

**Friedrich-Alexander-Universität Erlangen-Nürnberg**



**Lehrstuhl für Informationstechnik  
(Schwerpunkt Kommunikationselektronik)**

**LIKE**

Masterarbeit mit dem Thema:

**Modellfehler in optimierungsbasierter kombinierter  
Planung und Regelung für Rennwagen**

Bearbeiter	Weller Sebastian
Matrikelnr.	21777345
Studiengang	Informations und Kommunikationstechnik
Betreuer	Prof. Dr.-Ing. Jörn Thielecke Henrik Bey, M. Sc.
Beginn	08. Januar 2018
Ende	09. Juli 2018

---

# Bestätigung

---

## Erklärung:

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und, dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, den (hier Datum eintragen)\_\_\_\_\_

---

# **Danksagung**

---

Ich möchte mich bei meinen Betreuern und meiner Familie bedanken.....

---

# Thema und Aufgabenstellung

---

## **Thema:**

Modellfehler in optimierungsbasierter kombinierter Planung und Regelung für Rennwagen

## **Aufgabenstellung:**

Die Automatisierung des Fahrens schließt sowohl die Planung als auch die Regelung des Fahrzeugs mit ein. Häufig werden beide Bestandteile hierarchisch voneinander getrennt. Dies ist sinnvoll, solange das kontrollierte Fahrzeug sicher innerhalb der Aktuatorlimitierungen betrieben werden soll, oder wenn die Trennung bereits durch die Problemstellung gegeben ist (Zieltrajektorie bereits vorgegeben) [WDG<sup>+</sup>16].

In anderen Fällen, z.B. wenn die gewünschte Dynamik wie in einer Rennsituation im Grenzbereich liegt, bietet sich eine kombinierte Planung und Regelung an. In diesem Beispiel würde die Kostenfunktion eine Minimierung der Rundenzeit beinhalten, während gleichzeitig die Beschränkungen des Fahrzeugs berücksichtigt werden.

Für derartige Probleme ist die modellprädiktive Regelung (MPC) bzw. eines ihrer Derivate besonders geeignet. Dabei kommt es immer zu einem sogenannten Modellfehler, der von der Komplexität und Genauigkeit des verwendeten Modells abhängt.

Das Ziel dieser Arbeit ist es, den Abfall bei der Leistung des Regelungsansatzes durch den Modellfehler zu untersuchen. Dafür soll eine Simulation verwendet werden. Die Arbeit soll folgende Punkte beinhalten:

- Auswahl einer passenden Simulationsumgebung und deren Inbetriebnahme
- Implementierung verschiedener (gegebener) Modelle für die Simulation
- Implementierung des MPC-Ansatzes
- Entwicklung einer einfachen Evaluationsmethode, um die Leistungsfähigkeit des Reglers zu untersuchen
- Vergleich verschiedener Kombinationen aus Regler- und Simulationsmodellen

---

# Kurzzusammenfassung

---

Mit zunehmender Rechenleistung und Erfahrung der Automobilbranche mit autonomen Fahrzeugen rückt auch das Thema der selbstfahrenden Rennautos immer mehr in den Fokus. Das ROBORACE Projekt ist hier Vorreiter mit seiner ausgefeilten Hardwareplattform und dem bereits in öffentlichen Events gezeigten Fahrleistungen. Auch die Formula Student (FS) verschließt sich nicht diesem Trend und hat 2017 die Rubrik Driverless ins Leben gerufen.

Diese Masterarbeit beschreibt einen Ansatz zur Echtzeitregelung und Trajektorienplanung für ein Rennauto das in diesem Wettbewerb antreten soll. Die Basis hierfür ist ein Model Predictive Control (MPC) Algorithmus. Er vereint die Regelung und Trajektorienplanung und ist adaptiv bezüglich verschiedener Fahrsituationen und Ziele. Als Ausgangssituation wird angenommen, dass das Fahrzeug bereits eine Runde auf einem unbekannten Kurs absolviert und nun eine genaue Karte des Kurses errechnet hat. Um das MPC nutzen zu können, muss ein Fahrzeugmodell hinterlegt werden. Je genauer dieses ist, desto näher können die Grenzen des realen Fahrzeugs angenähert werden. Neben der Auslegung für das aktuellste FS-Fahrzeug des High Octane Motorsports Teams für die Driverless Umrüstung, wird untersucht, ab welchem Punkt ein kinematisches Modell nicht mehr ausreicht, um das Fahrzeug sicher auf dem Rennkurs zu halten.

---

# Abstract

---

In recent years with more and more computational power and sophisticated algorithms, autonomous cars are close to becoming a reality. An ancillary effect of this development is the emerging of driverless racecars. For this idea to become a reality, three key ingredients have to be met. A very good approximation of the location of the racecar in a virtual map, a trajectory planning algorithm and a controller that will drive the car on the computed path. In this thesis the model predictive control(MPC) algorithm is used, to combine the trajectory planning and controlling for an autonomous race car, build for the formula student driverless competition. A simulation environment will be developed to evaluate the effectiveness of the approach. One key part of this thesis is to investigate how prone to model errors the MPC-approach is and what performance can be expected for the use of a kinetic bicycle model as system description in the (MPC)-algorithm.

---

# Inhaltsverzeichnis

---

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Formula Student Driverless . . . . .	1
1.2	Problemstellung . . . . .	3
<b>2</b>	<b>Stand der Technik</b>	<b>4</b>
<b>3</b>	<b>Grundlagen</b>	<b>6</b>
3.1	Model Predictive Control . . . . .	6
3.2	Optimierung . . . . .	9
3.2.1	Simplex-Verfahren . . . . .	11
3.2.2	Innere-Punkte-Verfahren . . . . .	12
3.2.3	Automatische Ableitung . . . . .	13
3.3	Fahrzeugmodelle . . . . .	16
3.3.1	Kinematisches Modell . . . . .	16
3.3.2	Reifenmodell . . . . .	19
3.3.3	Dynamisches Fahrzeugmodell . . . . .	24
<b>4</b>	<b>MPC zur Trajektorienplanung und Regelung</b>	<b>30</b>
4.1	Fahrzeugmodell . . . . .	32
4.2	Kostenfunktionen . . . . .	33
4.3	Strecken-, und Positionsbeschränkung . . . . .	35
<b>5</b>	<b>Simulationsumgebung</b>	<b>41</b>
5.1	Python . . . . .	41
5.2	Julia . . . . .	42
5.3	Simple and Fast Multimedia Library . . . . .	45
5.4	Virtueller Rennkurs . . . . .	47
<b>6</b>	<b>Evaluierung</b>	<b>48</b>
6.1	Modellfehler . . . . .	48
6.1.1	Disziplin Acceleration . . . . .	48
6.1.2	Disziplin Skidpad . . . . .	49

6.1.3	Modellabhängige Veränderung der Orientierung . . . . .	51
6.1.4	Reifenmodelle . . . . .	53
6.2	Regelung des kinematischen Modells . . . . .	56
6.2.1	Mindestlänge des Prädiktionshorizontes . . . . .	56
6.2.2	Kostenfunktionen . . . . .	57
6.2.2.1	Verifizierung der Funktionen . . . . .	57
6.2.2.2	Berechnungszeit . . . . .	57
6.2.2.3	Rundenzeit . . . . .	59
6.3	Regelung des dynamischen Modells . . . . .	60
6.3.1	Berechnungszeit der Elastische Kostenfunktion . . . . .	60
6.3.2	Maximale Querbeschleunigung und Prädiktionshorizont . . . . .	61
6.3.2.1	Dynamisches Modell . . . . .	62
6.3.2.2	Kammischer Kreis . . . . .	63
6.4	Einfluss des Reifenmodells . . . . .	64
6.4.1	Maximale Geschwindigkeit . . . . .	65
6.4.2	Regelfrequenz . . . . .	66
6.4.3	Quadratische Kostenfunktion . . . . .	67
<b>7</b>	<b>Zusammenfassung</b>	<b>69</b>
7.1	Zusammenfassung . . . . .	69
7.2	Ausblick . . . . .	70
7.2.1	Simulator . . . . .	70
7.2.2	Dynamisches Fahrzeugmodell im MPC . . . . .	70
7.2.3	Zweispurmodell . . . . .	71
7.2.4	Implementierung in C++ . . . . .	71
7.2.5	Fahrzeugparameter anpassen . . . . .	72
<b>A</b>	<b>Anhang</b>	<b>73</b>
	<b>Abkürzungsverzeichnis</b>	<b>74</b>
	<b>Literaturverzeichnis</b>	<b>75</b>



---

# 1 Einleitung

---

Neben neu aufkommenden Trends in der Automobilindustrie, wie dem Elektroauto, Carsharing und dem aktuellen SUV-Boom, hat sich vor allem das autonome Auto in den letzten zwei Jahren mit dem Aufkommen von Teslas Autopilot stark in das Zentrum der Aufmerksamkeit von Firmen und Konsumenten gerückt. Das Jahr 2018 markiert dabei einen ganz besonderen Meilenstein. Ende des Jahres wird die Alphabet-Tochter Waymo die ersten vollautonomen Taxis in Phoenix für die Öffentlichkeit in Betrieb nehmen [Way]. Dies ist jedoch nur möglich, da Google im gesamten Gebiet, in dem sie ihren Service anbieten werden, eine hochgenaue Umgebungskarte in die Software der Fahrzeuge integriert. Sollen die Fahrzeuge in unbekannten Umgebungen agieren, müssen sowohl die Sensorsysteme, als auch die Algorithmen noch deutlich weiterentwickelt werden, bevor autonome Autos der sogenannten Stufe 5 frei verfügbar sind. Die Unterteilung dieser Level wurde von der Bundesanstalt für Straßenwesen zur Klassifizierung von selbst fahrenden Autos erstellt [BAS]. Die Spanne geht von Stufe 0, keinerlei Assistenz beim Fahren, bis Stufe 5, Vollautomatisierung. In dieser höchsten Stufe kann das Fahrzeug jederzeit ohne das Eingreifen des Fahrers navigieren. Nicht nur in den großen Automobilkonzernen wird deshalb zum Erreichen der höchsten Automatisierungsstufe geforscht. Bereits in der Schule, aber vor allem an Universitäten und Hochschulen, gibt es immer mehr Projekte, an denen die Studenten sich intensiv mit der Entwicklung autonomer Roboter und Fahrzeugplattformen beschäftigen. Ein solches Projekt ist der Formula Student (FS) Driverless-Wettbewerb.

## 1.1 Formula Student Driverless

Die Formula Student ist ein Ingenieurswettbewerb für Studenten. Er hat seine Wurzeln in den USA und wurde im Jahre 1981 das erste mal ausgetragen. Das Ziel des Wettbewerbes ist es, im Verlauf eines Jahres ein eigenes Rennauto zu konzipieren, designen, fertigen und sich schlussendlich mit anderen Teams zu messen. Dass der Wettbewerb sehr erfolgreich ist, machen nicht nur die inzwischen fast 700 Teams weltweit [FsW] deutlich, sondern auch die Anzahl der verschiedenen Events, die überall auf der Welt im Sommer stattfinden. Seit dem Jahr 2017 gibt es neben der ursprünglichen Combustion-Klasse und der vor 10 Jahren eingeführten Electric-Klasse auch noch die Driverless-Klasse. In dieser wird von

den Teams ein Altfahrzeug mit Sensoren und Aktoren so erweitert, dass das Rennauto die Kurse autonom befahren kann. Der Wettbewerb ist unterteilt in dynamische und statische Disziplinen. In letzteren werden verschiedene Präsentationen von den Teams ausgearbeitet. Diese beziehen sich auf die technische Realisierung, Softwaredesign, Kostenaufstellung und einen Businessplan. Die dynamischen Disziplinen, in denen das Fahrzeug selbstständig fährt, sind:

- Acceleration  
Ein 75 Meter langer Beschleunigungstreifen. Punkte werden nach der Zeit, nicht nach der Endgeschwindigkeit vergeben.
- Skidpad  
Eine liegende 8, bei der an der Engstelle eingefahren wird und jeweils zwei rechte und zwei linke Runden gefahren werden. Die zweite Runde geht jeweils in die Zeitmessung ein. Die Abmaße sind exakt vorgegeben.
- Trackdrive  
Ein bis zu 800 Meter langer Kurs mit maximal 80 Meter langen Geraden und Kurven mit einem minimalen Innenradius von neun Metern. Es werden elf Runden gefahren und die Teams erhalten im Vornherein keine Möglichkeit, Messungen am Kurs vorzunehmen.

Für die vorliegende Arbeit ist vor allem der Trackdrive von Interesse. Es wird davon ausgegangen, dass das Fahrzeug bereits die erste Runde absolviert und sich damit eine genaue Karte des Rennkurses erstellt hat. Die Messungen und Vergleiche beziehen sich damit auch immer auf einen Kurs, der so in einem FS-Event für die Driverless-Fahrzeuge vorkommen könnte. In der Abbildung 1.1 ist das autonome Rennauto von Zürich zu sehen, wie es den Trackdrive absolviert.



**Abbildung 1.1:** Zürichs FS-Driverless-Fahrzeug im Jahr 2017 während des Trackdrive [AMZ]

### 1.2 Problemstellung

Im Laufe dieser Arbeit soll ein Algorithmus entwickelt werden, der die Trajektorienplanung und Regelung eines FS Driverless Racecars berechnet. Der dafür gewählte Ansatz ist Model Predictive Control (MPC). Es handelt sich dabei um ein Verfahren, welches die Planung von Trajektorien und die Steuerung des Rennautos in einem Algorithmus vereint. Wie der Name schon sagt, wird hierfür ein Modell des zu kontrollierenden Systems benötigt, mit dem daraufhin das zukünftige Verhalten vorausberechnet werden kann. Dieses Wissen wird genutzt um mit Hilfe einer Kostenfunktion die Steuerparameter so zu variieren, dass das Rennauto eine möglichst schnelle Rundenzeit erreicht. In der Arbeit wird untersucht, wie groß die Unterschiede zwischen einem rein geometrischen Fahrzeugmodell und einem um Reifenkräfte erweiterten, für hohe Geschwindigkeiten realistischeren, Modell sind. Zudem wird evaluiert, unter welchen Einschränkungen der MPC-Ansatz mit hinterlegtem kinematischen Modell ein dynamisches Modell im Simulator steuern kann. Das Ziel ist also eine möglichst niedrige Rundenzeit zu erreichen unter der Voraussetzung, dass das Fahrzeug immer auf dem Kurs bleibt.

---

## 2 Stand der Technik

---

Obwohl es zum Zeitpunkt der Arbeit noch keine öffentliche, autonome Rennserie außerhalb der Formula Student gibt, ist das Interesse an den dafür benötigten Technologien sehr groß. Einen guten Rennfahrer zeichnet die Tatsache aus, dass er sein Fahrzeug in absoluten Grenzsituationen noch unter Kontrolle halten kann. Diese Eigenschaft ist nicht nur für autonome Rennautos wichtig, sondern ganz besonders auch für aktive Fahrassistenzsysteme. In kritischen Fahrsituationen, in denen ein ungeübter Fahrer einen Unfall nicht mehr verhindern kann, können diese Systeme noch eingreifen. Um einen guten Rennfahrer in Software nachstellen zu können, müssen drei Grundvoraussetzungen geschaffen werden:

- Genaue Kenntnis der Umgebung (dem Rennkurs) und der eigenen Position
- Möglichst viel Wissen über das Verhalten des Fahrzeugs (Fahrzeugmodell)
- Kurze Reaktionszeiten (engl.: muscle memory)

Um die Position des Fahrzeugs genau bestimmen zu können, werden verschiedene Sensortypen mit Hilfe von Filterverfahren fusioniert. Die Genauigkeit der Schätzung steigt im Vergleich zu den einzelnen Messsystemen durch die Kombination signifikant an [SCT07], [WDY16]. Um das Rennauto mit hoher Geschwindigkeit einen Rennkurs abfahren zu lassen, wird ein Modell benötigt, welches das dynamische Verhalten des Fahrzeugs abbilden kann. Erst dieses Wissen ermöglicht die Berechnung von Trajektorien, welche das Rennauto an die Grenzen seiner Traktion bringt. Es muss dabei ein guter Kompromiss aus Komplexität (Rechenaufwand) und Genauigkeit der Abbildung gefunden werden. Eine sehr genaue Systembeschreibung ermöglicht das Berechnen präziserer Trajektorien. Ist dies jedoch nicht mehr in Echtzeit berechenbar, kann das Modell nicht verwendet werden. Die Ansätze variieren je nach Komplexität und zu bewältigenden Fahrsituationen:

- Ein einfaches kinematisches Modell [Raj11], basierend auf einem Einspursystem, welches sehr schnell berechenbar ist, aber keine dynamischen Effekte berücksichtigt
- Die Nutzung einer dynamischen Fahrzeugbeschreibung die das Einspurmodell um ein lineares oder nichtlineares Reifenmodell erweitert [AAM], [PER16], [CUR18]

- Ein sehr genaues Zweispurmodell, die zehn Freiheitsgrade betrachtet und sogar das Einfedern des Fahrzeugs berücksichtigt [FGW02]

Um ein mathematisches Fahrzeugmodell in einem realen Fahrzeug einsetzen zu können, müssen die das System beschreibenden Parameter möglichst genau bestimmt werden. Dies kann mit Hilfe einer Regression sehr effizient und genau realisiert werden [WDG<sup>+</sup>16].

Sind die Grundvoraussetzungen geschaffen, gibt es unterschiedliche Ansätze, ein Rennauto möglichst schnell um einen Rennkurs fahren zu lassen. Der Einsatz von neuronalen Netzen [AGT12] ist eine Möglichkeit, um geeignete Steuerparameter zu berechnen. Ein Ansatz, wie in [KG10], hinterlegt für verschiedene Klothoiden die jeweilige Querschleunigung, welche wirken muss, um die Kurve fahren zu können. In einem Diagramm ist dann hinterlegt, wie groß die longitudinale Beschleunigung maximal sein darf, damit das Rennauto nicht instabil wird. Vor allem aber der in dieser Arbeit betrachtete MPC-Algorithmus wird in einer Vielzahl von Arbeiten untersucht und verwendet. Die Unterschiede bei den Verfahren beziehen sich auf die getrennte Berechnung von Trajektorie und Regelung, die Trennung von longitudinalem und lateralem Regler [PER16], [CUR18] oder eine kombinierte Betrachtung [AAM]. Im Vergleich hierzu, soll in dieser Arbeit die Regelfähigkeit des MPC-Ansatzes evaluiert werden, wenn das zugrunde liegende Modell stark von dem echten System abweicht.

Die Aktualisierungszeit für neue Steuerparameter sollte im Anwendungsfall eines autonomen Rennautos zwischen 20 und 100 Hz liegen, je nachdem wie schnell das Fahrzeug fahren soll [AAM], [WDG<sup>+</sup>16].

---

## 3 Grundlagen

---

Um für das Fahrzeug ideale Trajektorien zu berechnen und gleichzeitig das Rennauto in Echtzeit zu regeln wurde der MPC-Ansatz gewählt. Dieses Verfahren basiert auf der Optimierung eines nichtlinearen Programms.

### 3.1 Model Predictive Control

Bei MPC handelt es sich um einen Algorithmus, mit dem sich komplexe, multivariable Regelungsprobleme lösen lassen. Er eignet sich für Prozesse mit mehreren Ein-, und Ausgängen, die klar definierten Beschränkungen unterliegen. Vorausgesetzt es ist ein ausreichend genaues Modell des Prozesses vorhanden, können mit dem Einbeziehen des aktuellen Systemzustands zukünftige Zustände des Prozesses berechnet werden. Diese Information wird genutzt, um die Eingangsparameter abhängig von dem gewünschten zukünftigen Verhalten des Prozesses zu berechnen [SMED10].

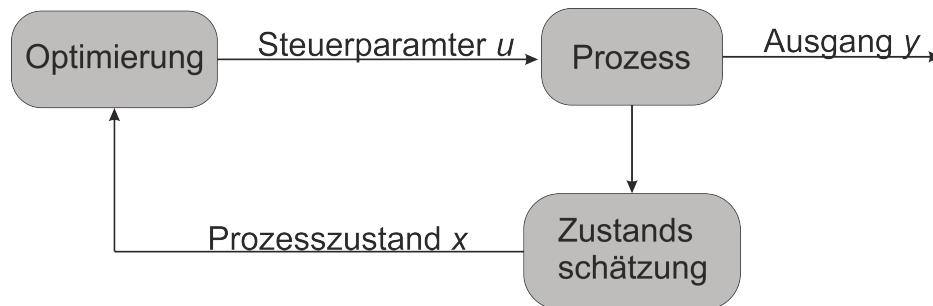
Einige wichtige Vorteile von MPC sind:

- Das sehr gute Annähern an Prozessgrenzen und damit ein hoher Durchsatz/Effizienz. Ein Rennauto versucht zum Beispiel den Kurs bis an die Bande auszunutzen, um die höchste Kurvengeschwindigkeit fahren zu können
- Die Einschränkungen der Eingangs- und Ausgangsgrößen werden berücksichtigt
- Die Vorhersage des Systemzustands kann genutzt werden, um mögliche Probleme frühzeitig zu detektieren. Zum Beispiel kann bei einer Blockade der Fahrbahn frühzeitig ermittelt werden, ob eine Notbremsung eingeleitet werden muss.

Seit den späten 1970ern bis in die frühen 2000er wurde MPC vor allem in der Chemiebranche genutzt, um die komplexen multivariablen Regelungsprozesse, zum Beispiel in einer Ölraffinerie, zu steuern. Für diese Aufgabengebiete war MPC hervorragend geeignet, da die Prozesse im Vergleich zu anderen Regelungsaufgaben sehr langsam sind und damit die geringe Rechenleistung der damaligen Zeit ausreichend war. Gerade in den letzten Jahren, mit stark gestiegener Prozessorleistung, sind die Einsatzgebiete vielfältiger geworden. Auch die Steuerung hoch dynamischer Systeme wie Quadrocopter [GD17] oder Rennautos [FB05] ist inzwischen möglich.

## Funktionsweise

Der systematische Ablauf eines MPC- Algorithmus ist in Abbildung 3.1 aufgezeigt.



**Abbildung 3.1:** Block Diagramm zu MPC

Die Funktion lässt sich in drei Schritte unterteilen:

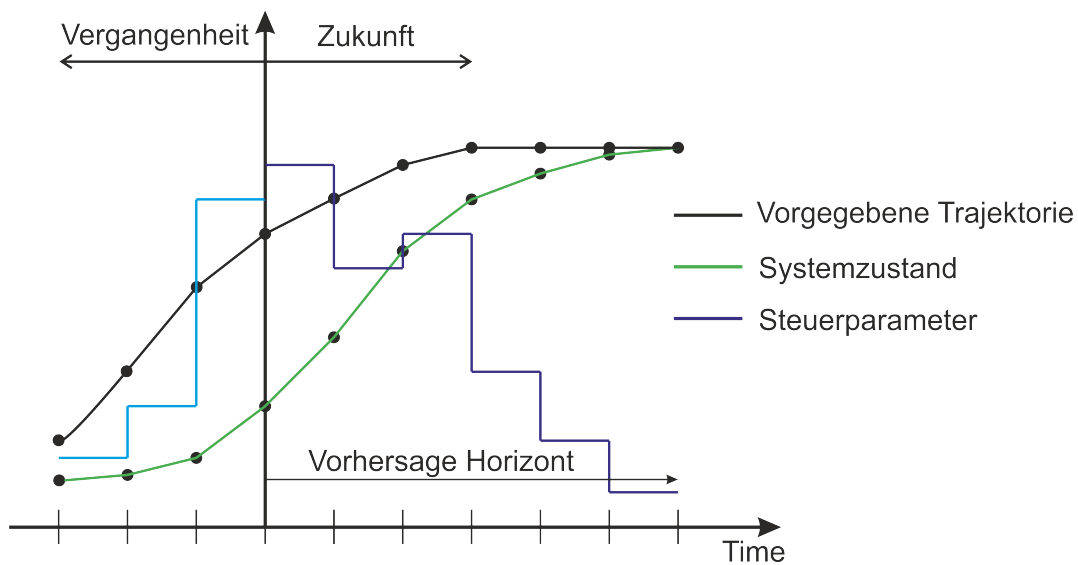
### **Optimierung** blabla

- **Zustandsschätzung**  
Im ersten Schritt wird der aktuelle Zustand des Systems erfasst und als Ausgangspunkt für die Berechnung des Optimierungsproblems festgelegt.
- **Optimierung**  
In diesem Schritt werden die zukünftigen Zustände des Systems berechnet und welche Steuerparameter vonnöten sind, um eine Kostenfunktion zu minimieren. Diese entspricht im industriellen Umfeld zum Beispiel einer Optimierung der Fertigungsrate, der Kostenreduktion, der Temperatursteuerung, etc.
- **Steuerung**  
Die berechneten Steuerparameter werden zum Regeln des realen Systems angewandt. Danach wird wieder beim ersten Schritt fortgefahren.

Die Stärke des MPC-Ansatzes basiert auf der Prädiktion des Systemverhaltens. Ein präzises Vorhersagen setzt eine möglichst gute Systembeschreibung voraus. Je genauer diese an die Realität heranreicht, desto weiter in die Zukunft können die Systemzustände, ohne große Abweichung von der Realität, berechnet werden. Zudem kann das System näher an seine Systemgrenzen geführt werden.

Bei MPC handelt es sich um ein Verfahren, welches immer bis zu einem festen Zeitpunkt in der Zukunft plant. Die Optimierung wird also immer für einen bestimmten Zeitraum

$[t_0, t_0 + T]$  durchgeführt. Der Bereich  $T$  wird in  $k$  Schritte unterteilt, für die jeweils ein Prädiktionsschritt berechnet wird. Dieser geht aus dem Zustand des vorherigen Schrittes  $k - 1$  und den dazu gehörigen Steuergrößen hervor. Nachdem der Optimierer die für eine Kostenfunktion idealen Steuergrößen für den gesamten Prädiktionsvektor berechnet hat, wird nur der *erste* Steuerwert ausgewählt und zur Regelung im realen System eingesetzt. Der Zusammenhang aus Prädiktion und Kostenfunktion ist im Schaubild 3.2 verdeutlicht. Für jeden Schritt im Vorhersagehorizont wird der Steuerparameter berechnet, der den Systemzustand möglichst schnell der vorgegebenen Trajektorie annähert. Das Wissen über das zu steuernde System hilft, die dafür passenden Steuerparameter zu finden und frühzeitig genug die Amplitude so anzupassen, dass kein Übersteuern auftritt.



**Abbildung 3.2:** Funktionsprinzip von MPC: Die Kostenfunktion ist die Reduktion der Distanz zur Referenztrajektorie. Durch das Wissen über das hinterlegte Modell können die Kosten sehr schnell minimiert werden ohne die Beschränkungen zu verletzen

Wie bereits ausgeführt wurde, berücksichtigt der MPC-Ansatz auch Eingangs-, Ausgangs- und Zustandsbeschränkungen, mit denen der Suchraum einschränkt wird. Diese sind im Falle eines Rennautos zum Beispiel die maximale Geschwindigkeit, Lenkeinschlag, Beschleunigung und Bremskraft. Ebenfalls ein sehr gutes Beispiel für eine Zustandsbeschränkung ist die Reisequalität in einem normalen Straßenauto. Ruckartiges Anfahren und allgemein plötzliche Änderung der lateralen wie longitudinalen Beschleunigung trübt bei einem autonomen Fahrzeug das Fahrgefühl. Eine Möglichkeit diesem Problem zu begegnen ist die Begrenzung der Änderungsrate der Beschleunigung des Fahrzeugs.



Das Grundgerüst des MPC-Ansatz bildet der Optimierer, der die optimale Lösung für die gewünschte Kostenfunktion sucht. Im Folgenden wird daher auf die Funktionsweise von diesem eingegangen.

## 3.2 Optimierung

Das Ziel der Optimierung besteht darin, eine gegebene Funktion  $f(\vec{x})$ , auch als Kostenfunktion bezeichnet, zu mini- oder maximieren. Dies wird in einer Vielzahl von Anwendungsgebieten genutzt: Zur Berechnung von Profit bzw. Verlust in einem Betrieb, der Geschwindigkeit oder Distanz in einem physikalischen Problem oder beispielsweise die erwartete Kapitalrendite für eine Geldanlage. Die Bezeichnung lineare Programmierung bezieht sich auf die Lösung eines linearen Optimierungsproblems und hat nichts mit dem eigentlichen Programm zu tun. Die allgemeine mathematische Definition eines Optimierungsproblems ist

minimiere  $f(\vec{x})$   
unter Berücksichtigung von  $g_i(\vec{x}) \leq 0, i = 1, 2, \dots, p$   
 $h_j(\vec{x}) = 0, j = 1, 2, \dots, m$

Der Eingabeparameter  $\vec{x}$  sei aus  $\mathbb{R}^n$ , das heißt, das Problem hängt von  $n$  Einflussparametern ab. Die Zielfunktion  $f : D \rightarrow \mathbb{R}$  sei zweifach stetig differenzierbar. Weiterhin sind die Nebenbedingungen in Ungleichheitsform  $g_i : D \rightarrow \mathbb{R}$  mit  $1 \leq i \leq p$  und in Gleichheitsform  $h_j : D \rightarrow \mathbb{R}$  mit  $1 \leq j \leq m$  gegeben. Wird  $f(\vec{x})$  durch  $-f(\vec{x})$  ersetzt, wird aus dem Mini- ein Maximierungsproblem. Der Unterschied zu einem nichtlinearen Programm ist, dass mindestens eine der Gleichungen des Optimierungsproblems nichtlinear ist.

Ein Optimierungsproblem hat nicht immer eine Lösung. Zum Beispiel können sich die Ungleichungen, welche den Suchraum einschränken, widersprechen ( $x \leq 0$  und  $x > 0$ ). Außerdem kann das Problem unbeschränkt sein. Soll zum Beispiel eine Optimierungsvariable minimiert werden, die nach unten nicht beschränkt ist, kann keine Lösung gefunden werden.

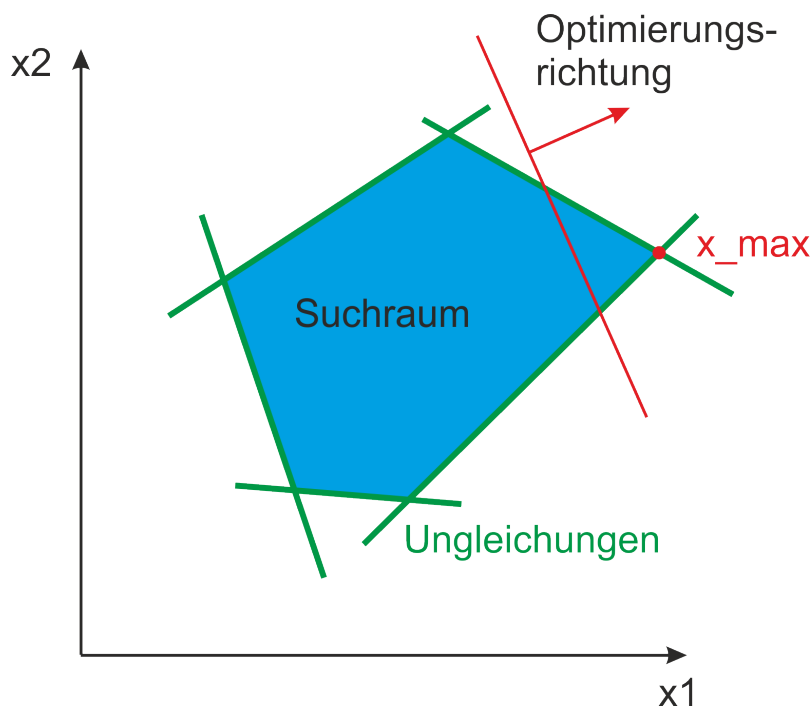
Ein Optimierungsproblem lässt sich in Matrixschreibweise darstellen. Es besteht aus der Matrix  $A \in \mathbb{R}^{m,n}$ , die zusammen mit den Vektoren  $b \in \mathbb{R}^{m,1}$  die Un- und Gleichheitsbedingungen enthält. Der Vektor  $c \in \mathbb{R}^{1,n}$  kodiert die Zielfunktion. Gesucht wird also der Vektor  $x \in \mathbb{R}^n$ , der die Bedingungen

$$\begin{array}{ccccccc}
a_{11}x_1 + & \dots & + a_{1n}x_n & \leq & b_1 \\
a_{21}x_1 + & \dots & + a_{2n}x_n & \leq & b_2 \\
\cdot & & \cdot & & \cdot \\
\cdot & & \cdot & & \cdot \\
a_{m1}x_1 + & \dots & + a_{mn}x_n & \leq & b_m
\end{array}$$

erfüllt und gleichzeitig die Zielfunktion  $cx = c_1x_1 + \dots + c_nx_n$  minimiert. Die Kurzschreibweise für dieses Gleichungssystem ist:

$$\min \{c^T x \mid Ax \leq b, x \geq 0\}$$

Veranschaulichen lässt sich die Lösung des Problems geometrisch im zweidimensionalen Raum, so dargestellt in Abbildung 3.3.



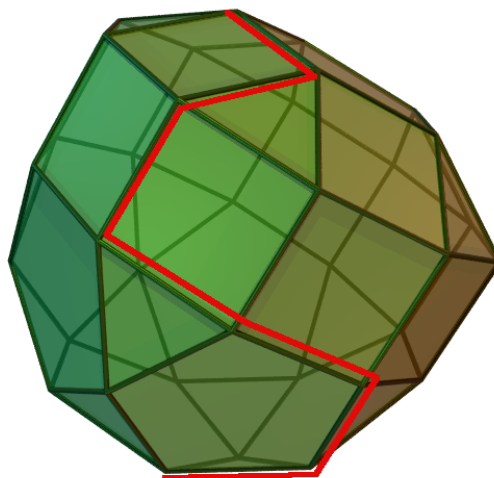
**Abbildung 3.3:** Lineare Optimierung: Suche des Maximums an den Schnittpunkten der Ungleichungen

Jede Ungleichung  $a_ix \leq b_i$  teilt den Suchraum in zwei Hälften, eine mit zulässigen Punkten und eine ohne. Die Punkte auf der Grenze sind ebenfalls zulässig. Die Menge der Punkte, welche alle Ungleichungen erfüllt, ist genau der Schnitt dieser Halbräume. Dieser Bereich ist in Abbildung 3.3 blau dargestellt. Der Punkt, der die Kostenfunktion  $c : x \rightarrow c^T x$  minimiert, liegt auf den Kanten der Ungleichungen und wird durch Verschiebung der

Hyperebene  $\{x | c^T x = 0\}$  in Richtung des Vektors  $c$  gefunden. Nachdem das Optimierungsproblem aufgestellt ist, gibt es verschiedene Ansätze dieses zu berechnen.

### 3.2.1 Simplex-Verfahren

Bei der in dieser Arbeit zu lösenden Optimierung handelt es sich um ein nichtlineares Optimierungsproblem. Um dieses zu lösen, wird auf ein Innere-Punkte-Verfahren zurückgegriffen. Für ein besseres Verständnis des Verfahrens soll zuerst das Simplex-Verfahren kurz vorgestellt werden. Es wird für lineare Optimierungsprobleme eingesetzt und veranschaulicht die Suche nach der optimalen Lösung sehr gut. Das Verfahren wurde 1951 von Dantzig [Dan51] entwickelt und basiert auf dem Zusammenhang, dass ein lösbares lineares Programm immer einen Knoten besitzt, an dem sich Kanten verschiedener Ungleichungen treffen und die Kostenfunktion minimieren. Das Verfahren nutzt diese Eigenschaft aus, indem es sich iterativ von einem Eckpunkt zum nächsten bewegt. Das Verfahren ist bildlich in der Grafik 3.4 dargestellt.



**Abbildung 3.4:** Simplex Methode: Ablaufen des Suchraums entlang der Kanten der Ungleichheitsbedingungen von Knoten zu Knoten [sim]

Die Kante, die gewählt wird, um sich zum nächsten Eckpunkt zu bewegen, hängt immer davon ab, auf welcher Flanke liegenden Punkte die Kostenfunktion weiter minimieren. Kann keine weitere Flanke gefunden werden, die die Kosten weiter reduziert, endet das Verfahren im Optimum. Die Anzahl der Kanten im Polyeder wächst exponentiell mit der

Anzahl der Optimierungsvariablen ( $n$ ). Da im schlechtesten Fall der Simplex-Algorithmus jeden Knotenpunkt einmal betrachten muss, hat das Verfahren daher ein exponentielles Wachstum  $\mathcal{O}(c^n)$  mit  $c > 1$  [FGW02]. Trotz dieser Problematik hat das Simplex-Verfahren in der Praxis eine sehr gute Laufzeit und kann für den wiederholten Aufruf einer nur leicht veränderten Optimierung (Anpassen von Parametern nach dem letzten Aufruf) einen sogenannten Warmstart nutzen. Dieser beschleunigt die Lösung ebenfalls deutlich.

### 3.2.2 Innere-Punkte-Verfahren

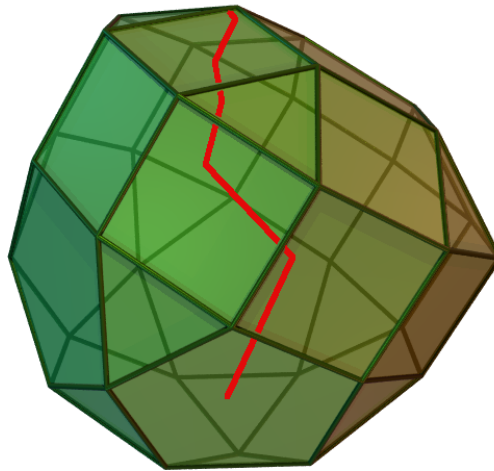
Die Suche nach noch besseren Verfahren, deren Komplexität nur polynomial mit der Dimension anwächst, führte dann in den Jahren 1979 bis 1984 zur Entwicklung eines neuen Algorithmus. Dieser wird als Innere-Punkte-Verfahren bezeichnet und eignet sich auch für nichtlineare Probleme. Die Un- und Gleichheitsbedingungen wie auch die Kostenfunktionen die im MPC-Ansatz zum Regeln des Rennautos vorkommen, sind nichtlinear. Sie setzen daher einen Optimierer voraus, der diese Problemen berechnen kann. Zusätzlich gibt es, wie im Kapitel 3.1 erwähnt, Beschränkungen, die bei der Suche der Lösung eingehalten werden müssen. Das Innere-Punkte-Lösungsverfahren erfüllt diese Anforderung für lineare, quadratische und nichtlineare Optimierungsprobleme und wird im Folgenden kurz vorgestellt. Der Ansatz des Verfahrens ist die Bewegung im Inneren des zulässigen Lösungsbereiches und nicht auf den Kanten wie beim Simplex-Verfahren. Diese Bewegung durch das Innere des Polytop ist in Abbildung 3.5 veranschaulicht.

Um dieses Verfahren umzusetzen, werden die Ungleichheitsbedingungen  $g(\vec{x}) \geq 0$  der Standardform mit Hilfe einer Substitution so umgeformt, dass sie der Form  $c(\vec{x}) = 0$  und  $\vec{x} \geq 0$  entspricht. Die neue Ungleichheitsbedingung kann nun mit einem logarithmischen Strafterm  $\mu \ln x$  ersetzt und in die Kostenfunktion integriert werden. Der Faktor  $\mu$  wird genutzt, um die Gewichtung des Strafterms zu verändern.

Die daraus resultierende Funktion wird als Barrierefunktion bezeichnet und ist definiert als

$$B(x, \mu) = c^T x - \mu \sum_i^n \ln x_i \quad (3.2.1)$$

mit  $\mu > 0$  und ersetzt die Kostenfunktion im ursprünglichen Optimierungsproblem. Der Strafterm verhindert, dass der Optimierer, wie beim Simplex-Verfahren, an den Grenzen der Un- und Gleichheitsbedingungen nach der Lösung sucht, sondern sich immer im



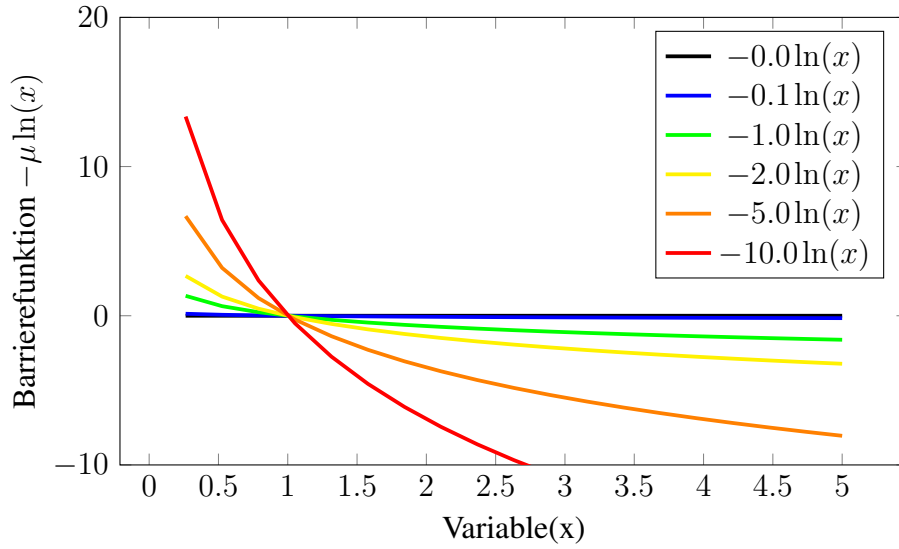
**Abbildung 3.5:** Beim Innere-Punkte-Verfahren bewegt sich der Optimierer durch das Innere des Polytops auf der Suche nach dem Minimum [Inn]

Inneren des zulässigen Lösungsbereiches bewegt. Wird  $\mu$  sehr klein (tendiert gegen null), entspricht die Lösung des Barriereproblems der gleichen wie beim ursprünglichen Optimierungsproblem. Diese Eigenschaft wird genutzt, von einem großen  $\mu$  als Startpunkt iterativ  $B(x, \mu)$  zu optimieren und dann vor der nächsten Iteration  $\mu$  zu verkleinern. Das schrittweise verkleinern von  $\mu$  dient dazu, sich anfangs grob und schnell, später mit kleinen Schritten und langsam der Lösung zu nähern. Die Auswirkung der stetigen Verkleinerung von  $\mu$  ist in der Abbildung 3.6 veranschaulicht.

Um die Lösung in jedem dieser iterativen Schritte zu finden, wird die Ableitung der Barrierefunktion gleich null gesetzt, da in diesem Punkt ein (lokales) Minimum existiert. Um diesen Punkt möglichst schnell zu finden, wird das Newton-Raphson-Verfahren angewandt. Bei diesem handelt es sich um eine Nullstellensuche, welche unter Zuhilfenahme der Ableitung der Funktion möglichst schnell ein Ergebnis findet. Der Algorithmus benötigt daher die Hesse-Matrix der Barrierefunktion. Die Berechnung dieser wird durch eine automatische Ableitung realisiert.

### 3.2.3 Automatische Ableitung

Ableitungen sind eine Grundvoraussetzung für viele numerische Algorithmen. Die Genauigkeit und Geschwindigkeit der Berechnung ist jedoch oftmals problematisch.



**Abbildung 3.6:** Auswirkung des Parameters  $\mu$  auf das Verhalten der Barrierefunktion

Ein möglicher Ansatz die Ableitung

$$\Delta f(x) = \left( \frac{\partial f}{\partial x_1}(x), \dots, \frac{\partial f}{\partial x_n}(x) \right) \quad (3.2.2)$$

der Funktion  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  zu berechnen ist

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \quad (3.2.3)$$

mit  $h \rightarrow 0$ . Dieser Ansatz ist jedoch mit einer Komplexität von  $\mathcal{O}(n)$  sehr ineffizient. Es entstehen zudem Rundungsfehler [jul]. Eine andere Möglichkeit wäre die Ableitung manuell zu berechnen, was jedoch für komplexere Systeme sehr aufwendig und vor allem fehleranfällig ist. Eine bessere Lösung ist die automatische Ableitung (eng.: automatic differentiation). Dieses System kann die Ableitung bis auf die maximal mögliche Genauigkeit eines float-Datentyps berechnen. Zu der höheren Präzision kommt noch die deutlich bessere Laufzeit hinzu. Im Idealfall entspricht die Komplexität  $\mathcal{O}(1)$  [jul]. Um diese Vorteile nutzen zu können, wird jedoch ein genaues Wissen über die abzuleitende Funktion  $f(x)$  benötigt. Realisiert wird dies durch den Zugriff auf den Sourcecode der Methode oder durch ein Überladen der Operatoren. Das Verfahren ist abhängig von den Möglichkeiten der Programmiersprache und welcher Ansatz bei der Implementierung der automatischen Ableitung verfolgt wurde.

Ein Verfahren der Umsetzung ist der Vorwärtsmodus. Dieser basiert auf der Kettenregel

$$(f \circ g)' = (f' \circ g)g' \quad (3.2.4)$$

Um dies auszunutzen wird eine abzuleitende Funktion  $y = f(g(h(x)))$  in seine Bestandteile  $y = f(g(h(w_0))) = f(g(w_1)) = f(w_2) = w_3$  aufgetrennt und anschließend in die Kettenregel eingesetzt.

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial w_2} \frac{\partial w_2}{\partial w_1} \frac{\partial w_1}{\partial x} \quad (3.2.5)$$

Zum besseren Verständnis wird die Ableitung  $\frac{\partial y}{\partial x_1}$  für die Funktion  $y = f(x_1, x_2) = x_1 x_2 + \sin(x_1)$  berechnet.

Dazu werden, wie oben bereits beschrieben, zuerst die Bestandteile substituiert:

$$y = w_1 w_2 + \sin(w_1)$$

$$y = w_3 + w_4$$

$$y = w_5$$

Danach muss der Seed errechnet werden. Er kodiert über welche Variable (hier  $x_1$ ) differenziert werden soll. Für  $x_1$  ist der Seed  $\dot{w}_1 = \frac{\partial x_1}{\partial x_1} = 1$  und  $\dot{w}_2 = \frac{\partial x_2}{\partial x_1} = 0$ . Im letzten Schritt werden nur noch von innen nach außen die Substitutionen ersetzt.

$$\dot{w}_3 = w_2 \dot{w}_1 + w_1 \dot{w}_2$$

$$\dot{w}_4 = \cos w_1 \cdot \dot{w}_1$$

$$\dot{w}_5 = \dot{w}_3 + \dot{w}_4$$

Dieses Verfahren kann im sogenannten Rückwärtsmodus auch von außen nach innen berechnet werden.

## 3.3 Fahrzeugmodelle

Nachdem die Grundlagen zur Optimierung gelegt wurden, wird nun auf den Teil im MPC-Algorithmus eingegangen, welcher die Un- und Gleichheitsbedingungen definiert. Wie der Name Model Predictive Control schon andeutet, wird eine Modellbeschreibung des zu regelnden Systems benötigt. Diese wird genutzt, um zukünftige Zustände zu berechnen und bildet damit einen wichtigen Bestandteil des Verfahrens. Je genauer die Beschreibung das reale System approximiert, desto besser ist die Vorhersage und damit auch die Regelung des Fahrzeugs. Im Folgenden wird zuerst ein kinematisches Fahrzeugmodell eingeführt und dann zu einem dynamischen Modell erweitert.

### 3.3.1 Kinematisches Modell

Unter gewissen Einschränkungen kann ein kinematisches Modell die laterale und longitudinale Bewegung eines Fahrzeugs mathematisch beschreiben. In diesem sehr stark vereinfachten Modell werden für die Berechnung der Bewegung keine wirkenden Kräfte berücksichtigt, sondern nur die geometrischen Beziehungen des Fahrzeugs betrachtet.

Im ersten Schritt werden die jeweils an einer Achse verbundenen Räder zu einem einzigen zusammengefasst. Diese Vereinfachung wird als Bicycle Modell bezeichnet und vereinfacht die Berechnungen [WQ01]. Obwohl auch für Hinterradlenkung möglich, wird im Folgenden nur die Vorderradlenkung betrachtet, da das Rennauto des High-Octane Motorsport e.V. keine Hinterradlenkung besitzt. Die Lenkwinkel, welche durch das Bicycle Modell berechnet werden, entsprechen nicht den Lenkwinkeln am echten Fahrzeug. Die kurveninneren und kurvenäußeren Räder bewegen sich auf zwei Kreisen mit unterschiedlichen Radien und damit auch verschiedenen Anstellwinkeln. Dies wird in Fahrzeugen durch die Ackermann-Lenkung mechanisch umgesetzt [Raj11].

Die nichtlinearen, zeitkontinuierlichen Gleichungen



$$\dot{x} = v \cos(\psi + \beta) \quad (3.3.1)$$

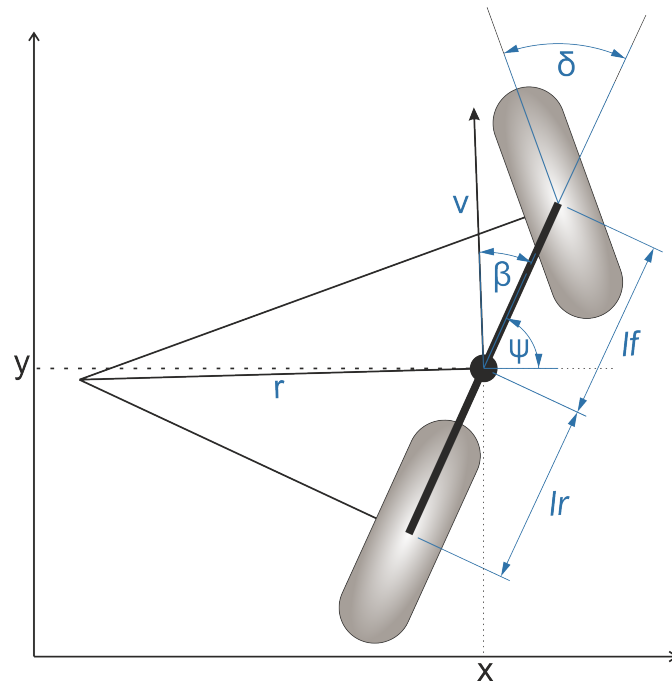
$$\dot{y} = v \sin(\psi + \beta) \quad (3.3.2)$$

$$\dot{\psi} = \frac{v}{l_r} \sin(\beta) \quad (3.3.3)$$

$$\dot{v} = a \quad (3.3.4)$$

$$\beta = \arctan\left(\frac{l_r}{l_f + l_r} \tan(\delta_f)\right) \quad (3.3.5)$$

basieren auf [Raj11, KPSB15] und beschreiben das kinematische Modell bezüglich eines Inertialsystems (siehe Abbildung 3.7), in dem  $x$  und  $y$  die Koordinaten des Schwerpunktes im Inertialsystem darstellen.



**Abbildung 3.7:** Kinematisches Modell

$\psi$  ist die Orientierung und  $v$  die Geschwindigkeit des Fahrzeugs.  $l_f$  und  $l_r$  sind die Abstände der vorderen ( $l_f$ ) und hinteren ( $l_r$ ) Achsen zum Schwerpunkt. Der Schwimmwinkel  $\beta$ , ist der Winkel zwischen der Bewegungsrichtung des Fahrzeugs im Schwerpunkt und der Fahrzeuglängsachse bei der Kurvenfahrt. Die Beschleunigung  $a$  bezieht sich ebenfalls

auf den Schwerpunkt und zeigt immer in die Richtung des Geschwindigkeitsvektors. Wird der Lenkeinschlag ruckartig geändert, verändert sich auch genauso ruckartig der Geschwindigkeitsvektor. Dies ist einer der Nachteile des kinematischen Modells.

Die Parameter lassen sich in zwei Bereiche unterteilen:

- Steuerparameter  
 $a, \delta$
- Zustandsgrößen  
 $x, y, v, \psi$

Die Annahme eines kräftefreien Modells, bei dem das Vorderrad genau in die Richtung rollt, in die es zeigt, ist nur bis etwa  $5 \frac{\text{m}}{\text{s}}$  plausibel [Raj11]. Danach müssen die Kräfte, welche die Reifen auf die Straße übertragen können, mitbetrachtet werden. Diese werden dann im dynamischen Modell genutzt, um eine genauere Vorhersage berechnen zu können.

Die für das kinematische Modell angenommenen Werte stammen vom Fahrzeug des High-Octane Motorsports Verein (siehe 3.2). Die Beschleunigung  $a$  wurde für einen Acceleration Durchgang mit der Zeit  $t = 4,5 \text{ s}$  und der Endgeschwindigkeit  $v = 33,6 \frac{\text{m}}{\text{s}^2}$  ( $121 \frac{\text{km}}{\text{h}}$ ) berechnet. Nach einsetzen in die Gleichung zum Berechnen der Geschwindigkeit

$$a = \frac{v}{t} \quad (3.3.6)$$

ergibt sich eine mittlere Beschleunigung von  $7,47 \frac{\text{m}}{\text{s}^2}$ .

#### Begrenzung des Kurvenradius

Da beim kinematischen Modell die Annahme getroffen wird, dass das Rad immer in genau die Richtung rollt, in die es zeigt, kann das Fahrzeug bei jeder Geschwindigkeit  $v$  den gleichen Kurvenradius durchfahren. Um auch in höheren Geschwindigkeiten eine akzeptable Regelung zu erhalten, wird eine Beschränkung eingefügt, die die maximale Querschleunigung beschränkt. Der Maximalwert entspricht dem höchsten g-Wert, den das Fahrzeug des High-Octane Motorsports mit Aerodynamik in einer Kurve erreichen kann:  $a_{max} \sim 2,0 \text{ g}$ . Mit der Gleichung

$$|\beta| \leq \arctan \left( \frac{l}{4} \frac{a_{max}}{v^2} \right) \quad (3.3.7)$$

wird dies über die Einschränkung des Schwimmwinkels, der wiederum aus dem Lenkwinkel (3.3.5) berechnet wird, erreicht. Bei höheren Geschwindigkeiten wird also der minimale Kurvenradius durch die künstliche Limitierung des maximalen Lenkwinkels erreicht.

### 3.3.2 Reifenmodell

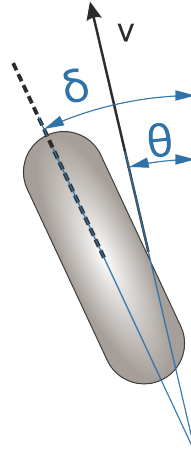
Da der Reifen der einzige Kontaktpunkt zwischen Fahrbahn und Fahrzeug ist, beeinflusst er das Fahrverhalten maßgeblich. Aufgabe des Reifens ist es, sämtliche Kräfte und Momente zu übertragen und eine optimale Straßenlage zu gewährleisten. Demzufolge ist der Reifen das Bauteil, das die Fahrleistungen des Rennautos am stärksten beeinflusst.

Die Kräfte, die ein Reifen auf die Straße übertragen kann, sind in longitudinalen und lateralen Anteil aufgespalten. Sie hängen ab von der Radlast, die das Fahrzeug auf die Straße presst, dem Schlupf und dem Schräglaufwinkel. Die Radlast  $F_z$  berechnet sich aus der Normalkraft und der Radlastverteilung und wird im Folgenden als konstant angesehen. Der Schlupf tritt in longitudinaler Richtung auf und ist die Differenz aus Umlaufgeschwindigkeit und Bewegungsgeschwindigkeit des Fahrzeugs. Nur wenn ein Schlupf vorhanden ist, übertragen Reifen Kraft auf ihren Untergrund. Dies ist bei Beschleunigungsrennen gut zu beobachten, wo die maximale Kraftübertragung im Bereich von 30 Prozent Schlupf liegt (leicht durchdrehende Reifen). Die Seitenführungskraft  $F_y$  wirkt bei einer Kurvenfahrt der Fliehkraft entgegen und hält das Fahrzeug auf der Spur solange ein Kräftegleichgewicht besteht. Für diese laterale Kraft muss der Schräglaufwinkel, genauso wie beim Schlupf, größer null sein. Als Schräglaufwinkel wird der von der Radmittelebene  $\delta$  (Lenkwinkel) und der Bewegungsrichtung  $\theta_{vf}$  des Reifens eingeschlossenen Winkel (siehe Abbildung 3.8)

$$\alpha_f = \delta - \theta_{vf} \quad (3.3.8)$$

bezeichnet. Der gleiche Zusammenhang gilt auch für das hintere Rad, welches jedoch in unserem Fall nicht gelenkt wird

$$\alpha_r = \theta_{vr} \quad (3.3.9)$$



**Abbildung 3.8:** Der Schräglaufwinkel ergibt sich aus Lenkwinkel  $\delta$  und Bewegungsrichtung  $\theta_v$

Für kleine Schräglaufwinkel besteht ein linearer Zusammenhang aus lateraler Kraft und Winkel.

$$F_{yf} = C_\alpha \alpha_f \quad (3.3.10)$$

$$F_{yr} = C_\alpha \alpha_r \quad (3.3.11)$$

Für größere Schräglaufwinkel kann kein linearer Verlauf mehr angenommen werden und es muss eine genauere Approximation der Kräfte gewählt werden. Hierfür wird die sogenannte Magic Formula (M.F.) [PB92] verwendet. Dabei handelt es sich um eine mathematische Gleichung, die sehr gut Messkurven approximiert, die auf Testständen gemessen werden. Sie wurde 1993 von Pacejka und Bakker entwickelt und eignet sich sowohl für die Berechnung der longitudinalen, wie auch der lateralen Kräfte. Die Querkräfte  $F_y$ , die der Reifen auf die Straße überträgt, ergeben sich durch den Schräglaufwinkel  $\alpha$ .

$$F_y = D \sin(C \arctan(B\alpha - E(B\alpha - \arctan(B\alpha)))) \quad (3.3.12)$$

Die Gleichung hängt von den Faktoren  $C$ ,  $B$  und  $E$  ab.

$$C = 1 + \left(1 - \left(\frac{2}{\pi} \arcsin\left(\frac{y}{D}\right)\right)\right) \quad (3.3.13)$$

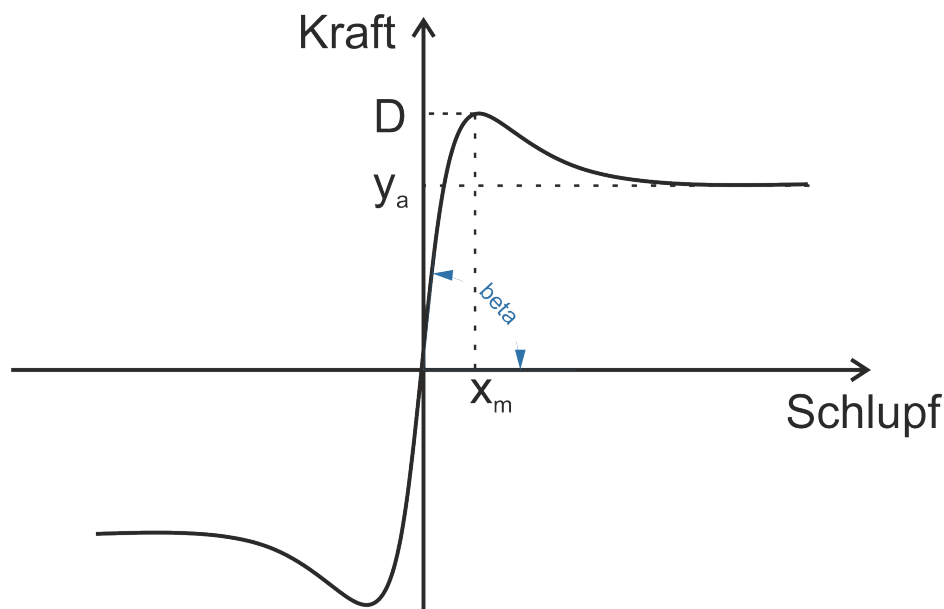
$$B = \frac{\tan(\beta)}{CD} \quad (3.3.14)$$

$$E = \frac{Bx_m - \tan\left(\frac{\pi}{2C}\right)}{Bx_m - \arctan(Bx_m)} \quad (3.3.15)$$

Diese werden abhängig von den folgenden Parametern berechnet.

- $D$   
Die maximale Kraft, die der Reifen im höchsten Punkt übertragen kann
- $\beta$   
Die Steigung im linearen Bereich
- $x_m$   
Der Punkt in rad, in dem die maximale Kraft übertragen wird
- $y_a$   
Die maximale Kraft, gegen die die Kurve für große Schräglaufwinkel tendiert

Die Parameter ändern sich mit der Normalkraft, die das Fahrzeug auf die Straße presst. Ist sie konstant wie in unserem Fall, müssen die Werte nur einmal berechnet werden und bleiben danach unverändert. Sie unterscheiden sich jedoch für das vordere  $(.)_f$  und hintere Rad  $(.)_r$ . Werden in einem komplexeren Modell auch die aerodynamischen Abtriebskräfte und die Steigung der Straße berücksichtigt, ändern sich  $D$ ,  $\beta$ ,  $x_m$  und  $y_a$  mit jeder Veränderung der Normalkraft, was die Neuberechnung von C, B und E zur Folge hat. Zur besseren Veranschaulichung ist in der Zeichnung 3.9 der Einfluss der einzelnen Werte auf die Kurve der Reifenkräfte dargestellt.



**Abbildung 3.9:** Parameter im Reifenmodell

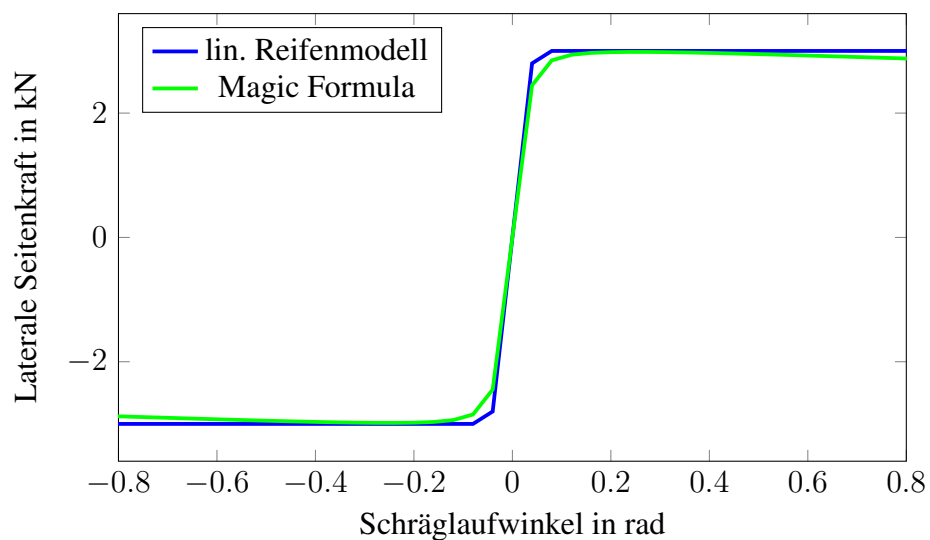
**Tabelle 3.1:** Magic Formula Parameter

Parameter	Wert	Einheit
$D_f$	2984	N
$D_r$	3274	N
$x_{mf}$	0.25	rad
$x_{mr}$	0.37	rad
$y_{af}$	2952	N
$y_{ar}$	3270	N
$\beta_{af}$	$\frac{\pi}{1.9}$	rad
$\beta_{ar}$	$\frac{\pi}{1.9}$	rad

Zusätzlich zu dem ursprünglichen Reifenmodell wird eine Vereinfachung betrachtet, die weniger Rechenaufwand erzeugt. Es wird eine lineare Funktion für die maximale laterale Kraft

$$F_y = C\alpha \quad (3.3.16)$$

angenommen, die durch die Maximalkraft  $|F_y| \leq F_{max}$  beschränkt ist. Nachteil der Vereinfachung ist die geringere Genauigkeit. Die zwei Modelle sind in der Abbildung 3.10

**Abbildung 3.10:** Laterale Kräfte der Reifen abhängig vom Schräglaufwinkel

dargestellt und wurden mit Parametern 3.1 berechnet, die vom High-Octane Motorsports e.V. zur Verfügung gestellt wurden.

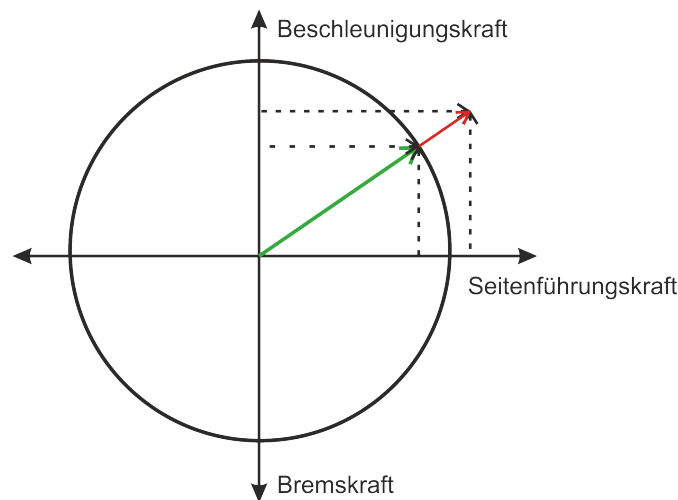
Nicht weiter spezifiziert wird im Folgenden davon ausgegangen, dass das genaueste Modell Verwendung findet.

#### Kammscher Kreis

Ähnlich wie bei  $F_y$  wird auch die Kraft  $F_x$ , welche das Fahrzeug in Längsrichtung beschleunigt, durch den Schlupf berechnet. Dieser hängt direkt von der Geschwindigkeit und Raddrehzahl ab. Da für die Bestimmung dieser jedoch eine Motorsimulation vonnöten wäre, wird  $F_x$  direkt aus der Motorleistung, Reibung und Luftwiderstand berechnet (siehe Abschnitt 3.3.3) und durch  $F_{max}$  begrenzt.  $F_{max}$  entspricht der maximalen Kraft, die der Reifen übertragen kann.

$$F_x \leq F_{max} \quad (3.3.17)$$

Der Zusammenhang zwischen lateraler und longitudinaler Kraft wird über den Kammschen Kreis (K.K.) modelliert und ist in Schaubild 3.11 dargestellt.



**Abbildung 3.11:** Kammscher Kreis: Beschränkung der maximal wirkenden Kraft.

Dieser schränkt die wirkenden Kräfte so ein, dass die Hypotenuse aus  $F_x$  und  $F_y$  (3.3.18) sich maximal auf einem Einheitskreis bewegen kann (3.3.19). Der Radius entspricht der maximalen Kraft, welche die Reifen übertragen können ( $F_{max}$ ).

$$F_{ll} = \sqrt{F_x^2 + F_y^2} \quad (3.3.18)$$

$$F_{ll} \leq F_{max} \quad (3.3.19)$$

Falls die Kraft  $F_{ll}$  größer als  $F_{max}$  ist, wird das Verhältnis der Kräfte

$$\alpha = \arctan(F_x, F_y) \quad (3.3.20)$$

berechnet und auf  $F_{max}$

$$F_x = F_{max} \sin(\alpha) \quad (3.3.21)$$

$$F_y = F_{max} \cos(\alpha) \quad (3.3.22)$$

herunterskaliert.

Bei einem gleichen Anteil an longitudinaler und lateraler Kraft können die Reifen also das größte Moment auf die Straße übertragen. Dieser Punkt wird von Rennfahrern versucht, in Kurven zu treffen, um das gesamte Potenzial der Reifen auszunutzen.

Mit dem Wissen wie die longitudinalen und lateralen Kräfte berechnet werden, kann nun ein genaueres Systemmodell genutzt werden.

### 3.3.3 Dynamisches Fahrzeugmodell

Die Basis ist, wie auch schon beim kinematischen Modell, das bicycle model. Es wird nun um die durch das zweite Newtonsche Gesetz entstehenden Kräfte entlang der  $y$ -Achse erweitert.

$$ma_y = F_{yf} + F_{yr} \quad (3.3.23)$$

Wobei  $a_y$  aus zwei Anteilen besteht, der Querbewegung  $\ddot{y}$  und der Zentripetalkraft  $\dot{x}\dot{\psi}$ . Die Kräfte  $F_{yf}$  und  $F_{yr}$  greifen jeweils am vorderen  $(.)_f$  und hinteren  $(.)_r$  Rad an. Die Zusammenhänge sind im Schaubild 3.12 veranschaulicht.



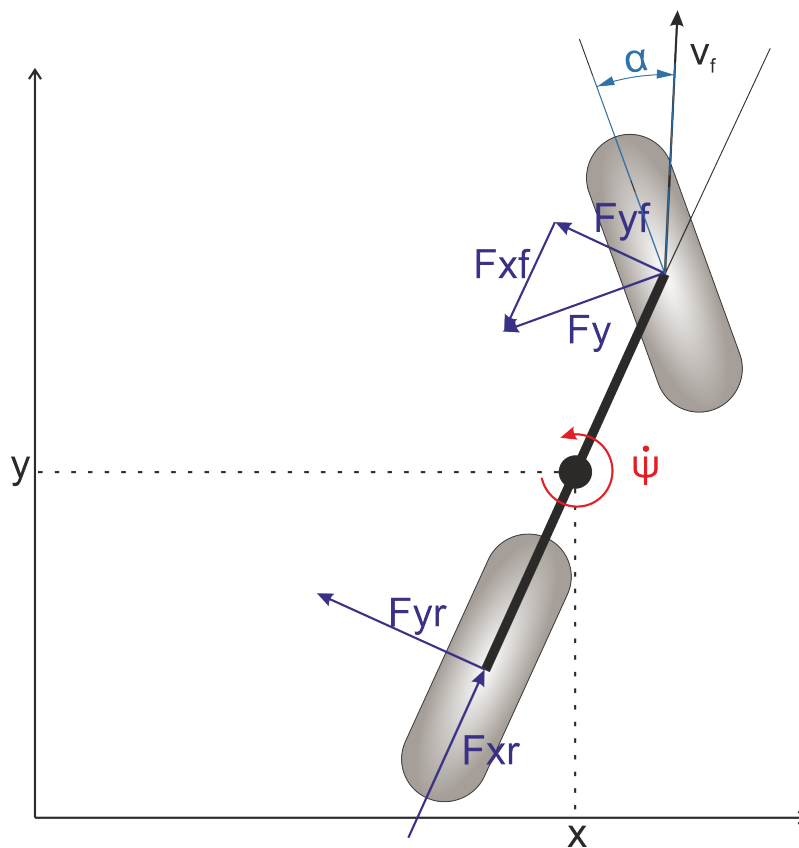


Abbildung 3.12: Dynamic Vehicle Model

Unter Berücksichtigung des Trägheitsmoments  $I_z$  des Fahrzeugs, kann das Drehmoment um die  $z$ -Achse betrachtet werden.

$$I_z \ddot{\psi} = l_f F_{yf} - l_r F_{yr} \quad (3.3.24)$$

Als Ergebnis lassen sich die Gleichungen für die Longitudinal-, Lateral- und Drehbewegung aufstellen.

$$m \ddot{x} = m \dot{\psi} + F_x \quad (3.3.25)$$

$$m \ddot{y} = -m \dot{x} \dot{\psi} + F_y \quad (3.3.26)$$

$$I(\ddot{\psi}) = l_f F_{yf} - l_r F_{yr} \quad (3.3.27)$$

Die Kräfte  $F_x$  und  $F_y$  wirken auf den Schwerpunkt des Fahrzeugs und setzen sich aus den Einzelkomponenten der Radkräfte zusammen.

$$F_x = F_{xf} + F_{xr} \quad (3.3.28)$$

$$F_y = F_{yf} + F_{yr} \quad (3.3.29)$$

Diese hängen ab von den lateralen  $(.)_C$  und longitudinalen  $(.)_l$  Radkräften und dem Lenkwinkel. Da das Vorderrad nicht angetrieben wird, besitzt es keinen longitudinalen Anteil. Aus den vom Reifenmodell errechneten Reifenkräften können die resultierenden Kräfte im dynamischen Modell abhängig von dem Lenkwinkel berechnet werden.

$$F_{xf} = -2F_{Cf} \sin(\delta_f) \quad (3.3.30)$$

$$F_{yf} = 2F_{Cf} \cos(\delta_f) \quad (3.3.31)$$

$$F_{xr} = F_{lr} \quad (3.3.32)$$

$$F_{yr} = 2F_{Cr} \quad (3.3.33)$$

Es ist zu beachten, dass das Fahrzeug in der Realität an jeder Achse zwei Reifen besitzt und daher die Kräfte verdoppelt werden müssen.

Die Kräfte  $F_{Cf}$  und  $F_{Cr}$  werden durch die M.F. wie im letzten Abschnitt 3.3.2 berechnet. Die dafür benötigten Schräglaufwinkel werden durch folgende Formeln bestimmt:

$$\alpha_f = \delta_f - \arctan\left(\frac{\dot{y} + l_f \dot{\psi}}{\dot{x}}\right) \quad (3.3.34)$$

$$\alpha_r = -\arctan\left(\frac{\dot{y} - l_r \dot{\psi}}{\dot{x}}\right) \quad (3.3.35)$$

### Longitudinale Kräfte

Da keine Simulation des Motors genutzt wird, müssen die Kräfte  $F_{lr}$ , welche die Reifen longitudinal auf die Straße bringen, anders als über den Schlupf berechnet werden. Die

longitudinale Kraft für Beschleunigung  $(.)_{acc}$  und Bremsvorgang  $(.)_{dec}$  wird unterschiedlich berechnet. Die Kraft, die der Motor beim Beschleunigen über die Achse an die Reifen überträgt

$$F_{lr_{acc}} = \frac{P_{engine}}{|\dot{x}|} \quad (3.3.36)$$

hängt von der Motorleistung  $P_{engine}$  und der Geschwindigkeit  $\dot{x}$  ab. Die Motorleistung wird über die Gaspedalstellung  $tp$  variiert und berechnet sich aus die Gleichung

$$P_{engine} = P_{engine_{max}} \cdot tp. \quad (3.3.37)$$

Für kleine Geschwindigkeiten begrenzen die Reifen die maximale auf die Straße übertragene longitudinal wirkende Kraft

$$F_{lr_{acc}} \leq 2F_{max}. \quad (3.3.38)$$

Beim Bremsvorgang, ergibt sich die longitudinale Kraft aus der Bremspedalstellung und den maximalen Reifenkräften.

$$F_{lr_{dec}} = -2F_{max} \cdot break \quad (3.3.39)$$

Solange sich das Fahrzeug vorwärts bewegt, wirken zusätzlich die Luft- und Rollreibung

$$F_{roll} = m\mu g \quad (3.3.40)$$

$$F_{aero} = \frac{1}{2}\rho C_d A_f \dot{x}^2 \quad (3.3.41)$$

der longitudinalen Kraft des Motors entgegen. Diese Gleichungen sind nur zulässig, solange von einem Rennkurs ohne Steigung ausgegangen wird. Die resultierende longitudinale Kraft

$$F_{lr} = F_{lr} - F_{aero} - F_{roll} \quad (3.3.42)$$

wird mit der lateralen Kraft im K.K. verrechnet und in der finalen Bewegungsgleichung verwendet.

$$\dot{x} = \dot{x} \cos(\psi) - \dot{y} \sin(\psi) \quad (3.3.43)$$

$$\dot{y} = \dot{x} \sin(\psi) + \dot{y} \cos(\psi) \quad (3.3.44)$$

$$m\ddot{x} = m\dot{y}\dot{\psi} + F_{xf} + F_{xr} \quad (3.3.45)$$

$$m\ddot{y} = -m\dot{x}\dot{\psi} + F_{yf} + F_{yr} \quad (3.3.46)$$

$$I\ddot{\psi} = l_f F_{yf} - l_r F_{yr} \quad (3.3.47)$$

Das Gleichungssystem wird hier in diskreter Form dargestellt.

$$x_{k+1} = X_k + \Delta t(\dot{x}_k \cos(\Psi_k) - \dot{y}_k \sin(\Psi_k)) \quad (3.3.48)$$

$$y_{k+1} = Y_k + \Delta t(\dot{x}_k \sin(\Psi_k) + \dot{y}_k \cos(\Psi_k)) \quad (3.3.49)$$

$$\Psi_{k+1} = \Psi_k + \Delta t\dot{\psi}_k \quad (3.3.50)$$

$$\dot{x}_{k+1} = \dot{x}_k + \Delta t \left( \frac{F_{xfk} + F_{xrk} - F_a}{m} + \dot{y}_k \dot{\psi}_k \right) \quad (3.3.51)$$

$$\dot{y}_{k+1} = \dot{y}_k + \Delta t \left( \frac{F_{yfk} + F_{yrk}}{m} - \dot{x}_k \dot{\psi}_k \right) \quad (3.3.52)$$

$$\dot{\psi}_{k+1} = \dot{\psi}_k + \Delta t \left( \frac{l_f F_{yf} - l_r F_{yr}}{I} \right) \quad (3.3.53)$$

$$(3.3.54)$$

Im dynamischen Fahrzeugmodell ändert sich damit auch der Zustandsvektor  $x, y, x_d, y_d, \Psi, \dot{\psi}$ .

Die Fahrzeugparameter 3.2 die zur Berechnung verwendet wurden, beziehen sich auf das aktuellste Fahrzeug des High-Octane Motorsports Verein.

**Tabelle 3.2:** Fahrzeug Parameter

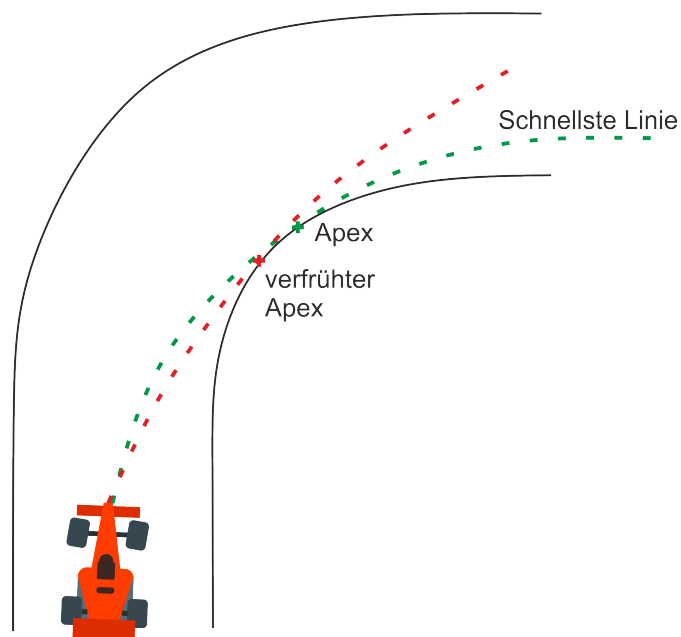
Formelzeichen	Wert	Einheit
$l_f$	0,66	m
$l_r$	0,97	m
$l_b$	1,53	m
$r$	0,2	m
$m$	220	kg
$I$	400	kgm <sup>2</sup>
$A_f$	0,6	m <sup>2</sup>
$P_{engine}$	53	kW
$C_d$	1.5	-
$\rho$	1,225	$\frac{\text{kg}}{\text{m}^3}$
$F_{max}$	1,6	kN

---

## 4 MPC zur Trajektorienplanung und Regelung

---

Ein schneller Rennfahrer zeichnet sich dadurch aus, dass er genau abschätzen kann, wie viel Kräfte die Reifen des Fahrzeugs auf die Straße übertragen können. Die Kunst liegt also darin, möglichst in jeder Fahrsituation das gesamte Potential voll auszunutzen. Zudem kann er sich dynamisch an Änderungen der Streckenverhältnisse anpassen und versucht immer die Idealtrajektorie zu treffen. Die Definition dieser ist die größtmögliche Streckenlänge in kleinstmöglicher Zeit zurückzulegen. Für eine Gerade oder eine einzige Kurve ist dies in Abbildung 4.1 dargestellt. Der sogenannte Scheitelpunkt markiert hierbei den Punkt des geringsten Radius und der kleinsten Geschwindigkeit. An diesem Punkt muss das Fahrzeug im besten Fall genau die Seitenbegrenzung tangieren. Bei komplexeren Streckengefügen kann es jedoch von Vorteil sein, eine Kurve eventuell nicht ideal auszufahren, um im späteren Verlauf einen größeren Geschwindigkeitsgewinn in einer anderen Kurve einfahren zu können.



**Abbildung 4.1:** Ideale Trajektorie für eine 90 Grad Kurve

Um all dies mit dem MPC-Algorithmus nachzustellen, wird im Folgenden auf die einzelnen Schritte bei der Entwicklung eingegangen.

Zu Beginn hätte die Möglichkeit bestanden die Regelung der lateralen und longitudinalen Führung des Fahrzeuges zu trennen. Vorteile hierfür wären gewesen:

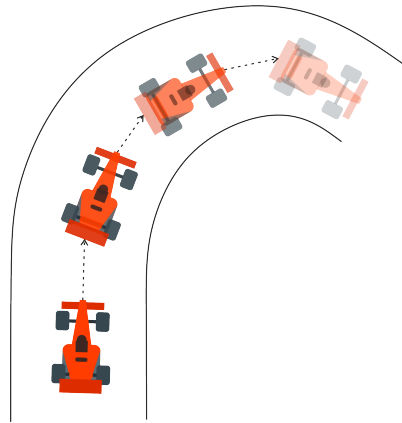
- Einfachere Anpassung der Parameter für das reale Fahrzeug
- Robustheit. Der Ausfall eines Reglers würde zumindest noch eine eingeschränkte Kontrolle ermöglichen
- Zwei einfacher zu berechnende Probleme. Da die Berechnungszeit nicht linear mit der Komplexität steigt, wären hier möglicherweise deutliche Geschwindigkeitssteigerungen möglich
- Ein weniger komplexer MPC-Ansatz ist einfacher zu testen und zu entwickeln

Die Vorteile werden aber mit dem Nachteil begleitet, dass bei einem Rennauto die longitudinale und laterale Bewegung des Fahrzeugs stark miteinander korrelieren. Durch die Trennung kann keine optimale Lösung mehr gefunden werden. Zudem ist die Integration komplexer, da beide Regler gleichzeitig laufen und ihre Berechnungen miteinander ausgetauscht werden müssen. Aufgrund des Anwendungsszenarios wurde sich gegen den zweigeteilten Ansatz entschieden.

Der erste Schritt ist das Erstellen des Vektors an Einflussparametern  $\vec{X}$ . Er setzt sich immer aus der Kombination von Systemzustand und Steuerparametern zusammen  $\vec{x} = [x, y, v, \psi, a, \delta]^T$  (siehe 3.3.1). Dieser Teilvektor wird dann für die Anzahl der gewünschten Prädiktionsschritte  $N$  mal in  $\vec{X}$  wiederholt. Da zusätzlich zu den Prädiktionsschritten auch der aktuelle Fahrzeugzustand benötigt wird, besteht der Vektor also aus  $N + 1$  Teilvektoren  $\vec{x}$ . In Vektordarstellung entspricht dies dann:

$$\vec{X} = \begin{bmatrix} \vec{x}_0 \\ \vec{x}_1 \\ . \\ . \\ \vec{x}_N \end{bmatrix} \text{ mit } \vec{x}_i = \begin{bmatrix} x_i \\ y_i \\ v_i \\ \psi_i \\ a_i \\ \delta_i \end{bmatrix}$$

In der Abbildung 4.2 ist grafisch dargestellt, wie die Prädiktion für  $N = 3$  aussieht. Zwischen jedem der einzelnen Schritte wird ein  $\Delta t$  angenommen, dass dem der Positionsschätzung entspricht, also  $\frac{1}{20}$  s. Obwohl immer nur der erste Steuerbefehl des berechneten Steuervektors



**Abbildung 4.2:** Grafische Visualisierung der Prädiktion abhängig von den Steuerparametern für drei Schritte in die Zukunft

im realen System angewandt wird, ist der Vektor trotzdem von Nutzen. Sollte die Optimierung, unterbrochen durch andere Prozesse, einmal nicht schnell genug sein, können die Steuerbefehle aus der vorherigen Optimierung verwendet werden. Dies funktioniert solange die Zeitverzögerung kleiner als die gesamte Zukunftsprädiktion ist. Obwohl diese Werte zum Regeln genutzt werden können, nimmt aufgrund von Modellfehlern die Güte der berechneten Werte mit der Anzahl der Prädiktionsschritte ab. Es ist also nicht sinnvoll, den MPC-Algorithmus langsamer als die Positionsschätzung berechnen zu lassen.

Nachdem der Vektor mit den Einflussparametern definiert ist, werden im nächsten Schritt alle Beschränkungen sukzessive ergänzt.

## 4.1 Fahrzeugmodell

Zu Beginn haben die Teilvektoren  $\vec{x}$  keinen Bezug zueinander. Da jedoch die einzelnen Prädiktionsschritte voneinander über den Fahrzeugzustand, Steuerparameter und das Fahrzeugmodell zusammenhängen, werden im zweiten Schritt die Beschränkungen hierfür definiert und als Ungleichheitsbedingungen in das zu optimierende nichtlineare Problem integriert. Dazu wird die diskretisierte Form des Systemmodells genutzt, um immer zwei aufeinander folgende Schritte miteinander zu verknüpfen.

Zusätzlich zu der Systembeschreibung fehlen noch Einschränkungen für den Optimierer, welche die physikalischen Eigenschaften des Rennautos abbilden. Dazu zählen die maximale Geschwindigkeit, Beschleunigung und Lenkwinkel. Diese Beschränkungen werden für jeden der Teilvektoren hinterlegt. Zusammen mit dem Fahrzeugmodell benötigt das MPC



noch eine Kostenfunktion, damit der Optimierer überhaupt einen Anlass hat das Rennauto zu Beschleunigen.

## 4.2 Kostenfunktionen

Erst eine geeignete Kostenfunktion führt zu der Planung einer Trajektorie, die das Rennauto um den Kurs führt. Es wurden drei verschiedene Kostenfunktionen implementiert:

### Maximalgeschwindigkeit

Mit der Funktion

$$f(\vec{X}) = \sum_1^N \frac{1}{v_i} \quad (4.2.1)$$

wird die Summe aller inversen Geschwindigkeitswerte der einzelnen Prädiktionsschritte addiert und als Kosten versucht zu maximieren. Die Berechnung bezieht den Geschwindigkeitswert aus dem aktuellen (0-ten) Fahrzeugzustand nicht ein, da der Optimierer keinen Einfluss mehr auf diesen hat. Der Vorteil dieser Funktion ist die einfache Berechnung und leichte Integration in den MPC-Algorithmus. Im weiteren Verlauf wird diese Kostenfunktion als max-speed bezeichnet.

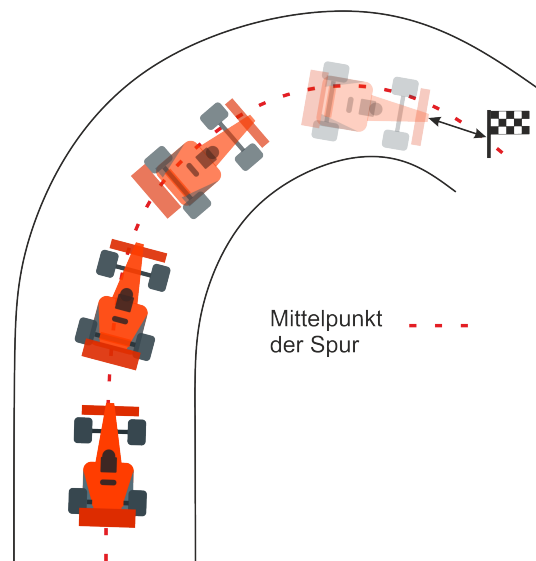
### Zielpunkt Distanzminimierung

Für diese Kostenfunktion wird vor dem letzten Prädiktionsschritt ein virtuelles Ziel definiert und die Distanz des  $N + 1$  Schrittes zu diesem Ziel minimiert. Der Optimierer wird also Steuerparameter suchen, die ihn möglichst dicht an dieses Ziel bringen. Das virtuelle Ziel wird nach jedem Update wieder so neu positioniert, dass das Rennauto den Punkt niemals erreichen kann und damit konstant schnell weiterfährt. Die Distanz  $d_{goal}$ , in der das virtuelle Ziel platziert wird, errechnet sich durch die maximale Geschwindigkeit und die Anzahl der Prädiktionsschritte durch die Gleichung

$$d_{goal} = (v_0 + N \cdot v_{max}) \Delta t. \quad (4.2.2)$$

Eine bildliche Darstellung der Kostenfunktion ist in Abbildung 4.3 zu sehen.

Im weiteren Verlauf wird diese Kostenfunktion mit min-dist bezeichnet.



**Abbildung 4.3:** Zielpunkt Distanzminimierung: Das virtuelle Ziel wird immer weit genug vor dem Fahrzeug hergeschoben, so dass es nie erreicht werden kann

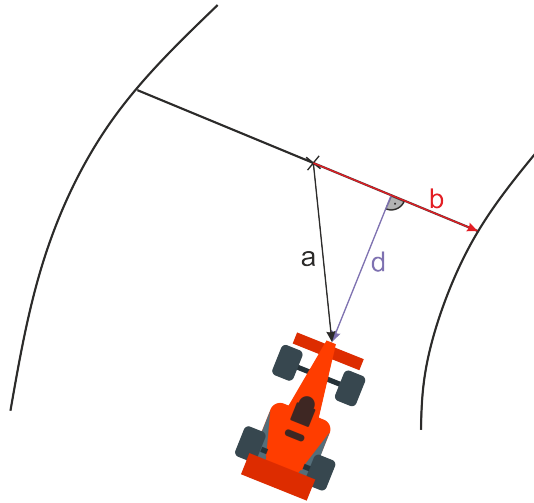
### Maximieren der Streckendistanz

Um eine Kostenfunktion zu finden, die einer Idealtrajektorie am nächsten kommt, muss überlegt werden wie diese gemessen werden kann. Die bestmögliche Trajektorie ist diejenige, welche für ein  $\Delta t$  die zurückgelegte Strecke, nicht bezogen auf das Fahrzeug, sondern auf den Rennkurs, maximiert. Würde die vom Rennauto zurückgelegte Strecke maximiert werden, würde das Rennauto einfach immer an der äußeren Begrenzung des Kurses entlang fahren. Die Streckenmaximierung lässt sich zu einer Minimierung der nach einem Zeitschritt übrig bleibenden Reststrecke umformen. Die Reststrecke wird einfachheitshalber auf einen Punkt bezogen, der in einer Distanz vor dem letzten Prädiktionsschritt liegt, den dieser nicht erreichen kann. Der Unterschied zur Kostenfunktion, welche die Distanz minimiert ist, dass sich die Differenz nicht auf einen Punkt, sondern auf eine Gerade senkrecht zur Rennkursmitte bezieht. Realisiert wird dies durch eine Vektorabbildung (eng.: vector rejection)

$$d = \left| a - \frac{ab}{bb} b \right|. \quad (4.2.3)$$

Der Vektor vom Mittelpunkt der Strecke zum Massenschwerpunkt des Fahrzeugs wird als  $a$  bezeichnet und  $b$  als der Vektor vom Mittelpunkt der Strecke zur Streckenbegrenzung. Der

Zusammenhang ist in der Grafik 4.4 grafisch veranschaulicht. Zur besseren Darstellung wurde die Nase des Rennautos und nicht der Massenschwerpunkt gewählt.



**Abbildung 4.4:** Maximieren der Streckendistanz

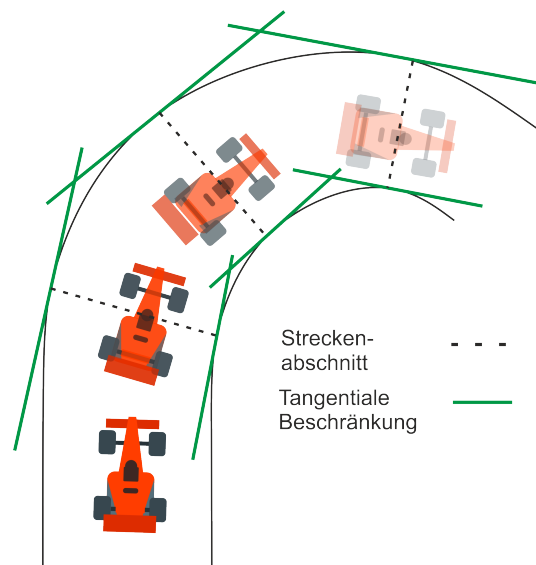
Im weiteren Verlauf wird diese Kostenfunktion als max-track bezeichnet.

### 4.3 Strecken-, und Positionsbeschränkung

In Folge der Kostenfunktionen werden die Steuerparameter durch den Optimierer so angepasst, dass die Kosten minimal werden. Dies würde auf einem Rennkurs zum Abkürzen bei Kurven führen, da in diesem Fall die Kosten geringer werden. Um das zu verhindern, wird eine zusätzliche Streckenbeschränkung eingeführt. Diese basiert auf zwei Tangenten, die für jeden Prädiktionsschritt an die Fahrbahnaußen- und innenseite projiziert werden. Die Tangenten sind in der Abbildung 4.5 dargestellt.

Solange sich jeder Prädiktionsschritt innerhalb seines Tangentenpaares bewegt, ist die Beschränkung erfüllt. Während des Optimierungsvorgangs wird die Position der Tangenten nicht verändert. Der Prädiktionsschritt  $n$  benötigt einen passenden Streckenabschnitt, für den die Beschränkungen ausgelegt werden. Dazu kann auf die Ergebnisse der vorausgehenden Optimierung zurückgegriffen werden. Die Position des  $n + 1$ -Schrittes wird gewählt und für diesen die Tangenten berechnet. Für den letzten Schritt  $N + 1$  ist dies nicht möglich, hier wird mithilfe der Geschwindigkeit und Orientierung ein virtueller Punkt projiziert.

Dieses Verfahren funktioniert nur durch die physikalischen Grenzen der Beschleunigung des Rennautos. Das Fahrzeug bewegt sich immer in einem bestimmten Bereich um



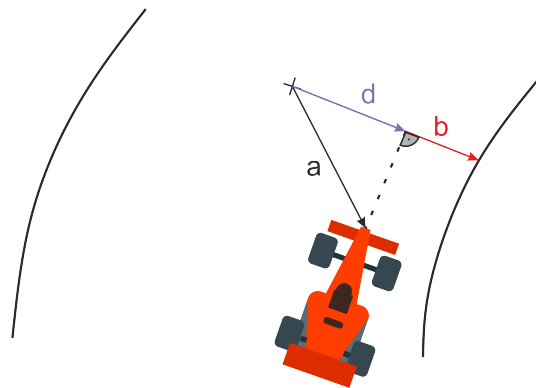
**Abbildung 4.5:** Tangentiale Begrenzung hält das Fahrzeug auf der Fahrspur

den Punkt herum, der bei gleichbleibender Geschwindigkeit  $\Delta t \cdot v$  Meter entfernt von seinem Vorgängerzustand liegt. Wenn sich dieser Punkt in einer Kurve direkt an der Streckenmarkierung befindet, könnte das Fahrzeug sich vom Kurs herunter bewegen, obwohl die Tangentialbeschränkung erfüllt ist. Dies wird dadurch verhindert, dass der nächste Prädiktionsschritt in diesem Fall seine Beschränkung nicht mehr erfüllen würde. Mathematisch lassen sich die Tangenten durch eine Vektorprojektion

$$d = \frac{a * b}{|b|} \quad (4.3.1)$$

darstellen. Der Vektor vom Mittelpunkt der Strecke zum Massenschwerpunkt des Rennautos entspricht  $a$  und der Vektor vom Mittelpunkt zum Rand des Rennkurses  $b$ . Mit der Beschränkung der Distanz  $|d|$  auf die Hälfte der Rennkursbreite wird also sichergestellt, dass das Rennauto immer auf dem Kurs bleibt. Die Vektorprojektion ist im Schaubild 4.6 dargestellt.

Für diese Beschränkung können sich die Räder des Fahrzeuges über den Rand hinausbewegen. Soll dies verhindert werden, muss diese Beschränkung auf jedes der einzelnen Räder erweitert werden. Das einfache verjüngen der Streckenbreite ist nicht ausreichend, da der Schwerpunkt des Fahrzeugs nicht in der Mitte liegt und das Fahrzeug auch quer stehen kann.



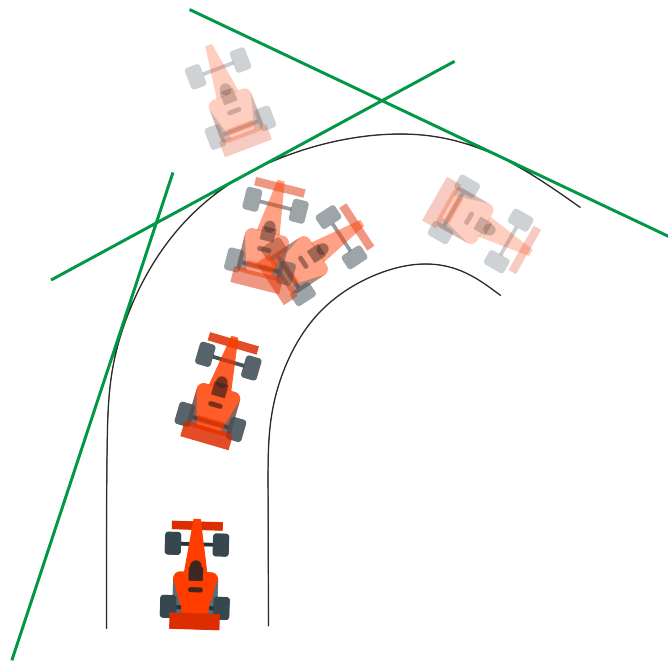
**Abbildung 4.6:** Vektor Projektion zum berechnen der tangentialen Beschränkung

Im letzten Schritt, bevor der MPC-Algorithmus funktionsbereit ist, wird der Fahrzeugzustand des aktuellen Zeitschritts  $t_0$  als Beschränkung verankert. Dies ist nötig um zu verhindern, dass der Optimierer einfach die  $x$ -, und  $y$ -Position auf den Punkt legt, welcher die Kostenfunktion ideal minimiert.

### Suchbereichsbeschränkung

Während dem Testen der Kostenfunktion zum Maximieren der Geschwindigkeit ist ein Phänomen aufgetreten. Der Optimierer fand augenscheinlich unmögliche Lösungen, bei denen sich das Rennauto links aus einer Rechtskurve und andersherum herausbewegt. Dies passiert jedoch nur in Kurven, die mehr als  $90^\circ$  aufweisen. Der Grund hierfür, liegt in den tangentialen Beschränkungen und darin, dass sie immer nur am Anfang des MPC-Zyklus aktualisiert werden. Werden die Tangenten wie in Abbildung 4.7 sehr lang eingezeichnet, ist ersichtlich, dass eine zweite mögliche Trajektorie entstehen kann. Diese ermöglicht eine höhere Geschwindigkeit. Bei der Berechnung der tangentialen Beschränkung für den nächsten MPC-Durchlauf liegen jetzt jedoch alle Tangenten nahezu auf einem Punkt und verhindern daher das Ausscheren des Rennautos. Die daher entstehende neue optimale Trajektorie entspricht wieder nahezu der ursprünglichen. Der resultierende Dreierzyklus wird im schlechtesten Fall solange durchlaufen, bis das Fahrzeug eine der Tangenten trifft und die Optimierung abgebrochen wird.

Um dieses Verhalten zu verhindern, wird für diese Kostenfunktion eine zusätzliche Suchbereichsbeschränkung eingeführt. Diese legt um die Prädiktionsschritte der letzten Optimierung einen Kreis zur Einschränkung des Suchbereiches. Ziel ist es, keine so stark



**Abbildung 4.7:** Fehlerhaftes Verhalten in Spitzkurven

abweichende Trajektorie mehr entstehen zu lassen. Es ist darauf zu achten, dass das Fahrzeug trotzdem in den Grenzen seines Fahrzeugmodells keine Beschränkung erfährt.

Die Suchbereichseinschränkung ist nur für die Kostenfunktion zum Maximieren der Geschwindigkeit benötigt worden.

### **Elastische Distanzminimierung**

Der Nachteil der oben aufgeführten Kostenfunktionen ist, dass sie keinen Spielraum beim Erfüllen der Streckenbeschränkung haben. Wenn sowohl das im Simulator wie auch im MPC hinterlegte Fahrzeugmodell exakt gleich sind, kann das virtuelle Rennauto in der Simulationsumgebung bis an die Streckenbegrenzung heranfahren. In der Realität würde eine solche Kostenfunktion das Fahrzeug jedoch zu nah an die Pylonen heranführen. Bei den kleinsten Regelfehlern, Kontaktverlust mit der Fahrbahn (Rutschen), Abweichungen vom Fahrzeugmodell etc. würde das Rennauto vom Kurs abkommen und damit der initiale Fahrzeugzustand ungünstig für den Optimierer werden. Um dieses Problem zu umgehen, wird eine neue Art der Kostenfunktion eingeführt, die die Ungleichheitsbedingung der Streckenbegrenzung ersetzt. Dafür wird eine der oberen Funktionen durch einen Kostenanteil

erweitert, der wächst, sobald sich das Fahrzeug vom Mittelpunkt der Fahrbahn nach außen hin bewegt. Das Verhältnis der einzelnen Kostenanteile wird der kombinierten Kostenfunktion

$$f(\vec{x}) = a \left( \sum_1^N v_i \right) + \left( \sum_1^N g(h(\vec{x}_i)) \right) \quad (4.3.2)$$

durch den Parameter  $a$  beeinflusst. Die Funktion  $g(d)$  dient als Platzhalter für verschiedene Gewichtungsfunktionen. Die Funktion  $h(\vec{x}_i)$  berechnet die Distanz  $d$  vom Mittelpunkt der Strecke genauso wie bei den tangentialen Beschränkungen in Abschnitt 4.3. Mögliche Funktionen sind:

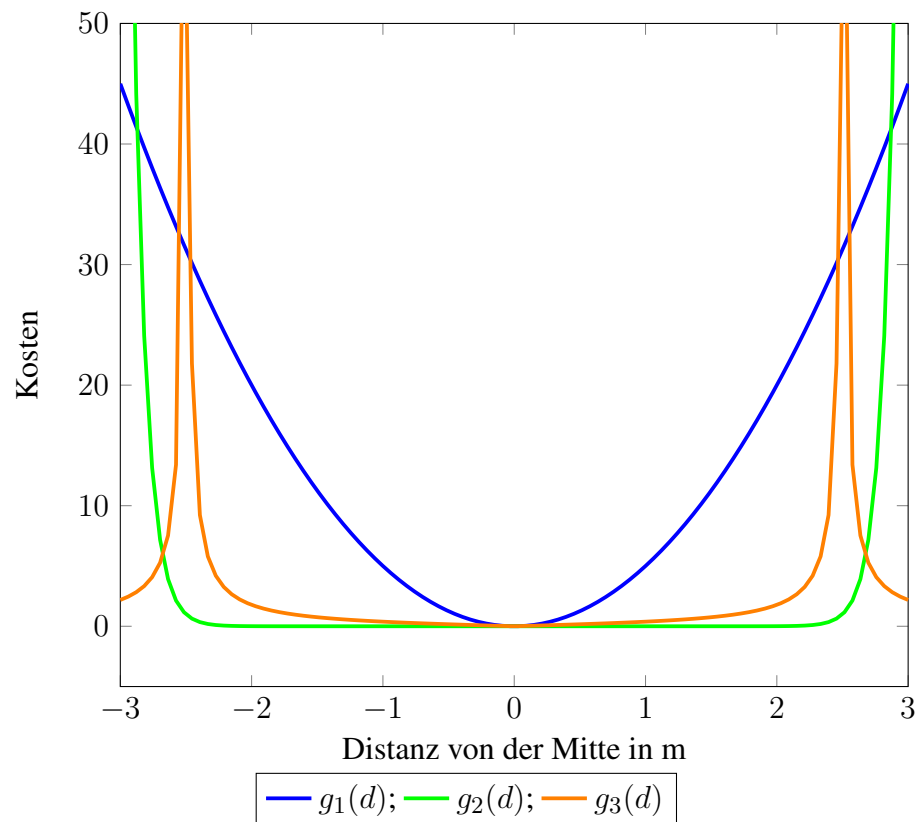
$$g_1(d) = \alpha d^2 \quad (4.3.3)$$

$$g_2(d) = e^{\alpha(k_1+d)} + e^{-\alpha(k_2+d)} \quad (4.3.4)$$

$$g_3(d) = \left| \frac{\alpha}{k_1-d} + \frac{\alpha}{k_2-d} \right| \quad (4.3.5)$$

Die Parameter  $\alpha$ ,  $k_1$  und  $k_2$  dienen zum Variieren der Scheitelpunkte und Steigung. Mögliche Graphen der Gleichungen sind zur Veranschaulichung in Abbildung 4.8 dargestellt.

Die Idee der Funktionen (4.3.4) und (4.3.5) ist die uneingeschränkte Bewegung des Rennautos auf der ganzen Breite der Strecke. Der Kostenanteil der Distanz von der Mitte wird also bis zum Erreichen der Streckengrenzung nahezu vernachlässigt. Erst beim Erreichen des Fahrbahnrandes werden die Kosten sehr schnell extrem groß. Bestenfalls stellen also diese Kostenfunktionen die tangentialen Begrenzungen ideal nach. Die quadratisch wachsende Funktion soll ebenfalls betrachtet werden. Es gilt zu untersuchen, ob bei gesenktem Rechenaufwand eine erfolgreiche Regelung möglich wäre.



**Abbildung 4.8:** Verschiedene Distanzmaße um das Verhalten der Trajektorienplanung zu beeinflussen



---

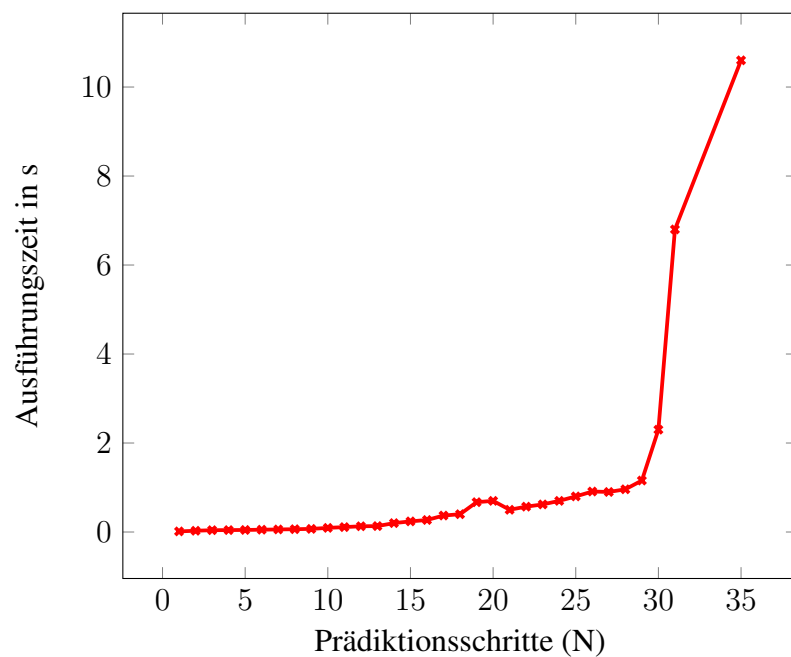
## 5 Simulationsumgebung

---

Um die Funktion des MPC-Algorithmus verifizieren zu können, wurde eine Simulationsumgebung entwickelt. Der Aufbau ist dreigeteilt: Eine Fahrzeugsimulation berechnet mit austauschbaren Fahrzeugmodellen die Bewegung des Rennautos abhängig von Steuereingaben und einem  $\Delta t$ . Der Regelanteil in Form des MPC-Algorithmus und eine Visualisierung auf Basis einer Spiele-Engine welche auch die zeitlichen Abläufe kontrolliert.

### 5.1 Python

Zu Beginn der Arbeit wurde die Programmiersprache Python verwendet, um den MPC-Ansatz zu programmieren. Der Vorteil dieser Sprache ist das schnelle Prototyping. Als Optimierer wurde ein Sequential Least Squares Programming (SLSQP)-Algorithmus verwendet. Dieser ist direkt in die SciPy-Bibliothek [Sci] integriert und unterstützt nichtlineare Nebenbedingungen mit denen sich das MPC-Problem zur Fahrzeugregelung abbilden lässt. Solange die Anzahl der Prädiktionsschritte zehn nicht überschreitet, ist die Ausführungszeit gering genug, um eine Aktualisierungsrate von 20 Hz zu erreichen. Bei größeren Prädiktionsvektoren steigt die Berechnungszeit sehr schnell an. Der Benchmark in Abbildung 5.1 ist für das fertige Optimierungsproblem gemessen worden. Dies liegt daran, dass der SLSQP-Algorithmus nicht für nichtlineare Probleme ausgelegt ist. Eine bessere Skalierung wird durch einen Lösungsalgorithmus erreicht, der das Innere-Punkte-Verfahren nutzt. Dieser Algorithmus setzt jedoch für eine gute Performance die Kenntnis über die Ableitung der Systemfunktion voraus. Auf diesen Sachverhalt wurde bereits im Abschnitt 3.2.2 eingegangen. Die automatische Ableitung kann mit Frameworks wie CasADi [Cas], PyADOL-C [PYA], PyCppAD [Cpp] berechnet werden [TT16]. Eine Implementierung des MPC-Ansatzes in CasADi wurde aufgrund der schlechten Dokumentation abgebrochen und es wurde nach einer Alternativlösung gesucht. Das Ergebnis dieser Suche führte zur Verwendung einer neuen Programmiersprache.



**Abbildung 5.1:** Ausführungszeiten für verschieden große Horizontlängen

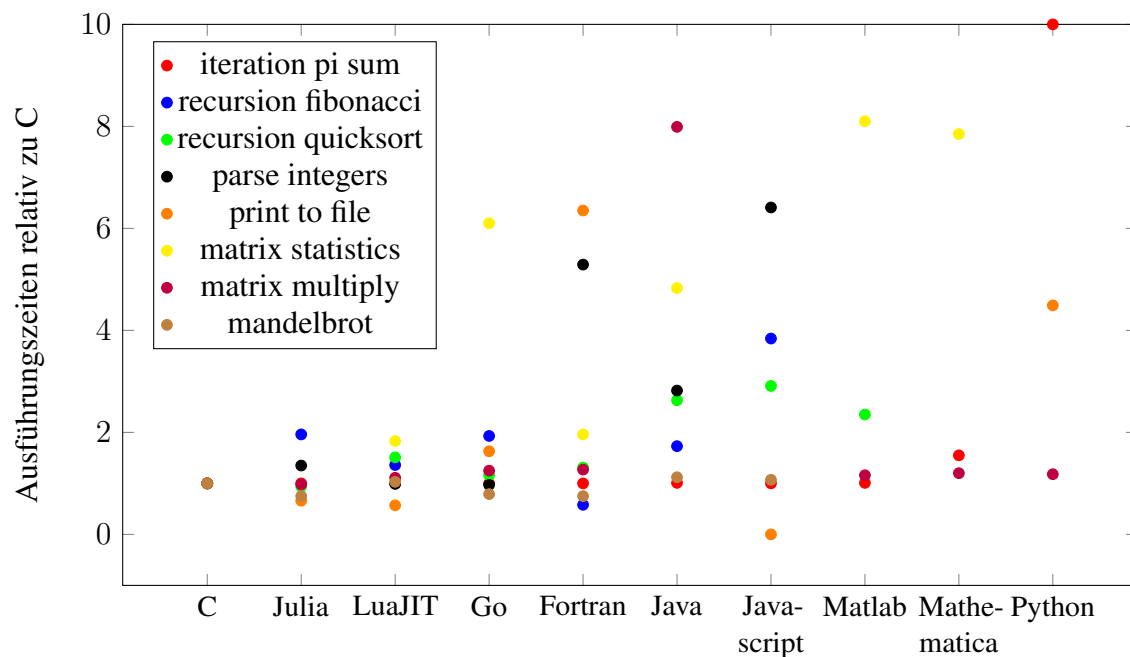
## 5.2 Julia

Julia ist die Programmiersprache, die eine Umsetzung des MPC-Algorithmus effizient und performant ermöglicht. Sie wurde im Jahr 2009 von Jeff Bezanson, Stefan Karpinski, Viral B. Shah, und Alan Edelman ersonnen und ist damit noch eine sehr junge Sprache. Sie ist vor allem für numerische Analysen und wissenschaftliche Berechnungen entworfen worden und bietet in diesen Bereichen viele Funktionalitäten. Ebenfalls ein wichtiges Ziel bei der Entwicklung war es, ohne vorheriges Kompilieren sehr schnell zu sein und trotzdem weiterhin das general-purpose Paradigma zu erfüllen. Ein Vergleich der Berechnungsgeschwindigkeit verschiedener Sprachen bezüglich viel genutzter Funktionen im wissenschaftlichen Umfeld ist in der Abbildung 5.2 zu sehen.

Die wichtigsten Features von Julia sind:

- Funktionsüberladung (eng.: multiple dispatch)

Wird genutzt um die gleiche Methode für verschiedene Kombinationen an Eingabeparametern oder Variablentypen zu überladen. Bei dem Funktionsaufruf wird die passende Methodendefinition aufgerufen.



**Abbildung 5.2:** Ausführungszeiten für verschiedene Sprachen

- Datentyp detektion (eng.:type inference)

Das automatische Detektieren des Datentyps. Dies befreit den Programmierer vom Festlegen des Typs und ermöglicht trotzdem Typprüfung.

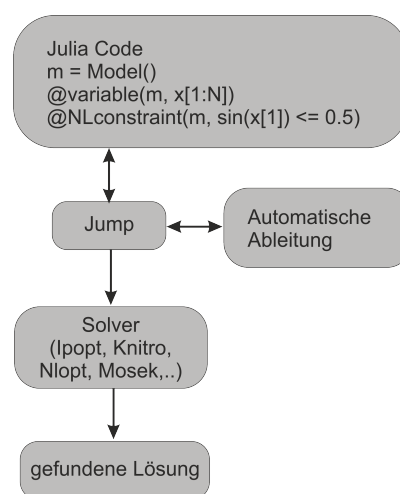
- just-in-time (JIT) Kompilierer

Ein System, in dem ein JIT Kompilierer implementiert ist, wandelt den Computer Code erst zur Laufzeit in Bytecode um. Der große Vorteil liegt in der kontinuierlichen Analyse während der Ausführung. Dadurch können Bereiche ausfindig gemacht werden, die durch ein Neukompilieren eine signifikante Verbesserung bei der Ausführungszeit erfahren würden.

Es wäre möglich die einzelnen benötigten Bestandteile für einen MPC-Algorithmus in Julia zu implementieren. Jedoch gibt es eine umfangreiche Erweiterung, welche eine passende Schnittstelle für solche Probleme zur Verfügung stellt.

## Jump

Julia for Mathematical Optimization (Jump) ist eine Bibliothek für Julia mit der sich mathematische Probleme modellieren und lösen lassen. Es besitzt Schnittstellen für mehrere sowohl frei verfügbare, wie auch kommerzielle Optimierer, mit denen lineare oder nichtlineare Probleme modelliert und gelöst werden können. Der große Vorteil von Jump ist, dass die Definition des mathematischen Problems unabhängig von dem verwendeten Optimierer ist und dieser damit leicht ausgetauscht werden kann (siehe Schaubild 5.3). Ebenfalls ein großer Vorteil ist die direkt berechnete automatische Ableitung in Jump. Der Programmierer spart sich dadurch sehr viel Arbeit.

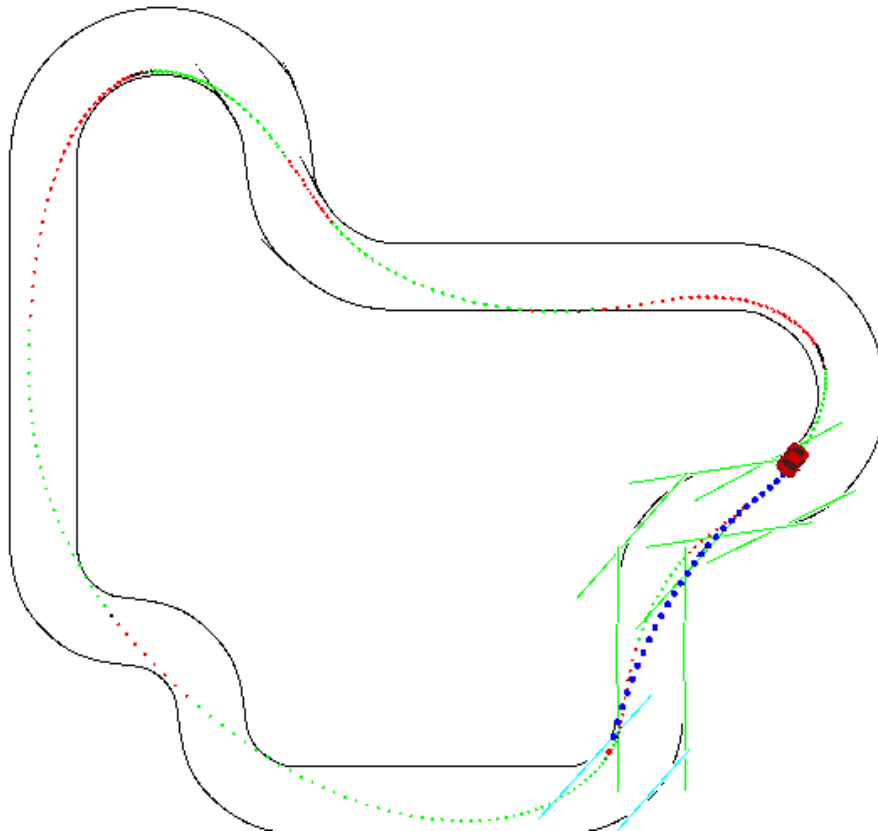


**Abbildung 5.3:** Blockdiagramm von Jump. Wichtige Vorteile sind die automatische Ableitung und das einfache Tauschen von Optimierern

Die Definition der Optimierungsprobleme wird in Jump mit einer Syntax vorgesehen, die der standardmäßigen mathematischen Definition ähnelt. Die Integration gestaltet sich daher sehr intuitiv. Zudem existiert eine ausführliche Dokumentation. Obwohl Jump Schnittstellen zu vielen verschiedenen Optimierern besitzt, ist nur der Interior Point OPTimizer (Ipopt) frei nutzbar und dafür ausgelegt nichtlineare Nebenbedingungen zu berücksichtigen. Die Verwendung von Jump und Ipopt führen zu einer deutlich besseren Skalierung und Laufzeit (siehe 6.2.2.2) als der SLSQP-Ansatz in Python. Die Abbruchbedingung in Ipopt wurde auf  $1e-1$  festgelegt.

### 5.3 Simple and Fast Multimedia Library

Die Basis für die Simulation bildet die Simple and Fast Multimedia Library (SFML), die für die grafischen Darstellungen des Kurses, Rennauto, Prädiktionsschritte und Fahrzeuginformationen genutzt wird. Ein Bildschirmfoto der Simulation ist in Abbildung 5.4 dargestellt. Zusätzlich



**Abbildung 5.4:** Grafische Darstellung des MPC - Algorithmus in der Simulationsumgebung

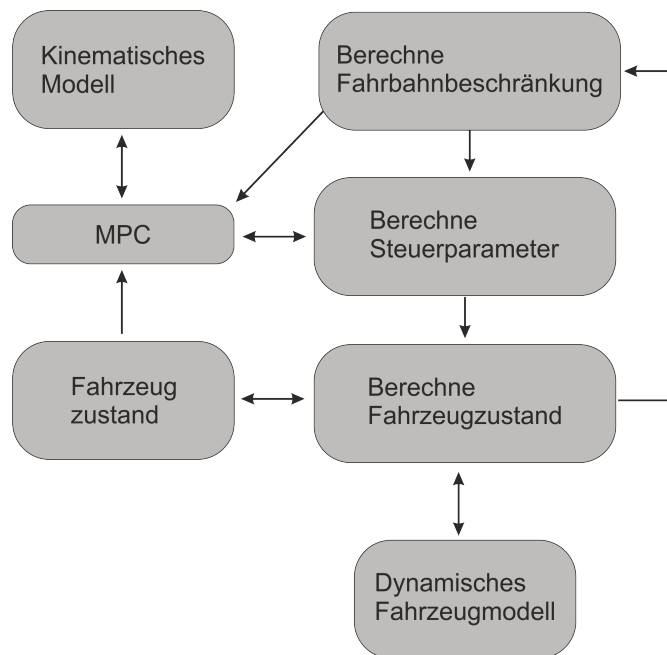
zur Visualisierung stellt die Spiele-Engine auch sicher, dass die zeitlichen Abläufe eingehalten werden. Das Grundprinzip ist eine einzige unendlich laufende Schleife, die mit einer vorher festgelegten Wiederholungsrate ausgeführt wird. Benötigen die Berechnungen innerhalb dieser Schleife länger als das angegebene  $\Delta t = \frac{1}{20}$  s, sinkt die Ausführungsrate, überschritten wird sie jedoch nie. Innerhalb dieser Schleife werden mögliche Eingaben der Nutzer oder Ereignisse der einzelnen Objekte abgearbeitet. Im zweiten Schritt werden alle Berechnungen der eigentlichen Fahrzeugsimulation und MPC-Algorithmus ausgeführt und im letzten Schritt werden die grafischen Elemente erstellt und angezeigt. Wie für eine Spiele-Engine üblich, befindet sich der Ursprung des Koordinatensystems in der linken

oberen Ecke. Die  $y$ -Achse ist daher entgegengesetzt zu dem in der Fahrzeugsimulation verwendeten Standardachsenaufbau orientiert. Zudem werden Distanzen in der Engine nur in Pixeln gemessen. Es wurden daher zwei Parameter eingeführt, die die Fenstergröße in Pixeln festlegen (in  $x$ - und  $y$ -Richtung) und zusätzlich eine Angabe wie vielen Metern dieser jeweils in Pixeln entspricht. Die daraus resultierenden Verhältnisse

$$scaleX = \frac{windowSizeXinPixel}{windowSizeXinM} \quad (5.3.1)$$

$$scaleY = -\frac{windowSizeYinPixel}{windowSizeYinM} \quad (5.3.2)$$

werden verwendet, um alle Größenverhältnisse einheitlich in der Simulation zu halten und eine realistische Visualisierung zu gewährleisten. Die Größe des Bereichs in dem der Rennkurs abgesteckt wird und die Fenstergröße zur Darstellung der Simulation können durch Parameter angepasst werden. Zusätzlich wurde ein Offset eingeführt, der die Nullpunkte der  $x$ - und  $y$ -Position im Koordinatensystem verschiebt. Damit kann der Ursprung des Koordinatensystems der Fahrzeugsimulation beliebig im Anzeigebereich verschoben werden. Der Aufbau ist in dem Blockdiagramm 5.5 nochmals zusammengefasst. Der Simulator berechnet abhängig vom aktuellen Fahrzeugzustand die Streckenbegrenzungen



**Abbildung 5.5:** Blockdiagramm der Simulation mit dem MPC-Algorithmus

und hinterlegt diese zusammen mit dem Fahrzeugzustand im MPC-Algorithmus. Im nächsten Schritt errechnet der Optimierer die Steuerparameter für den Prädiktionshorizont. Zusammen mit dem dynamischen Fahrzeugmodell und den Steuerparametern, wird der neue Fahrzeugzustand berechnet und der nächste Durchlauf der Simulationsschleife gestartet.

## 5.4 Virtueller Rennkurs

Wie in der Einführung bereits erwähnt, ist das Ziel der Arbeit ein MPC-Algorithmus zu entwickeln mit dem die Trackdrive-Disziplin möglichst schnell abgefahren werden kann. Die Grundvoraussetzung für diese Arbeit ist die bereits vollständig erstellte Karte des Rennkurses und eine Lokalisierung innerhalb dieser Karte. Die Updaterate für diese Positionsschätzung wird mit 20 Hz angenommen. Das Ziel ist es also, die Regelparameter mit einem  $\Delta t$  von  $\frac{1}{20}$  s für das Rennauto berechnen zu können. Zum Testen der Algorithmen wurde ein beliebiger Kurs definiert. Entscheidend war das Einhalten des Regelwerks mit einem minimalen Kurvenradius von 9 m, Maximallänge einer Geraden von 80 m und maximal 180° Spitzkurven. Die Strecke wird als kubischer Spline hinterlegt. Bei einem Spline (auch Polynomzug) handelt es sich um eine Funktion, die stückweise aus Polynomen zusammengesetzt ist. Kubische Splines werden unter anderem zur Berechnung des Bahnverlaufes bei Achterbahnen verwendet, um ruckartige Beschleunigungswechsel für die Fahrgäste zu vermeiden. Auch beim Entwurf von Kurven und Oberflächen sind Splines von Bedeutung. Die Stützpunkte werden durch das Zusammensetzen von verschiedenen Streckenelementen erzeugt. Der Vorteil ist das schnelle und einfache erstellen beliebiger Strecken.

---

## 6 Evaluierung

---

### 6.1 Modellfehler

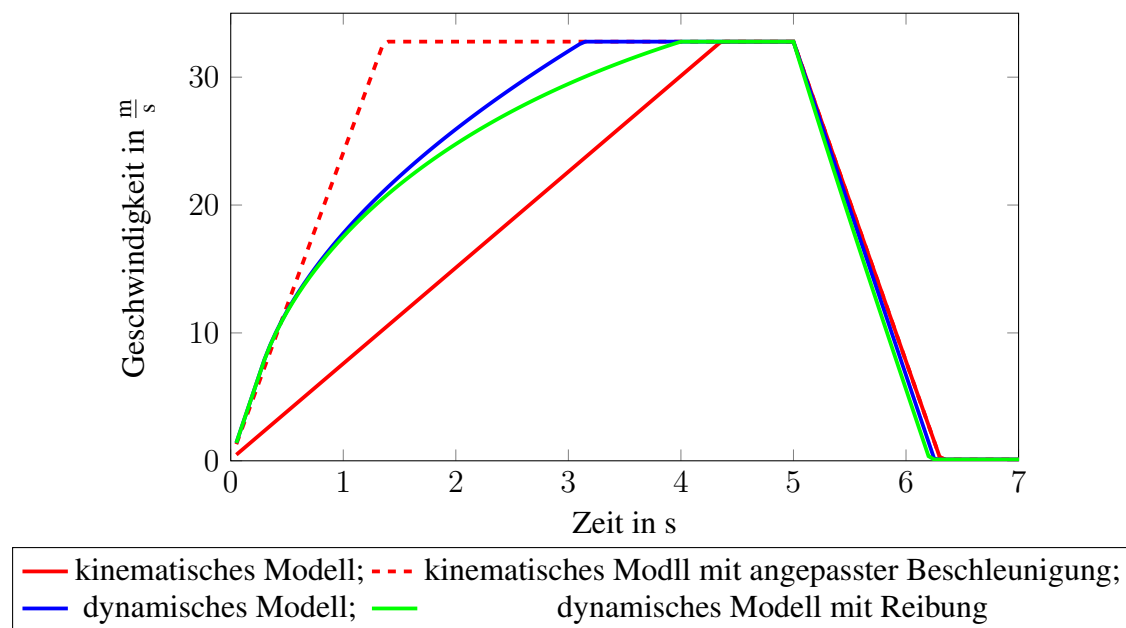
Ohne eine Verifizierung der Plausibilität der implementierten Modelle ist keine systematische Untersuchung möglich. Hierfür wurden die zwei Disziplinen acceleration und skidpad gewählt und die besten gefahrenen Ergebnisse des Rennautos des High-Octane Motorsports als Vergleichsbasis genutzt.

#### 6.1.1 Disziplin Acceleration

In dieser Disziplin ist es das Ziel der Teams eine 75 m lange, gerade Strecke in möglichst kurzer Zeit zurückzulegen. Es werden drei verschiedene longitudinale Konfigurationen des Fahrzeugmodells evaluiert. Zuerst das kinematische Modell in dem die maximale mittlere Beschleunigung (3.3.6) aus den Fahrzeugdaten errechnet wird. Das zweite untersuchte Modell basiert auf der Leistung des Motors und der maximal übertragbaren Kraft der Reifen (3.3.36). Das dritte Modell betrachtet zusätzlich zur Motorleistung die Rollreibung und den Luftwiderstand.

Zuerst werden die Beschleunigungskurven betrachtet. Hierfür wird mit allen hinterlegten Modellen fünf Sekunden lang maximal beschleunigt und direkt danach 2,5 s mit maximaler Kraft gebremst. In der Grafik 6.1 wird deutlich wie stark sich die einzelnen Modelle, obwohl sie sich auf das gleiche Fahrzeug beziehen, voneinander unterscheiden. Das Fahrzeug besitzt eine Maximalgeschwindigkeit von  $121 \frac{\text{km}}{\text{h}}$ , welche durch die Übersetzung im Getriebe limitiert wird. Das kinematische Modell trifft mit seiner durchschnittlichen Beschleunigung exakt die gefahrenen Testwerte. Da die dynamischen Modelle jedoch eine größere Anfangsbeschleunigung besitzen, wird die Anfangsbeschleunigung des im MPC-Algorithmus hinterlegt Modell auf die gleiche Steigung angepasst. Die Steigung beträgt  $24 \frac{\text{m}}{\text{s}^2}$ . Als nächstes wird die Zeit über eine Strecke von 75 m gemessen. Das Ergebnis für das kinematische Modell, mit angepasster Beschleunigungskurve, beträgt 3,0 Sekunden. Dieser Wert ist, wie bereits zu erwarten war, deutlich zu schnell. Realistische Werte liegen zwischen 3,8 und 4,5 Sekunden, je nach Fähigkeiten des Fahrers, Wetterbedingungen



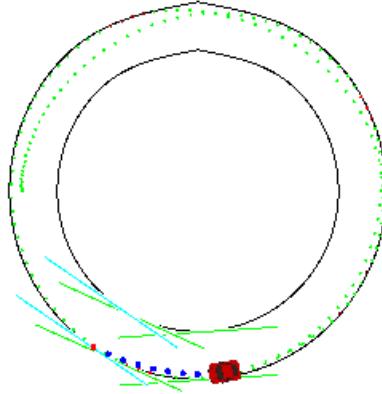


**Abbildung 6.1:** Geschwindigkeitsverlauf bei maximaler Beschleunigung für verschiedene longitudinale Modelle

und Reifentemperatur. Das dynamische Modell, das keinerlei Reibung berücksichtigt, ist ebenfalls mit 3,45 Sekunden zu schnell. Mit Reibung entsteht ein realistischer Wert von 4,3 Sekunden, was typischen Messzeiten für das Fahrzeug des High-Octane Motorsports e.V. entspricht. Für ein gutes longitudinales Modell des Fahrzeugs muss also der Luftwiderstand, der einen deutlich größeren Anteil auf die Gesamtreibung hat als der Rollwiderstand, berücksichtigt werden. Die Rollreibung ist bei maximaler Geschwindigkeit um den Faktor 100 kleiner als die aerodynamischen Kräfte und entspricht etwa 6 und 592 N respektive. Im Folgenden wird immer das um Reibung erweiterte longitudinalen Modell verwendet.

### 6.1.2 Disziplin Skidpad

In dieser Disziplin fährt das Fahrzeug eine liegende 8. Der limitierende Faktor ist also die im MPC-Algorithmus hinterlegte maximale Querbeschleunigung, bei der das Rennauto noch auf dem Kurs bleibt. In der Simulation wird vereinfacht von einem Rundkurs, welcher in Abbildung 6.2 dargestellt wird, mit dem Außendurchmesser von 21,25 m ausgegangen. Ob die Begrenzung der Querbeschleunigung (3.3.7) mit der Simulation übereinstimmt,



**Abbildung 6.2:** Rundkurs zum Messen der maximalen Querbeschleunigung

kann durch die wirkende Querbeschleunigung überprüft werden. Diese berechnet sich aus Bahngeschwindigkeit  $v$  und dem Radius der Kreisbahn  $r$ .

$$a = \frac{v^2}{r} \quad (6.1.1)$$

Die Bahngeschwindigkeit wird durch die Kreisfrequenz  $\omega$  berechnet.

$$\omega = 2\pi f \quad (6.1.2)$$

$$v = r \cdot \omega \quad (6.1.3)$$

Für das in der Simulationsumgebung hinterlegte kinematische Modell ist die gemessene Rundenzeit 4,65 s. Daraus lässt sich eine Durchschnittsgeschwindigkeit  $v = 13.9 \frac{\text{m}}{\text{s}}$  berechnen, was einer Querbeschleunigung von  $a = 19.37 \frac{\text{m}}{\text{s}^2}$  entspricht. Das im Fahrzeugmodell hinterlegte  $a_{max}$  ist mit  $19,62 \frac{\text{m}}{\text{s}^2}$  also sehr nah an dem in der Simulation erfahrenen Werten. Die Differenz zu der schnellsten gemessenen Runde mit dem echten Fahrzeug (4,77 s) lässt sich dadurch erklären, dass der MPC-Algorithmus die idealen Stellgrößen berechnet und damit auch einen idealen Rundkurs abfährt. Zudem wurde die maximal gemessene Querbeschleunigung des Fahrzeugs als  $a_{max}$  gewählt. Diese kann in der Realität nicht durchgängig gehalten werden. Der Wert wurde beim Einlenken mit hoher Geschwindigkeit ermittelt. In diesem Fall besitzt das Rennauto, aufgrund seiner Aerodynamik, einen hohen Abtrieb. Durch die höhere Normalkraft, die die Reifen auf die Straße presst, können

größere laterale Kräfte gewirkt werden. Für das dynamische Fahrzeugmodell wurde eine minimale Rundenzeit 3,35 s gemessen. Die damit erhaltene maximale Querbeschleunigung ist deutlich größer als die des kinematischen Modells. Die Differenz ist auf eine zu stark vereinfachte Annahme der Reifenkräfte zurückzuführen. Die von der M.F. errechneten lateralen Kräfte werden verdoppelt, da das Fahrzeug in der Realität jeweils zwei Räder pro Achse besitzt. In schnell gefahrenen Kurven verlagert sich jedoch die Gewichtskraft des Rennautos, die den Reifen auf die Straße presst, stark auf das kurvenäußere Rad. Die vom kurveninneren Rad maximal übertragbare Kraft bricht daher deutlich ein. Die maximal übertragbare Kraft des Reifens steigt zwar mit höherer Normalkraft an, jedoch ist der Zusammenhang nichtlinear und nähert sich immer mehr einem maximalen Wert an. Das kurvenäußere Rad kann daher den Verlust des inneren Reifens nicht kompensieren. Dieser Zusammenhang ist die Grundlage für das Verwenden eines Stabilisators im Fahrzeug. Er verringert den Effekt und führt zu einem besseren Fahrverhalten.

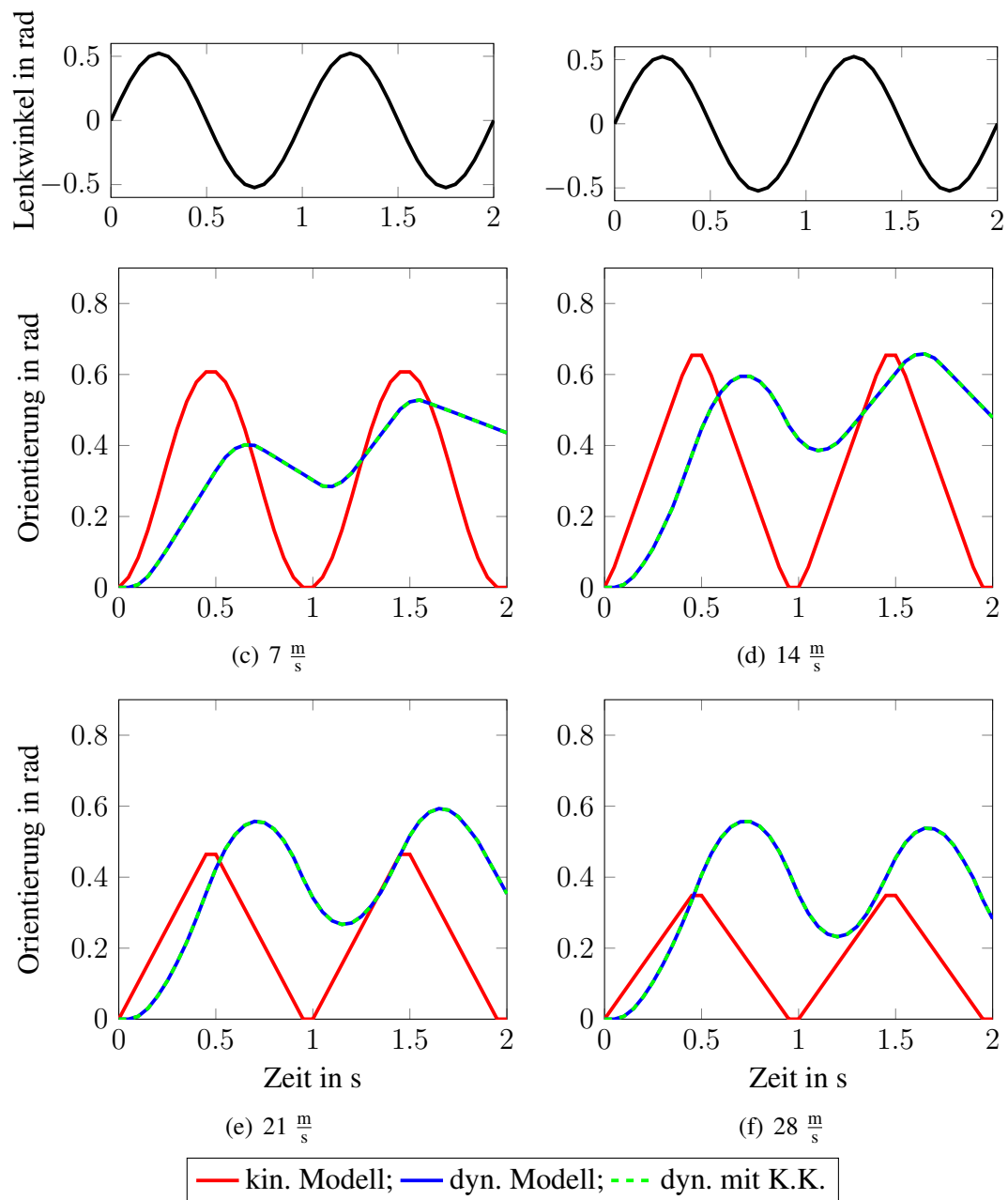
Da im Skidpad die Querbeschleunigung konstant ist, spielt das Trägheitsmoment in dieser Disziplin eine untergeordnete Rolle. Ein deutlich dynamischeres Szenario soll die Unterschiede zwischen kinematischem und dynamischen Szenario besser veranschaulichen.

### 6.1.3 Modellabhängige Veränderung der Orientierung

In der Darstellung 6.3 ist die Änderung der Orientierung für einen oszillierenden Lenkwinkel veranschaulicht. Es werden die vier verschiedenen Geschwindigkeiten 7, 14, 21 und  $28 \frac{\text{m}}{\text{s}}$  ohne jegliche longitudinale Beschleunigung getestet. Dies soll einen guten Überblick über den gesamten Geschwindigkeitsbereich ermöglichen.

Zuerst wird das Verhalten des kinematischen Modells betrachtet. Bei der geringsten Geschwindigkeit von  $7 \frac{\text{m}}{\text{s}}$  folgt die Orientierung noch genau dem Lenkwinkel nach und erreicht damit den Maximalwert im Nulldurchgang der Sinusfunktion des Lenkwinkels. Mit höheren Geschwindigkeiten begrenzt jedoch immer stärker die Beschränkung der maximalen Querbeschleunigung. Bei  $14 \frac{\text{m}}{\text{s}}$  ist dies deutlich durch den leicht abgerundeten Anfangspunkt der Kurve erkennbar. Dies ist bei 21 und  $28 \frac{\text{m}}{\text{s}}$  nicht mehr der Fall. Ab  $14 \frac{\text{m}}{\text{s}}$  begrenzt die maximale Querbeschleunigung auch immer stärker die maximal erreichte Orientierung.

Das Verhalten des dynamischen Modells und das des um den K.K. erweiterten sind in diesem Anwendungsfall genau gleich. Dies ist darauf zurückzuführen, dass keinerlei



**Abbildung 6.3:** Orientierung des Rennautos für eine Sinuskurve als Eingangswert des Lenkwinkels. Es werden verschiedene Geschwindigkeiten ohne longitudinale Beschleunigung betrachtet.

longitudinale Kraft an den Reifen anliegt. Die vom Reifen zu übertragenden Kräfte bewegen sich daher nur auf der x-Achse des K.K. und erfahren keine Einschränkung. Das Verhalten der dynamischen Modelle bei den unterschiedlichen Geschwindigkeiten ist nicht direkt aus den Gleichungen nachvollziehbar. Dem Verlauf der Diagramme kann entnommen werden, dass die maximale Orientierung deutlich dem kinematischen Modell und damit dem Lenkwinkel naheilt. Dieser Umstand ist auf das Trägheitsmoment zurückzuführen.

Wird nun das gleiche Szenario aber für eine zusätzliche longitudinale Beschleunigung betrachtet, ergibt sich ein gänzlich neues Bild (siehe 6.4). Zuerst lohnt sich nochmals der Blick auf das kinematische Modell. Hier ist nun sehr gut zu sehen, wie mit zunehmender Geschwindigkeit die Kurve immer weniger abgerundet ist und durch die maximale Querbeschleunigung begrenzt wird. Das Verhalten der beiden dynamischen Modelle ist in diesem Szenario nicht mehr kongruent. Für  $7 \frac{\text{m}}{\text{s}}$  ist die große Differenz klar durch die eingeschränkten lateralen Kräfte durch den K.K. ersichtlich. Dass die Differenz bei dem letzten Diagramm mit  $28 \frac{\text{m}}{\text{s}}$  nahezu verschwindet, ist auf  $F_x$  zurück zu führen. Dieses wird mit der Gleichung (3.3.36)

$$F_{lacc} = \frac{P_{engine}}{|\dot{x}|}$$

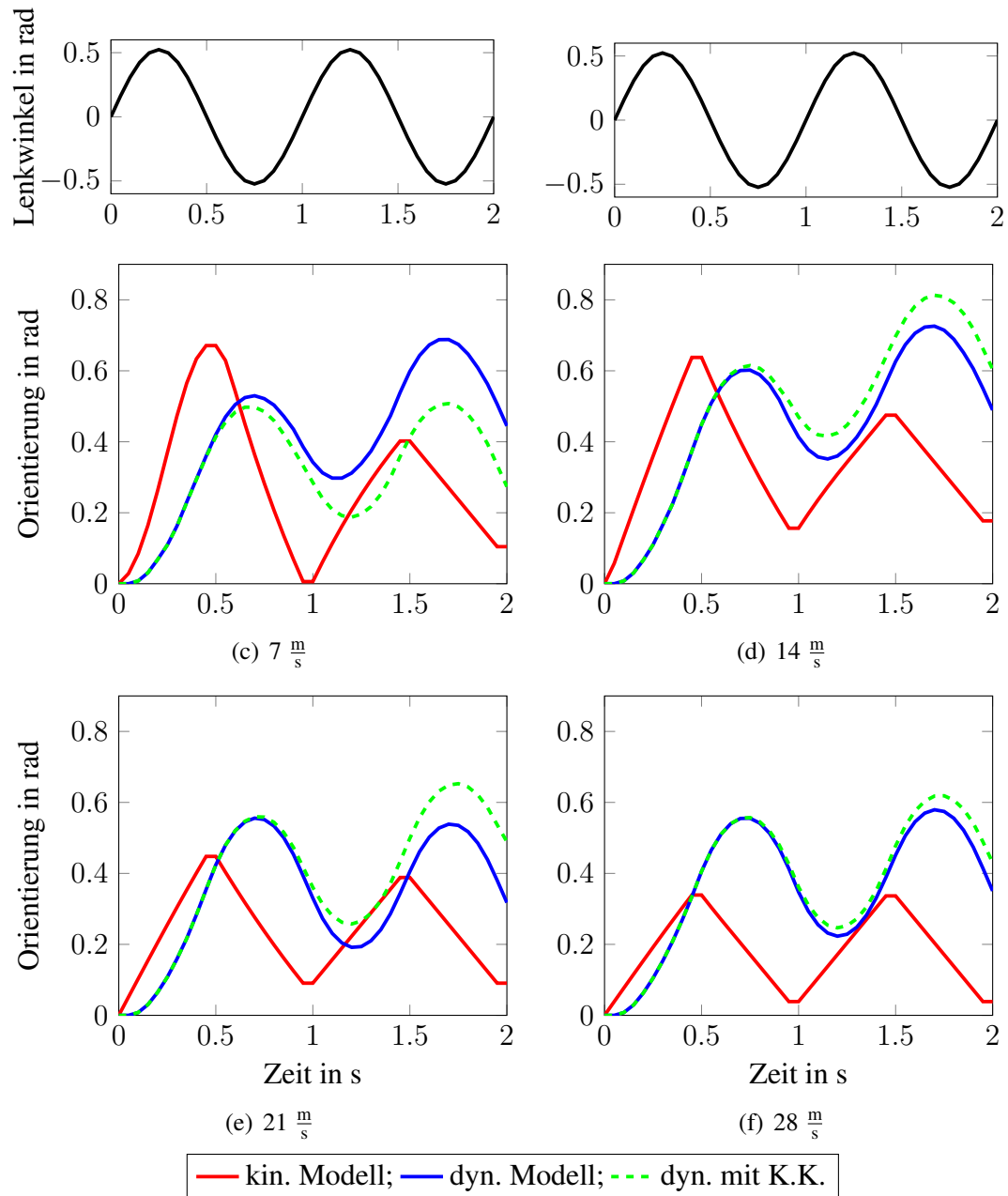
berechnet und mit zunehmender Geschwindigkeit immer kleiner.

#### 6.1.4 Reifenmodelle

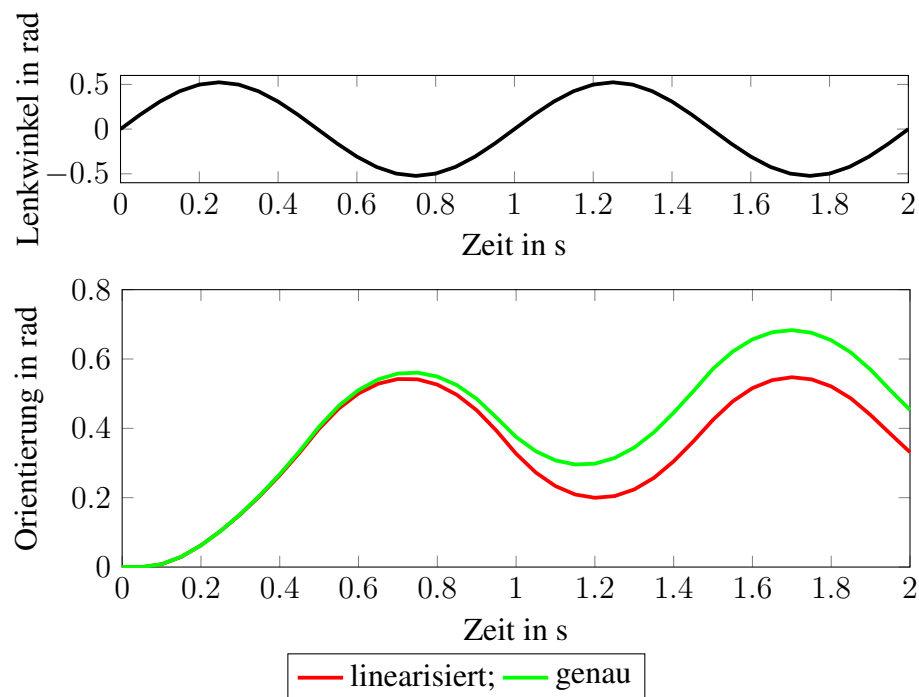
Wie bereits im Kapitel 3.3.2, wird für die Simulation standardmäßig die M.F. genutzt. Im Folgenden soll nun kurz darauf eingegangen werden, was für eine Auswirkung das linearisierte Modelle auf das Fahrzeugverhalten hat. Auch für diese Untersuchung wird eine Sinuswelle als Eingabe für den Lenkwinkel des Modells gewählt und die verschiedenen resultierenden Orientierungen des Fahrzeugs betrachtet (siehe Abbildung 6.5).

An der Orientierung lässt sich gut erkennen, dass die Modellierung der Reifen einen starken Einfluss auf das Verhalten des Fahrzeugs besitzt. Der Unterschied zwischen den Modellen beträgt im Maximum etwa  $0,2 \text{ rad}$ . Ob die Differenz auch einen Einfluss auf die Regelung des dynamischen Modells hat, wird im Abschnitt 6.3 näher beleuchtet.

Ist das Einsparen von Rechenzeit keine Priorität, sollte generell immer das beste Reifenmodell verwendet werden. Wie schon im Kapitel 3.3.2 angedeutet, hängt beim dynamischen Modell das Verhalten sehr stark von den Reifenkräften ab und sollte daher möglichst genau abgebildet werden. Soll trotzdem viel Rechenleistung gespart werden, entspricht das lineare Modell einer guten Alternative zur M.F.. Es liefert eine gute Annäherung und



**Abbildung 6.4:** Die gleiche Betrachtung wie in 6.3 mit maximaler longitudinaler Beschleunigung.



**Abbildung 6.5:** Fahrzeugorientierung, abhängig von den drei verschiedenen Reifenmodellen für einen sinusförmigen Lenkwinkel als Eingabe

ist zudem noch weniger komplex. Dies gilt jedoch nur für den Fall von Rennreifen, da sie über einen großen Bereich des Schräglaufwinkels hohe Seitenkräfte übertragen können. Normale Straßenreifen brechen deutlich stärker nach dem Erreichen ihrer maximalen Seitenkraft ein und werden daher nicht gut durch das linearisierte Modell repräsentiert.

## 6.2 Regelung des kinematischen Modells

In diesem Kapitel wird die Regelung des im Simulator hinterlegten kinematischen Modells mit Hilfe des MPC-Algorithmus untersucht. Zuerst wird evaluiert, was für Auswirkungen die einzelnen Kostenfunktionen auf die Rundenzeit besitzen und wie die Länge des Prädiktionshorizontes diese beeinflusst. Um dies tun zu können, muss zuerst die mindestens erforderliche Länge des Prädiktionshorizontes gefunden werden.

### 6.2.1 Mindestlänge des Prädiktionshorizontes

In einem Szenario im öffentlichen Verkehr muss der Horizont immer mindestens so lang gewählt werden, dass eine Vollbremsung von der aktuellen Geschwindigkeit durchgeführt werden könnte, bevor das Fahrzeug das Ende des Horizontes erreicht hat. Auf dem Rennkurs ist dies nicht nötig. Es ist nicht davon auszugehen, dass sich plötzlich Hindernisse auf der Strecke befinden (bei Formula Student Driverless befindet sich immer nur ein Fahrzeug auf der Strecke). Da im Simulator und im MPC-Algorithmus das gleiche Fahrzeugmodell hinterlegt ist, wird die tangentiale Beschränkung genutzt, um das Rennauto daran zu hindern die Strecke zu verlassen. Zuerst wird der minimale Prädiktionshorizont für einen sichere Runde auf dem Kurs gesucht. Um diesen Wert zu finden, wurde eine Teilstrecke definiert, die aus einer 80 Meter langen Gerade direkt in zwei aneinander gehängte 180° Kurven mit 9 m Außenradius übergeht. Diese Konstellation, entspricht der Fahrsituation die nach Regelwerk zulässig ist und das Fahrzeug sowohl longitudinal (bremsen) als auch lateral (enge Kurve) maximal beansprucht. Die Tests haben ergeben, dass bereits ein Horizont von 12 Schritten für die Kostenfunktionen, welche die Distanz max/-minimieren, ausreichend ist. Die Maximalgeschwindigkeit beträgt genau die vom Fahrzeugmodell begrenzten  $121 \frac{\text{km}}{\text{h}}$  auf der Geraden und eine maximalen Kurvengeschwindigkeit von  $32 \frac{\text{km}}{\text{h}}$ . Für die max-speed-Kostenfunktion wird ein anderer Horizont benötigt, als für die beiden anderen Kostenfunktionen. Der Algorithmus lenkt in Kurven nicht direkt so ein, dass das Fahrzeug am inneren der Kurve entlangfährt. Stattdessen fährt er möglichst lange geradeaus, um die maximale Geschwindigkeit zu erreichen. Der minimale Horizont ist daher im Vergleich zu den beiden anderen Funktionen 8 Zeitschritte länger und damit mindestens 20 Prädiktionsschritte lang.



## 6.2.2 Kostenfunktionen

Mit der Information wie groß der kleinste Prädiktionshorizont mindestens sein muss, wird nun untersucht, wie die unterschiedlichen Kostenfunktionen gegeneinander abschneiden. Um eine gute Vergleichbarkeit zu erhalten, wird ein zufälliger Rennkurs erstellt, auf dem die Messungen durchgeführt werden.

### 6.2.2.1 Verifizierung der Funktionen

Bevor die Messungen für die einzelnen Kostenfunktionen erstellt werden, wird sichergestellt, dass die Funktionen auch fehlerfrei funktionieren. Auf einer im Simulator definierten 75 m Geraden wird überprüft, ob die Zeit zum Abfahren dieser Strecke für alle Funktionen gleich ist. Abweichungen könnten zum Beispiel entstehen, wenn die Zielposition für die min-dist-Kostenfunktion nicht groß genug gewählt ist und beim Beschleunigen die Distanz zu null wird. In diesem Fall würde das Fahrzeug für die 75 m mehr Zeit benötigen als die max-speed-Kostenfunktion. Ebenfalls wurde untersucht, ob das Rennauto für eine der Kostenfunktionen einen Slalom fährt um die Geschwindigkeit oder den zurückgelegten Weg zu maximieren. Die Funktionen haben alle wie erwartet maximal beschleunigt und den 75 m Kurs in der gleichen Zeit absolviert. Die Werte sind in der Tabelle 6.1 aufgelistet.

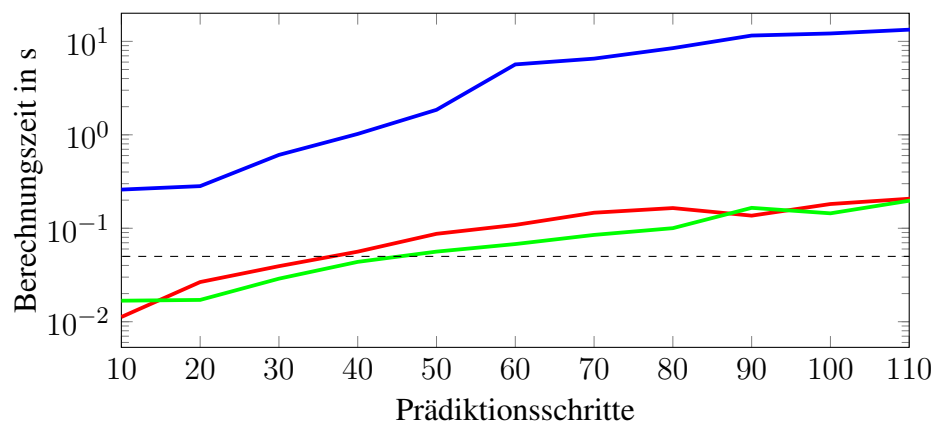
Kostenfunktion	Zeit
max-speed	3,05 s
min-dist	3,05 s
max-track	3,05 s

**Tabelle 6.1:** Beschleunigungszeit für unterschiedliche Kostenfunktionen

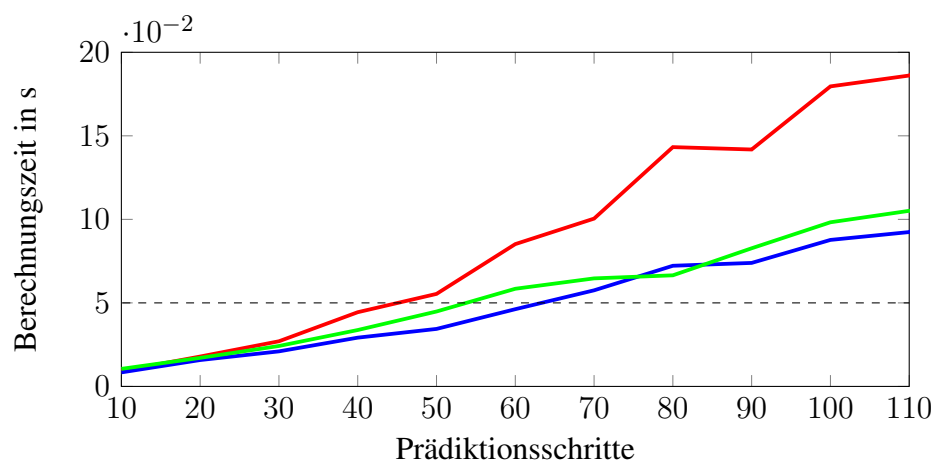
### 6.2.2.2 Berechnungszeit

Mit der Gewissheit, dass alle Kostenfunktionen die gewünschten Ergebnisse erzeugen, wird nun die Dauer untersucht, die benötigt wird das MPC-Problem zu berechnen und wie stark es von der Länge des Prädiktionshorizontes abhängt. Die Ergebnisse der Messung sind in der Abbildung 6.6 dargestellt.

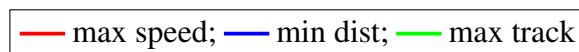
Bei einigen Kostenfunktionen dauert die Berechnung des initialen Optimierungsvektors, im Vergleich zu allen darauf folgenden Schritten, extrem lange. Dies liegt an dem sogenannten



(a) Mittelwert der ersten 40 Optimierungen



(b) Mittelwert sobald Warmstart aktiv

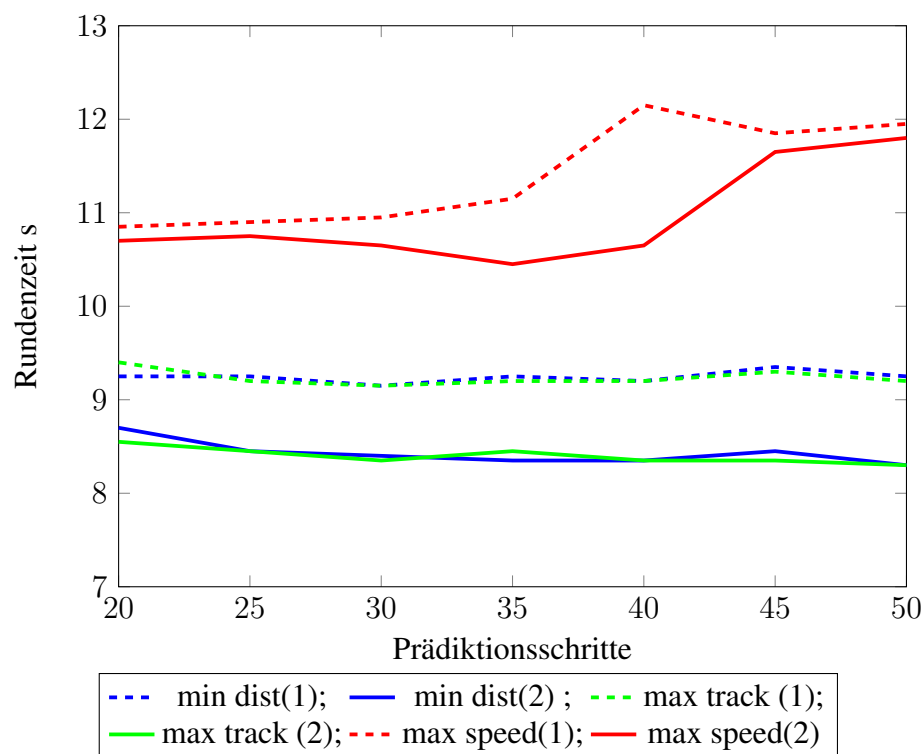

**Abbildung 6.6:** Berechnungszeit der Kostenfunktionen für verschieden große Horizontlängen

Warmstart. Solange die Parameter der Beschränkungen (neue Tangenten und neue Startposition) nicht signifikant verändert werden, findet der Optimierer in aufeinander folgenden Schritten sehr schnell eine passende Lösung. Dies ist mit einem Schieberegister vergleichbar, in dem die Lösung einen Schritt nach links verschoben wird und nur für den letzten Schritt eine neue Lösung gesucht werden muss. Aus diesem Grund wurden auch Messungen erstellt, die erst anfangen, sobald der Warmstart durchgeführt wird. Trotz der mathematischen Ähnlichkeit der beiden Kostenfunktionen max-track und min-dist ist die Ausführungszeit der Optimierungsschritte ohne Warmstart sehr unterschiedlich. Die max-track Funktion ist deutlich besser und liegt ohne Warmstart auf einem Niveau mit der max-speed Funktion.

Sobald der Warmstart aktiv ist, dauert die Berechnungszeit für min-dist und max-track ungefähr gleich lange und skaliert besser als die max-speed Kostenfunktion für große Horizontlängen.

### 6.2.2.3 Rundenzeit

Um die Qualität der Kostenfunktionen einschätzen zu können, wird im Folgenden die Rundenzeit betrachtet. Die Zeit wird abhängig von den Prädiktionsschritten untersucht. Gemessen werden immer die ersten zwei Runden, da in der ersten Runde die Anfangsbeschleunigung und die eventuell ungünstige Startposition die Rundenzeit beeinträchtigen. Es wurde auch jeweils noch eine dritte Rundenzeit ermittelt, diese entspricht aber nahezu genau der vorhergehenden. Die Ergebnisse sind in der Grafik 6.7 visualisiert.



**Abbildung 6.7:** Rundenzeit für verschiedene Kostenfunktionen und abhängig von der Anzahl der Prädiktionsschritte. Erste (1) und zweite (2) Runde

Die max-speed-Kostenfunktion wird für größere Vorwärtsprädiktion immer langsamer. Dieses Phänomen lässt sich mit dem größeren Wissen über den Kursverlauf erklären.

Desto weiter in die Zukunft der Optimierer schauen kann, desto mehr fängt er an Kurven voll auszufahren, also den zurückgelegten Weg länger zu planen. Dadurch kann zwar die Geschwindigkeit erhöht werden, die Rundenzeit leidet allerdings deutlich darunter. Die Differenz zwischen den beiden Kostenfunktionen, welche die Distanz zu einem Punkt minimiert beziehungsweise die gefahrene Streckendistanz maximiert, ist sehr klein und auch bei geringeren Horizontlängen bereits sehr gut. Wird zusätzlich die Berechnungszeit betrachtet, setzt sich die max-track Kostenfunktion deutlich als die bessere ab.

Nachdem die Grundlagen für die Kostenfunktionen und Vergleichswerte bezüglich der Rundenzeit erstellt wurden, wird im folgenden Abschnitt die Regelfähigkeit der komplexeren Modelle im Simulator überprüft.

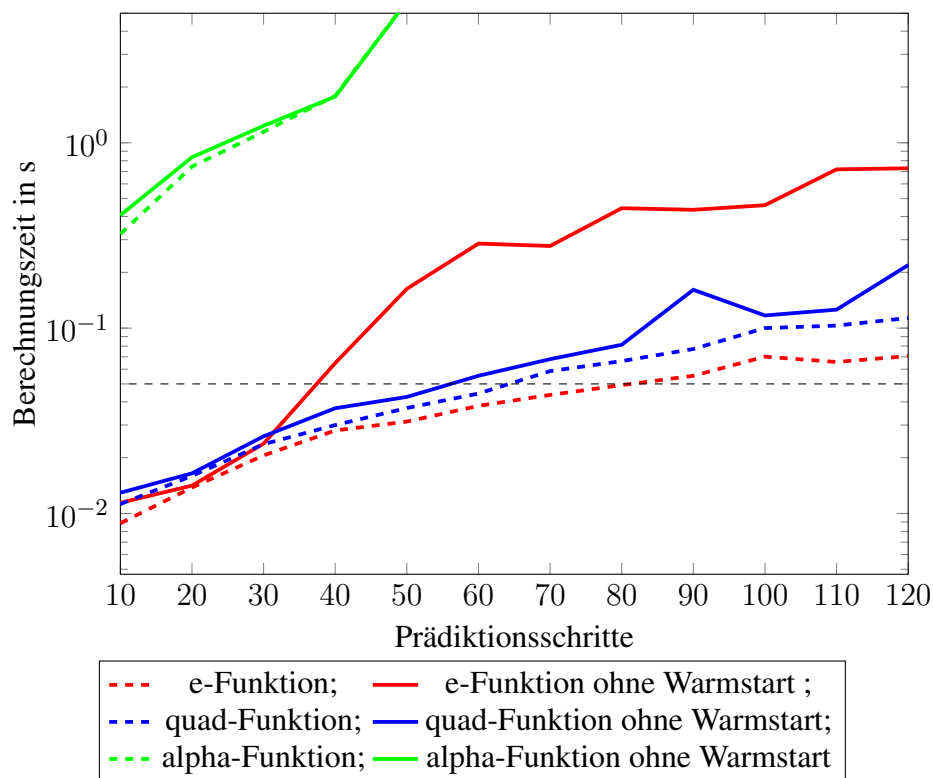
## **6.3 Regelung des dynamischen Modells**

In der Realität ist das kinematische Modell nur für geringe Geschwindigkeiten genau genug, um die Bewegung des Fahrzeugs zu modellieren. Im Folgenden wird deswegen das Modell im Simulator durch das dynamische Fahrzeugmodell mit berechneten Reifenkräften ersetzt. Das Ziel ist es zu evaluieren, wie gut sich das deutlich komplexere System mit dem weiterhin im MPC-Algorithmus hinterlegten, sehr einfachen kinematischen Modell, regeln lässt. Die deutlichen Unterschiede der Modelle wurden bereits im Kapitel 6.1 evaluiert. Für diesen Abschnitt wird vom, um Reibung erweiterten, dynamischen Fahrzeugmodell ausgegangen.

### **6.3.1 Berechnungszeit der Elastische Kostenfunktion**

Die max-track-Kostenfunktion hat sich im letzten Abschnitt, bezogen auf Berechnungszeit und Rundenzeit, als die Beste herausgestellt. Daher wird sie als Grundlage für die elastische Kostenfunktion genutzt und zusätzlich noch durch die Kosten für die Position auf der Spur erweitert. Im Folgenden wird nun zuerst die Berechnungszeit untersucht. Die Messungen werden mit dem kinematischen Modell im Simulator vorgenommen, um eine möglichst gute Vergleichbarkeit mit den Ergebnissen aus 6.2.2.2 zu erhalten. Die Ergebnisse der Messung sind in der Abbildung 6.8 dargestellt.

Die Auswahl der in Frage kommenden Kostenfunktionen ist schnell getroffen. Die e-Funktion und die quadratische-Funktion haben eine deutlich bessere Laufzeit und lassen sogar die einfacheren Kostenfunktionen aus der vorherigen Messung hinter sich. Dies ist



**Abbildung 6.8:** Berechnungszeit für verschiedene Kostenfunktionen mit elastischer Beschränkung.

darauf zurückzuführen, dass keine tangentialen Beschränkungen mehr vorhanden sind und damit das MPC-Problem eine geringere Komplexität aufweist.

Die deutlich höhere Ausführungszeit bei der alpha-Funktion liegt an deren Form. Jeweils auf der Rückseite der Erhöhung fallen die Kosten wieder gegen null. Der Optimierer muss also einen sehr großen Suchraum betrachten. Es werden auch mögliche Trajektorien evaluiert, die die hohen Kosten für zwei bis drei Schritte in Kauf nehmen und anschließend das Rennauto außerhalb des Kurses steuern. Diese Situation kann nachgestellt werden, indem das Kostenverhältnis zu Gunsten der elastischen Beschränkung angepasst wird, woraufhin das Fahrzeug den Kurs verlässt.

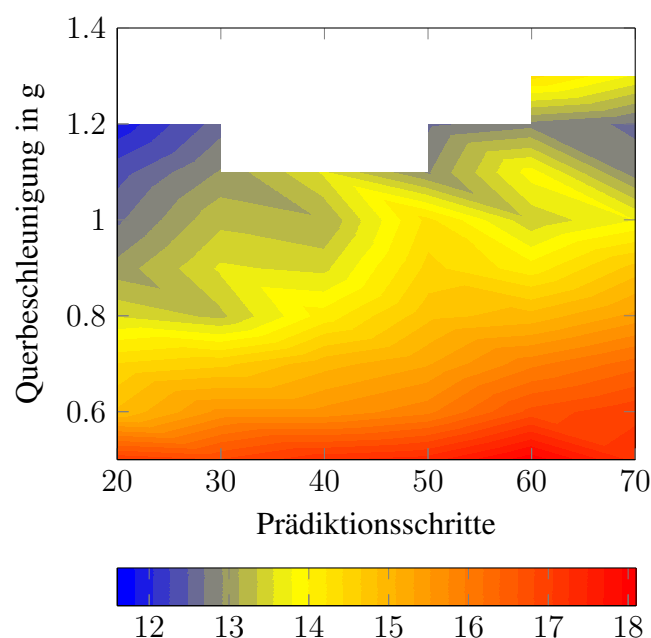
### 6.3.2 Maximale Querbeschleunigung und Prädiktionshorizont

Zuerst wird ermittelt, bis zu welcher maximalen Querbeschleunigung das Rennauto stabil im Rennkurs geregelt werden kann. Um diesen Wert zu finden, wurde die elastische

Beschränkung basierend auf der e-Funktion gewählt. Ein sehr großer Kostenfaktor wurde eingestellt, um sicherzustellen, dass das Fahrzeug den Kurs nicht verlässt um abzukürzen. Erst wenn der Regler die Kontrolle verliert und das Fahrzeug nicht mehr auf dem Rennkurs halten kann, ist die maximal regelbare Querbeschleunigung erreicht.

### 6.3.2.1 Dynamisches Modell

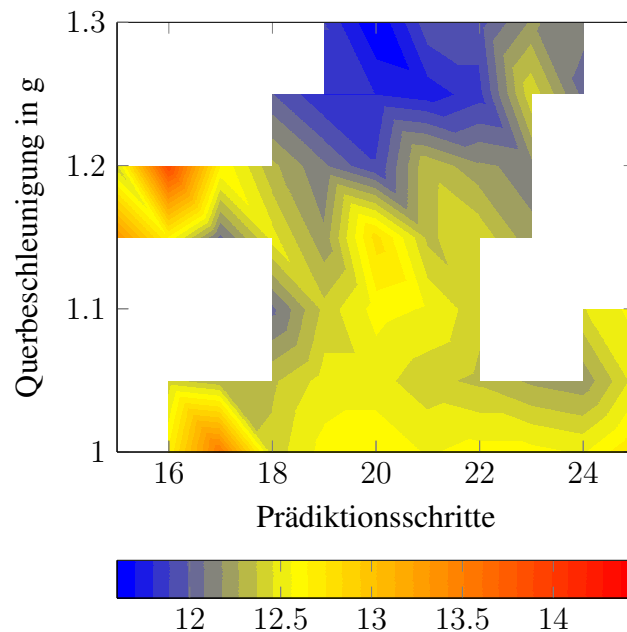
Für dieses Anwendungsszenario wird zuerst das dynamische Modell evaluiert. Daraufhin werden dann die Messungen ebenfalls für das Modell, das durch den K.K. erweitert wurde, erstellt. Der in Abbildung 6.9 dargestellte Graph zeigt den Zusammenhang aus maximaler Querbeschleunigung und Prädiktionsschritten an.



**Abbildung 6.9:** Die Rundenzeit in Sekunden abhängig von Querbeschleunigung und Prädiktionsschritten. Weiße Bereiche entsprechen dem Abkommen des Fahrzeug von der Strecke

Die beste erreichte Zeit mit 11,6 s ist entgegen der Erwartung für einen kurzen Horizont von 20 Schritten erreicht worden. Dies hängt vermutlich damit zusammen, dass bei einem großen Prädiktionshorizont die Bahn des Rennautos sehr nah an den Streckenbegrenzungen vorbeiführt. Dadurch sind höhere Geschwindigkeiten in engen Kurven möglich. Das dynamische Modell schießt jedoch deutlich öfter über die geplante Trajektorie hinaus als

bei einem kurzen Horizont. Die ideale Querbeschleunigung für den MPC-Regler liegt bei 1,3 g. Die Rundenzeiten wurden jeweils für die zweite Runde gemessen. Nachdem die erste Messung nur recht grob durchgeführt wurde um einen großen Bereich abzudecken, ist in der Grafik 6.10 der Bereich von 15-25 Prädiktionsschritten genauer untersucht worden.

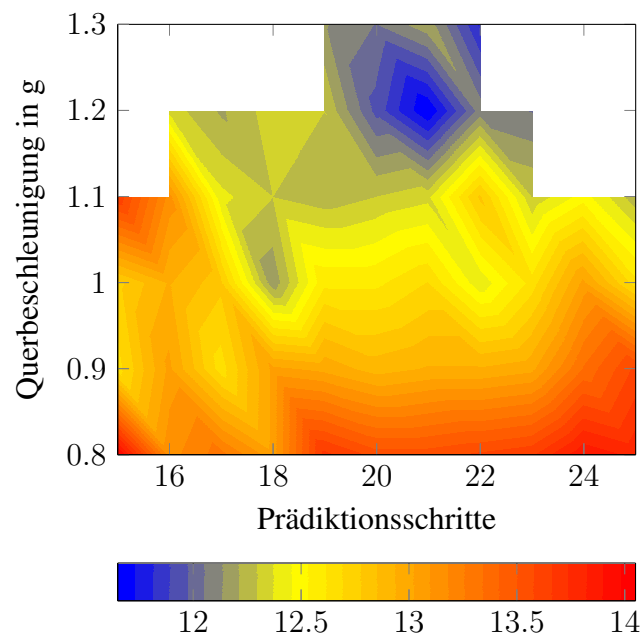


**Abbildung 6.10:** Die Rundenzeit in Sekunden abhängig von Querbeschleunigung und Prädiktionsschritten

Dadurch, dass die Geschwindigkeit nicht begrenzt ist, kann das Fahrzeug bei zu geringen wie auch bei zu großen Querbeschleunigungen nicht auf dem Kurs gehalten werden. Erst bei einem größeren Prädiktionshorizont bremst der MPC-Algorithmus früh genug ab, um bei geringen Querbeschleunigungen die Kurvenradien durchfahren zu können. Der Einfluss der maximalen Geschwindigkeit wird daher im Abschnitt näher betrachtet.

### 6.3.2.2 Kammscher Kreis

Die gleichen Messungen wurden nun auch für das dynamische Modell mit hinterlegtem K.K. durchgeführt. Die beste erreichte Rundenzeit (siehe 6.11) liegt bei 11,65 Sekunden. Dieser Wert wurde für einen Prädiktionshorizont von 21 Schritten bei der maximalen Querbeschleunigung von 1,2 g erreicht. Dieses Ergebnis überrascht im Hinblick auf die deutlichen Unterschiede der Modelle. Das dynamische Modell, das in Situationen, in denen



**Abbildung 6.11:** Die Rundenzeit in Sekunden abhängig von Querbeschleunigung und Prädiktionsschritten für das dynamische Modell erweitert durch den K.K.

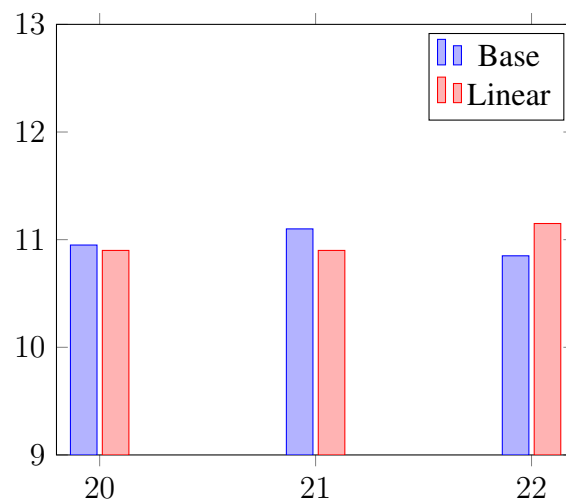
stark beschleunigt und gebremst wird deutlich höhere Kräfte übertragen kann, kann sich in der schnellsten Runde nur um einen Zeitschritt mit 0,05 s absetzen. Dieses Ergebnis entsteht dadurch, dass durch das hinterlegte kinematische Modell und die Beschränkung der Querbeschleunigung der Regler sehr früh das Bremsen anfängt, um in der Kurve möglichst nah an den Apex zu kommen. Die longitudinalen Kräfte sind daher bereits klein wenn die maximalen lateralen Kräfte auftreten. Die Einschränkung durch den K.K. tritt daher auch nur geringfügig auf.

## 6.4 Einfluss des Reifenmodells

Der Einfluss des Reifenmodells soll auch kurz untersucht werden. Hierfür wird die Grenzsituation aus dem vorherigen Abschnitt gewählt. Für die maximale Querbeschleunigung wurde 1,2 g gewählt und verschiedenen Prädiktionszeiten evaluiert.

Das Ergebnis der Messung sind in Abbildung 6.12 dargestellt und entsprechen den Erwartungen. Die Unterschiede der Kurven (siehe 3.10) beider Gleichungen zum Berechnen der Reifenkräfte sind nur gering. Die Differenz bei den Rundenzeiten fällt daher auch klein





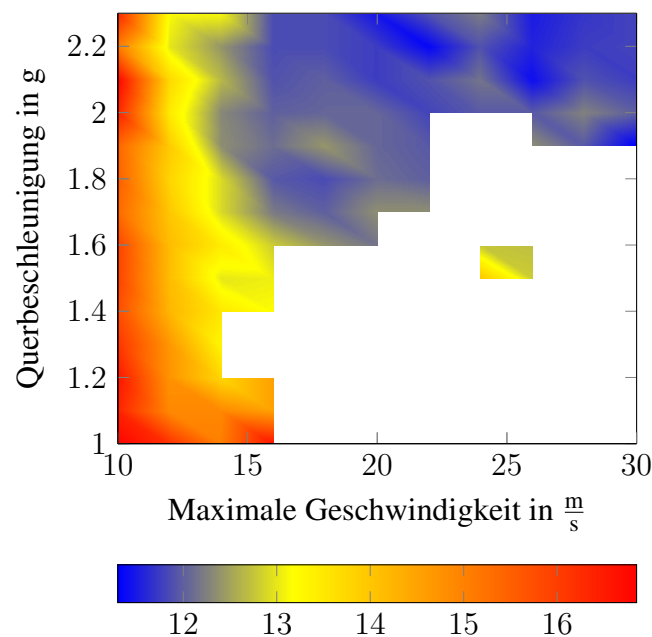
**Abbildung 6.12:** Rundenzeit abhängig von den Reifenmodellen für 1,2 g und verschieden lange Prädiktionshorizonte

aus. Wie bereits erwähnt, muss die Vereinfachung abhängig von den Reifen des Fahrzeugs gemacht werden. Generell ist dazu zu raten, immer das genaueste Modell zu verwenden.

### 6.4.1 Maximale Geschwindigkeit

In diesem Abschnitt soll untersucht werden, welchen Einfluss die Beschränkung der maximalen Geschwindigkeit auf die maximal mögliche Querbeschleunigung und die Rundenzeit besitzt. Gerade für kurze Prädiktionshorizonte macht die Begrenzung einen signifikanten Unterschied. Wenn das Rennauto auf einer Geraden viel Geschwindigkeit aufnimmt und dann in eine Kurve einfährt, kann für geringe Querbeschleunigungen keine Trajektorie geplant werden. Für zu große Querbeschleunigungen wird das Rennauto instabil. Es wird daher untersucht, wie das Rennauto sich bei einem Prädiktionshorizont von 15 Schritten für verschiedene Querbeschleunigungen und maximale Geschwindigkeiten verhält. Die Messergebnisse sind in Abbildung 6.13 dargestellt.

Sobald die maximale Geschwindigkeit geringer als  $15 \frac{\text{m}}{\text{s}}$  ist, kann das Rennauto auch für höhere Querbeschleunigungen sicher um den Kurs gefahren werden. Die minimale Rundenzeit in diesem Punkt beträgt 12.7 s bei 1.6 g. Das das Fahrzeug für einige Parameterkonfigurationen auch bei höherer Geschwindigkeit den Kurs absolvieren kann, liegt an dem Layout der im Simulator verwendeten Rennstrecke. Manchmal rutscht das Rennauto für höhere Geschwindigkeiten genau so um eine Kurve dass es die sofort darauf Folgende Kurve noch

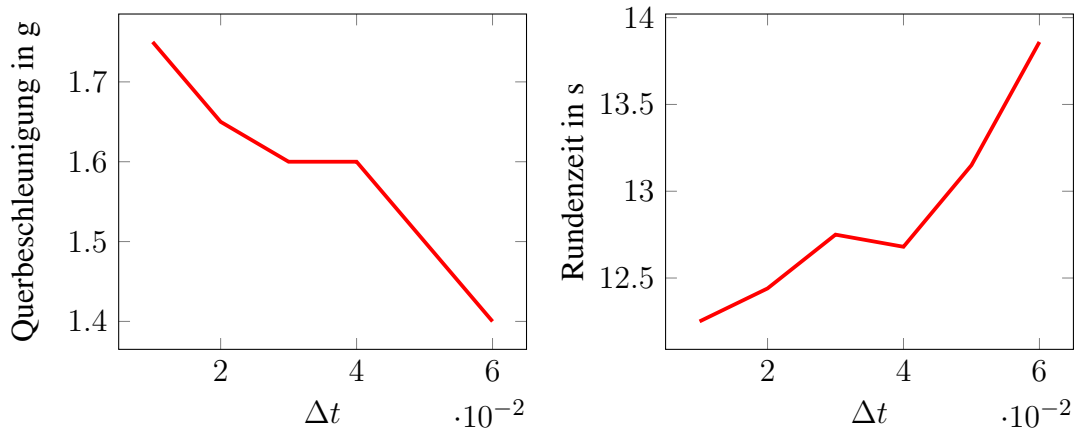


**Abbildung 6.13:** Rundenzeit, abhängig von der maximalen Geschwindigkeit und der Querbeschleunigung für einen 15 Schritte langen Horizont

durchfahren kann. Eine Veränderung des Rennkurses würde also ein zum Teil verändertes Bild der Ergebnisse mit sich bringen. Der Trend ist dennoch ersichtlich.

### 6.4.2 Regelfrequenz

Abschließend soll noch eine Betrachtung durchgeführt werden, ob eine höhere Aktualisierungsrate die Stabilität des Systems erhöht. Um dies zu überprüfen wird eine Parameterauswahl getroffen, in der das Rennauto beginnt auszuberechnen. Bei Abweichungen des Fahrzeugs von der erwarteten Trajektorie wird mit einer höheren Aktualisierungsrate weniger Zeit bis zu einer neu geplanten Trajektorie vergehen. Um eine gute Vergleichbarkeit zu erzielen, wird für jedes  $\Delta t$  die Anzahl der Prädiktionsschritte angepasst, alle Messungen beziehen sich also auf einen gleich langen vorausplanten Horizont. Wenn der Zeitwert welcher Vorausprädiziert werden soll, keinen ganzen Teiler für das betrachtete  $\Delta t$  besaß, wurde der nächst mögliche Wert gewählt. Die Messungen sind in der Abbildung 6.14 zusammengefasst. Je kleiner das  $\Delta t$  gewählt wurde, desto höhere Querbeschleunigungen konnte der Regler noch kontrollieren, bevor das Rennauto den Kurs verlässt. Die höhere Querbeschleunigung korreliert jedoch nicht mit einer besseren Rundenzeit. Das Rennauto



(a) Maximal mögliche Querbeschleunigung bei der das Fahrzeug noch den Kurs komplett abfahren kann (b) Beste Rundenzeit die gemessen wurde ()

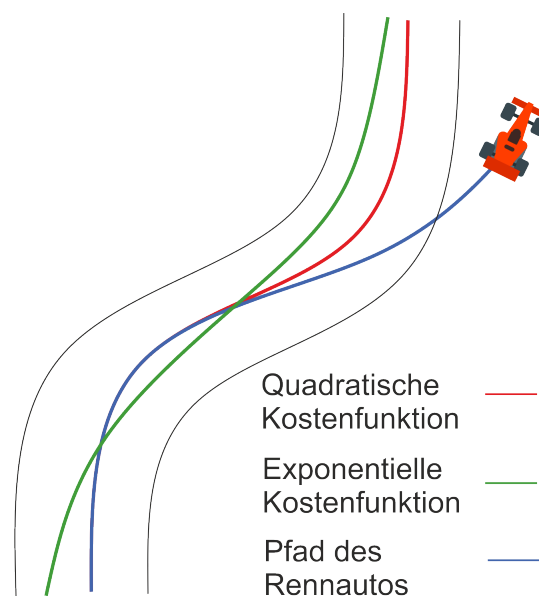
**Abbildung 6.14:** Abhängigkeit der Fahrzeugregelung von der Frequenz mit der das MPC-Verfahren ausgeführt wird

verlässt bei Querbeschleunigungen ab 1,5 g in jeder Kurve die Ideallinie. Die beste Rundenzeit wurde daher für alle Aktualisierungsraten bei etwa 1,4 g gemessen. Die hohe Aktualisierungsrate für ein  $\Delta t = \frac{1}{100}$  s erzeugt die beste Rundenzeit. Die Berechnungsdauer ist jedoch auch signifikant größer und deutlich langsamer als für Echtzeit nötig wäre. Betrachtet man das im vorherigen Abschnitt gemessene Verhalten der Rundenzeit für verschieden lange Prädiktionshorizonte, sollte der Fokus also auf eine höhere Aktualisierungsrate der Positionsschätzung und des MPC-Algorithmus gelegt werden. Die verwendete Aktualisierungsrate von 20 Hz ( $\Delta t = 0,05$ ) stellt einen guten Mittelweg aus Berechnungszeit und Rundenzeit dar. Ab einem  $\Delta t$  von 0,07 s war es ohne eine Einschränkung der Maximalgeschwindigkeit nicht mehr möglich das Fahrzeug zu kontrollieren. Unabhängig von der gewählten Querbeschleunigung.

### 6.4.3 Quadratische Kostenfunktion

Die Idee hinter der quadratischen Funktion ist, dass das Rennauto etwas mehr in der Mitte der Spur bleibt und daher im besten Fall auch bei höheren Geschwindigkeiten noch kontrollierbarer bleibt. Da es jedoch keinen abrupten Kostenanstieg an den Kursgrenzen gibt, muss evaluiert werden, für welchen Faktor  $\alpha$  und das Verhältnis  $a$  der Regler die besten Rundenzeiten liefert, ohne den Kurs zu verlassen. Für  $N$  wurde ein konstanter Wert von 25 Schritten mit einem  $\Delta t$  von 0,05 s gewählt. In den Messungen hat sich der

Ansatz nicht bestätigen lassen. Die Trajektorien werden immer möglichst in der Mitte des Kurses angelegt. Bei aufeinander folgenden Kurven, in denen das dynamische Modell stark vom kinematischen abweicht, verliert das Fahrzeug dadurch sehr schnell die Kontrolle. Auch geringe Querbeschleunigungen ändern an diesem Problem nichts, da die maximale Geschwindigkeit des Fahrzeugs nicht eingeschränkt ist. Die Folge daraus ist, dass das Rennauto mit zu hohen Geschwindigkeiten in die Kurven einfährt. Ohne ein genaueres Fahrzeugmodell im MPC-Algorithmus ist eine Regelung des Rennautos, trotz der schnellen Berechnungszeit, mit dieser Kostenfunktion nicht zu empfehlen. Die problematische Stelle im Kurs ist schematisch für die zwei verschiedenen Verhalten der quadratischen und der exponentiellen Kostenfunktion dargestellt.



**Abbildung 6.15:** Kurs auf dem sich das Fahrzeug für die quadratische Kostenfunktion bewegt

---

## 7 Zusammenfassung und Ausblick

---

Dieses Kapitel fasst noch einmal in kurzen Worten die Ergebnisse der Arbeit zusammen und bietet anschließend noch einige Anregungen für Verbesserungen des Algorithmus, Simulators und was in der Zukunft noch untersucht werden kann.

### 7.1 Zusammenfassung

In dieser Arbeit wurde eine Simulationsumgebung erstellt, die auf verschiedene Fahrzeugmodelle zurückgreifen kann und genutzt wird, um die Regelbarkeit eines Rennautos mithilfe eines MPC-Ansatzes zu überprüfen. Dies soll die Grundlage für die Entwicklung der Regelung und Trajektorienplanung des ersten High-Octane Motorsports Driverless Fahrzeugs bilden. Es handelt sich dabei um ein hoch performantes Rennauto, welches für niedrige Geschwindigkeiten (bis  $121 \frac{\text{km}}{\text{h}}$ ) und hohe Kurvendynamik ausgelegt ist. Besonderes Augenmerk wurde auf die Echtzeitfähigkeit und die Unterschiede zwischen den einzelnen Fahrzeugmodellen gelegt. Es ist nicht nur wichtig, ein möglichst realistisches Modell des Rennautos zu besitzen, der Algorithmus muss auch performant genug sein um eine Aktualisierungsrate von mindestens 20 Hz realisieren zu können. Im MPC-Algorithmus wurde ein einfaches kinematisches Fahrzeugmodell hinterlegt, um die Bewegung des Rennautos vorausberechnen zu können. Dieses wurde im Simulator durch ein deutlich realistischeres, dynamisches Fahrzeugmodell simuliert. Die Unterschiede zwischen den Modellen wurden genutzt, um zu evaluieren wie robust der Algorithmus noch ist, wenn die hinterlegte Systembeschreibung von der Realität abweicht. Um das Fahrzeug auf der Spur zu halten ohne dass der Algorithmus bei einer Überschreitung der Seitenlinien abbricht, wurden verschiedene Kostenfunktionen untersucht, die die tangentialen Beschränkungen in die Kostenfunktion integrieren. Das beste Ergebnis lieferte eine e-Funktion, die an den Grenzen sehr schnell groß wird. Sie zeichnet sich durch eine schnelle Berechnungszeit und eine gute Approximation der tangentialen Beschränkung aus. Die Untersuchungen der Länge des Prädiktionshorizontes und der Aktualisierungsrate verdeutlichen, dass das gewählte  $\Delta t$  einen größeren Einfluss auf die schnellste Rundenzeit hat als die Zeit mit der in die Zukunft prädictiert wird. Hier muss abhängig von der zur Verfügung stehenden Rechenleistung ein Kompromiss gesucht werden. Die höchste Priorität hat jedoch ein möglichst genaues Fahrzeugmodell. Durch den Umstieg im Simulator vom kinematischen auf das dynamische Fahrzeugmodell ist

die Rundenzeit signifikant schlechter geworden. Mit einem genaueren Modell im MPC-Algorithmus ist also auch mit einer besseren Regelung zu rechnen.

## 7.2 Ausblick

Im nächsten Abschnitt wird kurz auf die Verbesserungsmöglichkeiten eingegangen, um den MPC-Ansatz noch leistungsfähiger zu machen und die Simulation zu verbessern.

### 7.2.1 Simulator

In der aktuellen Version des Simulators bezieht sich die Beschränkung durch die Streckenbegrenzung auf den Mittelpunkt des Fahrzeugs. Diese Einschränkung kann erweitert werden, indem die tangentialen Beschränkungen auf allen vier Rädern angewandt werden. Ein weiterer Punkt, der noch keine Beachtung gefunden hat, ist die Definition der verschiedenen Rennkurse. Die Stützpunkte mit denen der virtuelle Rennkurs definiert ist, besteht aus Geraden und Punkten, die auf Kreisen mit verschiedenen Radien liegen. Im echten Straßenbau wird für den Übergang zwischen der Geraden und dem Kreis auf Klothoiden gesetzt. Das Ergebnis sind Kurven die ruckfrei bzw. ihre Krümmung eine stetige Funktion der Länge sind. Eine Vereinfachung die in der Realität eine entscheidende Rolle spielt, ist die Änderungsrate des Lenkwinkels. Im Simulator kann das Fahrzeug beliebig schnell zwischen Lenkeinschlägen wechseln. Dies ist natürlich im echten Rennauto nicht möglich. Eine Einschränkung dieser Änderungsrate für die dynamischen Modelle ist daher eine sinnvolle Erweiterung. Ebenfalls vorstellbar wäre die Berücksichtigung der Bewegung des Fahrzeugs während die Regelparameter vom MPC-Algorithmus berechnet werden. Die Stellgrößen beziehen sich auf eine Position, die nicht mehr der aktuellen entspricht sobald die Berechnung abgeschlossen ist.

### 7.2.2 Dynamisches Fahrzeugmodell im MPC

In dieser Arbeit wurde untersucht, unter welchen Bedingungen ein autonomes Rennauto mit einem MPC-Algorithmus geregelt werden kann. Das für das Fahrzeug hinterlegte Modell, welches zur Prädiktion zukünftiger Systemzustände genutzt wird, entspricht einem kinematischen Modell. Trotz dieser Einschränkung sind bereits hohe Geschwindigkeiten gut kontrollierbar. Performance an der Leistungsgrenze des Rennautos ist jedoch erst

zu erwarten, wenn das dynamische Modell im MPC-Ansatz hinterlegt wird, um die zukünftigen Zustände zu berechnen. An diesem Punkt würde der Regler auch die ganze Querdynamik des Fahrzeugs mit in die Lösung einbeziehen. Es muss untersucht werden, welche Einschränkungen das für die Länge des prädizierten Horizontes bedeutet, da die Komplexität der Berechnung steigt. Die Messungen in Kapitel 6.2.2.3 haben jedoch ergeben, dass der größere Prädiktionshorizont weniger Einfluss auf die Rundenzeit hat als das Modell.

### 7.2.3 Zweispurmodell

Im High-Octane Motorsports Verein ist eine Rundenzeitsimulation in der Entwicklung. Für diese wird ein genaues Zweispurmodell genutzt, welches das Fahrwerk, Abtrieb, Nicken, Rollen, Federkräfte und ein sehr genaues Reifenmodell berücksichtigt. Die Berechnungszeit ist nicht mehr in einem Bereich, der in Echtzeit realisierbar ist. Für die Steuerparameter könnte das Modell trotzdem mit dem MPC-Ansatz kombiniert werden, um die idealen Steuerparameter zu finden.

### 7.2.4 Implementierung in C++

Im Driverless-Projekt des High-Octane Vereins wird die Middleware Robot Operating System (ROS) verwendet um die Software des autonomen Rennautos zu entwickeln. ROS unterstützt nativ nur Python und C++. Daher sollte der Algorithmus in C++ implementiert werden, um eine möglichst hohe Performance zu gewährleisten. Es gibt es für C++ kein Framework wie Jump. Die Kombination aus Optimierer und der automatischen Ableitung muss vom Entwickler selbst programmiert werden. Für die automatische Ableitung kann zum Beispiel das CppAd Paket verwendet werden, welches unter der Common Public License veröffentlicht wird. Der Optimierer Ipopt ist in C++ implementiert und daher direkt integrierbar. Ein anderer Weg wäre es, den MPC-Ansatz in Python mit dem Jump ähnlichen Framework CasADi zu implementieren. Wie der Benchmark 5.2 in Kapitel 5.2 jedoch gezeigt hat, ist eine höhere Performance von einer C-Nahen Implementierung zu erwarten.

### **7.2.5 Fahrzeugparameter anpassen**

Wie bereits im Stand der Technik ausgeführt, kann nichtlineare Regression dazu verwendet werden die Werte, mit denen das Modell parametrisiert ist, zu verbessern. Das Rennauto wird von einem menschlichen Fahrer in verschiedenen Szenarien gefahren und sowohl die Steuerparameter (Gas, Bremse, Lenkung) wie auch der Fahrzeugzustand mit möglichst hoher Frequenz abgespeichert. Die so erfassten Daten werden genutzt, um die Parameter des Modells anzupassen und die Differenz zwischen Modell und Realität zu minimieren.



---

## **A Anhang**

---

---

# Abkürzungsverzeichnis

---

<b>M.F.</b>	Magic Formula
<b>Ipopt</b>	Interior Point OPTimizer
<b>ROS</b>	Robot Operating System
<b>Jump</b>	Julia for Mathematical Optimization
<b>K.K.</b>	Kammschen Kreis
<b>MPC</b>	Model Predictive Control
<b>FS</b>	Formula Student
<b>JIT</b>	just-in-time
<b>SLSQP</b>	Sequential Least SQuarez Programming
<b>SFML</b>	Simple and Fast Multimedia Library

---

# Literaturverzeichnis

---

- [AAM] Liniger Alexander, Domahidi Alexander, and Morari Manfred. Optimization-based autonomous racing of 1:43 scale rc cars. *Optimal Control Applications and Methods*, 36(5):628–647.
- [AGT12] C. Athanasiadis, D. Galanopoulos, and A. Tefas. Progressive neural network training for the open racing car simulator. In *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 116–123, Sept 2012.
- [AMZ] Success story amz racing. <https://www.sbg-systems.com/news/amz-racingellipse-n>. Accessed on 2018-07-05.
- [BAS] Rechtsfolgen zunehmender  
fahrzeugautomatisierung. [https://www.bast.de/BASt\\_2017/DE/Publikationen/Foko/2013-2012/2012-11.html](https://www.bast.de/BASt_2017/DE/Publikationen/Foko/2013-2012/2012-11.html). Accessed on 2018-06-22.
- [Cas] Casadi, a symbolic framework. <https://github.com/casadi/casadi/wiki>. Accessed on 2018-07-01.
- [Cpp] Python algorithmic differentiation using cppad. <https://github.com/b45chl/pycppad>. Accessed on 2018-07-01.
- [CUR18] FLORIAN CURINGA. Autonomous racing using model predictive control. 2018.
- [Dan51] G. B. Dantzig. *Maximization of a Linear Function of Variables Subject to Linear Inequalities*, in *Activity Analysis of Production and Allocation*, chapter XXI. Wiley, New York, 1951.
- [FB05] T. Keviczky J. Asgari D. Hrovat F. Borrelli, P. Falcone. Mpc-based approach to active steering for autonomous vehicle systems. *International Journal of Vehicle Autonomous Systems (IJVAS)*, 3(2/3/4), 2005.
- [FGW02] Anders Forsgren, Philip E. Gill, and Margaret H. Wright. Interior methods for nonlinear optimization. *SIAM Review*, 44(4):525–597, 2002.

- [FsW] Formula student - world ranking lists. <https://mazur-events.de/fs-world/>. Accessed on 2018-05-14.
- [GD17] G. Ganga and M. M. Dharmana. Mpc controller for trajectory tracking control of quadcopter. In *2017 International Conference on Circuit ,Power and Computing Technologies (ICCPCT)*, pages 1–6, April 2017.
- [Inn] Innere punkte verfahren. <https://de.wikipedia.org/wiki/Innere-Punkte-Verfahren>. Accessed on 2018-05-28.
- [jul] Juliadiff. <http://www.juliadiff.org/>. Accessed on 2018-06-01.
- [KG10] Krisada Kritayakirana and J. Christian Gerdes. Autonomous cornering at the limits: Maximizing a “g-g” diagram by using feedforward trail-braking and throttle-on-exit. *IFAC Proceedings Volumes*, 43(7):548 – 553, 2010. 6th IFAC Symposium on Advances in Automotive Control.
- [KPSB15] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099, June 2015.
- [PB92] Hans B. Pacejka and Egbert Bakker. The magic formula tyre model. *Vehicle System Dynamics*, 21(sup001):1–18, 1992.
- [PER16] GONÇALO COLLARES PEREIRA. Model Predictive Control for Autonomous Driving of Over-Actuated Research Vehicle. 2016.
- [PYA] Pyadolc, a wrapper for adol-c. <https://github.com/b45ch1/pyadolc>. Accessed on 2018-07-01.
- [Raj11] R. Rajamani. *Vehicle Dynamics and Control*. Mechanical Engineering Series. Springer US, 2011.
- [Sci] Scipy. <https://scipy.org/index.html>. Accessed on 2018-06-23.
- [SCT07] M. Spcmgenberg, V. Calmettes, and J. Y. Tourneref. Fusion of gps, ins and odometric data for automotive navigation. In *2007 15th European Signal Processing Conference*, pages 886–890, Sept 2007.
- [sim] Simplex verfahren. <https://de.wikipedia.org/wiki/Simplex-Verfahren>. Accessed on 2018-05-31.

- [SMED10] D.E. Seborg, D.A. Mellichamp, T.F. Edgar, and F.J. Doyle. *Process Dynamics and Control*. John Wiley & Sons, 2010.
- [TT16] Andrei Turkin and Aung Thu. Benchmarking python tools for automatic differentiation. *CoRR*, abs/1606.06311, 2016.
- [Way] First autonomous taxi in 2018 says waymo. [https://www.theregister.co.uk/2018/05/09/first\\_autonomous\\_vehicles/](https://www.theregister.co.uk/2018/05/09/first_autonomous_vehicles/). Accessed on 2018-05-15.
- [WDG<sup>+</sup>16] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos Theodorou. Aggressive driving with model predictive path integral control. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440, 2016.
- [WDY16] Shiyao Wang, Zhidong Deng, and Gang Yin. An accurate gps-imu/dr data fusion method for driverless car based on a set of predictive models and grid constraints. *Sensors*, 16(3), 2016.
- [WQ01] Danwei Wang and Feng Qi. Trajectory planning for a four-wheel-steering vehicle. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 4, pages 3320–3325 vol.4, 2001.

---

# Abbildungsverzeichnis

---

1.1	Zürichs FS-Driverless-Fahrzeug im Jahr 2017 während des Trackdrive [AMZ] . . . . .	2
3.1	Block Diagramm zu MPC . . . . .	7
3.2	Funktionsprinzip von MPC: Die Kostenfunktion ist die Reduktion der Distanz zur Referenztrajektorie. Durch das Wissen über das hinterlegte Modell können die Kosten sehr schnell minimiert werden ohne die Beschränkungen zu verletzen . . . . .	8
3.3	Lineare Optimierung: Suche des Maximums an den Schnittpunkten der Ungleichungen . . . . .	10
3.4	Simplex Methode: Ablaufen des Suchraums entlang der Kanten der Ungleichheitsbedingungen von Knoten zu Knoten [sim] . . . . .	11
3.5	Beim Innere-Punkte-Verfahren bewegt sich der Optimierer durch das Innere des Polytops auf der Suche nach dem Minimum [Inn] . . . . .	13
3.6	Auswirkung des Parameters $\mu$ auf das Verhalten der Barrierefunktion . . .	14
3.7	Kinematisches Modell . . . . .	17
3.8	Der Schräglaufwinkel ergibt sich aus Lenkwinkel $\delta$ und Bewegungsrichtung $\theta_v$ . . . . .	20
3.9	Parameter im Reifenmodell . . . . .	21
3.10	Laterale Kräfte der Reifen abhängig vom Schräglaufwinkel . . . . .	22
3.11	Kammischer Kreis: Beschränkung der maximal wirkenden Kraft. . . . .	23
3.12	Dynamic Vehicle Model . . . . .	25
4.1	Ideale Trajektorie für eine 90 Grad Kurve . . . . .	30
4.2	Grafische Visualisierung der Prädiktion abhängig von den Steuerparametern für drei Schritte in die Zukunft . . . . .	32
4.3	Zielpunkt Distanzminimierung: Das virtuelle Ziel wird immer weit genug vor dem Fahrzeug hergeschoben, so dass es nie erreicht werden kann . . .	34
4.4	Maximieren der Streckendistanz . . . . .	35
4.5	Tangentiale Begrenzung hält das Fahrzeug auf der Fahrspur . . . . .	36
4.6	Vektor Projektion zum berechnen der tangentialen Beschränkung . . . . .	37
4.7	Fehlerhaftes Verhalten in Spitzkurven . . . . .	38

4.8	Verschiedene Distanzmaße um das Verhalten der Trajektorienplanung zu beeinflussen . . . . .	40
5.1	Ausführungszeiten für verschieden große Horizontlängen . . . . .	42
5.2	Ausführungszeiten für verschiedene Sprachen . . . . .	43
5.3	Blockdiagramm von Jump. Wichtige Vorteile sind die automatische Ableitung und das einfache Tauschen von Optimierern . . . . .	44
5.4	Grafische Darstellung des MPC - Algorithmus in der Simulationsumgebung	45
5.5	Blockdiagramm der Simulation mit dem MPC-Algorithmus . . . . .	46
6.1	Geschwindigkeitsverlauf bei maximaler Beschleunigung für verschiedene longitudinale Modelle . . . . .	49
6.2	Rundkurs zum Messen der maximalen Querbeschleunigung . . . . .	50
6.3	Orientierung des Rennautos für eine Sinuskurve als Eingangswert des Lenkwinkels. Es werden verschiedene Geschwindigkeiten ohne longitudinale Beschleunigung betrachtet. . . . .	52
6.4	Die gleiche Betrachtung wie in 6.3 mit maximaler longitudinaler Beschleunigung.	54
6.5	Fahrzeugorientierung, abhängig von den drei verschiedenen Reifenmodellen für einen sinusförmigen Lenkwinkel als Eingabe . . . . .	55
6.6	Berechnungszeit der Kostenfunktionen für verschieden große Horizontlängen	58
6.7	Rundenzeit für verschiedene Kostenfunktionen und abhängig von der Anzahl der Prädiktionsschritte. Erste (1) und zweite (2) Runde . . . . .	59
6.8	Berechnungszeit für verschiedene Kostenfunktionen mit elastischer Beschränkung.	61
6.9	Die Rundenzeit in Sekunden abhängig von Querbeschleunigung und Prädiktionsschritten. Weiße Bereiche entsprechen dem Abkommen des Fahrzeug von der Strecke . . . . .	62
6.10	Die Rundenzeit in Sekunden abhängig von Querbeschleunigung und Prädiktionsschritten . . . . .	63
6.11	Die Rundenzeit in Sekunden abhängig von Querbeschleunigung und Prädiktionsschritten für das dynamische Modell erweitert durch den K.K.	64
6.12	Rundenzeit abhängig von den Reifenmodellen für 1,2 g und verschieden lange Prädiktionshorizonte . . . . .	65
6.13	Rundenzeit, abhängig von der maximalen Geschwindigkeit und der Querbeschleunigung für einen 15 Schritte langen Horizont . . . . .	66
6.14	Abhängigkeit der Fahrzeugregelung von der Frequenz mit der das MPC-Verfahren ausgeführt wird . . . . .	67

6.15 Kurs auf dem sich das Fahrzeug für die quadratische Kostenfunktion bewegt 68



---

# Tabellenverzeichnis

---

3.1	Magic Formula Parameter . . . . .	22
3.2	Fahrzeug Parameter . . . . .	29
6.1	Beschleunigungszeit für unterschiedliche Kostenfunktionen . . . . .	57

---

# Sebastian Weller

---

## Persönliche Daten

Adresse	An der Kühruh 13 96123 Litzendorf
Mobil	0170 - 9732890
Email	sebastian.weller01@gmail.com
Geburtsdatum	01.04.1992
Staatsangehörigkeit	deutsch

## Studium und Schulbildung

01/2013 - 07/2018	Friedrich-Alexander-Universität Erlangen-Nürnberg Studium: Informations und Kommunikationstechnik
01/2011 - 01/2013	Ohm-Fachhochschule Nürnberg Studium: Elektrotechnik

## Berufliche Erfahrungen / Praktika

01/2016 - 07/2016	Wissenschaftlicher Hilfsmitarbeiter am Fraunhofer IIS
01/2016 - 07/2016	Praktikum bei Siemens Erlangen

## Zusatzqualifikationen

Sprachen	Deutsch (Muttersprache) Englisch (fließend in Wort und Schrift)
Programmiersprachen	Java   C++   C Julia   Python

Erlangen, den (Datum eintragen)

---

Sebastian Weller