Ruprecht-Karls-Universität Heidelberg

Optimal Control of Vehicles in Driving Simulators

Diplomarbeit

von
Florian Kehrle

Betreuer:
Dr. Sebastian Sager

March 11, 2010

Abstract

The focus of this thesis is on the mathematical modeling, simulation, and optimal control of racing cars under realistic conditions. A mathematical model yielding a compromise between being suited for optimization in real-time and an adequate approximation of physical reality is derived, based on previous work from literature and an intrinsic model used in an open source racing simulation software. State-of-the-art methods for nonstandard optimal control problems are applied. To our knowledge, for the first time an offline solution for a time-optimal drive on the *Hockenheimring* including non-automatic gear shifts is calculated. The intention of our work is to test algorithmic improvements in the optimization software under highly realistic conditions. The racing simulation supplies a visualization tool for the complex, nonlinear processes to be optimized. The combination of an optimization software and a racing simulation furthermore establishes the basis to apply *Nonlinear Model Predictive Control (NMPC)*. Solution techniques for the described *mixed integer optimal control problems (MIOCPs)* are discussed, including an *Outer Convexification* approach. These techniques are implemented in the optimization software MUSCOD-II, which has been developed by the working group of *Prof. Dr. Dr. h.c. H.G. Bock* and *Dr. J.P. Schlöder* at the *Interdisciplinary Center for Scientific Computing (IWR)* of the *University of Heidelberg*. For interfacing an existing example for time optimal control of a vehicle with gear shifts to the racing simulator, a transformation of the coordinate system as well as modifications in the set of ordinary differential equations has been necessary. Hence, this thesis describes the optimal control problem not only on a short test track, but on realistic three-dimensional racing tracks. Therefore, an automatic import of the racing simulation track data into the standard form of the optimization software is implemented. The embedded mathematical car model improved in functionality and car specific parameters, which can be automatically included from various vehicles of the racing game as well. The numerical results of an optimal control on a complete Formula One Grand Prix racing circuit are presented, including two different car models. This optimal solution is achieved with an independently optimized gear selection, which is not implicitly defined to the actual state. The length and nature of the track, as well as the challenges appearing with the binary controls of gear shifting, result in a large-scale optimization problem. The computed optimal solution with a 1989 PORSCHE CLUBSPORT (max. velocity $248\frac{\text{km}}{\text{h}}$) of 118.85s for a complete lap on the *Hockenheimring*, can be compared to a real world lap time record driven by a 2006 PORSCHE 997 GT3 (max. velocity $310\frac{\text{km}}{\text{h}}$) in 116.41s. Additionally, the computed optimal control values are integrated to the racing simulator and illustrated with screenshots.

Mathematics Subject Classification (2010): 49J15.

Zusammenfassung

Der Fokus dieser Diplomarbeit liegt auf dem mathematischen Modellieren, der Simulation und der optimalen Steuerung von Rennfahrzeugen unter realistischen Bedingungen. Anhand eines integrierten Modells, das in einer open-source Rennsimulationssoftware benutzt wird, und früherer Arbeiten aus der Literatur, wird ein mathematisches Modell abgeleitet, welches einen Kompromiss zwischen Eignung für Echtzeitoptimierung und einer angemessenen Annäherung der physikalischen Realität bildet. Es werden modernste Methoden für nicht standardisierte Optimalsteuerungsprobleme angewendet und - nach unserem Kenntnisstand - zum ersten Mal eine Offline Lösung für zeit-optimales Fahren mit Gangschaltung auf dem *Hockenheimring* berechnet. Das Ziel dieser Arbeit ist, algorithmische Erweiterungen in der Optimalsteuerungssoftware zukünftig unter sehr realistischen Bedingungen testen zu können. Die vorhandenen Visualisierungen der Rennsimulation machen es möglich, die komplexen nichtlinearen Zusammenhänge, die man aus einer Optimierung erwarten kann, zu veranschaulichen. Die Verknüpfung einer Software zur Optimalsteuerung mit einer Rennsimulation zielt außerdem darauf ab, *Nichtlineare Modell-Prädiktive Steuerung (NMPC)* auf das Problem anwenden zu können. In diesem Zusammenhang werden spezielle Lösungsverfahren für *Gemischt-Ganzzahliger Optimalsteuerungsprobleme (MIOCPs)* vorgestellt, unter anderem die Methode der *Konvexifizierung*. Die präsentierten Lösungsverfahren sind in der Optimalsteuerungssoftware MUSCOD-II implementiert, welche in der Arbeitsgruppe von *Prof. Dr. Dr. h.c. H.G. Bock* und *Dr. J.P. Schlöder* am *Interdisziplinären Zentrum für wissenschaftliches Rechnen (IWR)* der *Universität Heidelberg* entwickelt wird. Um ein vorhandenes Beispiel zur zeit-optimalen Steuerung eines Fahrzeuges mit Gangschaltung mit dem Rennsimulator verknüpfen zu können, wurden Transformationen der vorhandenen Koordinatensysteme, sowie des eingebauten Differentialgleichungsmodells notwendig. Diese Arbeit beschreibt das untersuchte Optimalsteuerungsproblem nicht nur auf einer kurzen Teststrecke wie bisher, sondern auf realistischen drei-dimensionalen Rennstrecken. Dabei wurde ein Verfahren implementiert, welches einen automatischen Import der gewünschten Streckendaten aus der Rennsimulation in das Standardformat erlaubt. Das eingebaute mathematische Fahrzeugmodell wurde im Bezug auf Funktionalität ebenso verbessert, wie in der Genauigkeit fahrzeugspezifischer Parameter, welche nun für beliebige Fahrzeuge aus dem Rennspiel automatisch eingebunden werden können. Die numerischen Ergebnisse zur optimalen Steuerung zweier unterschiedlicher Fahrzeugmodelle werden für eine komplette Formel 1 Grand-Prix-Rennstrecke präsentiert. Dabei ist die Gangschaltung nicht implizit abhängig vom aktuellen Zustand, sondern wird unabhängig davon optimiert. Die Beschaffenheit und Länge der Strecke, sowie die auftretenden Schwierigkeiten der binären Steuerungen der Gangschaltung, führen zu einem großflächigen Problem. Die berechnete optimale Lösung eines 1989 PORSCHE CLUBSPORT (Maximalgeschwindigkeit $248\,\frac{\mathrm{km}}{\mathrm{h}}$) auf einer vollständigen Runde des *Hockenheimrings* in 118.85s kann mit dem aktuell bestehenden Rundenrekord eines 2006 PORSCHE 997 GT3 (Maximalgeschwindigkeit $310\,\frac{\mathrm{km}}{\mathrm{h}}$) von 116.41s verglichen werden. Die berechnete Lösung zur Optimalsteuerung wird außerdem noch in den Rennsimulator eingebaut und mittels Screenshots in der Arbeit präsentiert.

Contents

Chapter 0

Introduction

The simulation of driving vehicles has gained a lot of relevance over the past years. These simulations are applied in driving characteristics under extreme conditions, like in crash tests or in the "moose test". There are further applications in reduction of energy consumption and raising acceleration of a vehicle. In this connection, as well as primarily due to the computer gaming industry, several realistic mathematical models have been developed in the field of simulation software.

This thesis aims for coupling such racing simulation software, available as open source, to software for optimization of dynamic processes. Here the intention is on the ability to try algorithmic improvements in the optimization software under highly realistic conditions. Furthermore, the racing simulation supplies a great visualization tool for the complex, nonlinear processes to be optimized.

The combination of an optimization software and a racing simulation is designed with a view to time optimal control of a vehicle on a particular track. Most of the racing simulators come along with so-called *artificial intelligence* or AI drivers, which are normally controlled by a more or less simple algorithmic calculation of a racing line. The acceleration, as well as braking, and steering of the AI driver result from the basis of this racing line.

However, recently a more complex approach has been developed, as can be seen in the papers of Gerdts [15, 16], and Kirches [22]. Therein, a time optimal control of a vehicle with gear shifts on a short test track is presented, in the following simply called *testdrive*. Furthermore, a *fast solution of periodic optimal control problems in automobile test-driving with gear shifts* is given by Sager in [32]. Based on the same optimal control problem as in *testdrive*, the periodic version is applied on an elliptic track.

The difficulty of these problems lie in the integer controls of the gear shifts, which in combination with the continuous controls of acceleration, braking, and steering results in a problem class, called *mixed-integer optimal control problems* (MIOCP). To handle these problems, a combination of the particular solving techniques of *sequen-*

tial quadratic programming, the *direct multiple shooting method* and partly the MS MINTOC algorithm is used in the context of this thesis. These techniques are implemented in the mentioned optimization software called MUSCOD-II, which has been developed by the working group of H.G. Bock and J.P. Schlöder at the *Interdisciplinary Center for Scientific Computing (IWR)* of the University of Heidelberg.

In addition to the mathematical difficulties, appearing even for smaller problems as in the about 200m *testdrive* track, the main improvement of this thesis is the approach to optimize a vehicle control with gear shifts on a complete realistic Formula One racing track of a length of several kilometers. The complexity of this problem rises extremely with the track length as well as with its course.

As preliminary work, an extensive literature and software evaluation and reprocessing is made. We extend the existing model of *testdrive* by a modification of the *ordinary differential equations (ODEs)*. With the intention to integrate the computed optimal solution into the racing simulator, it is necessary to take a closer look at each of the embedded mathematical models of automobiles. Besides, a sensitivity analysis of each mathematical model is produced. We achieve three-dimensional track information in the standard format of the optimal control problem. This is implemented via automatic import of optional racing tracks from the racing simulation game. Besides, an appropriate track parametrization has to be attached to the optimization software. To actually integrate the optimal solution to the racing simulator, input/output operations are implemented within its source code.

0.1 Thesis Outline

Chapter 1 Chapter One gives a historical overview of racing video games. Different open-source games of that genre are presented, as well as the racing simulation we focus on: VDRIFT.

Chapter 2 Chapter Two introduces multibody systems and explains how mathematical models of real world procedures operate. The importance of tire dynamics in car modeling is presented, in particular with Pacejka's so-called *Magic Formula* [25], that shows how forces act on spinning wheels. We describe the functionality of the engine model in VDRIFT, in addition to transmission of energy to the wheels. Therefore, specific parameters of two realistic VDRIFT vehicles are illustrated, representing the same cars that are used in the following optimization part of the thesis. Then we focus on the buildup of a car model in the *mixed-integer optimal control problem testdrive* (short *testdrive*), as shown in [15, 22]. Furthermore, we demonstrate the necessity of the extensions and modifications in *testdrive*'s set of *ordinary differential equations*, for the interaction with the racing simulator VDRIFT. These modifications lead to the *extended* version of *testdrive*, finally presented in this chapter.

Chapter 3 Chapter Three starts with a track course formulation of the original *testdrive* problem. Besides, an overview in Bézier splines and patches [34] is used to give an idea of the functionality of the embedded coordinate system in VDrift. Due to the intention to apply the *extended testdrive* optimal control problem to an optional VDrift track, we use Bézier patches for a transformation between both embedded coordinate systems.

Chapter 4 Chapter Four presents the basic mathematical problems that are used in the context of this thesis. We define the general form of a *nonlinear problem*, as well as in particular a *mixed-integer optimal control problem* (MIOCP) and approaches how to treat those. Hence, the *direct multiple shooting* method in combination with a *sequential quadratic programming* algorithm is shown [37, 29, 17, 24]. Finally, the difficulty of integrality in combination with nonlinearity of the MIOCP formulation can be solved, by reformulating via the *Outer Convexification* approach [33].

Chapter 5 Chapter Five shows the numerical results of the original *testdrive* problem in comparison to the extended version. Due to the modifications in the ODE model, the track course had to be smoothed to solve the new, transformed optimal control problem. On the basis of the visualized numerical results, it is illustrated that the new model produces similar solutions in the driving behavior, but has some differences in velocity and steering, based on modifications in car and track model.

Chapter 6 Chapter Six illustrates the numerical results of the optimal control of two different vehicles, driving a lap of an official Formula One Grand Prix racing circuit. In this connection, the solution of a continuously optimized gear selection is presented, which not addicted to the actual state like in automatic transmission. This improvement is one of the main characteristics of this thesis. We give some information about the track as well as the initialization approach for the optimization and discuss the visualized numerical results.

Chapter 7 Chapter Seven represents in a way the inversion of the previous chapter, in which we achieved the optimal solution of a VDrift car and track within the "Muscod-II/*testdrive* world". This chapter shows the other way around. This means that the optimal control of a specific car and track calculated with Muscod-II, is applied to the control of an AI driver within VDrift. Therefore, we primarily have to implement some extensions in VDrift's source code. The results are illustrated as screenshots.

Chapter 8 Chapter Eight discusses the challenges appearing with the presented approaches, in linearization by Outer Convexification of the constraints, with the hugeness of the optimization problem, and with the accuracy while putting the numerical

results as control data into the racing game. Finally, proposals for solution are presented with model adjustment and *Nonlinear Model Predictive Control* closing with a summary of the achieved research.

Chapter 1

Racing Simulators

1.1 Commercial Racing Video Games

Racing video games can roughly be classified in two different genres. On the one hand there are racing simulations, on the other arcade racing games. Additionally you can divide every PC game in commercial and open source software. The main focus in racing simulations is on extremely realistic physics engines, whereas arcades are primarily concerned with gaming fun and a high speed experience with more "liberal" physics.

Due to limited hardware capabilities, mostly simple arcade games have been developed in the early stages of racing video games (*Gran Trak 10* released by Atari in 1974). The precursor of today's racing games with a third person view and AI drivers was *Pole Position* (1982), which included a real racing circuit for the first time. However, there were also racing simulations with focus on reality like *Revs - Formula Three simulation* (1984), which had only one selectable track due to hardware limitations, but with high track details compared to other games at that time.

Since the late 1990's faster CPUs allowed more and more realistic physics and of course better graphics (*Gran Turismo* - 1997). Higher grade of reality increases the difficulty of handling the car, which is one reason why arcade games generally are more popular than realistic simulations. Hence, most simulations have different levels of difficulty, which provide the possibility to activate little helpers like traction control, automatic gear shift, steering assistance or damage resistance.

1.2 Open-Source Software

Our intention is to couple the optimization algorithms we developed (presented in Chapter 4) with a car racing simulation game. We like to compare the solutions of our algorithms, which include optimal control of steering, accelerating, braking and gear shifting, to the existing AI drivers of the racing game. Furthermore, we want to

integrate those solutions to the racing simulator to have an excellent visualization of optimized results. Therefore, modifications in the source code of the racing game will be necessary. By using open source software, we are able to make required modifications in this source code. Besides, higher developed open source racing games place emphasis on detailed realistic physics. Most of these simulations try to exactly replicate the handling of the original automobile, often under different car names, when the acquisition of licenses would be too expensive.

We compared several open source racing simulators in reference to:

- physical models
- programming language
- coordinate systems
- computer controlled drivers
- graphics
- contact and support by the software engineer

Three different simulators were considered more precisely - *Racer*, *Torcs* and VDrift. We will briefly explain the decision against *Racer* and *Torcs*. Thereafter, we present our decision to work with the open source racing simulator VDrift in a more detailed report.

1.2.1 Racer

Racer, created by Ruud van Gaal (in 2000-2001)[1] is using professional car physics and an excellent render engine for graphical realism. It is written in C++ and OpenGL is used for rendering. Cars and tracks can easily be created and modified with tools available on the website. Therefore, there are many very good car and track models. *Racer* is using Pacejka's Magic Formula (see Section 2.2) for tire modeling.

According to the website, *Racer* is a free cross-platform car simulation project, however there is a comment by the author in a small section of the website saying "This is NOT an Open Source project...". We decided not to use that simulator due to prevention of license problems.

1.2.2 TORCS

"The Open Racing Car Simulator"[2], was initially written by Eric Espié and Christophe Guionneau in 1997. The current project leader is Bernhard Wymann. *Torcs* is a car

[1] www.racer.nl
[2] www.torcs.org

racing simulation which features more than 50 different cars, 20 tracks and 50 opponents to race against. You can also develop your own computer-controlled driver in C or C++. The graphics are good, but *Torcs* has some weaknesses in the physical model compared to the physics engine of VDrift.

1.3 VDrift

According to its website[3], VDrift is a cross-platform, open source driving simulation made with drift racing in mind. VDrift was created by Joe Venzon in early 2005. The driving physics engine was recently re-written from scratch, but it was inspired and owes much to the *Vamos*[4] physics engine. It is released under GNU General Public License (GPL) v2 and currently available for Linux, FreeBSD, Mac OS X, and Windows.

Figure 1.1 – Screenshot VDrift (VDrift website [36]).

Joe Venzon, developer of VDrift, describes the creation of the project in Auto Sim Sport Magazine[5]:

> "I started experimenting with breaking traction during turns, not so much interested in getting a fast time, but with getting the car sideways [...]

[3] www.vdrift.net

[4] http://vamos.sourceforge.net

[5] Joe Venzon in Auto Sim Sport Magazine - http://issuu.com/autosimsportmediallc/docs/vol5num1/1?viewMode=magazine

> Unfortunately the racing games I was playing at the time did a poor job of modeling car behavior when traction is lost [...] I found a great open-source simulator called Vamos that offered solid physics but little else [...] worked on improving the graphics and adding some small layers of gameplay."

Since the beginning of the project in 2005

> "[...] VDrift has become a pretty good grip racing game, and even allows a decent F1 driving experience. Drift is still in the name, but it's more of a general driving simulation game now."

The tire model of VDrift started with the classic Pacejka system (presented in Chapter 2.2) and was refined based on the ideas from Brian Beckman's 'Physics of Racing' [3] papers. There are more than 30 cars and also 30 fully modeled tracks (scenery and terrain) in VDrift. Optional driver assistance like automatic shifting, traction control and anti-lock braking mode are implemented. You are able to switch between several different camera angles and even to replay the race with a Skip Forward / Skip Backward function. Furthermore, there is the possibility to show the optimal racing line during the race, calculated by Remi Coulom's K1999 Path-Optimization Algorithm [9]. VDrift is still in development and any question to code or game will be answered by Joe Venzon or other forum users right away.

Chapter 2

Mathematical Models of Car Driving

In this chapter, we start with a common description of multibody systems. Thereupon, we give a brief illustration of a widespread method in tire modeling. This method, which is called Pacejka's *Magic Formula* [25], is used in different varieties in all kinds of models of this thesis. After that, we will present three different car models.

First of all, there is the racing simulator VDrift. The layout of a vehicle engine is shown at the explicit example of VDrift's car model. Therefore we use a detailed description with parameters of two different vehicles.

The foundation of the model that is used in this thesis is given by Gerdts in [15]. Gerdts presents a mixed-integer optimal control problem of a vehicle with gear shift on a short test track, called *testdrive* (in the following denoted by "original *testdrive* example" or simply "*testdrive*"). This original version of *testdrive* involves discrete controls for the choice of gears and was compared by Kirches in [22] to a convexification and relaxation of the integer control constraints. The *testdrive* car model and its system of ordinary differential equations (ODEs) is defined as second model.

At last, an extended version of the original *testdrive* model is illustrated that was built in the context of this thesis. Our intention is to use the mixed-integer optimal control of the original *testdrive* car model with convexification and relaxation of the integer control constraints on an optional VDrift track. This was realized by a modification of the ODE system. In addition we enhanced the engine model and car parameters of *testdrive*, using car specific parameters of the two different VDrift cars, which we introduce in the "VDrift Car Model" section.

2.1 Multibody Systems

To describe physical or mechanical real world trajectories, for example of walking robots or driving vehicles, you have to replace the trajectories with an equivalent math-

ematical model. The models that are used in this thesis, consisting of connected rigid or deformable parts are called multibody systems. You can describe multibody systems by ODEs, which are normally highly non-linear and cannot be solved analytically. Therefore, you need to approach a solution with numerical methods.

In order to identify the position and orientation of a rigid body in three-dimensional space, six coordinates are required. Each three coordinates to describe translational and rotational motion of the body. Generally, in dealing with multibody systems two different kinds of coordinate systems are used. A fixed "global" coordinate system, which allows to identify relations between different bodies or translation of a body in a certain time. The origin of the second type of coordinate systems is stuck to the center of mass of each body called "local" coordinate system. This system naturally changes with every time shift and is used for orientation and angle calculations at the body.

Thus, for unconstrained motion a rigid body in three-dimensional space exhibits six degrees of freedom (DOF), as you can see in Figure 2.1. These DOF, as a number of system coordinates, are reduced by the number of independent equalities for constrained motion. If a body is fixed on a plane, as in two-dimensional car driving for example, only three DOF (one rotational, two translational) are left.

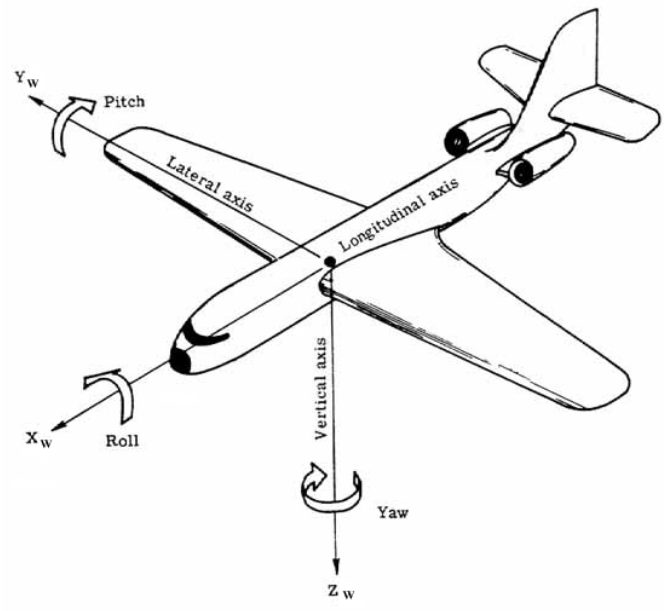


Figure 2.1 – Six degrees of freedom - yaw, pitch and roll angle for rotational and x, y and z for translational motion of an airplane (website [1]).

2.2 Pacejka's *Magic Formula* Tire Model

For the dynamic behavior of a road vehicle, tire characteristics are of essential importance. However, accurate friction models of road and tire interfaces are very difficult to obtain. For Example: A vehicle does not follow the road precisely for a specific steered angle of the wheel, due to lateral sliding forces. Most of today's racing simulations, just as professional tire research, are using a version of Pacejka's so-called *Magic Formula*. It is the state-of-the-art in realistic tire modeling. In the following, we give an overview over this *Magic Formula*, as both models of *testdrive* and VDrift are built with variations of it. For more detailed analysis, take a look at Pacejka's "Tyre and Vehicle Dynamics" [25].

The *Magic Formula* is a semi-empirical[1] tire model to calculate steady-state tire force and moment characteristics. The forces are generated by the model as a result of different wheel angles and parameters. The main input variables are:

camber angle γ, which is the inclination of the wheel to the vertical plane

side slip angle α, as the angle between the wheel's orientation and the actual direction of movement

slip ratio κ is used as longitudinal slip, while accelerating and braking

normal load F_z influences the grip of the tire on the road

As a result, three varying forces act on the wheel and accordingly affect the motion of the vehicle:

- F_x heading in **longitudinal direction**

  By pressing the throttle, the wheel speed increases and gets minimally higher than the current ground speed, so the car accelerates. If the wheel spins too fast, grip gets lost, resulting in less acceleration. For braking instead, the same force exists in opposite direction.

- **Lateral force** F_y

  Depending on the side slip angle, which describes the direction of the wheel compared to the actual vehicle direction on the ground.

- The **(self) aligning moment** M_z

  Acts on the steered wheel, trying to center the tire back to straight ahead driving.

[1] Based on measured data, but with physical structures

The *Magic Formula*

The general form of the formula for given values of side slip, slip ratio, vertical load and camber angle reads as follows. A more detailed description of the composed factors of longitudinal force, lateral force, and the aligning moment is given at the particular example of VDrift in Equation 2.14, 2.15, and 2.16.

$$y(x) = D \sin\Big(C \arctan\Big(Bx - E\,(Bx - \arctan Bx)\Big)\Big) \tag{2.1a}$$

with

$$Y(X) = y(x) + S_V \tag{2.1b}$$

where

$$x = X + S_H \tag{2.1c}$$

Y : output variable $F_x,\ F_y$ or possible M_z

X : input variable $\tan\alpha$ or κ

and including input variables $\gamma,\ F_z$

B stiffness factor

C shape factor

D peak value

E curvature factor

S_H horizontal shift

S_V vertical shift

The function describes a curve that matches the measurement data quite well, as can be seen in Figure 2.2 and 2.3. The forces are displayed for different loads F_z and variable camber or slip angles. For varying parameters of stiffness, shape, peak or curvature at each tire set, the curve shows force F_x, F_y or the aligning moment M_z.
In the following sections we will display how the *Magic Formula* parameters and variables are composed depending on the particular car model.

2.3 VDrift

Unfortunately the VDrift documentation includes no mathematical car model. Hence, we had to search the code to discover detailed information, like the operation method of the engine. However, VDrift's "Documentation Wiki"[2] gives some reference about car parameters. An overview of the global functionality of an engine and its

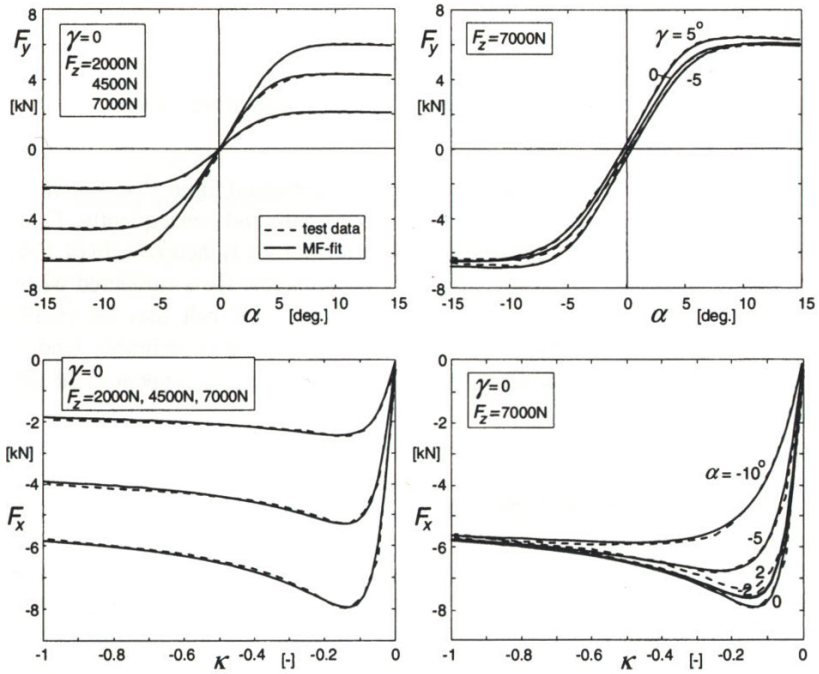


Figure 2.2 – Force characteristics of a 195/65 R15 car tire. *Magic Formula* computed results compared with data from measurements (dotted curves) (Pacejka [25]).

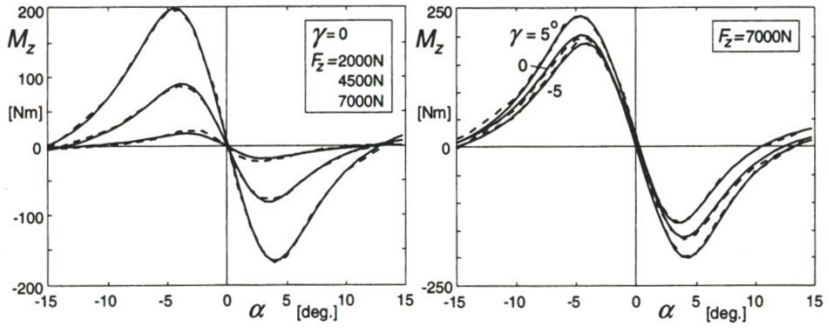


Figure 2.3 – Aligning torque characteristics of a 195/65 R15 car tire. *Magic Formula* computed results compared with data from measurements (dotted curves) (Pacejka [25]).

transmission of energy to the wheels, specifically at the example of VDrift's car model, is presented in Figure 2.5. For a closer look, an extensive description of the individual car parts follows.

2.3.1 VDrift Car Model

Within the diagram, blue rhombuses illustrate the individual vehicle parts with their parameters. For each car model there are different parameters shown in the gray, rounded boxes. Controls are displayed in form of a red circle. The green rectangles show the actual calculations of the vehicle parts, which combine the different input values. States needed in the evaluation are enclosed by a yellow rounded box (see Figure 2.4).

Figure 2.4 – Legend of the diagram

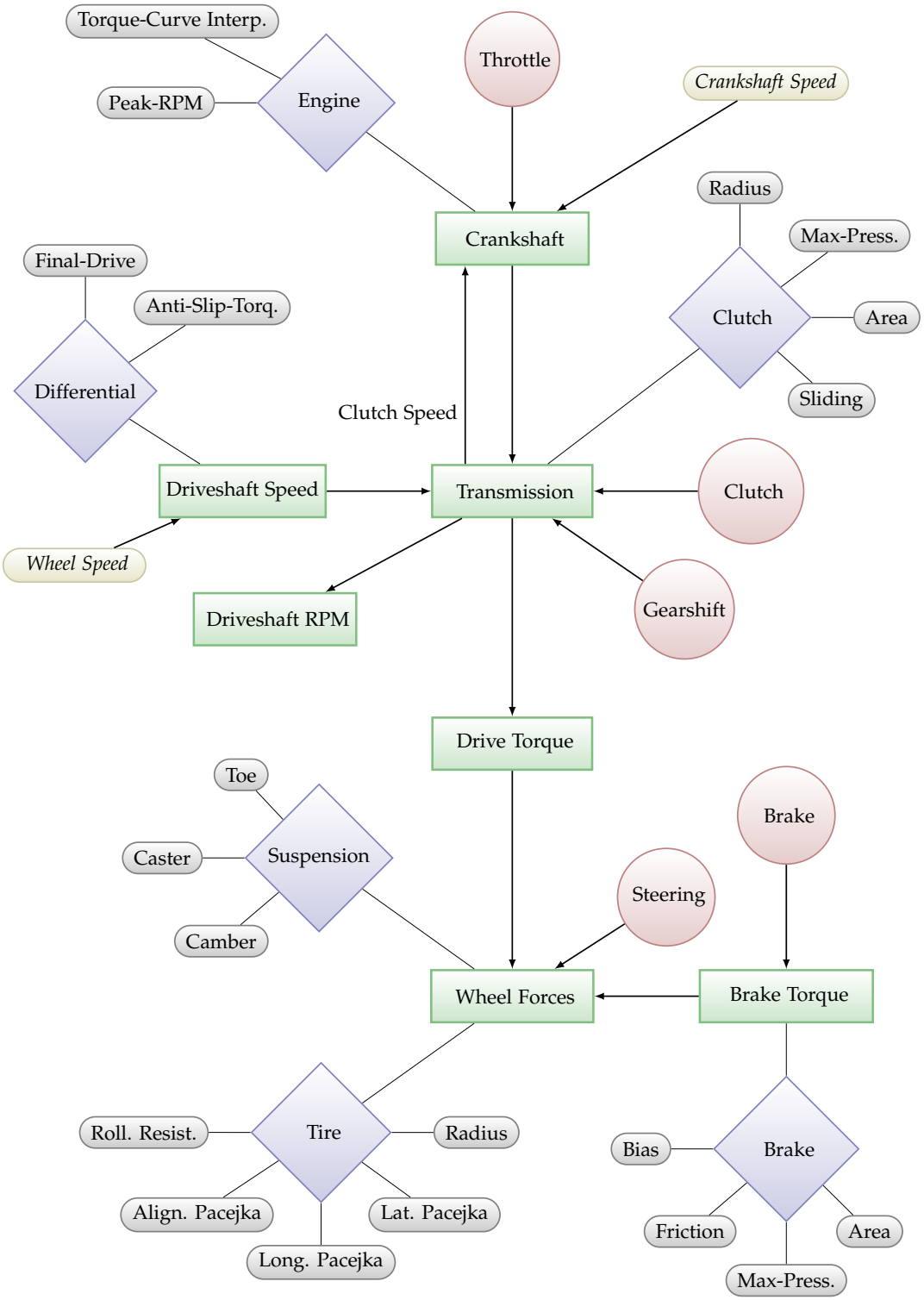


Figure 2.5 – Overview of VDrift's car model

Figure 2.6 – Porsche 911 Club Sport (VDrift website [36])

Figure 2.7 – 2002 Formula 1 car (VDrift website [36])

We use two different cars in the optimization of the test tracks (see Chapter 6). The important parameters for both vehicles, a Porsche 911 Club Sport ("Porsche", Figure 2.6) as well as a Formula 1 car ("F1-02", Figure 2.7), are listed in the following relevant tables. The remaining parameters can be seen in Appendix A.

The Porsche CS, built in the years 1987 till '89, is a licensed street car, but the CS-series come with a racing engine. The F1-02 is a realistic version of 2002's Formula One season, with values according to the official *FIA Regulations*, like weight distributions or tire restrictions.

Control	Range Porsche	Range F1-02	Unit	Description
ϕ	$[0, 1]$	$[0, 1]$	–	Accelerator pedal position
χ	$[0, 1]$	$[0, 1]$	–	Clutch (engaged at 1)
ξ	$[0, 1]$	$[0, 1]$	–	Brake pedal position
μ	$\{1, \ldots, 5\}$	$\{1, \ldots, 7\}$	–	Selected gear
δ	$[-1, 1]$	$[-1, 1]$	–	Steering ratio to max. angle
δ_{max}	$[-33.19, 33.19]$	$[-42, 42]$	deg	Max. steering wheel angle

Table 2.1 – Controls used in the VDrift model.

Engine To accelerate a car by pressing the throttle, initially gas is put into a piston[2] and ignited. The amount of energy, which is released in such a small space in form of expanding gas, can be transmitted by the crankshaft from linear motion of the pistons into rotational motion. The engine is controlled by accelerator pedal position ϕ (Listed in Table 2.1 with all the other control values).

[2] A piston is a cylindrical piece of metal that moves up and down inside the cylinder

Parameter	Value Porsche	Value F1-02	Unit	Description
$P_{\max}$	$1.72 \cdot 10^5$	633844	W	Max. power
I	0.35	0.2	kg m^2	Moment of inertia
ν_{p}	$5.50 \cdot 10^3$	$1.90 \cdot 10^4$	$\frac{1}{\min}$	Peak engine rpm
ν_{u}	$6.84 \cdot 10^3$	$1.95 \cdot 10^4$	$\frac{1}{\min}$	Upper rpm-limit
ν_{l}	$4.00 \cdot 10^2$	$3.50 \cdot 10^3$	$\frac{1}{\min}$	Lower (stall) rpm-limit

Table 2.2 – Engine parameters used at the crankshaft.

Total engine torque M_{E} is applied to the crankshaft from combustion M_{cb}, internal friction M_{fr}, and the clutch M_{Cl}:

$$M_{\mathrm{E}} = M_{\mathrm{cb}} + M_{\mathrm{fr}} - M_{\mathrm{Cl}}. \tag{2.2}$$

To compute the accelerating combustion torque, a cubic spline interpolates torque curve $g(\nu_{\mathrm{E}})$ at the current number of revolutions-per-minute (*rpm*) of the crankshaft (See control values for interpolation in Appendix Table A.1). Combined with the throttle it results in the combustion torque

$$M_{\mathrm{cb}} = \phi \; g(\nu_{\mathrm{E}}). \tag{2.3}$$

The engine's number of rpm comes with its angular velocity ω_{E}, implemented in VDrift as a modified Euler method (a survey of the general idea can be seen in section Numerical Integration in VDrift's Documentation Wiki [2]). Peak engine speed is achieved at ω_{p},

$$\nu_{\mathrm{E}} = \omega_{\mathrm{E}} \frac{30}{\pi}, \qquad \omega_{\mathrm{p}} = \nu_{\mathrm{p}} \frac{\pi}{30}.$$

This leads to the friction factor f_{E}, which allies with the engine's angular velocity to friction torque

$$f_{\mathrm{E}} = \frac{g(\nu_{\mathrm{p}})}{\omega_{\mathrm{p}}^2}, \tag{2.4}$$

$$M_{\mathrm{fr}} = -1.3 \cdot 10^3 \; \omega_{\mathrm{E}} \; f_{\mathrm{E}} \; (1 - \phi). \tag{2.5}$$

Parameter	Value Porsche	Value F1-02	Unit	Description
f_{Cl}	0.27	0.10	–	Sliding (friction)
p_{Cl}	11079.26	$7.0 \cdot 10^4$	$\frac{\mathrm{N}}{\mathrm{m}^2}$	Max. pressure
R_{Cl}	0.1	0.15	m	Clutch radius
A_{Cl}	0.75	0.90	m^2	Clutch area
i_{g}^1	3.50	4.50	–	Transmission ratio of first gear
i_{g}^2	2.06	3.60	–	Transmission ratio of second gear
i_{g}^3	1.41	3.00	–	Transmission ratio of third gear
i_{g}^4	1.13	2.60	–	Transmission ratio of fourth gear
i_{g}^5	0.89	2.30	–	Transmission ratio of fifth gear
i_{g}^6	–	2.05	–	Transmission ratio of sixth gear
i_{g}^7	–	1.90	–	Transmission ratio of seventh gear
i_{t}	3.88	3.7	–	Differential transmission ratio

Table 2.3 – Clutch and transmission parameters

Before we are able to calculate the total engine torque subject to the throttle position, current driving speed is necessary for backward computation of the clutch's angular velocity via driveshaft. Assuming a rear wheel drive, the current driving speed can be computed from the average of the rear wheel's rotational velocity ($\omega_{\mathrm{rl}},\ \omega_{\mathrm{rr}}$), differential ratio, and gear transmission ratio

$$\omega_{\mathrm{D}} = i_{\mathrm{t}}\ \frac{1}{2}(\omega_{\mathrm{rl}} + \omega_{\mathrm{rr}}), \tag{2.6}$$

$$\omega_{\mathrm{Cl}} = \omega_{\mathrm{D}}\ i_{\mathrm{g}}^{\mu}. \tag{2.7}$$

The clutch torque represents the friction at any time one side of the clutch is spinning faster than the other side. The sign of the friction depends on which side is spinning faster. If the engine speed is faster than the driveshaft, the clutch friction will generate torque to slow the engine and accelerate the wheels. Otherwise, if the engine speed is slower than the driveshaft, the clutch friction will generate torque to speed up the engine and slow the wheels[3]:

$$M_{\mathrm{Cl}} = \chi\ (\omega_{\mathrm{E}} - \omega_{\mathrm{Cl}})\ (A_{\mathrm{Cl}}\ R_{\mathrm{Cl}} \cdot f_{\mathrm{Cl}}\ p_{\mathrm{Cl}}). \tag{2.8}$$

The actual force transmitted by resulting total engine torque, over driveshaft M_{Dr}, to the wheel, can be computed with the current gear transmission ratio. Finally this leads

[3]Happens every time you shift to a lower gear

to the drive torque at rear right and left wheel M_{rr}, M_{rl} for a rear wheel driven car

$$M_{\mathrm{Dr}} = M_{\mathrm{E}}\, i_{\mathrm{g}}^{\mu}, \tag{2.9}$$

$$M_{\mathrm{rr}} = \frac{1}{2} i_{\mathrm{t}}\, M_{\mathrm{Dr}}. \tag{2.10}$$

The maximum brake torque is counteractive at this point. Calculated with parameters of Table 2.4 and acting on the one hand at the front wheels as

$$M_{\mathrm{Bf}} = A_{\mathrm{Bf}}\, R_{\mathrm{Bf}} \cdot f_{\mathrm{Bf}}\, \rho_{\mathrm{f}}\, p_{\mathrm{Bf}} \tag{2.11}$$

and on the other as rear wheel torque M_{Br}, applied accordingly with its rear parameters. To get the current brake torque, the input control value of the brake pedal[4] is required. In addition to effective rolling radius R_{e}[5] braking force F_{B} at front and rear wheels is

$$F_{\mathrm{Bf}} = \frac{M_{\mathrm{Bf}}\, \xi}{R_{\mathrm{ef}}}, \qquad F_{\mathrm{Br}} = \frac{M_{\mathrm{Br}}\, \xi}{R_{\mathrm{er}}}. \tag{2.12}$$

Parameter	Value Porsche	Value F1-02	Unit	Description
f_{Bf}	1.13	1.4	–	Friction of front brake
f_{Br}	1.13	1.4	–	Friction of rear brake
p_{Bf}	$2.0 \cdot 10^6$	$3.0 \cdot 10^6$	$\frac{\mathrm{N}}{\mathrm{m}^2}$	Max. pressure
p_{Br}	$2.0 \cdot 10^6$	$3.0 \cdot 10^6$	$\frac{\mathrm{N}}{\mathrm{m}^2}$	
ρ_{f}	0.57	0.60	–	Fraction front
ρ_{r}	0.43	0.40	–	to rear brake
R_{Bf}	0.14	0.15	m	Radius front brake disc
R_{Br}	0.14	0.14	m	Radius rear brake disc
A_{Bf}	$1.5 \cdot 10^{-2}$	$2.5 \cdot 10^{-2}$	m^2	Brake area front
A_{Br}	$1.5 \cdot 10^{-2}$	$1.5 \cdot 10^{-2}$	m^2	Brake area rear

Table 2.4 – Brake parameters

Tire Forces Due to the complexity of tire modeling, there are different approaches to use Pacejka's *Magic Formula*, at full length shown in [25]. We will simply give an overview of the composition of different tire force equations used in VDrift. For a summary see Beckman [3], who picked up the basic ideas of Genta [14]. All parameters are listed in Table 2.5 and Appendix A.

[4]Handbrake is also available in VDrift, but neglected in this thesis

[5]Radius dependent on velocity

Parameter	Value Porsche	Value F1-02	Unit	Description
γ_f	-1.0	-1.0	deg	Camber angle at front suspension
γ_r	-0.5	-0.5	deg	Camber angle at rear suspension
R_f	0.31265	0.33	m	Front tire radius
R_r	0.3179	0.33	m	Rear tire radius

Table 2.5 – Suspension and tire parameters

For computation of **longitudinal force** F_x, two variables are necessary: The dynamic parameter F_z, which is the normal load in kilo-newtons on the tire, as well as the instantaneous slip angle κ[6], given as

$$\kappa = \frac{\omega R_e - v}{v}, \tag{2.13}$$

with the wheel's angular velocity ω, current wheel speed v with respect to the ground and effective tire rolling radius R_e. See the resulting curve for longitudinal F_x (as well as lateral force F_y) with different loads in Figure 2.2.

The friction factor $f_s \in [0, 1]$ is a simplified way to model friction, depending on what surface the car is driving on. Maximum grip is reached at 1. If the car is off the road on a grass track surface, for example, it could be set to 0.5. Further corresponding parameters of Porsche CS and F1-02 are listed in Table A.4 and A.5.

Longitudinal force F_x:

$$F_x = D \ \sin\Big(b_0 \ \arctan\Big(SB + E\big(\arctan(SB) - SB\big)\Big)\Big) \tag{2.14a}$$

$$B = \frac{(b_3 \ F_z + b_4) \ e^{-b_5 F_z}}{(b_1 \ F_z + b_2) \ b_0} \tag{2.14b}$$

$$D = (b_1 \ F_z + b_2) \ F_z \ f_s \tag{2.14c}$$

$$E = b_6 \ F_z^2 + b_7 \ F_z + b_8 \tag{2.14d}$$

$$S = 100 \ \kappa + b_9 \ F_z + b_{10} \tag{2.14e}$$

[6]Variable name from Pacejka [25], varying in Beckman [3] and Genta [14], where it is denoted by σ

Lateral force F_y with side slip angle α as input variable and camber angle γ. Remaining parameters are listed in A.2 and A.3:

$$F_\mathrm{y} = D\ \sin\Big(a_0\ \arctan\Big(SB + E\big(\arctan\left(SB\right) - SB\big)\Big)\Big) + S_v \tag{2.15a}$$

$$B = \frac{a_3\ \sin\Big(2\arctan\Big(\frac{F_\mathrm{z}}{a_4}\Big)\Big)\ (1 - a_5|\gamma|)}{a_0\ (a_1\ F_\mathrm{z} + a_2)\ F_\mathrm{z}} \tag{2.15b}$$

$$D = (a_1\ F_\mathrm{z} + a_2)\ F_\mathrm{z}\ f_\mathrm{s} \tag{2.15c}$$

$$E = a_6\ F_\mathrm{z} + a_7 \tag{2.15d}$$

$$S = \alpha + a_8\ \gamma + a_9\ F_\mathrm{z} + a_{10} \tag{2.15e}$$

$$S_\mathrm{v} = \big((a_{11}\ F_\mathrm{z} + a_{12})\ \gamma + a_{13}\big)\ F_\mathrm{z} + a_{14} \tag{2.15f}$$

Aligning moment M_z is illustrated in Figure 2.3, example parameters specified in Table A.6 and A.7:

$$M_\mathrm{z} = D\ \sin\Big(c_0\ \arctan\Big(SB + E\big(\arctan\left(SB\right) - SB\big)\Big)\Big) + S_v \tag{2.16a}$$

$$B = \frac{\big(c_3\ F_\mathrm{z}^2 + c_4\ F_\mathrm{z}\big)\ (1 - c_6\ |\gamma|)\ e^{-c_5\ F_\mathrm{z}}}{c_0\ D} \tag{2.16b}$$

$$D = (c_1\ F_\mathrm{z} + c_2)\ F_\mathrm{z}\ f_\mathrm{s} \tag{2.16c}$$

$$E = \big(c_7\ F_\mathrm{z}^2 + c_8\ F_\mathrm{z} + c_9\big)\ (1 - c_{10}\ |\gamma|) \tag{2.16d}$$

$$S = \alpha + c_{11}\ \gamma + c_{12}\ F_\mathrm{z} + c_{13} \tag{2.16e}$$

$$S_\mathrm{v} = \big(c_{14}\ F_\mathrm{z}^2 + c_{15}\ F_\mathrm{z}\big)\ \gamma + c_{16}\ F_\mathrm{z} + c_{17} \tag{2.16f}$$

2.4 Testdrive

A mixed-integer optimal control problem with its origin in automobile test driving called *testdrive* was published by Gerdts in 2005, 2006 [15, 16]. The original benchmark problem involves discrete controls for the choice of gears. Our approach is based on a convexification and relaxation of the integer control constraint.

We will introduce the original *testdrive* car model as presented by Kirches in [22], in which an Outer Convexification (see Section 4.4.2) is used in comparison to an inner convexification of Gerdts [15]. Thereafter, the subsequent section shows the actual modifications of the *testdrive* car model and ODE system that have been made in the context of this thesis.

2.4.1 Testdrive Car Model

We consider a single-track model, derived under the simplifying assumption that rolling and pitching of the car body can be neglected. Consequentially, only each one single front and rear wheel is modeled, located in the virtual center of the original two wheels. Motion of the car body is considered on the horizontal plane only.

Control	Range	Unit	Description
ω_δ	$[-0.5, 0.5]$	$\frac{\text{rad}}{\text{s}}$	Steering wheel angular velocity
F_B	$[0, 1.5 \cdot 10^4]$	N	Total braking force
ϕ	$[0, 1]$	–	Accelerator pedal position
μ	$\{1, \dots, 5\}$	–	Selected gear

Table 2.6 – Controls used in the car model.

The model includes four control values for the vehicle, which are the steering wheel's angular velocity ω_δ and the total braking force F_B. Additionally, the effective engine torque's transmission ratio is controlled by the selected gear μ, which is in conjunction with the accelerator pedal position ϕ translated into an accelerating force. All controls are listed in Table 2.6.

The equations of motion are described by a system of ODEs. The individual system states are listed in Table 2.7, while Figure 2.8 visualizes the choice of coordinates, angles and forces.

State	Unit	Description
c_x	m	Horizontal position of the car
c_y	m	Vertical position of the car
v	$\frac{\text{m}}{\text{s}}$	Magnitude of directional velocity of the car
δ	rad	Steering wheel angle
β	rad	Side slip angle
ψ	rad	Yaw angle
ω_z	$\frac{\text{rad}}{\text{s}}$	Yaw angle velocity

Table 2.7 – Coordinates and states used in the car model.

The coordinate pair (c_x, c_y) denotes the position of the car's center of gravity on the horizontal plane. Then, let v be the magnitude of the car's velocity in the direction of driving. The steering wheel angle (the angle of the front wheel with respect to the general orientation of the car's longitudinal axis) is named δ, while the slip angle β

gives the angle of the car's direction of movement against that axis. Finally, ψ and ω_z denote the car's yaw angle and its angular velocity, representing the orientation of the car's longitudinal axis against the horizontal coordinate axis.

The center of gravity is denoted by the coordinate pair (c_x, c_y) which is obtained by integration over the directional velocity

$$\dot{c}_x(t) = v(t)\ \cos\big(\psi(t) - \beta(t)\big), \tag{2.17}$$
$$\dot{c}_y(t) = v(t)\ \sin\big(\psi(t) - \beta(t)\big). \tag{2.18}$$

Acceleration is obtained from the sum of forces attacking the car's mass m in the direction of driving

$$\begin{aligned}\dot{v}(t) = \frac{1}{m}\Big(&\big(F_{lr}^{\mu} - F_{Ax}\big)\ \cos\beta(t) + F_{lf}\ \cos\big(\delta(t) + \beta(t)\big) \\ &- \big(F_{sr} - F_{Ay}\big)\ \sin\beta(t) - F_{sf}\ \sin\big(\delta(t) + \beta(t)\big)\Big).\end{aligned} \tag{2.19}$$

The steering wheel's angle is given by the corresponding controlled angular velocity

$$\dot{\delta}(t) = \omega_\delta. \tag{2.20}$$

The slip angle's change is controlled by the steering wheel and counteracted by the sum of forces attacking perpendicular to the car's direction of driving

$$\begin{aligned}\dot{\beta}(t) = \omega_z(t) - \frac{1}{m\ v(t)}\Big(&\big(F_{lr} - F_{Ax}\big)\ \sin\beta(t) + F_{lf}\ \sin\big(\delta(t) + \beta(t)\big) \\ &+ \big(F_{sr} - F_{Ay}\big)\ \cos\beta(t) + F_{sf}\ \cos\big(\delta(t) + \beta(t)\big)\Big).\end{aligned} \tag{2.21}$$

The yaw angle is obtained by integrating over its change ω_z

$$\dot{\psi}(t) = \omega_z(t), \tag{2.22}$$

which in turn is the integral over the sum of forces attacking the front wheel in direction perpendicular to the car's longitudinal axis of orientation

$$\dot{w}_z(t) = \frac{1}{I_{zz}}\big(F_{sf}\ l_f\ \cos\delta(t) - F_{sr}\ l_{sr} - F_{Ay}\ e_{SP} + F_{lf}\ l_f\ \sin\delta(t)\big). \tag{2.23}$$

In the following we describe the engine model and explain the resulting forces translated to the tire and used in the ODE system.

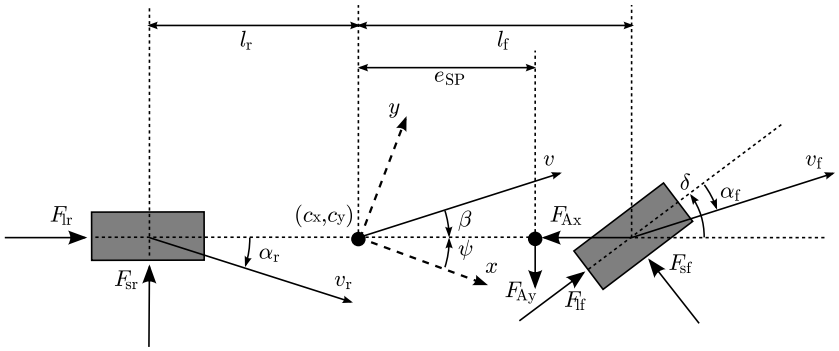


Figure 2.8 – Coordinates and forces in the single-track car model (Gerdts [16]).

Engine

The engine's rotary frequency ν^{μ}_{mot} in Hertz can be computed from the car's speed. We denote the gearbox transmission ratio corresponding to the selected gear μ by i^{μ}_{g} and the axle drive's fixed transmission ratio by i_{t}. Assuming a rear wheel drive, hence for a rear wheel radius R and a given gear μ, the rotary frequency is:

$$\nu^{\mu}_{\text{mot}} := \frac{i^{\mu}_{\text{g}}\, i_{\text{t}}}{R}\, v(t). \tag{2.24}$$

The effective engine torque M^{μ}_{mot}, depending on the acceleration pedal's position ϕ, is modeled as follows:

$$M^{\mu}_{\text{mot}}(\phi) := f_1(\phi)\; f_2(\nu^{\mu}_{\text{mot}}), + \Big(1 - f_1(\phi)\Big)\; f_3(\nu^{\mu}_{\text{mot}}), \tag{2.25}$$

$$f_1(\phi) := 1 - \exp(-3\;\phi), \tag{2.26}$$

$$f_2(\nu_{\text{mot}}) := -3.78 \cdot 10^1 + 1.54\;\nu_{\text{mot}} - 1.9 \cdot 10^{-3}\;\nu^2_{\text{mot}}, \tag{2.27}$$

$$f_3(\nu_{\text{mot}}) := -3.49 \cdot 10^1 - 4.775 \cdot 10^{-2}\;\nu_{\text{mot}}. \tag{2.28}$$

Tire Forces

The **longitudinal force** at the rear wheel is given directly by the transmitted engine torque and reduced by braking force F_{Br} and resistance due to rolling friction F_{Rr}

$$F^{\mu}_{\text{lr}} := \frac{i^{\mu}_{\text{g}}\, i_{\text{t}}}{R}\; M^{\mu}_{\text{mot}}(\phi) - F_{\text{Br}} - F_{\text{Rr}}. \tag{2.29}$$

Whereas the longitudinal force at the front wheel is simply composed of braking force and rolling friction

$$F_{\text{lf}} := -F_{\text{Bf}} - F_{\text{Rf}}. \tag{2.30}$$

Parameter	Value	Unit	Description
m	$1.239 \cdot 10^3$	kg	Mass of the car
g	9.81	$\frac{\mathrm{m}}{\mathrm{s}^2}$	Gravity constant
l_f	1.19016	m	Front wheel distance to center of gravity
l_r	1.37484	m	Rear wheel distance to center of gravity
R	0.302	m	Wheel radius
I_{zz}	$1.752 \cdot 10^3$	kg m^2	Moment of inertia
c_w	0.3	–	Air drag coefficient
ρ	1.249512	$\frac{\mathrm{kg}}{\mathrm{m}^3}$	Air density
A	1.4378946874	m^2	Effective flow surface
i_g^1	3.09	–	Transmission ratio of first gear
i_g^2	2.002	–	Transmission ratio of second gear
i_g^3	1.33	–	Transmission ratio of third gear
i_g^4	1.0	–	Transmission ratio of fourth gear
i_g^5	0.805	–	Transmission ratio of fifth gear
i_t	3.91	–	Differential transmission ratio
B_f	$1.096 \cdot 10^1$	–	Pacejka coefficients (stiffness)
B_r	$1.267 \cdot 10^1$	–	
C_f	1.3	–	Pacejka coefficients (shape)
C_r	1.3	–	
D_f	$4.5604 \cdot 10^3$	–	Pacejka coefficients (peak)
D_r	$3.94781 \cdot 10^3$	–	
E_f	-0.5	–	Pacejka coefficients (curvature)
E_r	-0.5	–	

Table 2.8 – Parameters used in the original *testdrive* car model.

The total braking force F_B is controlled by the driver. For its distribution to front and rear wheels we choose:

$$F_\mathrm{Bf} := \frac{2}{3}\, F_\mathrm{B}, \qquad\qquad F_\mathrm{Br} := \frac{1}{3}\, F_\mathrm{B}. \tag{2.31}$$

The braking forces F_Rf and F_Rr due to rolling resistance are obtained from

$$F_\mathrm{Rf}(v) := f_\mathrm{R}(v)\, \frac{m\, l_\mathrm{r}\, g}{l_\mathrm{f} + l_\mathrm{r}}, \qquad\qquad F_\mathrm{Rr}(v) := f_\mathrm{R}(v)\, \frac{m\, l_\mathrm{f}\, g}{l_\mathrm{f} + l_\mathrm{r}}, \tag{2.32}$$

where the velocity-dependent amount of friction is modeled by

$$f_\mathrm{R}(v) := 9.0 \cdot 10^{-3} + 7.2 \cdot 10^{-5}\, v + 5.038848 \cdot 10^{-10}\, v^4. \tag{2.33}$$

The **side (lateral) force** on the front wheel as a function of the front slip angle α_f is calculated by Pacejka's *Magic Formula* (see Section 2.2), with the fixed Pacejka parameters of Table 2.8:

$$F_{sf}(\alpha_f) := D_f \ \sin\Big(C_f \ \arctan\big(B_f \ \alpha_f - E_f(B_f \ \alpha_f - \arctan(B_f \ \alpha_f))\big)\Big) \tag{2.34}$$

The front slip angle itself is obtained from

$$\alpha_f := \delta(t) - \arctan\left(\frac{l_f \ \dot{\psi}(t) - v(t) \ \sin\beta(t)}{v(t) \ \cos\beta(t)}\right). \tag{2.35}$$

The same approximation is used for the lateral force at the rear wheel depending on the rear slip angle α_r

$$F_{sr}(\alpha_r) := D_r \ \sin\Big(C_r \ \arctan\big(B_r \ \alpha_r - E_r(B_r \ \alpha_r - \arctan(B_r \ \alpha_r))\big)\Big) \tag{2.36}$$

with the rear slip angle being

$$\alpha_r := \arctan\left(\frac{l_r \ \dot{\psi}(t) + v(t) \ \sin\beta(t)}{v(t) \ \cos\beta(t)}\right). \tag{2.37}$$

The **aligning moment** is being neglected in this model. Furthermore we assume that there is no sidewards drag force (e.g. side wind) present, but drag force due to air resistance is given by

$$F_{Ax} := \frac{1}{2} \ c_w \ \rho \ A \ v^2(t), \qquad F_{Ay} := 0. \tag{2.38}$$

2.5 Extended Testdrive Model

The previous sections illustrate the mathematical model of the original *testdrive* example. It allows to compute an optimal control of a vehicle with gear shift on a fixed track[7]. However, the embedded set of ODEs is only applicable for calculations on tracks which are built in one direction along the x-axis[8]. Hence, for optimization of an optional track while coupling *testdrive* to VDrift, modifications in the ODE system, as well as a transformation of the coordinate system are required. Our approach is to change the time-dependent set of ODEs to a system dependent on a car's position to the midline of the track. So the first step is to modify the ODEs and afterwards essentially transform the coordinate system in the way that the midline of the track is mapped to the x-axis of the coordinate system (described in Section 3).

[7] *testdrive* track course presented in Section 3.1

[8] With maximum variance of less than 90 degrees

In addition, we tried to attach specific car parameters and parts of VDrift's tire and engine model. Therefore, the *extended testdrive* version is now closer to the "real" world, which in this case is designed by VDrift. One major improvement is the possibility to select an optional car from VDrift with its realistic parameters. The accuracy of the lateral force F_y improved due to a greater extent of Pacejka parameters. Furthermore, the engine model has been rebuilt with car specific torque curve and transmission ratio. The 2-D model expands up to three dimensions, so one can drive not only on even tracks.

2.5.1 Transformation of ODE System

We denote the state vector of the ODE system by x and the corresponding right-hand side function by f in the original *testdrive* example. The continuous controls are described in vector u. The separate vector μ contains the integer controls.

$$x := \Big(c_{\mathrm{x}}, c_{\mathrm{y}}, v, \delta, \beta, \psi, \omega_{\mathrm{z}}\Big)^{\mathrm{T}}, \qquad u := \Big(\omega_\delta, F_{\mathrm{B}}, \phi\Big)^{\mathrm{T}}, \qquad \mu \in \{1, \ldots, 5\}.$$

The time-dependent differential equation reads as

$$\frac{dx}{dt} = f_t\Big(t, x(t), u(t), \mu(t)\Big). \tag{2.39}$$

σ denotes the independent variable in the new set of ODEs as current advance on the midline. We transform Equation 2.39 by multiplying a factor, which can also be seen as the inverse of the magnitude of the car's velocity v:

$$\frac{dt}{d\sigma} = \frac{1}{v} \tag{2.40}$$

This leads to a position-dependent system, in which every state of the previous system has to be divided by v:

$$\frac{dx}{d\sigma} = f_\sigma\left(\sigma, x(\sigma) \cdot \frac{1}{v(\sigma)}, u(\sigma), \mu(\sigma)\right) \tag{2.41}$$

The center of gravity of the vehicle in the original *testdrive* example is denoted by (c_x, c_y). So far c_x specifies the horizontal position of the car. Due to a transformation of the coordinate system, illustrated in Chapter 3.3, the midline of the track is mapped to the horizontal x-axis. This means that after the modification, the advance on the midline equals the advance in horizontal direction. Accordingly the state c_x drops out after the ODE transformation, because it equals σ now.

$$\frac{dc_x}{dt} \hat{=} \frac{d\sigma}{dt} \qquad \Rightarrow \qquad \frac{d\sigma}{dt}\,\frac{dt}{d\sigma} = 1$$

The current deviation of the midline is labeled with $d(\sigma)$, which equals c_y in the old coordinate system. To compute $d(\sigma)$ for an optional track course, we have to reduce the yaw angle ψ by the "curvature" of the track course at the current car position. For that purpose the dimension of the track data is decreased by one. Then the curvature of the track course is computed by the *arc tangent 2* operating on a trajectory, which is represented by the midline of the track. One can imagine this trajectory by looking from above on the track course. As desired, this means that the difference $(\psi - curv)$ is now representing the orientation of the car's longitudinal axis against the midline of the track.

$$d'(\sigma) = \sin\Big(\psi(\sigma) - curv(\sigma) - \beta(\sigma)\Big) \tag{2.42}$$

Finally, we have to add the time to the state vector in the new ODE system:

$$t'(\sigma) = \frac{dt}{dt} \cdot \frac{dt}{d\sigma} = \frac{1}{v} \tag{2.43}$$

2.5.2 Modifications in Car Model

Most of the car specific parameters of a chosen VDrift car are now assumed by *testdrive*. Parameters like mass of the car, front or rear wheel drive, measurement data, or air drag coefficient are added to the *testdrive* car model.

Controls We changed the control of the maximum braking force, as described in *testdrive* Equation 2.31, to a control of the braking pedal position $\xi \in [0, 1]$ like presented in VDrift's car model. Therefore, we calculated a car specific maximum braking torque (See Equation 2.11[9]) with the parameters of Table 2.4 and multiplied the brake pedal position (Equations 2.12) for the resulting braking force at front and rear wheels.

Due to the fact that a Formula 1 has seven possibilities for the choice of gears, the extended version of *testdrive* is able to model a car with an optional number of gears.

The state and control vectors of the new set of ODEs are

$$x := \Big(d, v, \delta, \beta, \psi, \omega_z, t\Big)^T, \qquad u := \Big(\omega_\delta, \xi, \phi\Big)^T, \qquad \mu \in \{1, \ldots, 5\}.$$

Note that $\mu \in \{1, \ldots, 7\}$ for the Formula One car F1-02.

Engine Model The engine model of *testdrive* is fixed to a certain car type. However, our intention is the possibility to use different car models of VDrift in the *extended*

[9]Though with fixed tire radius instead of effective rolling radius

testdrive version. The attached engine parameters of the PORSCHE CS worked reasonably well in the old engine model. But as shown in Figure 2.9, this engine model in combination with the transmission ratios of the F1-02 creates an unrealistic accelerating force, with respect to a Formula One car. This is caused by a much higher number of revolutions per minute of the Formula One car in comparison to the original *testdrive* engine model. Hence, we decided to include the engine model of VDRIFT to the *testdrive* optimal control problem.

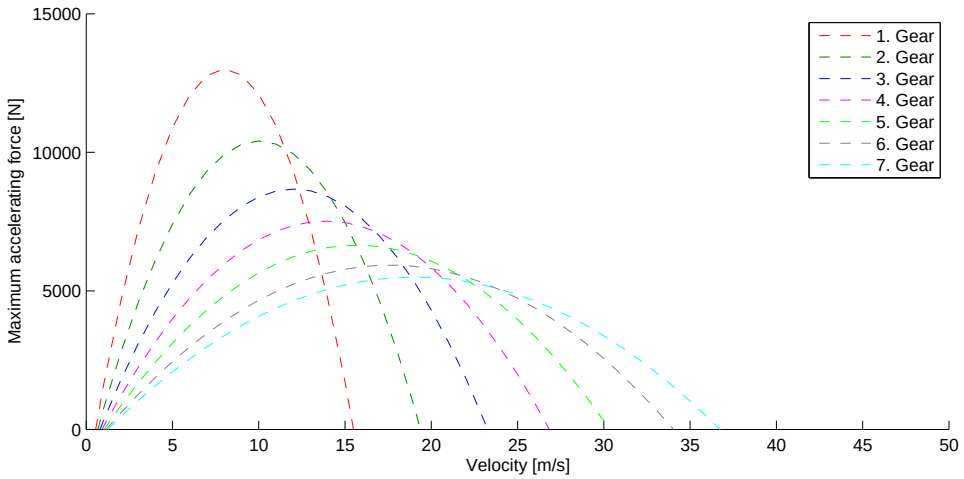


Figure 2.9 – Maximum acceleration force of the original *testdrive* engine model with transmission ratio of the F1-02.

Total engine torque is now applied by combustion and friction torque as described in VDRIFT's car model, Equation 2.3 and 2.5, with a car specific torque curve and engine parameters. The clutch torque is being neglected in this context, due to the fact that an additional state in the ODE system would have been necessary for its calculation.

However, one problem appears in the conjunction with the included torque curve. Within VDRIFT, the engine constraints are formulated as upper and lower bounds of the engine's number of revolutions per minute. This is easily realized by an "if query" formulation in VDRIFT's source code. Therefore, the torque curve data is only formulated within these bounds. For this purpose, see the gear independent torque curve of PORSCHE's engine in Figure 2.10 and for the F1-02 engine in Figure 2.11.

The difficulties raise with the new *testdrive* optimal control problem, in the way that the torque curve has to be continuously differentiable beyond the upper and lower constraints. This is caused by the fact that the curve is embedded in the ODE system's right hand side. Additionally, the engine constraints are not activated until the engine torque is computed.

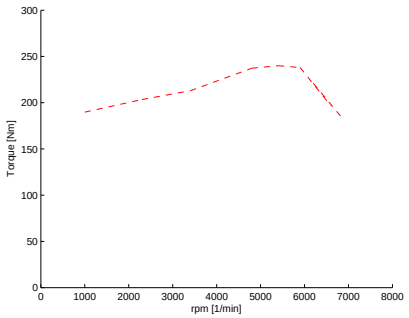


Figure 2.10 – Engine revolution dependent torque curve of the Porsche 911 Club Sport of VDrift

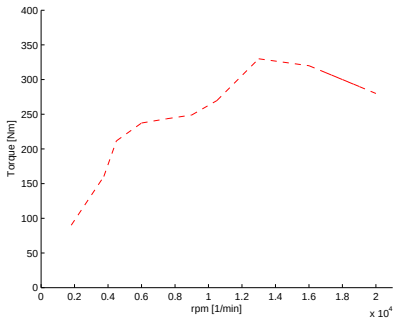


Figure 2.11 – Engine revolution dependent torque curve of a 2002 Formula 1 car of VDrift

Thus, we extended the torque curve, which is given as two-dimensional point data and interpolated with Bézier splines (explained in Section 3.2), by adding further points at both ends. The gear dependent maximum acceleration force of the F1-02 with respect to velocity, is illustrated in Figure 2.12. It is based on the new *testdrive* engine model. The vertical lines are representing the upper and lower gear addicted engine constraints. Furthermore, the dashed/dotted lines describe the original VDrift torque curve, whereas the simply dashed lines show the required extensions of the curve.

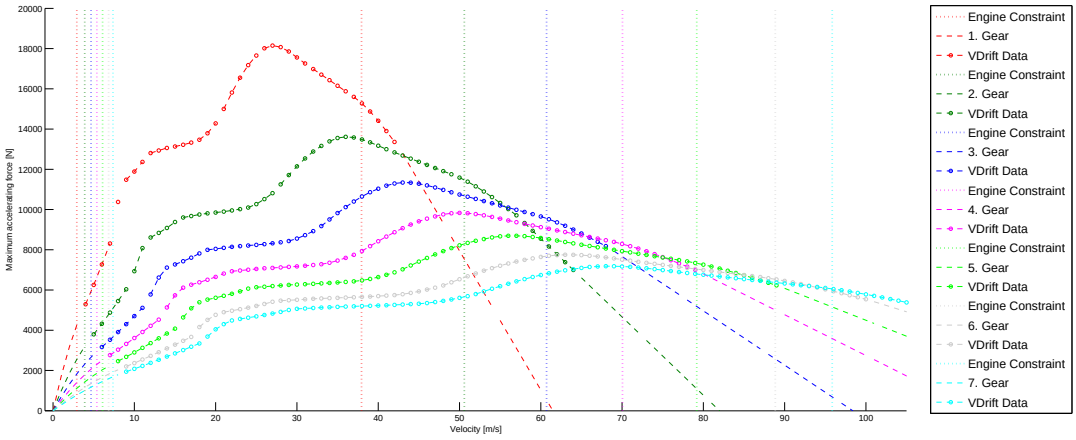


Figure 2.12 – Maximum acceleration force of the new F1-02 engine model with respect to velocity. Included from VDrift, illustrating the upper and lower engine constraints of every gear.

The acceleration force of the PORSCHE CS, including the new engine model as well, is shown in Figure 2.13.

Summing up, the new engine model produces significantly more realistic results for both presented vehicles. More information about the optimal control problem, the set of ODEs, as well as the formulation of the engine constraints is given in Chapter 4.

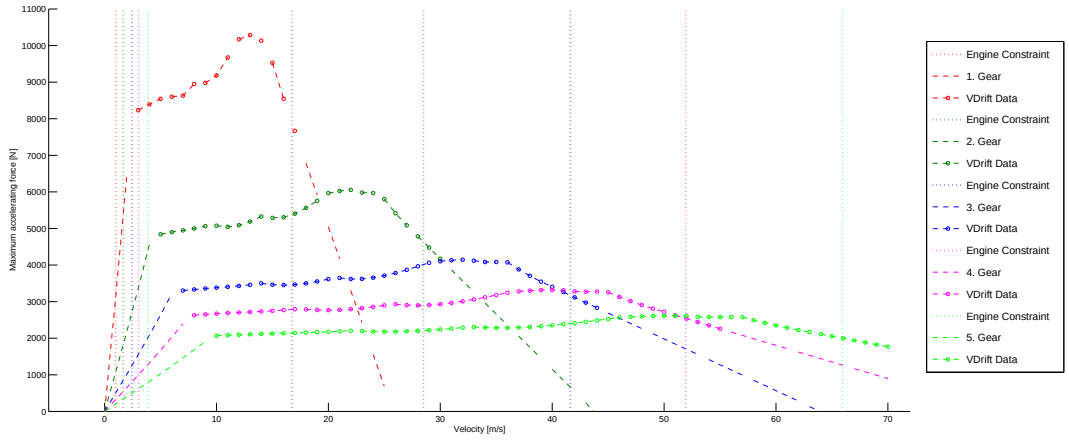


Figure 2.13 – Maximum acceleration force of the new PORSCHE CS engine model with respect to velocity. Included from VDRIFT, illustrating the upper and lower engine constraints of every gear.

Tire Model The original *testdrive* uses fixed Pacejka parameters for B, C, D, E (see Table 2.8), as well as the side slip angle of Equation 2.35 to get the lateral tire force. In the extended version we implemented a tire specific calculation like in VDRIFT (see Equation 2.15), depending on varying Pacejka parameters for each vehicle (Appendix A) and the normal load:

$$F_{\text{zf}} = \frac{1}{1000} \frac{m\, l_{\text{r}}\, g\, \cos\big(\arctan(\text{slope})\big)}{l_{\text{f}} + l_{\text{r}}} \tag{2.44}$$

$$F_{\text{zr}} = \frac{1}{1000} \frac{m\, l_{\text{f}}\, g\, \cos\big(\arctan(\text{slope})\big)}{l_{\text{f}} + l_{\text{r}}} \tag{2.45}$$

The current slope is calculated from the level curve of the track with the aid of Bézier splines, which are presented in detail in the following Chapter 3.

Chapter 3

Coordinate Systems

The embedded coordinate system and the method of calculation of the track course are very important for an exact and realistic track model of a racing simulation. The coordinate system is essential for computation of the current car position and whether the car fulfills track boundary conditions. Our ambition is to change the current test track of *testdrive* to an optional VDrift racing track, according to the modifications in the ODE system of Section 2.5.1.

First of all, we start with a short explanation of the original *testdrive* track. After that, we take a look at the definition of Bézier splines and patches, due to the fact that VDrift uses Bézier patches for calculation of the track data. The last section of this chapter shows how the transformation of the coordinate system is realized. This transformation permits to compute an optimal control using the modified *testdrive* set of ODEs on an optional VDrift racing track.

3.1 Testdrive Track

The test track of *testdrive* (as presented in Gerdts [15] and Kirches [22]) is a standardized driving process in automobile industry called double-lane-change maneuver. It is realized by constraining the car's position onto a prescribed track at any time $t \in [t_0, t_f]$, as seen in Figure 5.1. The driver has to manage to avoid hitting an obstacle suddenly appearing on the starting lane by a change of lanes with an offset of 3.5 meters. Starting in the left position with an initial velocity and ending in the same lane the maneuver can also be regarded as an overtaking move.

From a mathematical point of view, the test track is described by setting up piecewise cubic spline functions $P_l(x)$ and $P_r(x)$ with given horizontal position x. The functions are modeling the top and bottom track boundaries (compare Figure 3.1) and

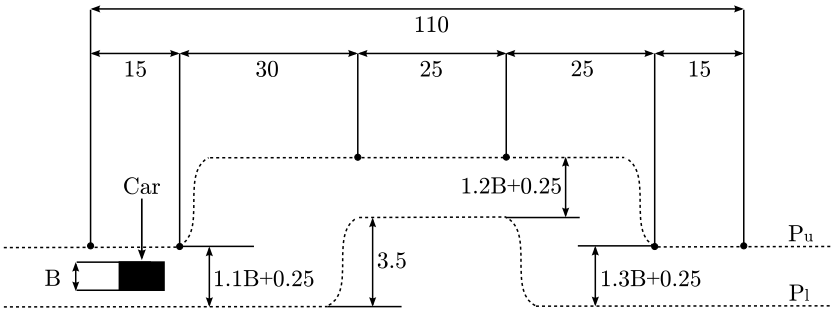


Figure 3.1 – Layout of test course (Gerdts [15]).

read as follows:

$$P_{\mathrm{l}}(x) := \begin{cases} 0 & \text{if} \quad x \leq 44, \\ 4\,h_2\,(x-44)^3 & \text{if} \quad 44 < x \leq 44.5, \\ 4\,h_2\,(x-45)^3 + h_2 & \text{if} \quad 44.5 < x \leq 45, \\ h_2 & \text{if} \quad 45 < x \leq 70, \\ 4\,h_2\,(70-x)^3 + h_2 & \text{if} \quad 70 < x \leq 70.5, \\ 4\,h_2\,(71-x)^3 & \text{if} \quad 70.5 < x \leq 71, \\ 0 & \text{if} \quad 71 < x, \end{cases} \tag{3.1}$$

$$P_{\mathrm{u}}(x) := \begin{cases} h_1 & \text{if} \quad x \leq 15, \\ 4\,(h_3-h_1)\,(x-15)^3 + h_1 & \text{if} \quad 15 < x \leq 15.5, \\ 4\,(h_3-h_1)\,(x-16)^3 + h_3 & \text{if} \quad 15.5 < x \leq 16, \\ h_3 & \text{if} \quad 16 < x \leq 94, \\ 4\,(h_3-h_4)\,(94-x)^3 + h_3 & \text{if} \quad 94 < x \leq 94.5, \\ 4\,(h_3-h_4)\,(95-x)^3 + h_4 & \text{if} \quad 94.5 < x \leq 95, \\ h_4 & \text{if} \quad 95 < x, \end{cases} \tag{3.2}$$

where $B = 1.5$ m is the car's width and

$$h_1 := 1.1\,B + 0.25, \qquad h_2 := 3.5, \qquad h_3 := 1.2\,B + 3.75, \qquad h_4 := 1.3\,B + 0.25.$$

3.2 Bézier Patches in VDrift

The track coordinates in VDrift are embedded as bicubic Bézier points. We briefly introduce Bézier splines and patches in the following section. A more extensive look at Bézier spline interpolation is given in Stoer [34](pages 100–112).

Bézier patches are a surface extension of Bézier curves. One advantage of both polynomial maps is the simple evaluation of optional points on the curve or surface by interpolation. Furthermore, Bézier curves permit a fast run time evaluation.

Definition 3.1. ***Bézier Curve***
A Bézier curve of order n, with given $n+1$ control or Bézier points $(P_i)_{i=0}^n$ is defined as

$$C(u) = \sum_{i=0}^{n} P_i \; B_i^n(u), \tag{3.3}$$

for $u \in [0,1]$, with the Bernstein polynomial

$$B_i^n(u) = \binom{n}{i} \; u^i \; (1-u)^{n-i}. \tag{3.4}$$

In general, composed continuous splines are used for fixed-order curves due to the global influences, while modifying a single point. For a three-dimensional world coordinate system, a **cubic Bézier spline** is given for $u \in [0,1]$ with the following equation:

$$\begin{aligned} C(u) &= \sum_{i=0}^{3} \binom{3}{i} \; P_i \; u^i \; (1-u)^{3-i} \\ &= P_3 u^3 + 3P_2 \; u^2(1-u) + 3P_1 \; u(1-u)^2 + P_0(1-u)^3. \end{aligned} \tag{3.5}$$

To imagine a three-dimensional Bézier patch, one can think of two orthogonal Bézier curves. Each bicubic patch is defined by a set of 16 control points. With these control points you are able to generate any point on the surface.

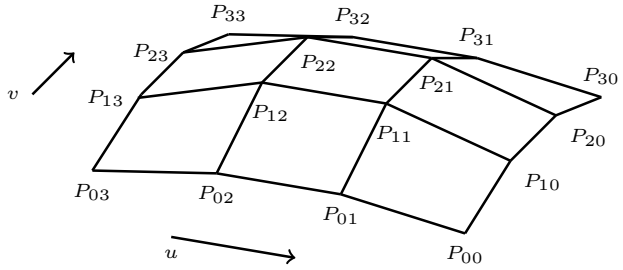


Figure 3.2 – An array of 16 control points creates a bicubic Bézier patch.

Definition 3.2. ***Bézier Patch***
A given Bézier patch of order (n,m) is defined by a set of $(n+1)(m+1)$ control points $P_{i,j}$

$$C(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} P_{i,j} \; B_i^n(u) \; B_j^m(v) \tag{3.6}$$

with the Bernstein polynomials $B_i^n(u)$ and $B_j^m(v)$ as described in equation 3.4.

To obtain $P_0, \ldots, P_3$ with the given control points, one has to choose $u \in [0, 1]$. An additional $v \in [0, 1]$ then produces any desired position on the surface.

$$P_0 = P_{30}u^3 + 3P_{20}\, u^2(1-u) + 3P_{10}\, u(1-u)^2 + P_{00}(1-u)^3 \tag{3.7a}$$
$$P_1 = P_{31}u^3 + 3P_{21}\, u^2(1-u) + 3P_{11}\, u(1-u)^2 + P_{01}(1-u)^3 \tag{3.7b}$$
$$P_2 = P_{32}u^3 + 3P_{22}\, u^2(1-u) + 3P_{12}\, u(1-u)^2 + P_{02}(1-u)^3 \tag{3.7c}$$
$$P_3 = P_{33}u^3 + 3P_{23}\, u^2(1-u) + 3P_{13}\, u(1-u)^2 + P_{03}(1-u)^3 \tag{3.7d}$$

$$P_{car} = P_3 v^3 + 3P_2\, v^2(1-v) + 3P_1\, v(1-v)^2 + P_0(1-v)^3 \tag{3.8}$$

3.3 Transformations between Coordinate Systems

The three-dimensional track data of the world coordinate system in VDrift (called *CSI* or Coordinate System I), creates an excellent replication of the real world track (compare Figure 3.3 and 3.4).

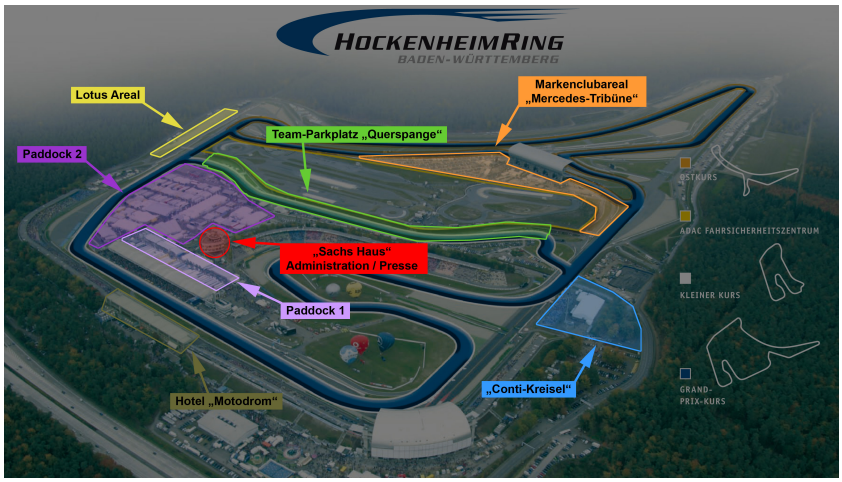


Figure 3.3 – Aerial picture of the *Hockenheimring* (website [19]).

For the ability to compute optimal controls and states of the new position-dependent ODE system presented in Chapter 2.5.1, a transformation of *CSI* is required. Hence, our intention is to map the track's midline onto the x-axis of the coordinate system. Therefore, we calculate the arc length of every patch by setting up a frequency polygon to the midline of the track. After that we are able to transform all patches from *CSI* to a second coordinate system, short *CSII*. The track boundaries can now be regarded as upper and lower constraints, as in the original *testdrive* example in Section 3.1.

Moreover the track course results from the curvature of the two-dimensional track, looking from above. It is calculated with the *arc tangent 2* and interpolated with Bézier

splines. The slope of the track is described by a level curve and implemented as one-dimensional Bézier splines as well.

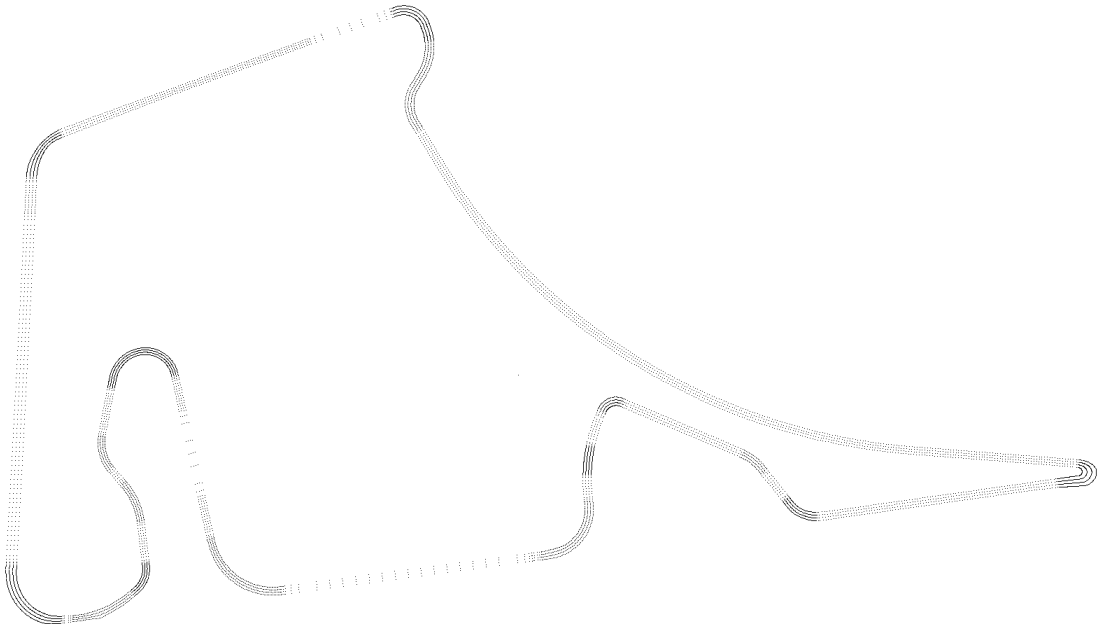


Figure 3.4 – Track data of the *Hockenheimring*, given as Bézier points in VDrift

3.3.1 Locate Car Position with Newton's Method

To translate the current vehicle position between the two different coordinate systems, the patch number as well as car's position data $P_{car}(x, y, z)$ is required. If it is possible to get $u, v \in [0, 1]$ of a given patch in one coordinate system with the input values of P_{car}, the actual car position arises within the other coordinate system from Equation 3.7d and 3.8.

At first, one requires u in longitudinal direction of the patch to compute four control points on this level. Therefore, we take the car position and search for a u that gives the euclidean distance to a point on the track boundaries. Without loss of generality, we use the inner boundary for this purpose and convert Equation 3.7d for easier calculation of the derivatives:

$$\begin{aligned} 3.7d \Leftrightarrow P_3 = (P_{33} - 3P_{23} + 3P_{13} - P_{03})\, u^3 + \\ 3(P_{23} - 2P_{13} + P_{03})\, u^2 + 3(P_{13} - P_{03})\, u + P_{03}. \end{aligned} \tag{3.9}$$

For clarity reasons, we define $a, b, c,$ and d as

$$a := P_{33} - 3P_{23} + 3P_{13} - P_{03}, \tag{3.10a}$$
$$b := 3(P_{23} - 2P_{13} + P_{03}), \tag{3.10b}$$
$$c := 3(P_{13} - P_{03}), \tag{3.10c}$$
$$d := P_{03}, \tag{3.10d}$$

whereas the resulting function with its derivatives reads as follows

$$f(u) := au^3 + bu^2 + cu + d, \tag{3.11a}$$
$$f'(u) = 3au^2 + 2bu + c, \tag{3.11b}$$
$$f''(u) = 6au + 2b. \tag{3.11c}$$

Subsequently with the aid of the Pythagorean theorem, we define a distance function (3.12) to the inner boundary and look for it's minimum

$$d^2(u) := (f(u)_x - x)^2 + (f(u)_z - z)^2. \tag{3.12}$$

Based on monotony, the minimum of d^2 equals the minimum of d. Hence, we set d^2 to $\hat{d}$ for simplification

$$\hat{d}(u) := (f(u)_x - x)^2 + (f(u)_z - z)^2, \tag{3.13a}$$
$$\hat{d}'(u) = 2\,(f(u)_x - x)\,f'(u)_x + 2\,(f(u)_z - z)\,f'(u)_z \stackrel{!}{=} 0. \tag{3.13b}$$

This extremum problem leads to a polynomial F of grade five, which in general cannot be solved analytically. Therefore, we decided to use Newton's method to find the zero point of the polynomial:

$$\Rightarrow F(u) := (f(u)_x - x)\,f'(u)_x + (f(u)_z - z)\,f'(u)_z, \tag{3.14}$$
$$F'(u) = (f(u)_x - x)\,f''(u)_x + f'(u)_x^2 + (f(u)_z - z)\,f''(u)_z + f'(u)_z^2, \tag{3.15}$$
$$u_{n+1} = u_n + \frac{F(u_n)}{F'(u_n)}. \tag{3.16}$$

The same procedure executed in the other direction of the patch, leads to a $v \in [0, 1]$. Thereafter, it is possible to compute the position of the car in *CSII* respectively *CSI*, like shown in Equation 3.7 and 3.8.

Chapter 4

Mixed-Integer Optimal Control Problems

In this chapter we give an overview of the basic mathematical problems that are solved in the context of this thesis. In particular, we focus on a *mixed-integer optimal control problem* (MIOCP). The ability to deal with integer controls is required for the optimization of a car's gear selection.

There are two key problems in dealing with MIOCP. On the one hand, the difficulty is nonlinearity and on the other its discrete nature. Therefore, we present solving techniques for general *nonlinear programs*, as well as in particular possibilities to treat integrality like in discrete *mixed-integer nonlinear programs* (MINLPs).

First of all the definition of a general *nonlinear problem* (NLP) with equality and inequality constraints is presented. Then we take a closer look at the variables, which assign it to a MIOCP. To treat this kind of infinite-dimensional optimal control problem, we use the *direct multiple shooting* method, developed by Bock and Plitt in 1984 [7], in combination with a discretization of the controls and parametrization of the states. To achieve an exact and efficient solution for the resulting NLP, the sequential quadratic programming (SQP) approach is introduced. Further information on this is given by Wilson [37], Powell [29], Han [17], Nocedal and Wright [24], and combined with the *direct multiple shooting* method by Leineweber [23].

Generally, the solution of MINLPs is more difficult to obtain than the solution of a continuous problem. There are several algorithms which could be applied to MINLPs, like *Sum Up Rounding*, *Outer Convexification*, *Branch&Bound* or *Bender's Decomposition*. In the last section, we take a closer look at an approach first suggested by Sager [30, 31], using a reformulation of the problem by Outer Convexification combined with a transformation and relaxation of the constraints.

4.1 Problem Formulation

Primarily, we look at the general type of problem in nonlinear programming:

Definition 4.1. ***Nonlinear Problem (NLP)***

$$\min_{x \in D} \quad f(x) \tag{4.1a}$$

$$\text{subject to} \quad g(x) = 0, \tag{4.1b}$$

$$h(x) \geq 0, \tag{4.1c}$$

with a subset $D \subset \mathbb{R}^{n_x}$, objective function $f \in \mathscr{C}^2(D, \mathbb{R})$, equality constraints $g \in \mathscr{C}^2(D, \mathbb{R}^{n_g})$, and inequality constraints $h \in \mathscr{C}^2(D, \mathbb{R}^{n_h})$.

This definition is obviously not restricted to minimum problems. To solve maximum problems instead, it is mentioned that a maximization of $f(x)$ equates a minimization of $-f(x)$.

The most common examples of optimization problems with dynamical systems are constrained by differential equations, like differential algebraic equations (DAEs), partial differential equations (PDEs) or by ODEs, as presented in Section 2.4 and 2.5.1. These problems are called *optimal control problems*. We will now take a closer look at the variables of a NLP described by ODEs, with continuous and additional binary controls to obtain a mixed-integer problem.

The optimization vector of variables is denoted by x. Some of the variables are defined as states

$$x_i : [t_0, t_\mathrm{f}] \to \mathbb{R},\ t \mapsto x_i(t), \quad t_0, t_\mathrm{f} \in \mathbb{R},\ i \in \{1, \ldots, n_x\}. \tag{4.2}$$

The corresponding time-dependent set of ODEs, including this state vector is given by

$$\dot{x}(t) = f\big(t, x(t), u(t), \mu(t), p, \rho\big), \quad t \in [t_0, t_\mathrm{f}]. \tag{4.3}$$

Further right-hand side variables may be parameters $p \in \mathbb{R}^{n_p}$, as well as continuous controls

$$u_i : [t_0, t_\mathrm{f}] \to \mathbb{R},\ t \mapsto u_i(t), \quad i \in \{1, \ldots, n_u\}. \tag{4.4}$$

Additionally, there can be discrete controls

$$\mu_i(t) \in \Omega := \{\mu^1, \mu^2, \ldots, \mu^{n_\mu}\}, \quad \Omega \subset \mathbb{R}^{n_\mu}, \tag{4.5}$$

and parameters $\rho \in \mathbb{N}^{n_\rho}$, such as integer or binary variables. Path- and control constraints may exist as

$$0 \leq c\big(t, x(t), u(t), \mu(t), p, \rho\big), \quad t \in [t_0, t_\mathrm{f}], \tag{4.6}$$

and further interior point constraints as

$$0 = e^g\big(x(t_0), x(t_\mathrm{f}), p, \rho\big), \tag{4.7}$$
$$0 \leq e^h\big(x(t_0), x(t_\mathrm{f}), p, \rho\big). \tag{4.8}$$

The complete mixed-integer optimal control problem, with an objective function Φ reads as follows:

$$\min_{x,u,\mu,p,\rho,t_\mathrm{f}} \quad \Phi\big(t_\mathrm{f}, x(t), p, \rho\big), \tag{4.9a}$$
$$\text{s.t.} \quad \dot{x}(t) = f\big(t, x(t), u(t), \mu(t), p, \rho\big), \quad t \in [t_0, t_\mathrm{f}] \tag{4.9b}$$
$$0 \leq c\big(t, x(t), u(t), \mu(t), p, \rho\big), \quad t \in [t_0, t_\mathrm{f}] \tag{4.9c}$$
$$0 = e^g\big(x(t_0), x(t_\mathrm{f}), p, \rho\big), \tag{4.9d}$$
$$0 \leq e^h\big(x(t_0), x(t_\mathrm{f}), p, \rho\big), \tag{4.9e}$$
$$\mu(t) \in \Omega, \tag{4.9f}$$
$$\rho \in \mathbb{N}^{n_\rho}. \tag{4.9g}$$

The objective function Φ could be available as *Mayer term* Φ_M, evaluated at the end time t_f

$$\Phi_M = \Phi_M\big(t_\mathrm{f}, x(t_\mathrm{f}), p, \rho\big), \tag{4.10a}$$

as *Lagrangian term* Φ_L, evaluated on the whole time interval (t_0, t_f)

$$\Phi_L = \int_{t_0}^{t_\mathrm{f}} L\big(t, x(t), u(t), \mu(t), p, \rho\big)\ dt, \tag{4.10b}$$

or as a sum of both, called *Bolza type functional*

$$\Phi = \Phi_M + \Phi_L. \tag{4.10c}$$

The variables and constraints of the MIOCP can be described at the specific example of *testdrive*. The right hand side $f(\cdot)$ is given by the physics. In the original *testdrive* example $f(\cdot)$ is described by the time derivative of the position, which results in velocity. Additionally, by the time derivative of the velocity by the acting forces. Otherwise the transformed ODE model is position-dependent on σ, but with minimum end time as objective function as well.

Example of a MIOCP at the *extended testdrive* version

Objective function Φ:

$\min_{t_f}$ $\Phi_M\big(\sigma_f, x(\sigma_f)\big)$, whereas σ denotes the current advance on the midline, see Section 2.5.1.

States $x(\cdot)$:

d position to midline,

v velocity,

ψ car orientation,

ω_z car orientation velocity,

δ steering wheel angle,

β side slip angle,

t time.

Continuous controls $u(\cdot)$:

ϕ acceleration,

ξ braking,

ω_δ steering.

Discrete controls $\mu(\cdot)$:

$\mu \in \{1, \dots, n_\mu\}$ describes the gear choice.

Parameters p, ρ:

No parameters.

Path constraints $c(\cdot)$:

Track boundaries.

Interior point constraints e^g, e^h:

$x(\sigma_0)$ is the fixed initial car position or other initial values.

In the following, we present an efficient and reliable way to treat optimal control problems in general, as well as in the specific case of a mixed-integer problem. Due to the fact that the parameters p, ρ can also be defined as functions and added to the state or control vector, from now on we renounce on the notation of their explicit dependency. Furthermore, we assume a time-dependent set of ODEs, whereas the position-dependent system could be treated equally after the transformation of Section 2.5.1.

4.2 The Direct Multiple Shooting Method

In this section we present Bock's *direct multiple shooting* method, developed by Bock and Plitt in the early eighties [26, 7]. We describe a discretization and parametrization approach of the infinite-dimensional optimal control problem 4.9. This treatment is necessary to solve any type of optimal control problems numerically, for example using the SQP method, which is shown in the subsequent section.

The main idea of *direct* optimization methods like in multiple shooting is that discretization comes previous to optimization. By contrast, *indirect methods* operate the other way using *Pontryagin's maximum principle*, see e.g. [27]. A third approach how to treat this kind of problems is *dynamic programming* based on Bellman [4]. However, the indirect approach and dynamic programming both have trouble dealing with large-scale optimal control problems, which is also known as the *curse of dimensionality*. Thus, direct methods have become more and more common in optimization, in particular with large ODE control problems.

The key practices in direct methods are *single shooting*, Bock's *direct multiple shooting* method, and *collocation*. Further references are given in Betts [5], Binder [6].

4.2.1 Control Discretization

For the discretization of the controls, the time horizon is separated in a finite number of intervals. These intervals have to be neither of equal length, nor equate the parametrization grid of the states. Due to advantages in efficiency we use the same grid for states, as well as continuous and integer controls. Besides, for clarity reasons we combine the continuous and integer controls in vector u in this section, due to the fact that both can be treated equally for the control discretization. The selected time grid reads as

$$t_0 = \tau_0 < \tau_1 < \ldots < \tau_m = t_f, \quad m \in \mathbb{N}, \tag{4.11}$$

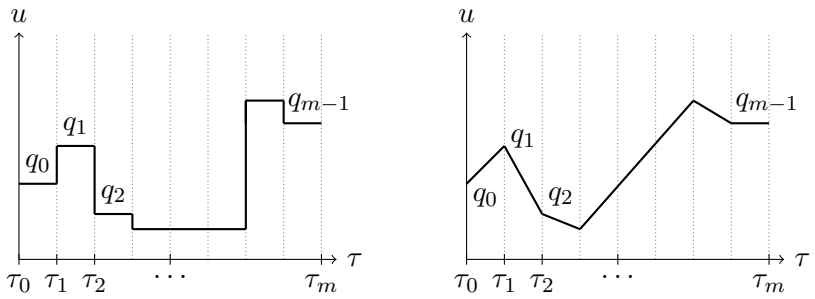
with intervals

$$I_i := [\tau_i, \tau_{i+1}], \quad \forall i \in \{0, \ldots, m-1\}. \tag{4.12}$$

The control functions are now replaced by basis functions

$$b_i : I_i \to \mathbb{R},\ t \mapsto b_i(t), \quad \forall i \in \{0, \ldots, m-1\}, \tag{4.13}$$

which could be of the type of piecewise constant functions (Figure 4.1(a)), piecewise linear functions (Figure 4.1(b)) or for example cubic splines. However, spline functions are rather unsuitable, because $b_i(t)$ should only have local influences.



(a) Piecewise constant controls. (b) Piecewise linear controls.

Figure 4.1 – Examples of control discretizations.

Now we are able to describe the infinite-dimensional control function u by vector $q_j \in \mathbb{R}^{n_q},\ j = 1, \ldots, k$, which depends only on a finite number of control discretization parameters:

$$u(t)\Big|_{I_i} = b_i(t, q_i), \qquad q_i \in \mathbb{R}^{k_i \times n_u},\ \forall i \in \{0, \ldots, m-1\}. \tag{4.14}$$

4.2.2 State Parametrization

On the same time grid as before, we introduce $m+1$ variables $s_0, \ldots, s_m,\ s_i \in \mathbb{R}^{n_x}$ as initial values of the ODE's state vector x. One initial value stands for each interval I_i as well as one for the final state τ_m. This leads to m initial value problems of the form

$$\dot{x}(t; s_i, q_i) = f\big(t, x_i(t), b_i(t, q_i)\big), \quad t \in [\tau_i, \tau_{i+1}], \tag{4.15a}$$

$$x(\tau_i) = s_i. \tag{4.15b}$$

Additional matching conditions

$$s_{i+1} = x(\tau_{i+1}; s_i, q_i), \quad \forall i \in \{0, \ldots, m-1\} \tag{4.16}$$

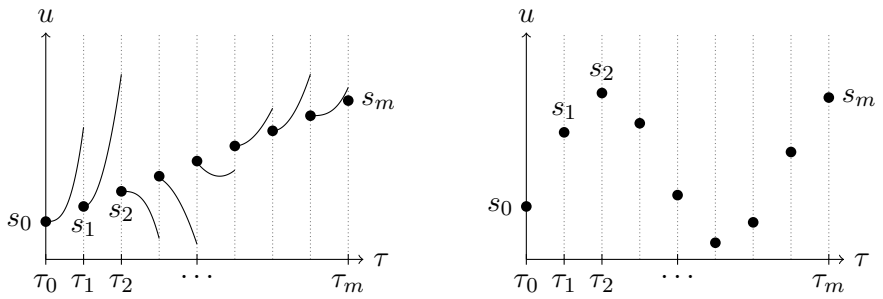
guarantee continuity of the assembled trajectory of $x(t)$, compare Figure 4.2.

4.2.3 Constraint Discretization

Furthermore, the path constraints of Equation 4.9c have to be discretized. That leads to an only pointwise enabled, finite number of inequality constraints

$$c\big(\tau_i, s_i, b_i(\tau_i, q_i)\big) \geq 0, \quad \forall i \in \{0, \ldots, m\}. \tag{4.17}$$

This discretization involves that the constraints between the grid knots could be violated. However, in most practical cases the approximation is sufficiently accurate, otherwise there are possibilities to reduce this constraint violation, see Potschka [28].



(a) Initial states s and assembled trajectory.

(b) Fulfilled matching conditions by the final trajectory.

Figure 4.2 – Examples of state parametrization of the multiple shooting method with matching conditions and the resulting trajectory.

4.2.4 Discrete Nonlinear Problem

The infinite dimensional optimal control problem of 4.9 is now discretized to a finite dimensional nonlinear problem as defined in 4.1.

$$\min_{q_i, s_i, s_m} \quad \Phi(t_\mathrm{f}, s_m) \tag{4.18a}$$

$$\text{s.t.} \quad \dot{x_i}(t, s_i, q_i) = f\big(t, x_i(t), b_i(t, q_i)\big) \tag{4.18b}$$

$$x_i(\tau_i) = s_i \tag{4.18c}$$

$$0 = s_{i+1} - x_i(\tau_{i+1}, s_i, q_i) \tag{4.18d}$$

$$0 = e^g\big(\tau_i, s_i, b_i(\tau, q_i)\big) \tag{4.18e}$$

$$0 \leq e^h\big(\tau_i, s_i, b_i(\tau, q_i)\big) \tag{4.18f}$$

$$0 \leq c\big(\tau_i, s_i, b_i(\tau, q_i)\big) \tag{4.18g}$$

This type of problem can be solved with a SQP method and converges to the solution of 4.9 for $m \to \infty$. The SQP approach is introduced in the following section.

4.3 Sequential Quadratic Programming Method

After discretizing and parametrizing, the resulting form of a NLP is described in Definition 4.1. In this section, we will briefly explain how to use the SQP method to solve this problem. For further information compare [37, 29, 17, 24, 23].

The basic idea is to approximate the solution of a NLP, starting from an initial value x^0 with the iteration

$$x^{k+1} = x^k + t^k \, \Delta x^k, \quad t^k \in (0, 1], \tag{4.19}$$

by solving constrained quadratic sub-problems of the following form:

Definition 4.2. ***Constrained Quadratic Sub-Problem***

$$\min_{\Delta x} \quad \nabla_x f(x^k)\Delta x + \frac{1}{2}\Delta x^T H^k \Delta x \tag{4.20a}$$

$$\text{s.t.} \quad g(x^k) + \nabla_x g(x^k)^T \Delta x = 0 \tag{4.20b}$$

$$h(x^k) + \nabla_x h(x^k)^T \Delta x \geq 0 \tag{4.20c}$$

$\nabla_x f(x^k)$ displays the gradient of the function f in iteration k, whereas H^k is typically an approximation of the Hessian of the Lagrangian function in this iteration step. Hence, as precondition the functions f, g, h have to be at least two times continuously differentiable. In general, the termination condition of iteration step $k+1$ is reached if x^k differs only negligible of x^{k+1}.

4.4 Outer Convexification

4.4.1 Motivation

The solution of a MIOCP as presented in 4.9 is difficult to obtain due to the combination of its nonlinear and discrete nature. In problems with integer controls for example, the behavior between two close, feasible points of the objective may significantly change, compared to a smooth continuous function. There are several algorithms which are able to treat this kind of problems. Negative fact in most of these algorithms is that they probably find the optimal solution of the problem on the underlying discretization grid, but only with an expensive calculation. Additionally, the computed solution might not be very useful for the continuous problem.

However there have been some improvements recently, which show the connection between rigorous bounds on the optimal integer solution value and results of relaxed, continuous control problems - first suggested by Sager in [30].

The intention on this is to equivalently reformulate a NLP to obtain a so-called *binary-control-affine* system. Afterwards, a relaxation of the binary controls produces an optimal solution which yields the lower bound of the NLP. Further information on this in Sager [31, 33].

4.4.2 Outer Convexification

We present the idea of the Outer Convexification approach, in combination with a relaxation of the binary controls. The $\mu(\cdot)$ of the MIOCP in 4.9f are denoted by *integer controls*. For every possible control of $\mu^i \in \Omega$, we now introduce a control function $w_i(\cdot)$ with the special attribute $w(t) \in \{0,1\}^{n_w}$. These particular $w(\cdot)$ are called *binary*

controls. The transformation of the integer into binary controls results in a *binary nonlinear* problem.

However, the reformulation of the integer controls results in additional control constraints. At the specific examples of a gear selection of a vehicle with five gears, this would lead to five different binary controls instead of one five-dimensional integer control. Hence, after eliminating one control with the linear equality constraint of the SOS1 condition 4.22d, we have four additional controls.

We are now able to formulate a *binary convex* problem by a convexification, including a modification of the ODEs (4.9b $\rightarrow$ 4.21b). For clarity reasons, we neglect path constraints c, as well as interior point constraints e^g, e^h in the following formulation. They remain just as in the MIOCP 4.9:

$$\min_{x,u,\mu} \quad \Phi\big(x(t)\big), \tag{4.21a}$$

$$\text{s.t.} \quad \dot{x}(t) = \sum_{i=1}^{n_w} f\big(t, x(t), u(t), \mu^i\big) \cdot w_i(t), \qquad t \in [t_0, t_\mathrm{f}], \tag{4.21b}$$

$$w(t) \in \{0,1\}^{n_w}. \tag{4.21c}$$

Furthermore, we add a *special ordered set* type one (SOS1) condition with

$$1 = \sum_{i=1}^{n_w} w_i(t). \tag{4.21d}$$

To achieve the following *convex relaxed* problem, the binary controls 4.21c are replaced by the relaxation 4.22c.

$$\min_{x,u,\mu} \quad \Phi\big(x(t)\big), \tag{4.22a}$$

$$\text{s.t.} \quad \dot{x}(t) = \sum_{i=1}^{n_w} f\big(t, x(t), u(t), \mu^i\big) \cdot w_i(t), \qquad t \in [t_0, t_\mathrm{f}], \tag{4.22b}$$

$$w(t) \in [0,1]^{n_w}, \quad t \in [t_0, t_\mathrm{f}], \tag{4.22c}$$

$$1 = \sum_{i=1}^{n_w} w_i(t). \tag{4.22d}$$

First important issue of these transformations is the existing bijection between the solutions of MIOCP 4.9 and the convexified problem 4.21. Furthermore, a solution of a binary convex problem, can be arbitrarily close approximated by the solution of a relaxed convex problem. A more detailed explanation as well as proofs are given in Sager [31, 30]. The SOS1 condition in this connection is necessary whenever the binary solution is constructed from the relaxed solution.

The additional controls resulting from the Outer Convexification, in this thesis given as number of gears, only enter linearly in the ODE system just as in most practical applications, compare Sager [31] as well. Hence as a conclusion, the negative aspect of an increased number of control functions is clearly out-weighted by the advantages of the presented method.

4.5 MUSCOD-II

The software package MUSCOD-II - *multiple shooting code for direct optimal control* - has been developed in the Simulation and Optimization group of H.G. Bock and J.P. Schlöder at the University of Heidelberg. MUSCOD-II is able to solve dynamic optimization problems, by the use of a multiple shooting approach in combination with the sequential quadratic programming method. Both of these methods have been presented in the previous sections, for further details of their usage in conjunction with MUSCOD-II see Leineweber [23], Diehl [13] and Bock [7].

In the context of this thesis, we use MUSCOD-II to solve mixed-integer nonlinear optimal control problems, which arise with the intention to achieve an optimal control of racing vehicles. Additionally, we use the Outer Convexification approach of Section 4.4.2, with a relaxation of the integer controls to handle the difficulties appending in gear shifting. This idea is part of the MS MINTOC algorithm - *multiple shooting based mixed-integer optimal control* - first proposed by Sager in [30, 31] and implemented in MUSCOD-II as well. MS MINTOC combines different MINLP algorithms, like Grid-Adaptivity, Switching Time Optimization, Sum Up Rounding, and the Outer Convexification approach.

Chapter 5

Comparison of Models on Test Track

In this chapter we present a comparison of the original *testdrive* optimal control problem to the modified one, which is called *extended testdrive* problem. Due to the improvements in ODE, car, and track model shown in the previous context of this thesis, one can see the basic differences of the optimal control problems at the relation of their numerical results.

The time-optimal control of a double-lane-change maneuver (see Section 3.1), computed with the car model of the original problem is described in Section 2.5. Numerical results for this purpose are presented in Gerdts [15, 16], as well as in Kirches [22] including the outer convexification approach. In this chapter, we use the latter case as solution data of the original *testdrive* example.

In the extended version we use a new car model, more precisely the PORSCHE CS, shown in Section 2.4. On the other hand, there is a slightly modified track course, due to non-differentiability in the computation of the track within the new coordinate system.

We briefly review both mixed-integer optimal control problems. Afterwards, initial values are introduced which are used as multiple shooting variables in the context of the direct multiple shooting method, presented in Chapter 4. Finally, the similarities and differences of both models are illustrated as visualized numerical results of states and controls of the ODE system.

5.1 Modification of the Track Course

The original track data is given by piecewise cubic Bézier splines, described in Section 3.1. After the transformation of the original to the *extended testdrive* version and the resulting modifications in the ODE system 2.5.1, it is necessary to transform the original track course to another coordinate system (*CSII*), compare Section 3.3. There-

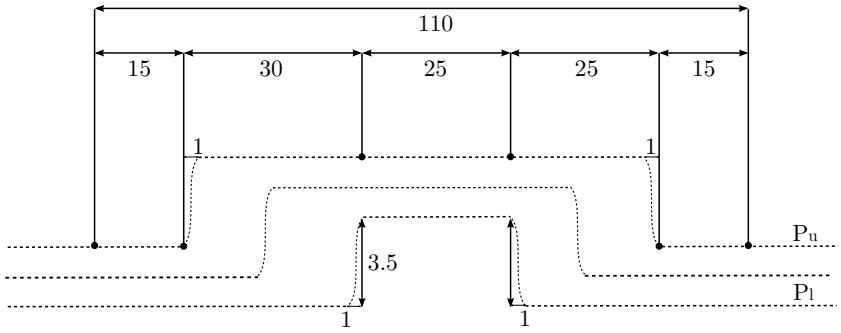


Figure 5.1 – Layout of test course with non differentiable midline.

fore, we first have to convert the piecewise splines into control points of cubic Bézier patches. For the transformation of the coordinate system, we have to compute the midline out of the Bézier patches. Then the curvature of the midline is used to get the routing of the track at a finite number of points.

The problem of the modifications in this case is that the track boundaries, given as piecewise splines, are not differentiable. Hence, the midline which is defined as half of the distance between the upper and lower track boundary is also not differentiable, as can be seen in Figure 5.1. However, the function that describes the curvature of the midline essentially has to be at least two times continuously differentiable, due to the fact that it is used for the calculation of the new ODE right-hand side function (see Section 4). To fix this, we decided to smooth the track boundaries, which leads to a smoothed midline as well. The smoothed track is comparable with the old one, due to the remaining key points at the transition curves of the track course, see Figure 5.2.

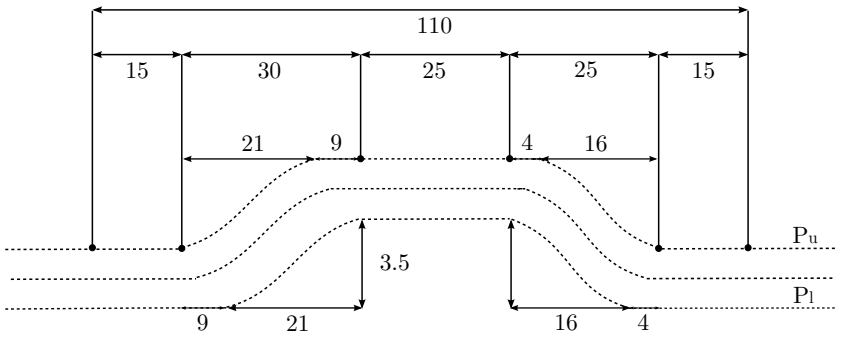


Figure 5.2 – Layout of smoothed test course

5.2 Optimal Control Problems

We briefly review the two different mixed-integer optimal control problems of *testdrive* at a glance.

5.2.1 Original Testdrive Optimal Control Problem

The original *testdrive*, as shown in [22, 16], strives to minimize the total time t_f while doing minimal steering effort $\omega_\delta(t)$. At any time, the car of width b must be positioned within the test course's boundaries, which are formulated by the double inequality path constraint in Equation 5.1c. The system's initial values are fixed in all states in Equation 5.1h, with the exception of the car's initial vertical position on the track, which remains a free variable only constrained by the track's boundary. Finally, constraints of Equation 5.1i, 5.1j guarantee that the car actually arrives at the end of the test course, driving straight ahead. The entire problem reads as

$$\min_{t_\mathrm{f},x(\cdot),u(\cdot),\mu(\cdot)} \quad t_\mathrm{f} + \int_0^{t_\mathrm{f}} \omega_\delta^2(t)\ \mathrm{d}t \tag{5.1a}$$

$$\begin{aligned}
\text{s.t.} \quad \dot{x}(t) &= f\big(t, x(t), u(t), \mu(t)\big), && \text{(5.1b)}\\
c_\mathrm{y}(t) &\in \big[P_\mathrm{l}(c_\mathrm{x}(t)) + \frac{b}{2},\ P_\mathrm{u}(c_\mathrm{x}(t)) - \frac{b}{2}\big], && \text{(5.1c)}\\
\omega_\delta(t) &\in [-0.5, 0.5], && \text{(5.1d)}\\
F_\mathrm{B}(t) &\in [0, 1.5 \cdot 10^4], && \text{(5.1e)}\\
\phi(t) &\in [0, 1], && \text{(5.1f)}\\
t &\in [t_0, t_\mathrm{f}], && \text{(5.1g)}\\
x(t_0) &= \big(-30, \text{free}, 10, 0, 0, 0, 0\big)^T, && \text{(5.1h)}\\
c_\mathrm{x}(t_\mathrm{f}) &= 140, && \text{(5.1i)}\\
\psi(t_\mathrm{f}) &= 0, && \text{(5.1j)}
\end{aligned}$$

with state vector $x(\cdot)$, vector of the continuous controls $u(\cdot)$, and integer control vector $\mu(\cdot)$

$$x := \Big(c_\mathrm{x}, c_\mathrm{y}, v, \delta, \beta, \psi, w_\mathrm{z}\Big)^T, \qquad u := \Big(\omega_\delta, F_\mathrm{B}, \phi\Big)^T, \qquad \mu(t) \in \{1, \ldots, 5\}.$$

5.2.2 Extended Testdrive Optimal Control Problem

The extended MIOCP strives to minimize the total time t_f as well, but with a position-dependent set of ODE's. In the following formulation the advance on the midline is

given at the starting position by $\sigma_0 = 0$, while the final value reads as $\sigma_\mathrm{f} = 170.637$.

$$\min_{x(\cdot),u(\cdot),\mu(\cdot)} \quad t(\sigma_\mathrm{f}) + \int_{\sigma_0}^{\sigma_\mathrm{f}} \omega_\delta^2(\sigma)\ \mathrm{d}\sigma \tag{5.2a}$$

$$\text{s.t.} \quad \dot{x}(\sigma) = f\big(\sigma, x(\sigma), u(\sigma), \mu(\sigma)\big), \tag{5.2b}$$

$$d(\sigma) \in \big[-P_\mathrm{up}(\sigma) + \frac{b}{2},\ P_\mathrm{up}(\sigma) - \frac{b}{2}\big], \tag{5.2c}$$

$$\omega_\delta(\sigma) \in [-0.5, 0.5], \tag{5.2d}$$

$$\xi(\sigma) \in [0, 1], \tag{5.2e}$$

$$\phi(\sigma) \in [0, 1], \tag{5.2f}$$

$$\sigma \in [\sigma_0, \sigma_\mathrm{f}], \tag{5.2g}$$

$$x(\sigma_0) = \big(\text{free}, 10, 0, 0, 0, 0, 0\big)^T, \tag{5.2h}$$

$$\psi(\sigma_\mathrm{f}) = 0. \tag{5.2i}$$

The track boundaries are described by the double inequality path constraint of Equation 5.2c using only the upper bound and the car's width b, due to the same difference between upper and lower bound to the midline. The initial values in Equation 5.2h are fixed, except the difference d of the start position to the midline. The track length equates the original problem, but is now described by the length of the midline with 170.637, starting in 0. At the end of the track, straight ahead driving is guaranteed as well (5.2i). Braking is now controlled by a model of the brake pedal, see Equation 5.2e, which influences the resulting vehicle dependent braking force. In the following, state vector $x(\cdot)$, continuous control vector $u(\cdot)$, as well as integer control vector $\mu(\cdot)$ are illustrated:

$$x := \Big(d, v, \delta, \beta, \psi, \omega_\mathrm{z}, t\Big)^\mathrm{T}, \qquad u := \Big(\omega_\delta, \xi, \phi\Big)^\mathrm{T}, \qquad \mu(\sigma) \in \{1, \ldots, 5\}.$$

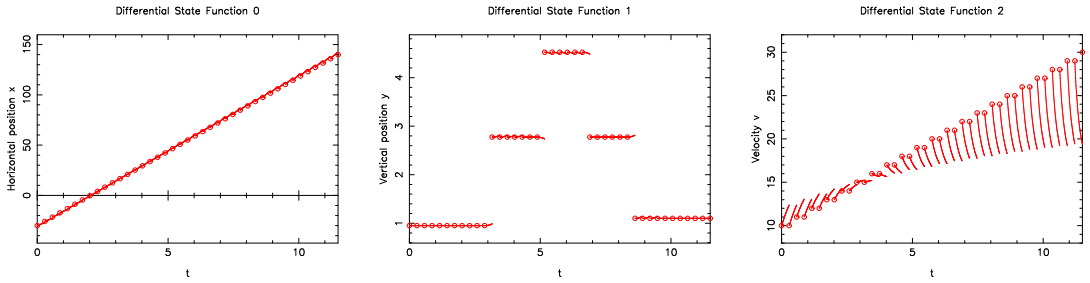
5.3 Numerical Results

To solve these mixed-integer optimal control problems, we use the techniques presented in Chapter 4. Therefore, the introduced variables for discretization and parametrization in the multiple shooting method at first have to be initialized. Afterwards, we compare the achieved solutions of both *testdrive* problems.

5.3.1 Variable Initialization

The initial control values of an optimal control problem are often difficult to obtain. However, the direct multiple shooting method allows to supply additional information about the state vector. The following figure shows the initial state variables $(c_x, c_y, v)(t)$ of the original *testdrive* problem, for 40 multiple shooting nodes ($N_\mathrm{shoot} = 40$). With

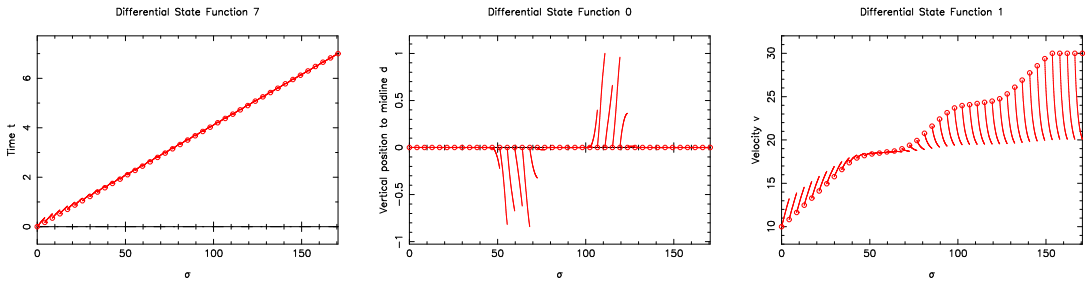
the initial guess of these state and control variables, the number of iterations which are needed to obtain convergence differs consequently. For unphysical initializations however, the whole procedure may fail.



(a) Horizontal position c_x. (b) Vertical position c_y. (c) Velocity v.

Figure 5.3 – Initialization of the shooting variables of the original *testdrive* example, chosen to start the solution process.

We have initialized the end time $t_f = 11.5\text{s}$ in the previous time-dependent set of ODEs, whereas in the following position-dependent problem the time has to be initialized as additional state, compare Figure 5.4(a). On the other hand, the horizontal position c_x in Figure 5.3(a) is comparable with the advance on the midline σ, after the coordinate transformation and ODE modification. The vertical position c_y, shown in 5.3(b) of the original example has to be initialized, depending on the track course. The smoothed track course however, is attached to the state d in Figure 5.4(b) as position to the midline and can easily be initialized by 0.

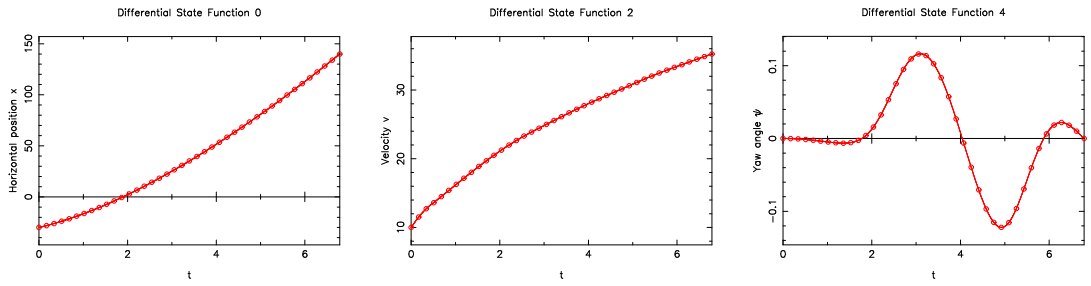


(a) Time t. (b) Vertical position to midline d. (c) Velocity v.

Figure 5.4 – Initialization of the shooting variables of the *extended testdrive* example on the smoothed track course, chosen to start the solution process.

5.3.2 Comparison of the Solutions

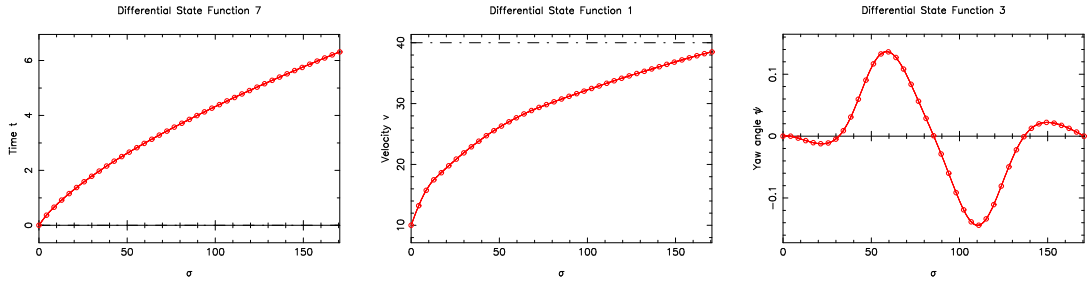
The driving behavior of the optimal solution in both examples is very similar. During the entire time, respectively space horizon the brake is inactive and the vehicle is driving with maximum acceleration. Although the PORSCHE CS accelerates faster and has a higher final velocity (compare Figure 5.5(b), 5.6(b)), the gear shift acts nearly identical for the old engine model and the PORSCHE CS. This is caused by different gear ratios and the resulting higher range of revolutions per minute.



(a) Horizontal position c_x. (b) Velocity v. (c) Yaw angle ψ.

Figure 5.5 – States for the original *testdrive* problem with $N_{\text{shoot}} = 40$.

Logically speaking, the PORSCHE CS results by the use of a higher amount of velocity, within a faster end time. This is represented in the old model explicitly in combination with the horizontal position, resulting at the end condition $c_x = 140$ within the optimal time $t_f = 6.786794$ (Figure 5.5(a)). On the other hand, the additional state of time in the extended model results in $t_f = 6.315952$ (5.6(a)). Furthermore, the illustrations of the different yaw angles 5.5(c), 5.6(c) display that the behavior of the car's orientation is very similar - the slight differences are caused by the variations in the track course.

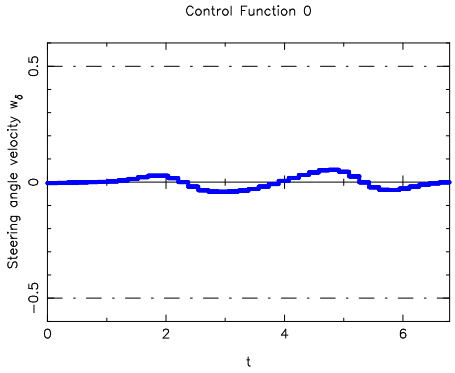


(a) Time t. (b) Velocity v. (c) Yaw angle ψ.

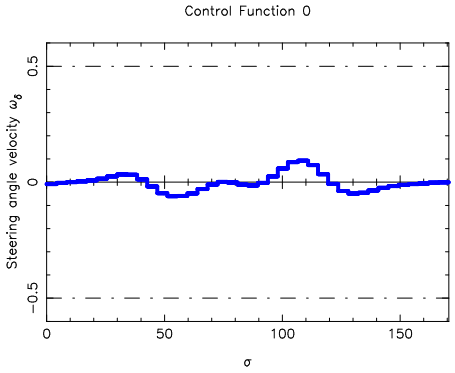
Figure 5.6 – States for the *extended testdrive* problem on the smoothed track course with $N_{\text{shoot}} = 40$.

In spite of the similarities of both *testdrive* versions, the differences of car and track

models lead to a slightly varying steering behavior. Figure 5.7(a) and Figure 5.8(a) clarify that the vehicle is able to round the obstacle within one arc. Whereas the steering angle velocity of the extended version in Figure 5.7(b) shows that the steering wheel shortly remains in a certain position within the turn, followed by an increased steering effort.



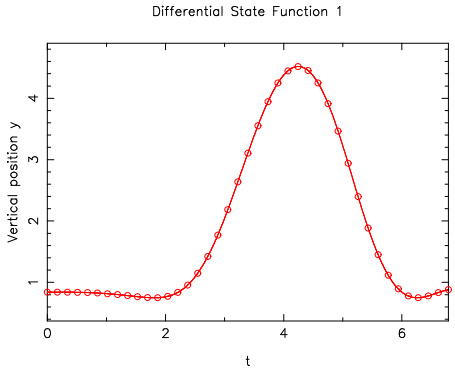
(a) Steering angle velocity ω_δ of the original *testdrive* example.



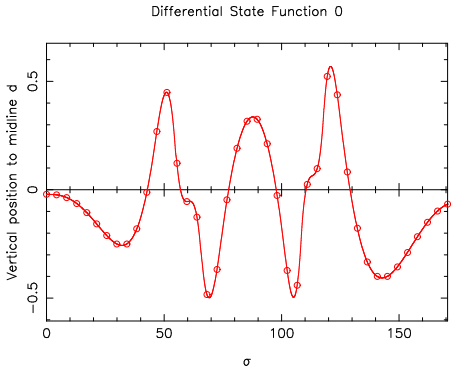
(b) Steering angle velocity ω_δ of the *extended testdrive* model.

Figure 5.7 – Slightly different steering behavior due to differences in velocity.

The vertical position to the midline (Figure 5.8(b)) can be tracked in Figure 5.2, which shows the smoothed track. The car closely passes the key points for the optimal control on that test course.



(a) Vertical position c_y of the original *testdrive* example.



(b) Vertical position to midline d of the *extended testdrive* model.

Figure 5.8 – Differences in vertical amplitude of the vehicle due to different embedded coordinate systems.

Chapter 6

Optimal Control of Vehicles on a Race Track

In this chapter, we finally present the optimal control of different car models on a realistic race track. We introduced the optimal control problem *testdrive*, included in the optimization software package MUSCOD-II, as basic idea in the previous course. So far, MUSCOD-II has been capable to optimize a certain car on a short test track with the integrated problem formulation.

With intent to enlarge the test track to an realistic race track, first of all *testdrive*'s set of ODEs had to be modified. We used the track data of the racing simulation game VDRIFT and transformed the coordinate system. This allows to optimize a vehicle on an optional VDRIFT track within the *testdrive* problem. Furthermore, we are able to use optional VDRIFT vehicles and improved *testdrive*'s engine model, wheel forces, and car parameters, so that the computed solution is more realistic and gets closer to the "real" VDRIFT world.

In particular, we show the solution of an optimal controlled lap, of a PORSCHE CLUBSPORT as well as a Formula One car of the year 2002 F1-02, both presented in Chapter 2, driving on Germany's official Formula One racing track the *Hockenheim-ring*. We demonstrate not only the optimization of the racing line but especially the gear-shifting of the vehicle. Here, the gear selection is not optimized addicted to the actual state like in automatic transmission, but continuously optimized which to our knowledge never has been accomplished in this form yet.

The huge potential of this approach is shown for energy-optimal controlled trucks in Buchner [8], Terwen [35], and Hellstrom [18], which allows the assumption to gain performance for time-optimal control of racing vehicles as well.

6.1 Hockenheimring - Track Information

The *Hockenheimring, Baden-Württemberg*[1] was originally built in 1932 at a length of almost 8km as racing circuit and furthermore to test *Mercedes Benz* vehicles. The racing track became more and more successful over the years, especially with motorcycle events until the 1960's. In 2002 the whole track was redesigned and shortened, to create new overtaking possibilities for Formula One cars as well as more seating capacity.

There are two different track courses, which is the official Formula One racing circuit with a length of 4574m, as well as a shorter lap where other racing events are battled. The current lap record on the Formula One track was made by *Kimi Räikkönen* in 2004 with a *McLaren Mercedes* in 73.780s, which equals an average speed of $223.182\frac{\text{km}}{\text{h}}$.

In Figure 6.1 an illustration of the *Hockenheimring* Formula One Grand Prix circuit is shown, including an advised gear shifting for a part of the track, given by *Sport Auto* magazine (see website [20]). The driving direction on the *Hockenheimring* is clockwise.

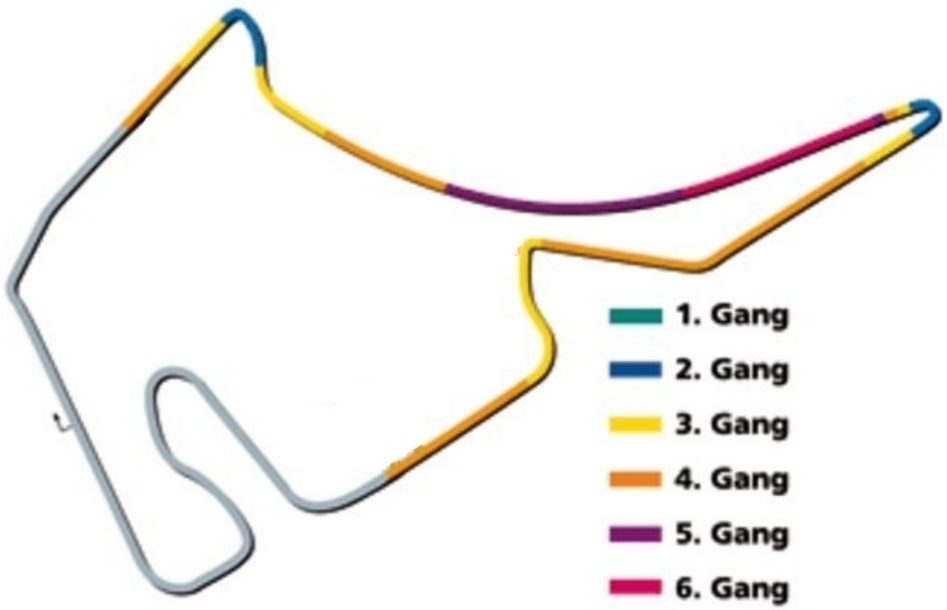


Figure 6.1 – Illustration of an advised gear shifting by "Sport Auto" magazine (website [20]) for a partial lap on the *Hockenheimring* driving a BMW M3 CSL, which lap time is about 2min. The starting line is represented by the small sign on the left straight track part.

[1]Former name up to 1947 was *Kurpfalzring*, later on simply *Hockenheimring* until in 2002 the official name became *Hockenheimring, Baden-Württemberg*.

6.2 Initialization Approach

In the previous Chapter 5 we presented the time-optimal control of a car on a short test track with a length of about 170 meters. In comparison to that, the optimization problem of the driving behavior on an original race track with a length of multiple kilometers is significantly more complex. The dimension of this problem makes it impossible to solve, in desirable accuracy within the set constraints, all at once. Hence, our idea was to split the track in several overlapping parts, optimize those and put the segments back together. This leads to a partial optimal solution, which we can use as initial values of the complete optimization problem. Nevertheless, this problem with several hundred multiple shooting nodes will be huge.

Another difficulty lies in the position of the multiple shooting nodes. Naturally, in track parts like in hairpin curves for example, with a lot of steering, braking, or gear shifting, the multiple shooting grid should be more accurate than in straight track segments. Fortunately, the Bézier points of the track data in VDrift are arranged just in that way. Therefore, we decided to set one multiple shooting node on the beginning in every second Bézier patch, which results in 324 multiple shooting nodes for a complete lap of the *Hockenheimring*.

The initial values of the track segments are estimated depending on the track's curvature. The several segments are in general initialized by about 40 multiple shooting nodes, although differing with the track course. Every segment within the track course has an overlap of 5 to 15 nodes of the optimized previous track part's state and control values.

The first multiple shooting point is set onto the starting line, which also equates the zero point in *CSII* (see Chapter 3). We assume that the vehicle launches out of a specific starting position, which is given by VDrift's track data, several meters before the actual starting line and shifted in relation to the midline. Hence, we have to assign the velocity as well as the vertical position on the starting line and fix them as first multiple shooting point. Therefore, we use the output data of VDrift in the zero point, driving straight ahead with maximum acceleration in the first gear, which is achieved by little improvements we made in VDrift's source code (further in Chapter 7).

Thus, the Porsche CS has a fixed initial velocity $v_0 = 11.4509\frac{\mathrm{m}}{\mathrm{s}}$ and vertical position as distance to the midline $d = 4.05983$m. The car's orientation is given by the yaw angle $\psi_0 = -1.55632$, which equals straight ahead driving on the *Hockenheimring* at this particular track position. For the F1-02 the initial variables are $v = 15.2314\frac{\mathrm{m}}{\mathrm{s}}$, $d = 4.05983$, and $\psi_0 = -1.55632$. All other states are set to 0.0 as fixed initial values.

The initial control values for both cars are given by maximum acceleration, no braking and steering, with the first gear engaged. All initial control values are free for optimization.

6.3 Numerical Results

In addition to the huge size of the optimization problem driving a complete lap on the *Hockenheimring*, difficulties appear in conjunction with the Outer Convexification of the binary control constraints. Further information on these problems is given in Section 8.1.

According to this, we primarily produce a relaxed optimal solution, which is presented subsequently using a PORSCHE CS. Then we apply specific heuristics to the relaxed solution, to finally achieve the optimal solution with binary constraints for gear shifting.

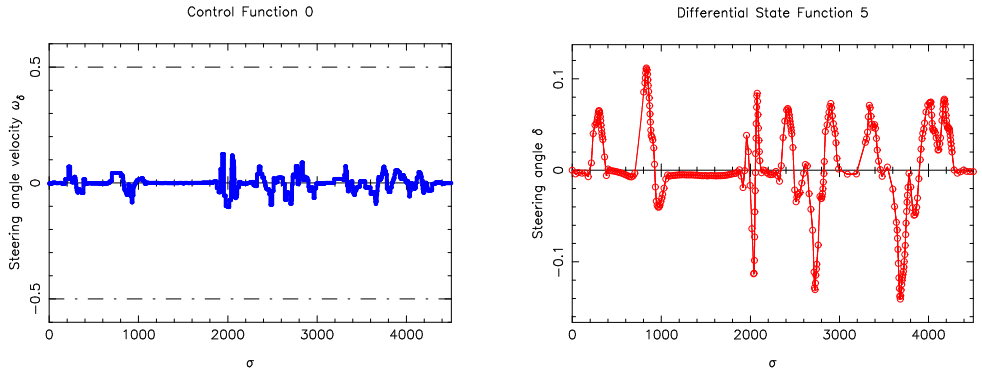
Afterwards, the optimal solution of the Formula One car F1-02 is illustrated, driving a partial lap on the *Hockenheimring*. Here, the mentioned problems appeared to be too complicated to achieve an optimal solution for a complete lap. Therefore, an idea to handle these challenges by a different approach is given in the outlook of the thesis, compare Chapter 8.

6.3.1 Porsche Clubsport - Relaxed Solution

The PORSCHE CLUBSPORT has been introduced in Chapter 2, including its parameters as well as the functionality of the embedded mathematical model. We now show the numerical results of the optimal control problem, which is presented in formulation 5.2 in the previous chapter. The only difference to this formulation is the underlying track course, used for the calculation of the vehicle's orientation, which is now given by the *Hockenheimring* instead of the test track data. Hence the new final value of the advance on the midline reads as $\sigma_f = 4503.23$m.

The formulation of the integer control constraints has been introduced with the Outer Convexification approach in Section 4.4.2. To actually achieve an optimal solution of a complete lap, initialized with the optimized parts of the track, we primarily have to target the solution with relaxed binary control constraints. This solution, computed by MUSCOD-II, is presented in the following. Here the x-axis shows the advance on the midline of the vehicle in meters, while the y-axis illustrates the specific control or state value.

In Figure 6.2(a) and 6.2(b) the course of a lap can be tracked by the control of the steering wheel angular velocity ω_δ, as well as the state of the steering wheel angle δ. The first curve appears after about 200m, followed by a sharper double curve, which results in an increased steering activity. An additional trajectory amplitude in the double curve can be seen in both figures, showing turn in, counter steering and a subsequent straightening back of the car. Then, the course yields a drawn-out curve of about 1000m with ω_δ near zero. However, the little activity of the steering angle δ, with the increasing vehicle velocity (see Figure 6.5(b)) it is enough to handle this curve. A very high steering amplitude appears logically within the hairpin curve at

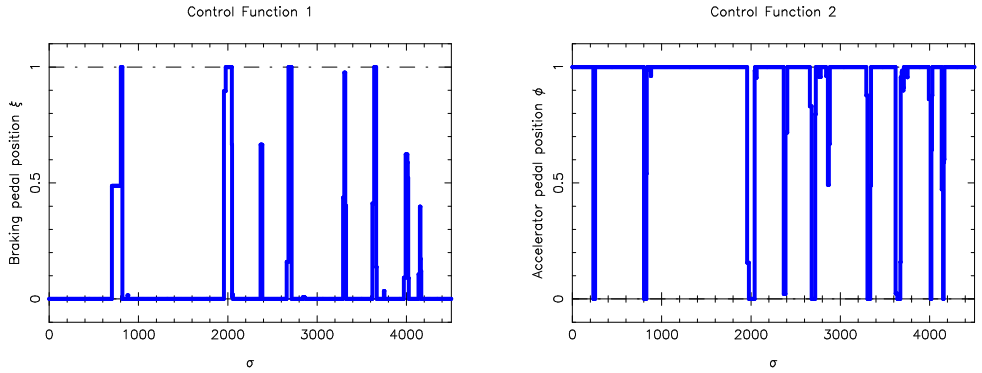


(a) Steering angle velocity ω_δ.

(b) Steering angle δ.

Figure 6.2 – Optimal control of the steering angle velocity and steering angle given as differential state of the relaxed optimal solution of the PORSCHE CS.

about 2000m. Likewise, this curve can be seen at the side slip angle β in Figure 6.5(c), with an at least fourfold amount of the slip angle compared to the remaining track. In the second half of the lap, a lot more steering is going on. This is mainly caused by shorter distances between the curves in this track part.



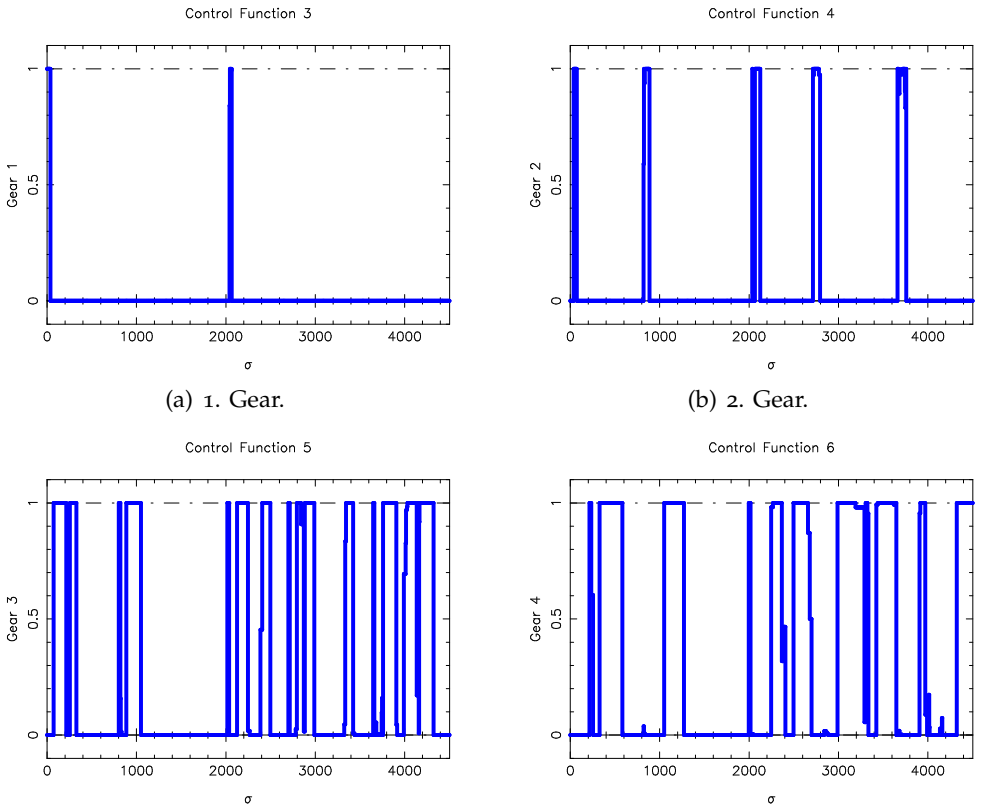
(a) Braking pedal position ξ.

(b) Acceleration pedal position ϕ.

Figure 6.3 – Differential states of braking and accelerating of the relaxed optimal solution of the PORSCHE CS.

Figure 6.3(a) and 6.3(b) illustrate the position of the braking and the accelerating pedal. It is enough to shortly get off the acceleration pedal without any braking use to handle the first curve, due to the little velocity which is reached since the starting position and the condition of the curve. Within the following double curve however the braking pedal is used. It can be seen very well that the braking and accelerating correlate.

Obviously the hairpin curve after the high speed track segment at 2000m appears to be the longest braking distance.



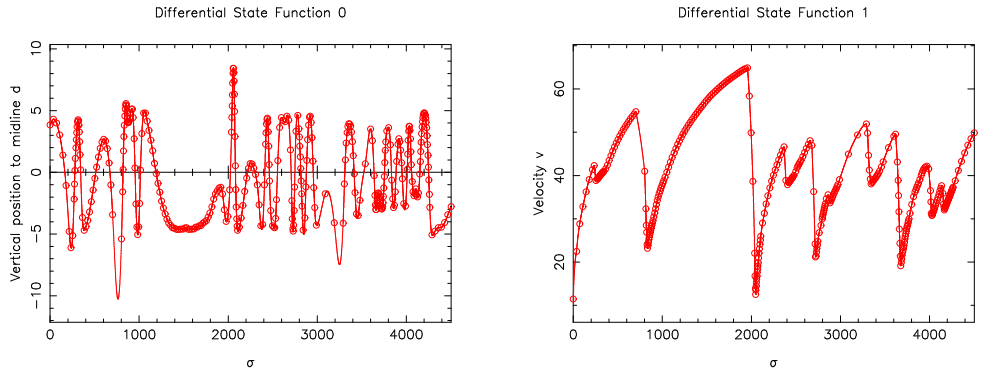
(a) 1. Gear. (b) 2. Gear.

(c) 3. Gear. (d) 4. Gear.

Figure 6.4 – Gear activity of the relaxed solution, showing the first four gears of the Porsche CS. The 5. Gear has been eliminated as specific binary control with the linear equality constraint of the SOS1 condition.

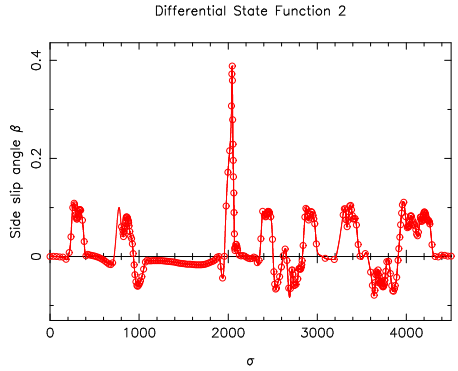
Figure 6.4 shows the relaxed binary controls of the gears in the optimization problem without the control of the 5. Gear, which is eliminated as specific binary control by the SOS1 condition of Equation 4.22d. That means, whenever no other gear is active in the above illustrations, the 5. Gear is engaged. This is the case, for the long high speed segment between 1200m to 2000m for instance. The relation of the gear shifting to the state trajectory of the velocity can be seen in Figure 6.5(b).

In the first half of the track the relaxed solution is often of bang-bang type, which means that the relaxed controls only take values at its bounds. As mentioned, the second half of the track is sticked with a higher amount of curves in a shorter distance. The velocity acts mainly within $20\frac{\mathrm{m}}{\mathrm{s}}$ to $50\frac{\mathrm{m}}{\mathrm{s}}$ with many amplitudes, which causes a lot of gear shifting. At this point, the relaxed solution is frequently not of bang-bang type.

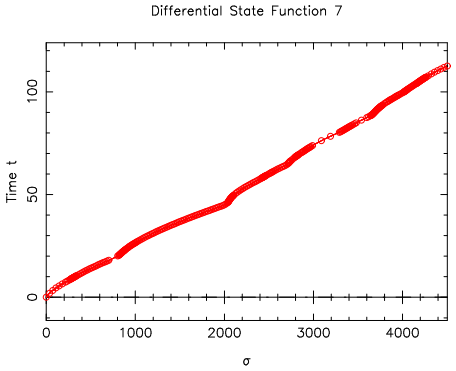


(a) Vertical position d to the midline.

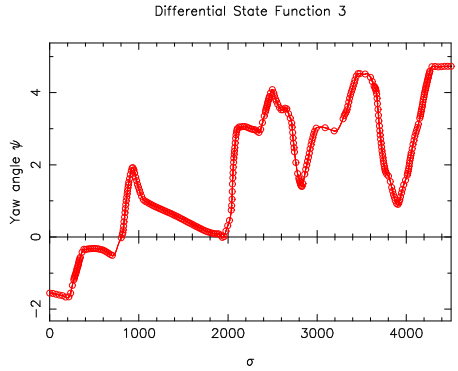
(b) Velocity v.



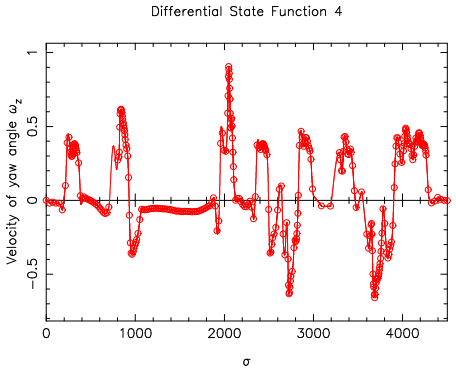
(c) Side slip angle β.



(d) Time t.



(e) Yaw angle ψ.



(f) Yaw angle ω_z.

Figure 6.5 – Differential states of the relaxed optimal solution of the PORSCHE CS.

The differential states of the relaxed optimal solution are illustrated in Figure 6.5. The orientation of the car is given as the yaw angle ψ in Figure 6.5(e). If the yaw angle changes its value within a short time period, this can be seen as indication for a curve. The time t with regard to the advance of the car increases continuously with the only noticeable point at the hairpin curve, where the vehicle needs more time to drive within less distance. Obviously this is caused by the extremely slow velocity at this point, see Figure 6.5(b).

The optimal objective function value of the presented relaxed solution is $t = 112.7175\text{s}$.

6.3.2 Porsche Clubsport - Integer Solution

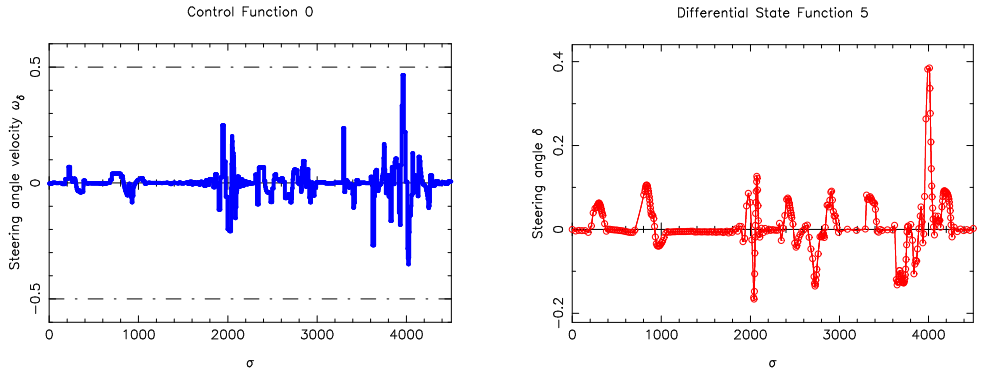
To achieve a feasible integer solution, there are several heuristics which can be applied to a relaxed optimal solution, like *Grid Adaptivity*, *Switching Time Optimization*, or different rounding strategies. For closer examination of all techniques, we refer to Sager [33]. All heuristics are implemented within the MS MINTOC extension of the software package MUSCOD-II.

The concept of **Grid Adaptivity** is based on a refining technique of the discretization of the control space. In this case, a bang-bang structured solution is assumed, although the relaxed solution is not of this form yet. This approach aims in reproducing the controls of the relaxed solution which are already $\in \{0, 1\}$ and refine the time grid anywhere else.

Switching Time Optimization can be applied to a bang-bang structured solution as well as to an already rounded solution which has been computed on a fixed time grid. Then, instead of free controls the time grid is considered free to achieve a better solution, for an exemplary application see Gerdts [16].

Grid Adaptivity followed by Sum Up Rounding would be the method of choice for which convergence towards a theoretically guaranteed feasible integer solution would be obtained, Sager [31]. However, for technical reasons (coupling of shooting notes to Bezier patches) and the high dimensionality this was not possible in a straightforward way. Therefore we used a **rounding strategy on a shrinking horizon**. Here, every control is rounded one by one, which means that the control of the relaxed solution at the first grid point is rounded, followed by a new optimization. If convergence is achieved, one advances with the control in the second grid point and so on.

Although the resulting solution improved much from the relaxed one, concerning the integrality of gear shifting, there are still few of the binary controls left which are not feasible. At this, we refer to Chapter 8, where the occurred problems are discussed and ideas for future work are presented. Anyway we stick to the name *integer* solution for the presented numerical results in this section.

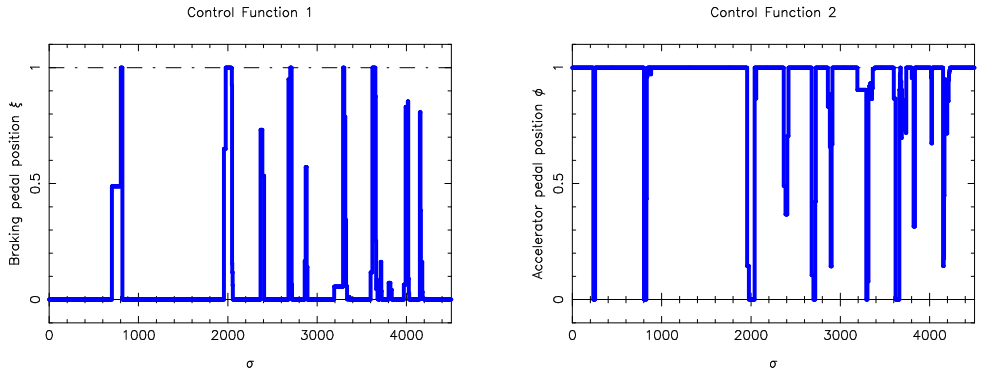


(a) Steering angle velocity ω_δ. (b) Steering angle δ.

Figure 6.6 – Optimal control of the steering angle velocity and steering angle given as differential state of the optimal integer solution of the PORSCHE CS.

Compared to the relaxed solution, the integer solution shows a lot more steering activity at the first sight. Although the trajectory's amplitude of the steering wheel angle velocity in Figure 6.6(a) increased, the single time periods of steering are shortened. Particularly noticeable at this point is the resulting vertical position to the midline in Figure 6.9(a), which is quite different now. This vertical position, which can be interpreted as optimal racing line is illustrated with respect to the actual track data of the *Hockenheimring* in Figure 6.10.

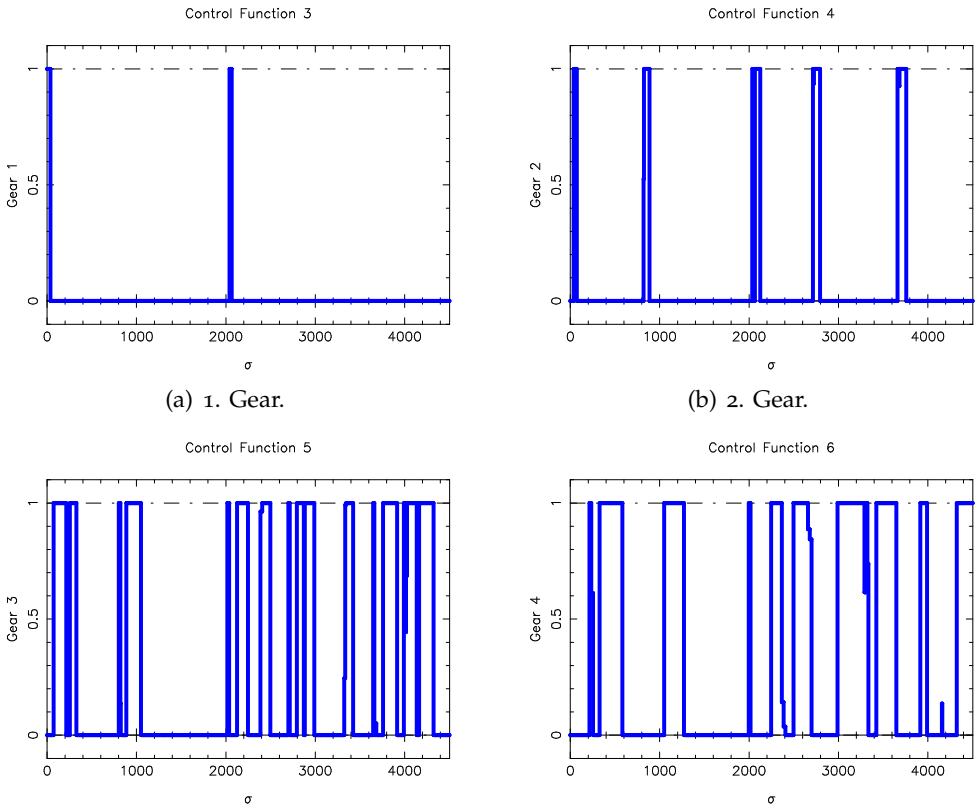
Furthermore, the use of the braking pedal increased especially in the second half of the track. At the same time the use of the acceleration pedal is reduced.



(a) Braking pedal position ξ. (b) Acceleration pedal position ϕ.

Figure 6.7 – Differential states of braking and accelerating of the optimal integer solution of the PORSCHE CS.

Figure 6.8 illustrates the gear shifting of the integer solution, achieved with the described rounding heuristic applied to the relaxed solution. An illustration of the optimal gear shifting points within the actual track data of the *Hockenheimring* is given in Figure 6.11. Here, the complete shifting controls can be tracked at downshifting in curves, compare the hairpin curve, and up shifting at the high-speed track parts. In the last curve the remaining difficulties with the integer solution can be seen, where the gear is switched from the third into the fifth gear. At that point, once again we refer to the ideas in the outlook in Chapter 8.



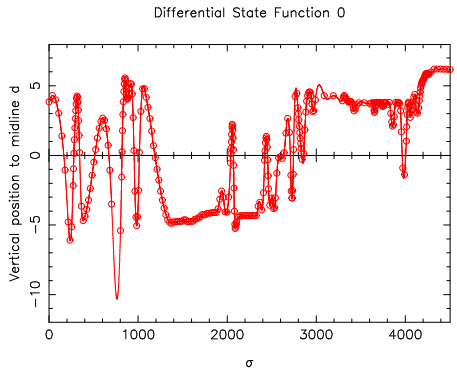
(a) 1. Gear.

(b) 2. Gear.

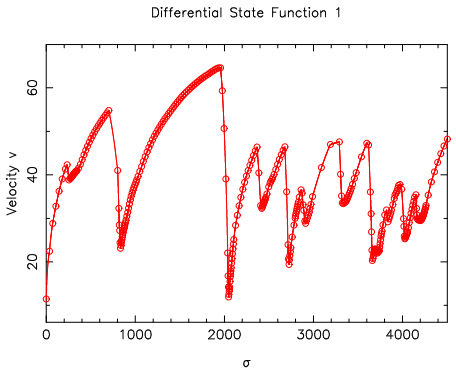
(c) 3. Gear.

(d) 4. Gear.

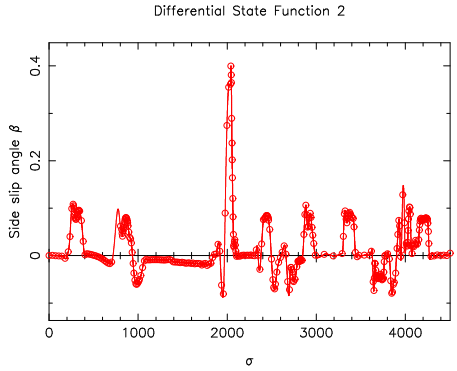
Figure 6.8 – Gear activity of the integer solution, showing the first four gears of the Porsche CS. The 5. Gear has been eliminated as specific binary control with the linear equality constraint of the SOS1 condition. Note that the solution is not integer feasible because of the problems mentioned in Section 8.1, and technical difficulties with an adaptation of the control grid size.



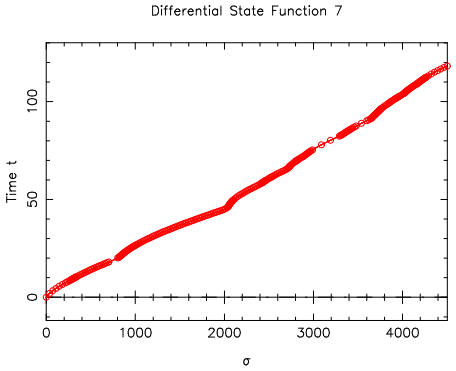
(a) Vertical position d to the midline.



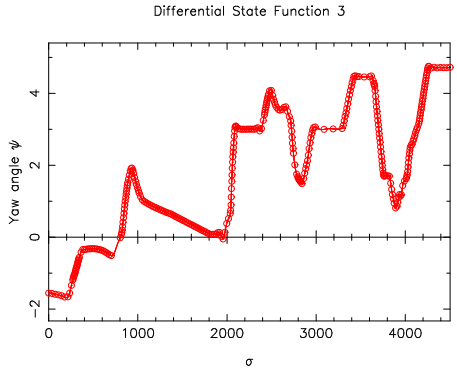
(b) Velocity v.



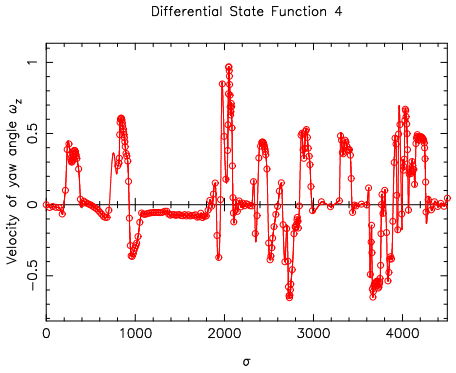
(c) Side slip angle β.



(d) Time t.



(e) Yaw angle ψ.



(f) Yaw angle ω_z.

Figure 6.9 – Differential states of the optimal integer solution of the PORSCHE CS.

The optimal objective function value of the integer solution is $t = 118.8537$s, which is 6.1362s slower than then relaxed solution, logically caused by fixing of the controls.

To classify that solution in real world, note the lap time driven by a 2006 Porsche 997 GT3. According to website *fastestlaps.com* [21], this lap time is the leading result of a non Formula One vehicle on the Grand Prix circuit of the *Hockenheimring*.

Car	$v_{\max}$ in $\frac{\text{m}}{\text{s}}$	$v_{\max}$ in $\frac{\text{km}}{\text{h}}$	Lap Time t
2006 Porsche 997 GT3	86.1	310	116.41s
1989 Porsche CS	68.8	248	118.85s

Table 6.1 – Comparison of the computed optimal solution to a real world lap time, driven on the *Hockenheimring*, see website [21].

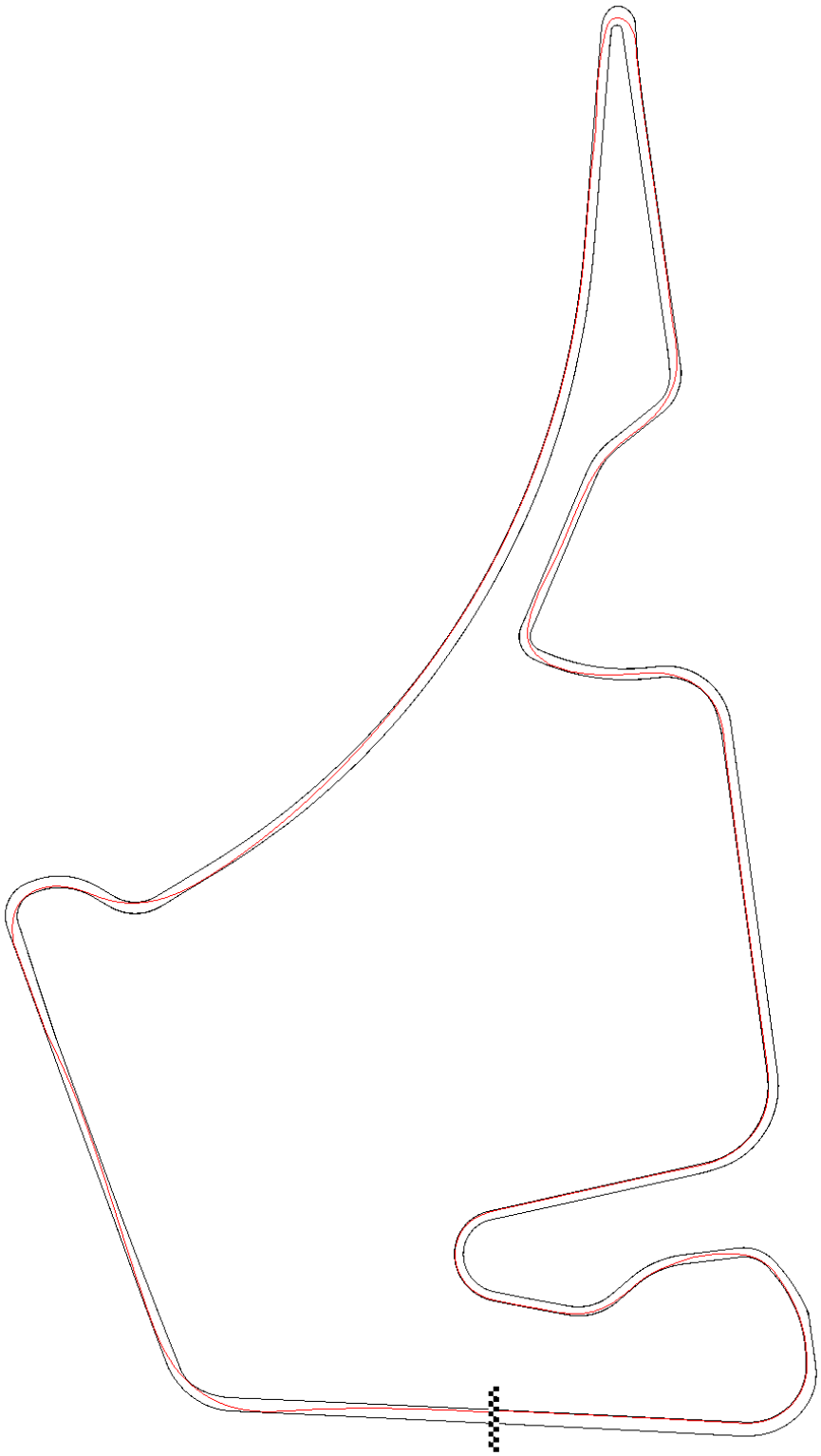


Figure 6.10 – Optimal racing line for a complete lap with a Porsche CS on the *Hockenheimring*, produced with the optimal integer solution of the vertical position to the midline. The driving direction is clockwise.

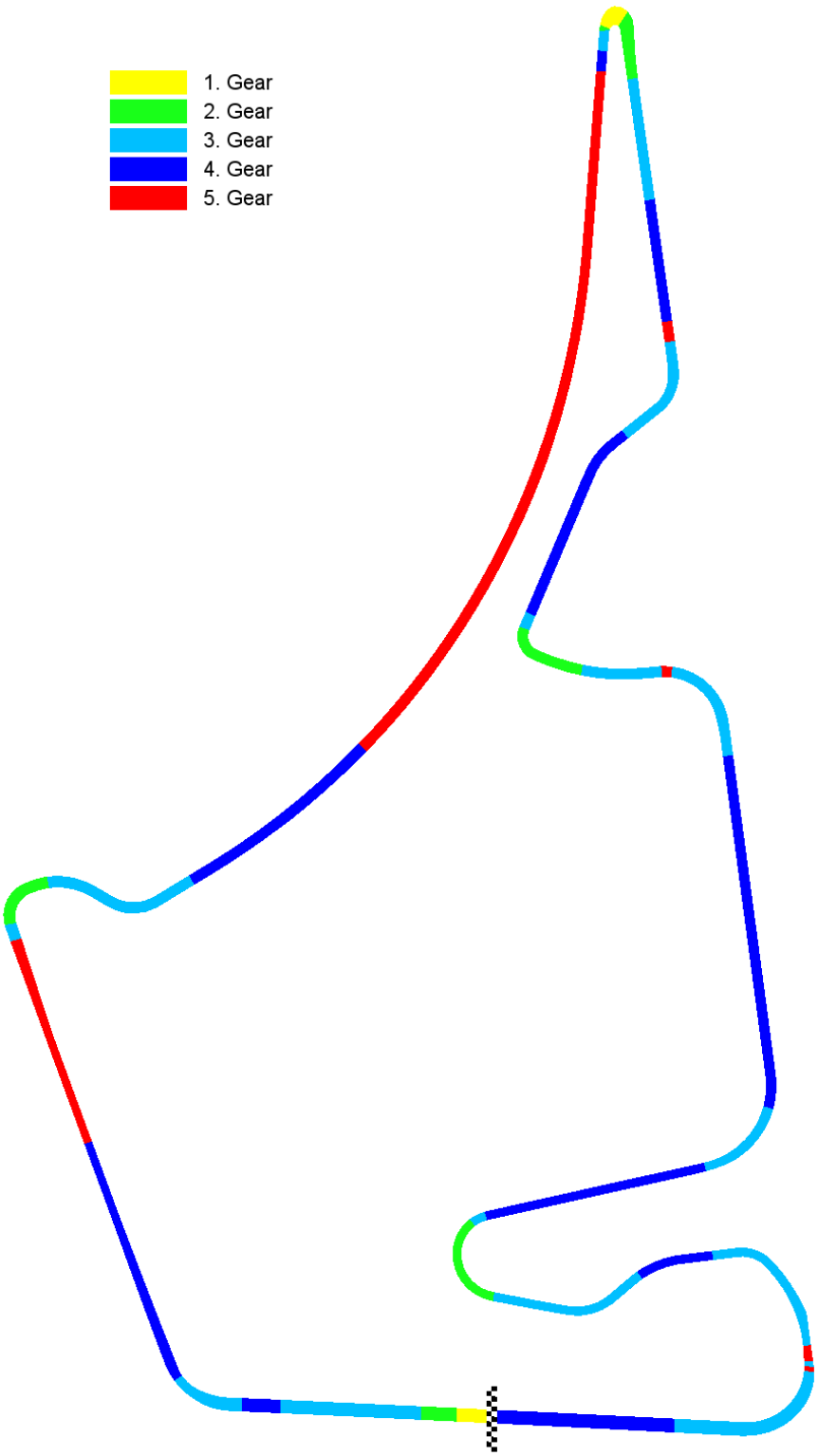
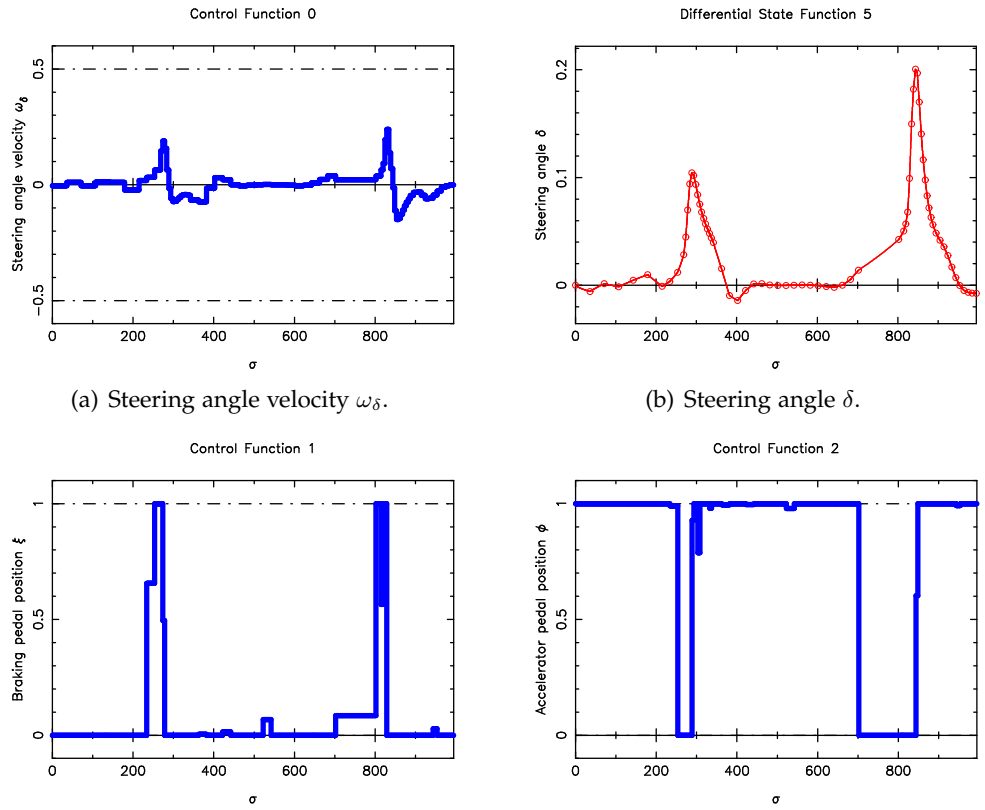


Figure 6.11 – Gear shifting for a complete lap with a PORSCHE CS on the *Hockenheimring*. The driving direction is clockwise. Compare Figure 6.1 on page 56.

6.3.3 2002 Formula One Car

In this section, the numerical results for an optimal control of a Formula One car are presented. The vehicle has been introduced as F1-02 in Chapter 2. As described in the previous sections, a particular heuristic is applied to the primarily computed relaxed optimal solution, which yields the optimal integer solution.

The integer solution is illustrated in the following, at a length of $\sigma_f = 943.396$m on the *Hockenheimring*. Due to the difficulties to get a feasible binary solution for the choice of seven gears (see Section 8.1) and the high dimensionality of the optimization problem, the solution is produced *only* on a partial lap.



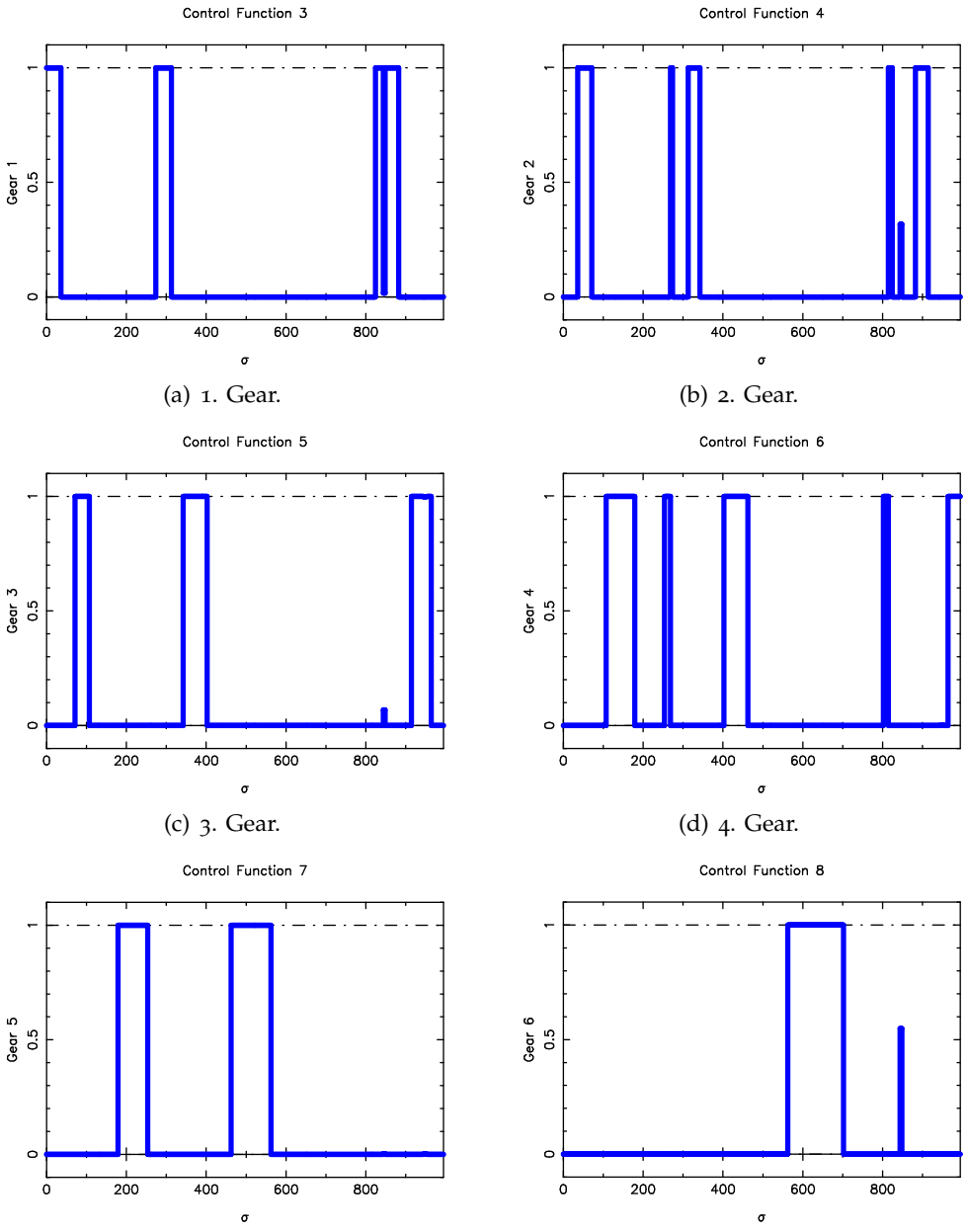
(a) Steering angle velocity ω_δ.

(b) Steering angle δ.

(c) Braking pedal position ξ.

(d) Acceleration pedal position ϕ.

Figure 6.12 – Optimal control of the steering angle velocity, as well as the differential states of steering angle, braking pedal position, and accelerating pedal position of the optimal integer solution of the F1-02.

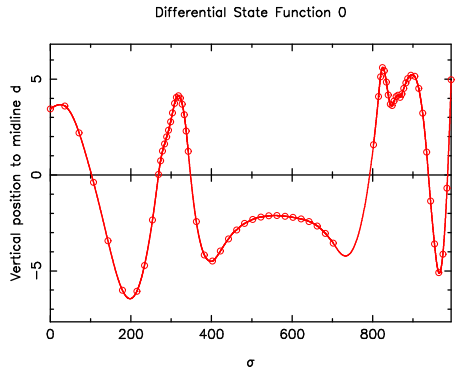


(a) 1. Gear. (b) 2. Gear.

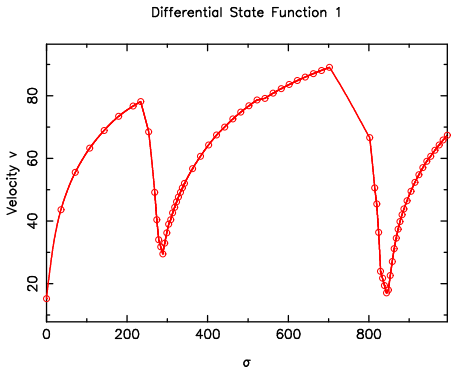
(c) 3. Gear. (d) 4. Gear.

(e) 5. Gear. (f) 6. Gear.

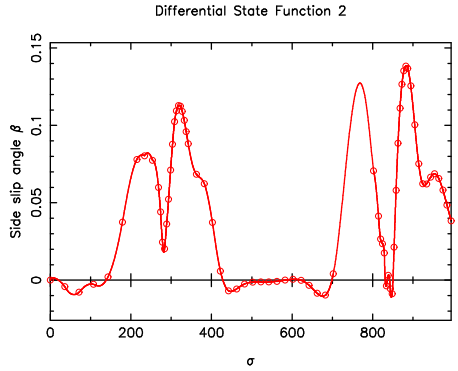
Figure 6.13 – Gear activity of the integer solution, showing the first six gears of the F1-02. The 7. Gear has been eliminated as specific binary control with the linear equality constraint of the SOS1 condition. Note that the solution is not integer feasible because of the problems mentioned in Section 8.1, and technical difficulties with an adaptation of the control grid size.



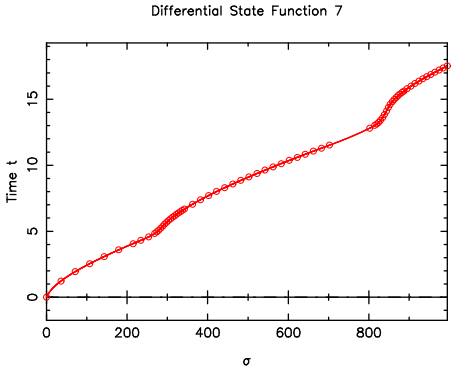
(a) Vertical position d to the midline.



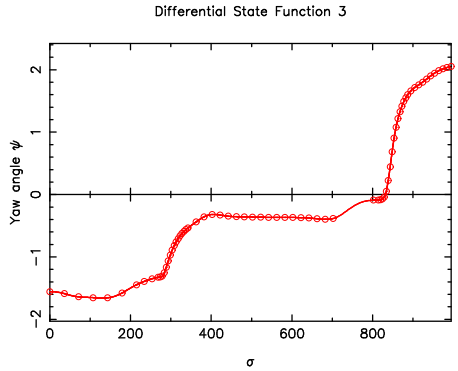
(b) Velocity v.



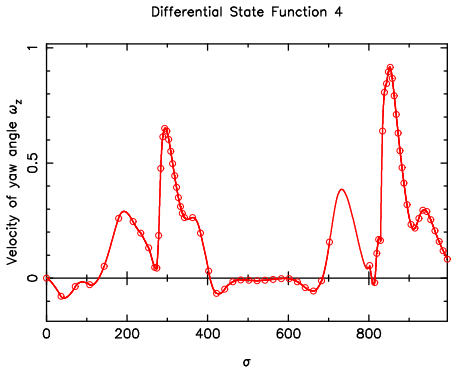
(c) Side slip angle β.



(d) Time t.



(e) Yaw angle ψ.



(f) Yaw angle ω_z.

Figure 6.14 – Differential states of the optimal integer solution of the F1-02.

Figure 6.15 – Gear shifting for a partial lap with the F1-02 on the *Hockenheimring*, compare to the gear shifting of the PORSCHE CS in the Figure 6.11.

Chapter 7

Integration of Numerical Results to Racing Simulator

So far we tried to attach parts of the racing simulation VDrift into the optimization tool Muscod-II to realize the optimal control of a VDrift car on a realistic racing track. This chapter shows the other way around, which means that we try to integrate the offline calculated optimal solution for a specific track into VDrift. In this context, "offline" means that once the optimization process started, there are no further information updates used in the calculation.

Therefore, we have to implement input possibilities within VDrift's source code for the control data of a vehicle. The optimized solution is illustrated in several exemplary screenshots of VDrift.

7.1 Input/Output Operations

First of all, the challenge is to find a way to include the computed optimal control of a vehicle into VDrift. There is the possibility to use the human controlled car, which is normally controlled e.g. by the keyboard, or to include the input data into the AI controlled driver. We decided to use the AI driver for this purpose, which results in the opportunity to directly challenge the optimized car. Additionally it is possible to add a VDrift controlled AI driver as third competitor.

Every input operation has to be at a particular time, which equates the position of the multiple shooting points in the optimization problem. For the Porsche CS this position is every second Bézier patch intersection.

Furthermore, the adjustment of the controls is important. The acceleration pedal in both VDrift and *testdrive* is controlled by $\phi \in [0, 1]$, just as the braking pedal position $\xi \in [0, 1]$, which we already changed within the *extended testdrive* formulation (see Section 2.5). However, the steering control differs. Given in the optimal control

problem by the steering angle velocity $\omega_\delta \in [-0.5, 0.5]$, it is formulated within the racing simulator by the steering wheel angle ratio $\delta_{\text{VD}} \in [-1, 1]$. At this, 1 represents the car dependent ratio to the maximum positive steering angle and -1 its negative equivalent. The essential transformation is made by the solution of the steering wheel angle δ_{tt}, formulated as state of the ODE system in *testdrive*. In this connection, δ_{tt} has to be transformed from radians into degrees:

$$\delta_{\text{VD}} = \frac{\delta_{\text{tt}}\ 180}{\delta_{\max}\ \pi} \tag{7.1}$$

The current optimal gear in VDrift is controlled in the same way as in *testdrive*, by binary variables.

Additionally, it is possible to compare "measured data" of the optimized VDrift vehicle to the solutions of controls and states of the optimization problem. Therefore, output data is written to a text file at every patch intersection.

7.2 Preliminary Considerations

We briefly want to consider the expected result, for including the offline calculated solution of Muscod-II into VDrift's AI driver. As described in Chapter 2, we tried to adjust the model of the optimal control problem *testdrive* to the VDrift car model. We included specific car dependent parameters, as well as mathematical models of different car parts of VDrift within Muscod-II. Nevertheless, the two models have still some differences, e.g. in the computation of the engine progress. This results in a slightly different velocity of the *extended testdrive* model compared to the VDrift car, due to the missing clutch torque (described in Equation 2.8).

However, the differences within one patch will probably not be very big, the error accumulates with the car's movement. Therefore, we expect the car with the offline calculated solution, to go off the track and accordingly crash into a constraining wall sooner or later.

7.3 Illustration of the Solution

The following screenshots illustrate two Porsche CS, driving on the *Hockenheimring* in the racing simulation VDrift. The red vehicle (see Figure 7.1) is controlled by input data, which is offline calculated as optimal solution of the *extended testdrive* example, implemented in Muscod-II. The black Porsche (Figure 7.1) is human controlled.

The "head-up display" (HUD) shows the current gear as well as a rotation speed indicator on the lower left screen and the actual vehicle velocity in $\frac{\text{km}}{\text{h}}$ on the lower right. On the upper left of the screenshots, the current lap time is displayed, starting on *GO* (Figure 7.2) and is turned back to zero, while the vehicle is crossing the starting

line (Figure 7.3). On the upper right, the track course is shown with the actual car positions. All HUD information refer to the optimal controlled vehicle.[1]

As previously mentioned, the AI driver starts in the first gear with maximum acceleration and no steering. While crossing the starting line, the first optimal control data is entered to the AI driver.

Figure 7.1 – Porsche CS in starting position on the *Hockenheimring*. The red car in the rear is controlled by the optimal solution, while the black car is human controlled.

[1] But the "Place" shows the human controlled car's position.

Figure 7.2 – Starting positions of the *Hockenheimring*. Due to engaging the first gear, the rotation speed indicator primarily declines.

Figure 7.3 – The optimal controlled Porsche is crossing the starting line with the velocity of $11.45\frac{\mathrm{m}}{\mathrm{s}} \mathrel{\widehat{=}} 41.22\frac{\mathrm{km}}{\mathrm{h}}$, which equates the fixed initial value of the first multiple shooting point (compare Section 6.2). The lap time is turned back to zero, while entering the first lap.

Figure 7.4 – Driving in the first gear with $58\frac{\text{km}}{\text{h}}$, while the revolutions per minute reaches the upper engine constraint.

Figure 7.5 – *Hockenheimring* starting line after about 50m with pit lane on the right - heading towards the first curve.

Figure 7.6 – Although the steering angle only slowly increases, the car starts to drive to the left with a higher amount of velocity.

Figure 7.7 – The optimized Porsche CS slightly touches the grass - driving in the fourth gear now - while steering right at the transition curve.

Figure 7.8 – Backspacing at maximum velocity within the curve. As can be seen, the optimal controlled car is a bit early in the curve, due to the slightly different mathematical engine models, which results in different velocity of the optimized offline model and the "real world".

Figure 7.9 – Consequently the AI car turns off the road, due to the early turn in to the curve.

Figure 7.10 – Due to different off-road steering characteristics, the AI driver is now late for straightening the vehicle. Furthermore, the off-road ride naturally reduced the car's velocity.

Figure 7.11 – Hence, the AI car touches the right, delimiting wall and additionally steers to the left, due to the intention to straighten the car.

Figure 7.12 – Consequently, the AI driver heads up way to much left.

Figure 7.13 – Finally, the AI car ends up crashing in the delimiting, left wall.

Chapter 8

Conclusion and Outlook

In Chapter 6, we presented the offline calculated optimal control of a PORSCHE CS, driving a complete lap on the *Hockenheimring*. In the previous chapter, we used this offline solution as input data in VDRIFT's AI driver. Although the mathematical model of the *extended testdrive* MIOCP has been adjusted to VDRIFT's car model, the previous section shows that there are still some differences. As suggested, the optimized vehicle crashes sooner or later. Unfortunately, this happens already within the first curve. Additional difficulties appeared in achieving the optimal control of a complete lap due to *ill-conditioning* and *vanishing constraints*, which we describe in the following.

Then we like to present two different approaches to solve some of these problems as an outlook of this thesis. This could be adjustment of the mathematical car models, which yield negligible differences. Otherwise, we illustrate the idea of *Nonlinear Model Predictive Control* (NMPC), compare Wirsching [38], in which we see a great opportunity to proceed with the results that have been achieved in this thesis. This final chapter is closed with a short summary of the main contributions, achieved in the context of this thesis.

8.1 Vanishing Constraints and Ill-Conditioning

This section should deliver an insight into the difficulties that could be seen in Chapter 6, on the basis of the numerical results. Therefore, we discuss some technical and numerical properties which appear by applying the Outer Convexification approach to integer or binary control constraints of a NLP. The problem class of *Mathematical Programs with Vanishing Constraints (MPVCs)* includes such NLPs. Furthermore, the issue of *ill-conditioning* often occurs with the linearized constraints by Outer Convexification and is described in the following as well.

Definition 8.1. ***Nonlinear Program with Vanishing Constraints***
The Nonlinear Program

$$\min_{x \in \mathbb{R}^n} \quad f(x) \tag{8.1a}$$

$$\text{s.t.} \quad g_i(x) \cdot h_i(x) \geq 0, \qquad i \in \{1, \ldots, m\} \tag{8.1b}$$

$$h(x) \geq 0, \tag{8.1c}$$

with $m \geq 1$ complementary inequality constraints, is called Mathematical Program with Vanishing Constraints (MPVC).

Furthermore, the following problem illustrates the specific case of a NLP, obtained from discretization of an optimal control problem and additional linearization of the integer controls by Outer Convexification.

The discretized system states are given by x, whereas u describes the discretized continuous control parameters, and the relaxed binary controls are denoted by w:

$$\min_{x,u,w} \quad f(x,u,w) \tag{8.2a}$$

$$\text{s.t.} \quad w_i \cdot (g_i(x,u) - g_i^{\text{lo}}) \geq 0, \qquad i \in \{1, \ldots, m\} \tag{8.2b}$$

$$w_i \cdot (g_i^{\text{up}} - g_i(x,u)) \geq 0, \qquad i \in \{1, \ldots, m\} \tag{8.2c}$$

$$w_i \in [0,1] \subset \mathbb{R} \qquad i \in \{1, \ldots, m\} \tag{8.2d}$$

The constraints g_i may be two-sided with lower bounds g_i^{lo} and upper bounds g_i^{up}, and vanish if the associated relaxed binary control w_i is zero. This happens at every gear shift point from one gear to another, at the example of the optimal control problem of a vehicle including gear shifts. However, the fact of ill-conditioning can trouble that.

Ill-conditioning can be explained at the following example. Consider the scalar constraint $w_1 \cdot g(x_2) \geq 0$ in combination with its associated simple bound $w_1 \geq 0$. The possibly nonlinear constraint g on x_2, given as the upper engine constraint for instance, vanishes if the particular gear is inactive with $w_1 = 0$. As w_1 approaches zero, the condition number of the problem approaches infinity.

Due to roundoff and truncation errors inevitably experienced by any numerical algorithm applied to solve MPVCs, the bound $x_2 \geq 0$ may be only *weakly* active. For example if $w_1 = 10^{-12}$ instead of $w_1 = 0$, the condition number is at least $\frac{1}{w_1} = 10^{12}$ if the vanishing constraint g is active. Hence, it is remarked that in a neighborhood of the area of violation, linearizations of constraints treated by Outer Convexification are ill-conditioned.

Besides ill-conditioning, a phenomenon known as "zig-zagging" could appear in active set methods for the solution of MPVCs. Here the angle between the constraints

approaches 0, which yields a spike-shaped form of the feasible set. This may result in many tiny steps being made by an active set method if the stationary point is located near the spike's pinpoint.

Zig-zagging could possibly appear in the neighborhood of a gear shift point. The optimal gear shift point at a specific vehicle velocity lies in a small feasible set. It is mentioned that this could be one reason, why the solution for an optimal control on the *Hockenheimring* with the Formula One car F1-02 caused considerably more trouble compared to the PORSCHE CS, see Figure 2.11.

8.2 Model Adjustment

The main difference of the two mathematical car models of *testdrive* and VDRIFT can be reduced to the engine. As described in Section 2.5, the engine's clutch torque is neglected in the *testdrive* model.

The clutch torque represents the friction whenever one side of the clutch is spinning faster than the other side. As mentioned, within VDRIFT this is calculated with the difference of the engine's angular velocity and the clutch's angular velocity and further clutch parameters (compare Equation 2.8). At the engine, the angular velocity comes with a numerical integration, by a "forward" calculation with an implemented Euler method. On the other hand, the clutch's angular velocity is calculated "backwards", out the actual wheel speed of the car. Hence, to model the clutch torque within the *extended testdrive* version, it is necessary to introduce a new ODE state for the forward calculation of the engine's angular velocity.

The tire forces vary in longitudinal direction described in Equations 2.29, 2.30 to VDRIFT's longitudinal Pacejka tire model, which is described in Equation 2.14. Additionally, the aligning moment is modeled in VDRIFT compared to *testdrive*.

It would be interesting to see in which way these adjustments improve the in VDRIFT entered solution. Although the two mathematical models would be close now, they would still not match exactly. Therefore we suppose that those adjustments which are possible to include to *testdrive* with a practicable effort, still would not have the ability to compute a complete optimal controlled lap on a VDRIFT track.

Hence, we intend to use *Nonlinear Model Predictive Control* (NMPC), which is presented in the following section. Nevertheless, the model adjustments would be an additional improvement, while doing NMPC.

8.3 Nonlinear Model Predictive Control

The idea of an offline calculated optimal control of a dynamic system, involves the solution of the entire problem on a fixed time horizon $[t_0, t_f]$, as seen in Chapter 6.

Afterwards, the offline calculated control is applied to the dynamic system.

The problems appearing with this approach are illustrated at the example of VDrift screenshots in this chapter, caused by a not exactly coincident mathematical model. Further problems could be subjected to disturbances appearing during the process, for example to avoid hitting an obstacle while driving a car or countersteer after a slight contact with an opponent vehicle.

Hence, we now want to describe the idea of a predictive control approach, which operates in a way to control the system during the process. The main applications in *Model Predictive Control* (MPC) are permanent processes, which are not stuck to a finite time horizon. Most common examples are permanent heating or cooling processes or chemical procedures.

A wide spread field of theory research as well as many industrial applications can be found in *Linear Model Predictive Control* (LMPC), in which a linear model is used to predict the system dynamics. However, many real world applications include nonlinearities, which can be treated by a *Nonlinear Model Predictive Control* (NMPC). NMPC is a much younger and less explored sector of MPC. This is mainly caused by the computational challenge, which appears with nonlinear models and constraints. To fasten up the solving process, so far the nonlinear systems often have been replaced by linear differential equations and constraints. This works well for systems which nonlinearity isn't that drastic.

However, to solve nonlinear optimal control problems including a higher control performance, the major difficulty lies in the real-time requirements of the solution. Hence, faster nonlinear MPC algorithms have been developed recently, see Diehl [10]. Furthermore, the time horizon on which the optimal control problem is solved has to be considered more precisely.

8.3.1 Moving Horizon

Ideally the control problem should be solved on the complete time horizon $[t_0, t_f]$ or for $T = \infty$ for permanent processes with a long time horizon $[t_0, T]$. However, this would lead to large-scale NLPs, which could not be solved in real-time. Therefore, the concept of NMPC is generally formulated on a moving horizon. At this, the idea is to solve one optimal control problem after another, each independently with the required accuracy on a short real-time capable time horizon.

We now want to solve an optimal control problem, formulated as in 4.9, with the system states x_i, measured at the current time t_i, for a predictive time horizon $[t_i, t_i+T]$ and based on the ODE model

$$\dot{x}(t) = f\big(t, x(t), u(t), p\big). \tag{8.3}$$

The following explanations are illustrated in Figure 8.1.

The first optimized control is applied to the system at time t_i to control the first segment $[t_i, t_i + \epsilon],\ \epsilon < T$ of the time horizon $[t_i, t_i + T]$. The next system state is taken

at latest at $t_i + \epsilon$. Now a new optimization problem is solved for the next time horizon $[t_i + \epsilon, t_i + \epsilon + T]$ and given back to the system to control the next segment $[t + \epsilon, t + 2\epsilon]$. This procedure can be continued any number of times.

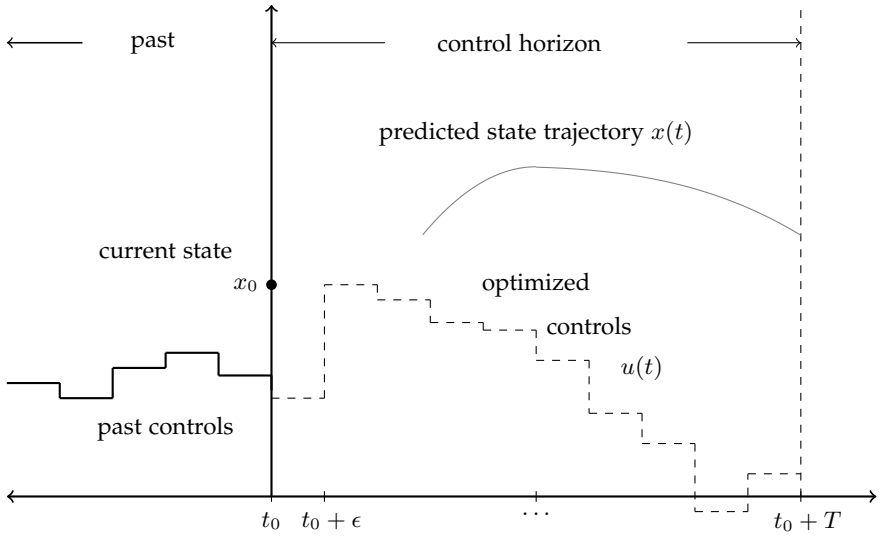


Figure 8.1 – Illustration of Model Predictive Control on a moving time horizon.

Obviously the computing time varies much, depending on the length of the time horizon, the difficulty of the optimization problem, as well as the quality of the initial values for the next iteration step. Hence, some ideas have been proposed by Diehl in [10, 12, 11], to fasten up the algorithms for real-time calculation in combination with a *direct multiple shooting* and SQP approach.

The basic difference in this approach in contrary to the conventional NMPC scheme, is to use the already calculated solution of the first time horizon $[t_i, t_i + T]$ as initial value x_{i+1} of the next interval $[t_i + \epsilon, t_i + \epsilon + T]$. Furthermore, the measured state $\hat{x}_{i+1}$ at the end of the first interval is added to the new optimization problem as constraint

$$x(t_{i+1}) - \hat{x}_{i+1} = 0. \tag{8.4}$$

That means the solution algorithm is iterating continuously, while the problem data is modified in every iteration step. In addition, the optimal control u_i can be used as initial control data for the next iteration step. This allows to reduce the number of iterations to one iteration per sampling time, to calculate the optimal control of the next time horizon.

While choosing the length of the time horizon T, two different options have to be considered. If T is chosen too big, the solution will probably not be reached in real-time, due to the fact that the difficulty increases with the problem progress. Otherwise,

if the time horizon is chosen too small, the calculate solution may no longer be optimal, examining the global problem.

8.4 Summary

At first, we presented a historical overview of racing video games, different open source racing simulations and the reasons why we actually chose the game VDrift for coupling to an optimization software. Afterwards, we modified the existing optimal control problem *testdrive* included in Muscod-II, using a transformation of the underlying ODE system. In this context, we improved the mathematical model of *testdrive* with certain car parts and more realistic parameters of VDrift.

The following transformation of the coordinate system allows to calculate the optimal control of a vehicle in the *testdrive* problem, using optional tracks of VDrift which are given as Bézier points. We gave an overview of the *direct multiple shooting* method in combination with a SQP algorithm to solve general NLPs. Furthermore, we showed the Outer Convexification approach with a relaxation of the binary control constraints, to solve the MIOCPs, which are treated in this thesis.

The model extensions are illustrated at the explicit example of a smoothed test course of the original *testdrive* track. These solutions are compared to the original *testdrive* version results.

In Chapter 6 we applied the described improvements in car and track model to optimize the control of a Porsche CS on a complete lap of the *Hockenheimring*. At this, the difficulties are explained appearing due to the complexity of this problem, on the one hand with the very high number of multiple shooting nodes, and otherwise with the shifting decisions related to the engine constraints. Additionally the offline calculated optimal control of the Formula One car F1-02 with seven gears is shown on the first half of the *Hockenheimring*. Here, the complexity of the problem made it impossible to solve it on a complete lap all at once, even with primarily partial optimized initial values.

Further difficulties appear, when the offline calculated optimal solution is entered into the control of VDrift's AI driver. Additional implementations within VDrift's source code, allow input/output operations in the racing simulator. At several exemplary screenshots can be seen that even for the adjusted car models of *testdrive* and VDrift, their mismatch is to big. Hence, the car crashes after the first curve.

Finally an outlook is given with further model adjustment as well as the idea of NMPC that should solve the previously described problems.

Now, our intention is to focus on NMPC, to solve the optimal control problem of a vehicle on a complete racing track not only as offline solution, but in real-time on a moving horizon within the racing simulator VDrift.

Appendix A

Car Parameters of VDrift

This appendix describes parameters of two car models of VDrift, read in from the car file (*/VDrift/data/cars/carname/carname.car*) and not yet listed in section 2.3.1. On the one hand, there is the Porsche 911 Club Sport, a street car from the late 1980's with a racing engine under the hood. On the other hand, a Formula 1 car of the year 2002, with realistic Formula One values according to the FIA Regulations.

- Front wheels must be between 305 and 355 mm wide, the rears between 365 and 380 mm
- With tyres fitted the wheels must be no more than 660 mm in diameter (670 mm with wet-weather tyres)
- "Cars must weight at least 605 kg (including the driver) at all times" (total weight 673 kg at start)
- weight distribution $\frac{\text{front}}{\text{rear}} = \frac{44}{56}$

These are the two main models, used in this thesis.

Parameter	Value Porsche	Value F1-02	Unit
torque-curve-00	1000, 189.81	1800, 90.00	$\frac{1}{min}$, Nm
⋮	2000, 196.59	2600, 110.59	
	2200, 196.59	3300, 135.59	
	2300, 203.37	3750, 160.21	
	2800, 196.59	4000, 190.47	
	3000, 203.37	4250, 202.15	
	3400, 212.86	4500, 211.56	
	3800, 203.37	5000, 218.78	
	4200, 221.0	5500, 225.23	
	4800, 237.27	6000, 237.37	
	5100, 238.62	6500, 238.89	
	5300, 242.69	7000, 242.69	
	5400, 239.98	7500, 243.43	
	5600, 235.91	8000, 244.91	
	5700, 237.27	8500, 246.27	
	5900, 237.95	9000, 248.95	
	6200, 221.0	9600, 250.20	
	6400, 207.44	10000, 259.65	
	6600, 196.59	10500, 269.78	
	6840, 184.39	11000, 295.51	
		12000, 320.00	
		13000, 330.00	
		14000, 347.00	
		15000, 330.00	
		16000, 320.00	
		17000, 310.00	
		18000, 300.00	
		19000, 290.00	
torque-curve-21		20000, 280.00	

Table A.1 – Torque-curve parameters (rpm, torque) used in the VDrift engine model.

Parameter	Value Porsche	Value F1-02	Unit	Description
a_0	1.4	1.39	–	Shape factor
a_1	-35	-90	$\frac{1}{\text{MN}}$	Load infl. on lat. friction coeff
a_2	1550	1900	$\frac{1}{1000}$	Lateral friction coefficient at load = 0
a_3	2400	2900	N	Maximum stiffness
a_4	6.5	7.2	kN	Load at maximum stiffness
a_5	0.014	0.024	$\frac{1}{\text{deg}}$	Camber influence on stiffness
a_6	-0.24	-0.24	kN	Curvature change with load
a_7	1.0	1.0	–	Curvature at load = 0
a_8	-0.03	-0.03	–	Horizontal shift because of camber
a_9	-0.0013	-0.0013	$\frac{\text{deg}}{\text{kN}}$	Load influence on horizontal shift
a_{10}	-0.15	-0.15	deg	Horizontal shift at load = 0
a_{11_1}	-8.5	-8.5	$\frac{1}{\text{MN}\cdot\text{deg}}$	Camber influence on vertical shift
a_{11_2}	-0.29	-0.29	$\frac{1}{\text{kdeg}}$	Camber influence on vertical shift
a_{12}	17.8	17.8	$\frac{1}{1000}$	Load influence on vertical shift
a_{13}	-2.4	-2.4	N	Vertical shift at load = 0

Table A.2 – Lateral Pacejka parameters used in the VDrift tire model acting at the front wheels, units according to [14].

Parameter	Value Porsche	Value F1-02	Unit	Description
a_0	1.3	1.55	–	Shape factor
a_1	-45	-50	$\frac{1}{\text{MN}}$	Load infl. on lat. friction coeff
a_2	1700	2000	$\frac{1}{1000}$	Lateral friction coefficient at load = 0
a_3	2500	2800	N	Maximum stiffness
a_4	6.5	10.0	kN	Load at maximum stiffness
a_5	0.014	0.024	$\frac{1}{\text{deg}}$	Camber influence on stiffness
a_6	-0.24	-0.24	kN	Curvature change with load
a_7	1.0	1.0	–	Curvature at load = 0
a_8	-0.03	-0.03	–	Horizontal shift because of camber
a_9	-0.0013	-0.0013	$\frac{\text{deg}}{\text{kN}}$	Load influence on horizontal shift
a_{10}	-0.15	-0.15	deg	Horizontal shift at load = 0
a_{11_1}	-8.5	-8.5	$\frac{1}{\text{MN}\cdot\text{deg}}$	Camber influence on vertical shift
a_{11_2}	-0.29	-0.29	$\frac{1}{\text{kdeg}}$	Camber influence on vertical shift
a_{12}	17.8	17.8	$\frac{1}{1000}$	Load influence on vertical shift
a_{13}	-2.4	-2.4	N	Vertical shift at load = 0

Table A.3 – Lateral Pacejka parameters used in the VDrift tire model acting at the rear wheels, units according to [14].

Parameter	Value Porsche	Value F1-02	Unit	Description
b_0	1.6	1.95	–	Shape factor
b_1	-70	-85	$\frac{1}{\mathrm{MN}}$	Load infl. on long. friction coeff
b_2	1600	1950	$\frac{1}{1000}$	Longitudinal friction coefficient at load = 0
b_3	23.3	24.3	$\frac{1}{\mathrm{MN}}$	Curvature factor of stiffness
b_4	350	390	$\frac{1}{1000}$	Change of stiffness with load at load = 0
b_5	0.05	0.07	$\frac{1}{\mathrm{kN}}$	Change of progressivity of stiffness/load
b_6	0.0	0.0	$\frac{1}{\mathrm{kN}^2}$	Curvature change with load
b_7	0.055	0.059	$\frac{1}{\mathrm{kN}}$	Curvature change with load
b_8	-0.024	-0.024	–	Curvature at load = 0
b_9	0.014	0.014	$\frac{1}{\mathrm{kN}}$	Load influence on horizontal shift
b_10	0.26	0.26	–	Horizontal shift at load = 0
b_11	0.0	0.0	$\frac{\mathrm{N}}{\mathrm{kN}}$	Load influence on vertical shift
b_12	0.0	0.0	N	Vertical shift at load = 0

Table A.4 – Longitudinal Pacejka parameters used in the VDrift tire model acting at the front wheels, units according to [14].

Parameter	Value Porsche	Value F1-02	Unit	Description
b_0	1.6	1.75	–	Shape factor
b_1	-90	-100	$\frac{1}{\mathrm{MN}}$	Load infl. on long. friction coeff
b_2	1600	2200	$\frac{1}{1000}$	Longitudinal friction coefficient at load = 0
b_3	23.3	23.3	$\frac{1}{\mathrm{MN}}$	Curvature factor of stiffness
b_4	375	410	$\frac{1}{1000}$	Change of stiffness with load at load = 0
b_5	0.05	0.075	$\frac{1}{\mathrm{kN}}$	Change of progressivity of stiffness/load
b_6	0.0	0.0	$\frac{1}{\mathrm{kN}^2}$	Curvature change with load
b_7	0.055	0.055	$\frac{1}{\mathrm{kN}}$	Curvature change with load
b_8	-0.024	-0.024	–	Curvature at load = 0
b_9	0.014	0.014	$\frac{1}{\mathrm{kN}}$	Load influence on horizontal shift
b_10	0.26	0.26	–	Horizontal shift at load = 0
b_11	0.0	0.0	$\frac{\mathrm{N}}{\mathrm{kN}}$	Load influence on vertical shift
b_12	0.0	0.0	N	Vertical shift at load = 0

Table A.5 – Longitudinal Pacejka parameters used in the VDrift tire model acting at the rear wheels, units according to [14].

Parameter	Value Porsche	Value F1-02	Unit	Description
c_0	2.2	2.2	–	Shape factor
c_1	-3.9	-2.3	$\frac{\text{Nm}}{\text{kN}^2}$	Load influence of peak value
c_2	-3.9	-2.4	$\frac{\text{Nm}}{\text{kN}}$	Load influence of peak value
c_3	-1.26	0.0	$\frac{\text{Nm}}{\text{deg}\cdot\text{kN}^2}$	Curvature factor of stiffness
c_4	-8.2	-1.5	$\frac{\text{Nm}}{\text{deg}\cdot\text{kN}}$	Change of stiffness with load at load = 0
c_5	0.025	0.0225	$\frac{1}{\text{kN}}$	Change of progressivity of stiffness/load
c_6	0.0	0.0	$\frac{1}{\text{deg}}$	Camber influence on stiffness
c_7	0.044	0.044	–	Curvature change with load
c_8	-0.58	-0.58	–	Curvature change with load
c_9	0.18	0.18	–	Curvature at load = 0
c_{10}	0.043	0.043	–	Camber influence of stiffness
c_{11}	0.048	0.048	–	Camber influence on horizontal shift
c_{12}	-0.0035	-0.0035	$\frac{\text{deg}}{\text{kN}}$	Load influence on horizontal shift
c_{13}	-0.18	-0.18	deg	Horizontal shift at load = 0
c_{14}	0.14	0.14	$\frac{\text{Nm}}{\text{deg}\cdot\text{kN}^2}$	Camber influence on vertical shift
c_{15}	-1.029	-1.029	$\frac{\text{Nm}}{\text{deg}\cdot\text{kN}}$	Camber influence on vertical shift
c_{16}	0.27	0.27	$\frac{\text{Nm}}{\text{kN}}$	Load influence on vertical shift
c_{17}	-1.1	-1.1	Nm	Vertical shift at load = 0

Table A.6 – Aligning Pacejka parameters used in the VDrift tire model acting at the front wheels, units according to VDrift's Wiki Documentation [2]

Parameter	Value Porsche	Value F1-02	Unit	Description
c_0	2.2	2.2	–	Shape factor
c_1	-4.1	-4.3	$\frac{\text{Nm}}{\text{kN}^2}$	Load influence of peak value
c_2	-3.9	-4.4	$\frac{\text{Nm}}{\text{kN}}$	Load influence of peak value
c_3	-1.36	-1.9	$\frac{\text{Nm}}{\text{deg}\cdot\text{kN}^2}$	Curvature factor of stiffness
c_4	-8.0	-9.6	$\frac{\text{Nm}}{\text{deg}\cdot\text{kN}}$	Change of stiffness with load at load = 0
c_5	0.025	0.0225	$\frac{1}{\text{kN}}$	Change of progressivity of stiffness/load
c_6	0.0	0.0	$\frac{1}{\text{deg}}$	Camber influence on stiffness
c_7	0.044	0.044	–	Curvature change with load
c_8	-0.58	-0.58	–	Curvature change with load
c_9	0.18	0.18	–	Curvature at load = 0
c_{10}	0.043	0.043	–	Camber influence of stiffness
c_{11}	0.048	0.048	–	Camber influence on horizontal shift
c_{12}	-0.0035	-0.0035	$\frac{\text{deg}}{\text{kN}}$	Load influence on horizontal shift
c_{13}	-0.18	-0.18	deg	Horizontal shift at load = 0
c_{14}	0.14	0.14	$\frac{\text{Nm}}{\text{deg}\cdot\text{kN}^2}$	Camber influence on vertical shift
c_{15}	-1.029	-1.029	$\frac{\text{Nm}}{\text{deg}\cdot\text{kN}}$	Camber influence on vertical shift
c_{16}	0.27	0.27	$\frac{\text{Nm}}{\text{kN}}$	Load influence on vertical shift
c_{17}	-1.1	-1.1	Nm	Vertical shift at load = 0

Table A.7 – Aligning Pacejka parameters used in the VDrift tire model acting at the rear wheels, units according to VDrift's Wiki Documentation [2]

Bibliography

[1] Picture of airplane from website. `http://mtp.jpl.nasa.gov/notes/pointing/Aircraft_Attitude2.png`, March 2010.

[2] Various Authors. VDrift Documentation Wiki. `http://wiki.vdrift.net/Numerical_Integration`, March 2010.

[3] Brian Beckman. The physics of racing, 1991-2008.

[4] R.E. Bellman. *Dynamic Programming*. University Press, Princeton, 1957.

[5] J.T. Betts. *Practical Methods for Optimal Control Using Nonlinear Programming*. SIAM, Philadelphia, 2001.

[6] T. Binder, L. Blank, H.G. Bock, R. Bulirsch, W. Dahmen, M. Diehl, T. Kronseder, W. Marquardt, J.P. Schlöder, and O.v. Stryk. Introduction to model based optimization of chemical processes on moving horizons. In M. Grötschel, S.O. Krumke, and J. Rambau, editors, *Online Optimization of Large Scale Systems: State of the Art*, pages 295–340. Springer, 2001.

[7] H.G. Bock and K.J. Plitt. A Multiple Shooting algorithm for direct solution of optimal control problems. In *Proceedings 9th IFAC World Congress Budapest*, pages 243–247. Pergamon Press, 1984. Available at http://www.iwr.uni-heidelberg.de/groups/agbock/FILES/Bock1984.pdf.

[8] A. Buchner. Auf dynamischer programmierung basierende nichtlineare modellprädiktive regelung für LKW. Diploma thesis, Ruprecht–Karls–Universität Heidelberg, January 2010.

[9] Rémi Coulom. *Reinforcement Learning Using Neural Networks, with Applications to Motor Control*. PhD thesis, Institut National Polytechnique de Grenoble, 2002.

[10] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*. PhD thesis, Universität Heidelberg, 2001.

[11] M. Diehl, H.G. Bock, and J.P. Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5):1714–1736, 2005.

[12] M. Diehl, H.G. Bock, J.P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *J. Proc. Contr.*, 12(4):577–585, 2002.

[13] M. Diehl, D.B. Leineweber, and A.A.S. Schäfer. MUSCOD-II Users' Manual. IWR-Preprint 2001-25, Universität Heidelberg, 2001.

[14] G. Genta. *Motor Vehicle Dynamics - Modeling and Simulation*. World Scientific Publishing Co. Pte. Ltd., Singapore, 1997.

[15] M. Gerdts. Solving mixed-integer optimal control problems by Branch&Bound: A case study from automobile test-driving with gear shift. *Optimal Control Applications and Methods*, 26:1–18, 2005.

[16] M. Gerdts. A variable time transformation method for mixed-integer optimal control problems. *Optimal Control Applications and Methods*, 27(3):169–182, 2006.

[17] S.P. Han. Superlinearly convergent variable-metric algorithms for general nonlinear programming problems. *Mathematical Programming*, 11:263–282, 1976.

[18] E. Hellström, M. Ivarsson, J. Aslund, and L. Nielsen. Look-ahead control for heavy trucks to minimize trip time and fuel consumption. *Control Engineering Practice*, 17:245–254, 2009.

[19] Aerial picture of the Hockenheimring from website. `http://archive2008.jimclark-revival.com/fileadmin/data/2008/img/Luftbild-Hockenheimring.jpg`, March 2010.

[20] Illustration of the Hockenheimring from website with advised gear shifting for a BMW M3 CSL. `http://www.sportauto-online.de/news/grand-prix-strecke-hockenheim-3-1051082.html`, March 2010.

[21] List of lap times driven by different cars on the Hockenheimring from website. `http://www.fastestlaps.com/index.php?page_id=track&track=46&filter1=true&filter2=true&filter3=true&filter4=true`, March 2010.

[22] C. Kirches, S. Sager, H.G. Bock, and J.P. Schlöder. Time-optimal control of automobile test drives with gear shifts. *Optimal Control Applications and Methods*, 30(5), September/October 2009. DOI 10.1002/oca.892.

[23] D.B. Leineweber. Analyse und Restrukturierung eines Verfahrens zur direkten Lösung von Optimal-Steuerungsproblemen. Master's thesis, Universität Heidelberg, 1995.

[24] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Verlag, Berlin Heidelberg New York, 2nd edition, 2006. ISBN 0-387-30303-0.

[25] Hans B. Pacejka. *Tyre and Vehicle Dynamics*. Elsevier Ltd., Oxford Burlington, 2nd edition, 2006.

[26] K.J. Plitt. Ein superlinear konvergentes Mehrzielverfahren zur direkten Berechnung beschränkter optimaler Steuerungen. Diploma thesis, Universität Bonn, 1981.

[27] L.S. Pontryagin, V.G. Boltyanski, R.V. Gamkrelidze, and E.F. Miscenko. *The Mathematical Theory of Optimal Processes*. Wiley, Chichester, 1962.

[28] A. Potschka, H.G. Bock, and J.P. Schlöder. A minima tracking variant of semi-infinite programming for the treatment of path constraints within direct solution of optimal control problems. *Optimization Methods and Software*, 24(2):237–252, 2009.

[29] M.J.D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In G.A. Watson, editor, *Numerical Analysis, Dundee 1977*, volume 630 of *Lecture Notes in Mathematics*, Berlin, 1978. Springer.

[30] S. Sager. *Numerical methods for mixed–integer optimal control problems*. Der andere Verlag, Tönning, Lübeck, Marburg, 2005. ISBN 3-89959-416-9. Available at http://sager1.de/sebastian/downloads/Sager2005.pdf.

[31] S. Sager. Reformulations and algorithms for the optimization of switching decisions in nonlinear optimal control. *Journal of Process Control*, 19(8):1238–1247, 2009.

[32] S. Sager, C. Kirches, and H.G. Bock. Fast solution of periodic optimal control problems in automobile test-driving with gear shifts. In *Proceedings of the 47th IEEE Conference on Decision and Control (CDC 2008), Cancun, Mexico*, pages 1563–1568, 2008. ISBN: 978-1-4244-3124-3.

[33] S. Sager, G. Reinelt, and Hans Georg Bock. Direct methods with maximal lower bound for mixed-integer optimal control problems. *Mathematical Programming*, 118(1):109–149, 2009.

[34] J. Stoer and R. Bulirsch. *Numerische Mathematik 1*. Springer-Verlag, Berlin Heidelberg New York, 10th edition, 2007.

[35] S. Terwen, M. Back, and V. Krebs. Predictive powertrain control for heavy duty trucks. In *Proceedings of IFAC Symposium in Advances in Automotive Control*, pages 451–457, Salerno, Italy, 2004.

[36] Joe Venzon. VDrift Website. `http://vdrift.net`, March 2010.

[37] R.B. Wilson. *A simplicial algorithm for concave programming*. PhD thesis, Harvard University, 1963.

[38] L. Wirsching, H.J. Ferreau, H.G. Bock, and M. Diehl. An online active set strategy for fast adjoint based nonlinear model predictive control. In *Proceedings of the 7th Symposium on Nonlinear Control Systems (NOLCOS), Pretoria*, 2007.

Erklärung

Hiermit versichere ich, dass ich meine Arbeit selbständig unter Anleitung verfasst habe, dass ich keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, und dass ich alle Stellen, die dem Wortlaut oder Sinne nach anderen Werken entlehnt sind, durch Angabe der Quellen als Entlehnungen kenntlich gemacht habe.