max_val,idx]=max(abs(AngleMatrix(:)));
% Iteratively combine halfspaces until conditions are met
while max_val > cos(tolAngle*pi/180) && ...
        size(SlopeList, 1) > nSymmetricConstraints
    % Determine which hyperplanes will be combined
    [row,col]=ind2sub(size(AngleMatrix),idx);
    %Check if they are in the same direction and combine
    if sign( AngleMatrix(row, col)) == 1
        NewSlope = (SlopeList(row,:)+SlopeList(col,:))./2;
        PosVertexList = [VertexArrayPos{row};VertexArrayPos{col}];
        NegVertexList = [VertexArrayPos{row};VertexArrayPos{col}];
    else
        NewSlope = (SlopeList(row,:)-SlopeList(col,:))./2;
        PosVertexList = [VertexArrayPos{row};VertexArrayNeg{col}];
        NegVertexList = [VertexArrayNeg{row};VertexArrayPos{col}];
    end
    % Determine new distance to the origin
    New_d = min([abs(NewSlope*PosVertexList'), abs(NewSlope*NegVertexList')]);
    % Change the starting matrices to enable iteration
    VertexArrayPos(max([row col])) = [];
    VertexArrayPos(min([row col])) = [];
    VertexArrayNeg(max([row col])) = [];
    VertexArrayNeg(min([row col])) = [];
    VertexArrayPos{end+1} = PosVertexList;
    VertexArrayNeg{end+1} = NegVertexList;
    SlopeList(max([row col]),:) = [];
    SlopeList(min([row col]),:) = [];
    SlopeList(end+1,:) = NewSlope;
    dList(max([row col])) = [];
    dList(min([row col])) = [];
    dList(end+1) = New_d;
    % Recalculate the angles between remaining hyperplanes
    AngleMatrix = zeros(size(SlopeList,1),size(SlopeList,1));
    for i = 1:size(SlopeList,1)
        AngleMatrix(:,i) = SlopeList*SlopeList(i,:)'./ ...
                (sqrt(sum(SlopeList.*SlopeList,2))...
                *sqrt(SlopeList(i,:)*SlopeList(i,:)'));
```

```
    end
    % Ignore diagonals
    AngleMatrix(eye(size(AngleMatrix))== true) = 0;
    %calculate the vectors that should be combined
    [max_val,idx]=max(abs(AngleMatrix(:)));
end
%% Rebuild the approximated polytope
P_out = Polyhedron([SlopeList; SlopeList.*(-1); H_SingleHyperplanes],...
    [dList; dList; h_SingleHyperplanes]);
end
```

C

Experimental Testing

C.1 CVXgen Optimization Problem Statement

```
dimensions
  m = 1  # Number of inputs.
  n = 4  # Number of states
  T_short = 5 # Near horizon.
  T_long = 20  # Far horizon.
end
parameters
  x[0] (n)  # Initial state.
  u0 (1) # Previous input

  # Linearized system dynmics in near horizon
  A_short (n,n)1,1 1,2 2,1 2,2 3,2 3,3 4,1 4,3 4,4
  B_short (n,m)1,1 2,1
  dc_short (n)1,1 2,1

  # Linearized system dynmics in correction step
  A_int (n,n)1,1 1,2 2,1 2,2 3,2 3,3 4,1 4,3 4,4
  B_int (n,m)1,1 2,1  # input matrix.

  # Linearized system dynmics in far step
  A_long (n,n)1,1 1,2 2,1 2,2 3,2 3,3 4,1 4,3 4,4
  B_long (n,m)1,1 2,1  # input matrix.

  # Input limits
  Fyf_max nonnegative  # Input limit.
  dFyf_max nonnegative  # Input slew rate limit.

  # Handling Envelope.  H_sh*x<=G_sh
  H_sh (4,n)1,1 1,2 2,2 3,1 3,2 4,2
  G_sh (4)

  # Environmental Envelope.  H_env*x <=G_env
  H_env (2,n)1,4 2,4
  G_env[i] (2), i=1..T_short+T_long
```

```
  sigma_env psd # Environment envelope cost term
  sigma_sh psd # Handling envelope cost term

  sigma_e nonneg # lane tracking cost
  sigma_psi psd # Heading cost

  sigma_u_near psd # Input force cost term
  sigma_u_far psd # Input force cost term

  sigma_du_near psd # Input derivative cost term
  sigma_du_far psd # Input derivative cost term
end
variables
  x[t] (n), t=1..T_short+T_long  # State
  u[t] (m), t=0..T_short+T_long-1  # Input
  s_sh[t] (1), t=1..T_short+T_long  # Safe handling envelope
  s_env[t] (1), t=1..T_short+T_long  # Environmental envelope
end
minimize
  sum[t=T_short+1..T_short+T_long](quad(s_env[t], sigma_env))
  +sum[t=1..T_short+T_long](sigma_sh*s_sh[t])
  +sum[t=T_short+1..T_short+T_long](sigma_e*abs(x[t][4])
        +quad(x[t][3], sigma_psi))
  +sum[t=0..T_short-1](quad(u[t], sigma_u_near))
  +sum[t=T_short..T_short+T_long-1](quad(u[t], sigma_u_far))
  +sum[t=0..T_short-1](quad(u[t]-u[t+1], sigma_du_near))
  +sum[t=T_short..T_short+T_long-2](quad(u[t]-u[t+1], sigma_du_far))
subject to
  # Motion model near horizon
  x[t+1] == A_short*x[t] + B_short*u[t]+ dc_short , t=0..T_short-1
  # Motion model correction step
  x[t+1] == A_int*x[t] + B_int*u[t] , t=T_short..T_short
  # Motion model far horizon
  x[t+1] == A_long*x[t] + B_long*u[t], t=T_short+1..T_short+T_long-1

  # Environmental envelope
  H_env *x[t] <= G_env[t] + s_env[t] , t=T_short+1..T_long

  # Safe handling envelope
  H_sh *x[t] <= G_sh + s_sh[t] , t= 1..T_short+T_long

  # Input limit constraint
  abs(u[t]) <= Fyf_max , t=0..T_short+T_long-1
  # Input slew rate constraint
  abs(u[t+1] - u[t]) <= dFyf_max , t=0..T_short
```

```
  abs(u[0] - u0) <= dFyf_max

  # Soft constraint slack variables
  s_sh[t] >= 0 , t=1..T_short+T_long
  s_env[t] >= 0 , t=1..T_short+T_long
end
```

C.2 Controller Parameters

σ_{env}	1000
σ_{sh}	60
σ_{e}	10
σ_{ψ}	5
$\sigma_{u_{near}}$	2×10^{-6}
$\sigma_{u_{far}}$	2×10^{-5}
$\sigma_{du_{near}}$	5×10^{-10}
$\sigma_{du_{far}}$	5×10^{-9}

Table C.1: Controller parameters used in experimental testing