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+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

.....the Abstract is here..... **Bitte nicht löschen oder auskommentieren - ist obligatorisch!**

Inhaltsverzeichnis

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 Phönix 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 Zutun 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 Beschleunigungsstreifen. 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 minimalem 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 wurde. Die Messungen und Vergleiche beziehen sich damit auch immer auf einen Kurs, der so in einem FS-Event für die Driverless-Fahrzeuge vorkommen könnte.

Abbildung 1.1: Zürichs FS-Driverless-Fahrzeug im Jahr 2017 während des Trackdrive

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. 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 Model Predictive Control (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 Vorraussetzung 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 [HB15], basierend auf einem Einspursystem, welches sehr schnell berechenbar ist, aber keine dynamischen Effekte berücksichtigt

- Die Verwendung einer dynamischen Fahrzeugbeschreibung welche das Einspurmodell um ein lineares oder nichtlineares Reifenmodell erweitert [AAM], [PER16], [CUR18]
- Ein sehr genaues Zweispurmodell, welches zehn Freiheitsgrade betrachtet und sogar das Einfedern des Fahrzeugs berücksichtigt [FGW02]

Um ein mathematisches Fahrzeugmodell in einem realen Fahrzeug einsetzen zu können, müssen die Parameter, welche das System beschreiben, möglichst genau bestimmt werden. Dies kann mit Hilfe einer linearen Regression sehr effizient und genau realisiert werden [WDG+16]. 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+16].
Sind die Grundvorraussetzungen geschaffen, gibt es unterschiedliche Ansätze, ein Rennauto möglichst performant um einen Rennkurs fahren zu lassen. Eine Möglichkeit ist der Einsatz von neuronalen Netzen ,um geeignete Steuerparameter zu berechnen, wie vom Autor in [AGT12] beschrieben wurde. Ein „g-g“-Ansatz, wie in [KG10], nutzt ein Diagramm von lateralen (g) zu longitudinalen (g) Kräften, abhängig von verschiedenen Euler Spiralen, um das Verhalten eines Rennfahrers zu Simulieren. 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 kombinierter Betrachtung [AAM].

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 Fahrzeugen [FB05] ist inzwischen möglich.

Funktionsweise

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

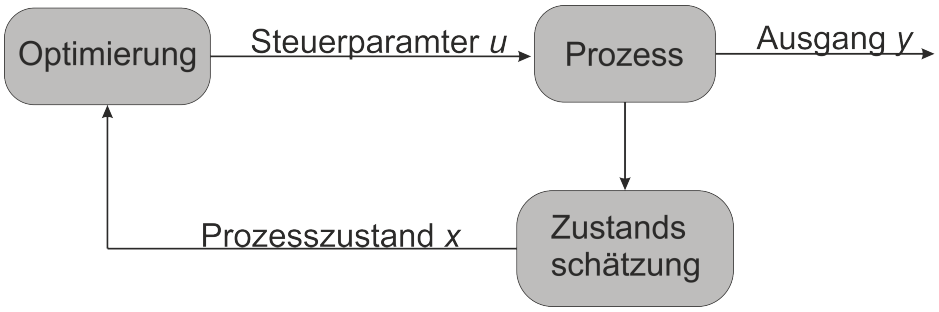

Abbildung 3.1: Block Diagramm zu MPC

Die Funktion lässt sich in drei Schritte unterteilen:

- 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 auf dem realen System 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, welcher 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 entsteht.

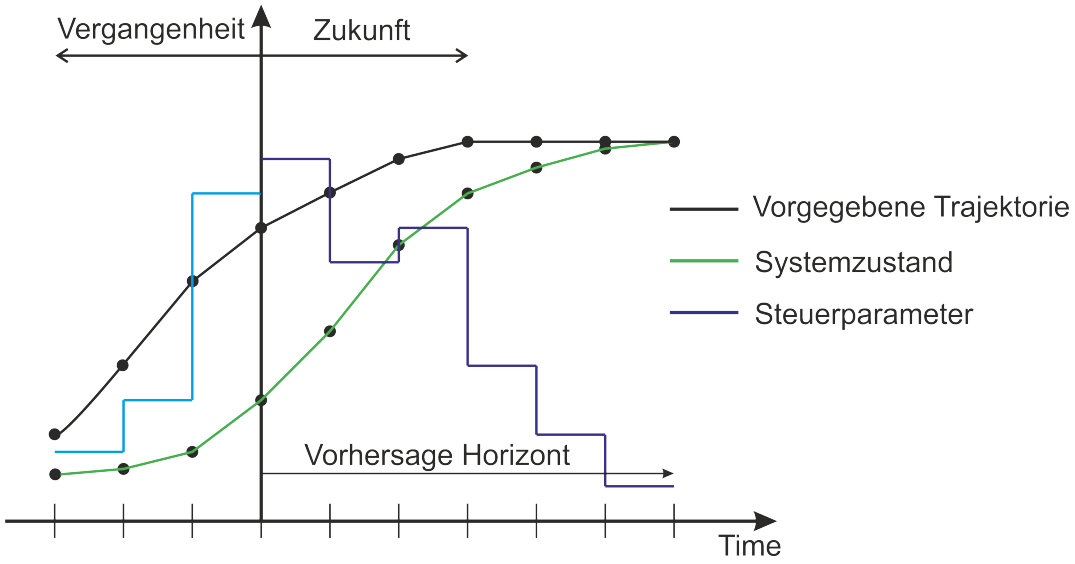

Abbildung 3.2: Funktionsprinzip von MPC: Die Kostenfunktion ist die Reduktion der Distanz zur Referenztrajektorie. Durch das Wissen über das hinterlegte Modell können die Kosten sehr schnell minimiert werden ohne die Beschränkungen zu verletzen.

Wie bereits ausgeführt wurde, berücksichtigt der MPC-Ansatz auch Eingangs-, Ausgangs- und Zustandsbeschränkungen, mit welchen man den Suchraum einschränkt. 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 zu 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

$$\begin{aligned} &\text{minimiere} \quad f(\vec{x}) \\ &\text{unter Berücksichtigung von} \quad g_i(\vec{x}) \leq 0,\ i = 1,2,...,p \\ &\qquad\qquad\qquad\qquad\quad h_j(\vec{x}) = 0,\ j = 1,2,...,m \end{aligned}$$

Der Eingabeparameter $\vec{x}$ sei aus $\mathbb{R}^{\mathrm{n}}$, das heißt, das Problem hängt von n Einflussparametern ab. Die Zielfunktion $f : D \to \mathbb{R}$ sei einmal stetig differenzierbar. Weiterhin sind die Nebenbedingungen in Ungleichheitsform $g_i : D \to \mathbb{R}$ mit $1 \leq i \leq p$ und in Gleichheitsform $h_j : D \to R$ 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 Optimierungsproblem nichtlinear ist.

Ein Optimierungsproblem hat nicht immer eine Lösung. Zum Beispiel können sich die Ungleichungen, welche den Suchraum einschränken, widersprechen ($x \leq 0$ und $x > 0$). In diesem Fall gibt es keine Lösung. 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 $A \in \mathbb{R}^{\mathrm{m,n}}$ und zwei Vektoren $b \in \mathbb{R}^{\mathrm{m,1}}$ und $c \in \mathbb{R}^{\mathrm{1,n}}$.

$$\begin{pmatrix} a_{11}x_1+ & ... & +a_{1n}x_n & \leq b_1 \\ a_{21}x_1+ & ... & +a_{2n}x_n & \leq b_2 \\ . & . & . & . \\ . & . & . & . \\ a_{m1}x_1+ & ... & +a_{mn}x_n & \leq b_m \end{pmatrix}$$

Das Optimierungsverfahren sucht eine Lösung für den Vektor $\vec{x}$, welcher sowohl die linearen Bedingungen erfüllt, als auch die Zielfunktion $cx = c_1x_1 + ... + c_nx_n$ minimiert. Die Kurzschreibweise für dieses Gleichungssystem ist:
min $\{c^{\mathrm{T}}x | Ax \leq b, x \geq 0\}$

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

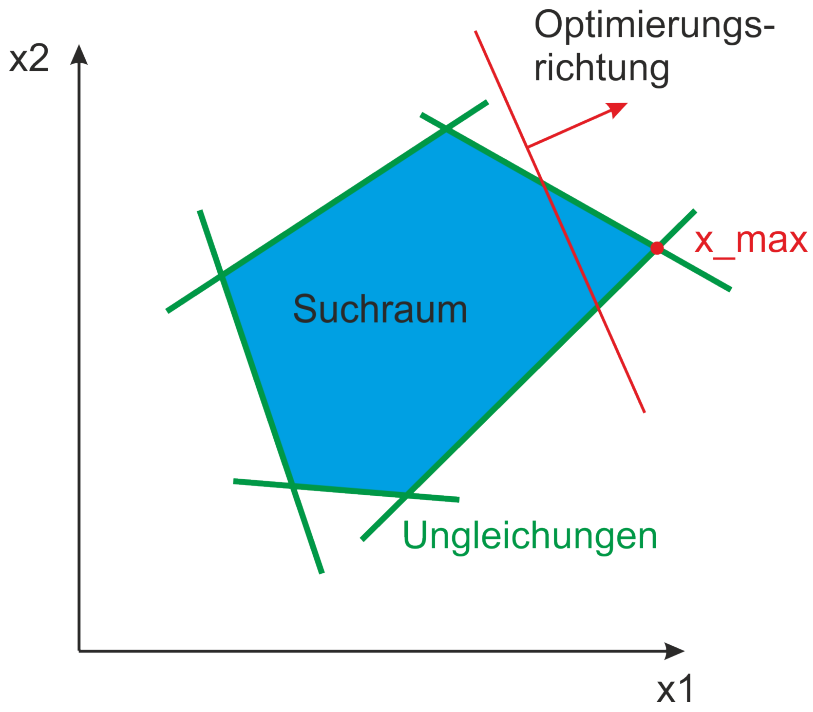

Abbildung 3.3: Lineare Optimierung

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 $c : x \to c^{\mathrm{T}}x$

minimiert, liegt auf den Kanten der Ungleichungen und wird durch Verschiebung der Hyperebene (siehe 3.3) $\{x|c^{\mathrm{T}}x = 0\}$ 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 nutzt die Eigenschaft, 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.

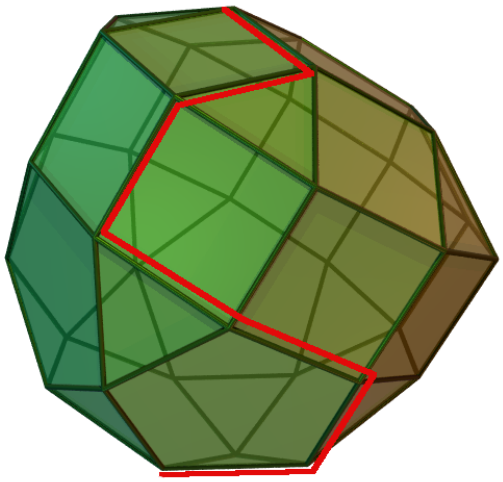

Abbildung 3.4: Simplex Methode: Ablaufen des Suchraums entlang der Kanten der Ungleichheitsbedingungen von Knoten zu Knoten

Die Kante, welche 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 besuchen muss, hat das Verfahren daher ein exponentielles Wachstum $\mathcal{O}(c^{\mathrm{n}})$ 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, welcher auch für nichtlineare Probleme genutzt werden kann. Das Innere-Punkte-Verfahren. Die Un- und Gleichheitsbedingungen wie auch die Kostenfunktionen, welche für den MPC-Ansatz zum regeln des Rennautos benötigt werden, sind nichtlinear. 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 es, sich nicht mehr wie im Simplex-Verfahren auf den Kanten zu bewegen, sondern immer im Inneren des zulässigen Lösungsbereiches. 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 Subsitution so umgeformt, dass sie der Form $c(\vec{x}) = 0$ und $\vec{x} \geq 0$ entspricht. Die neue Ungleichheitsbedingung, kann nun mit einem logarithmische Strafterm $\mu \ln x$ ersetzt und in die Kostenfunktion integriert werden. Der Faktor μ wird genutzt, um die Gewichtung des Strafterms zu verändern.

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

$$B(x, \mu) = c^{\mathrm{T}} x - \mu \sum_{i}^{n} \ln x_i \tag{3.2.1}$$

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

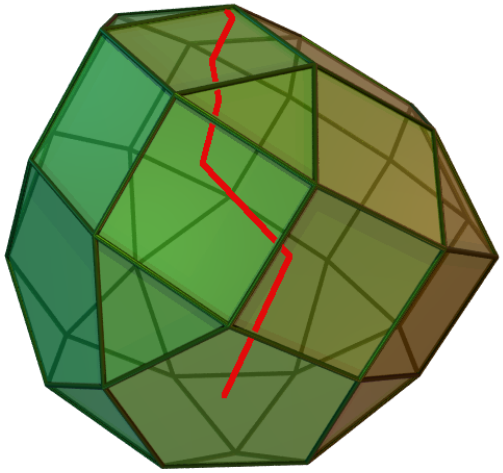

Abbildung 3.5: Beim Innere-Punkte-Verfahren bewegt sich der Optimierer durch das Innere des Polytops auf der Suche nach dem Mimimum

Inneren des zulässigen Lösungsbereiches bewegt 3.5. Wird μ sehr klein (tendiert gegen 0) entspricht die Lösung des Barriereproblems der gleichen wie beim ursprüngliches Optimierungsproblem. Diese Eigenschaft wird genutzt, um von einem großen μ als Startpunkt iterativ $B(x, \mu)$ zu optimieren und dann vor der nächsten Iteration μ zu verkleinern. Das Schrittweise verkleinern von μ dient sorgt dafür dass man sich anfangs grob und schnell später mit kleinen Schritten und langsam der Lösung nähert. Die Auswirkung der stetigen Verkleinerung von μ 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, bedient man sich des Newton-Rapshon-Verfahrens. 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.

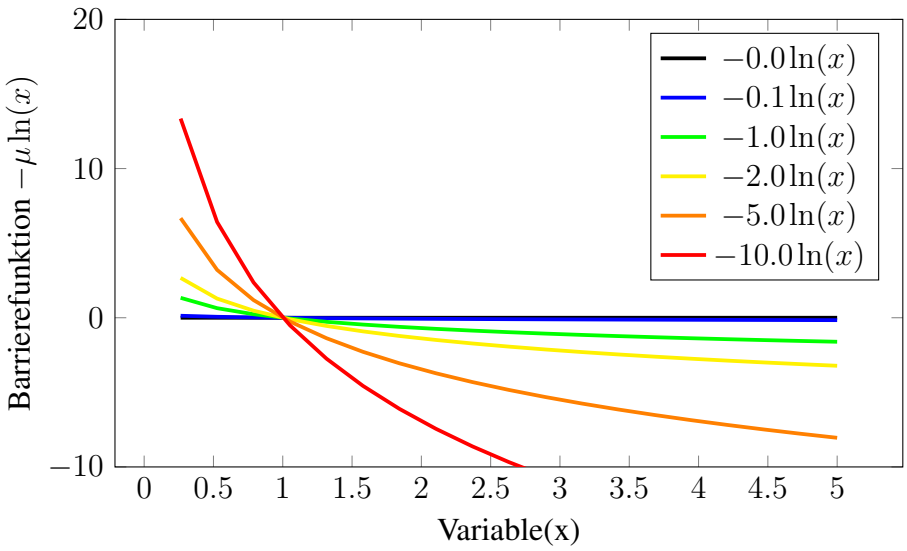

Abbildung 3.6: Beim Innere-Punkte-Verfahren bewegt sich der Optimierer durch das Innere des Polytop auf der Suche nach dem Mimimum

Ein möglicher Ansatz die Ableitung

$$\Delta f(x) = (\frac{\partial f}{\partial x_1}(x), ..., \frac{\partial f}{\partial x_n}) \tag{3.2.2}$$

der Funktion $f : \Re^n \to \Re$ zu berechnen ist

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

mit $h \to 0$. 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 (emg.: automatic differentiation). Dieses System kann die Ableitung bis auf die maximal mögliche Genauigkeit eines *float*-Datentyp berechnen. Zu der höheren Präzision kommt noch die deutlich bessere Laufzeit. Im Idealfall entspricht die Komplexität $\mathcal{O}(1)$ [jul]. Um diese Vorteile nutzen zu können, benötigt man jedoch ein genaues Wissen über die abzuleitende Funktion $f(x)$. 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

$$(f \circ g)' = (f' \circ g)g' \tag{3.2.4}$$

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.

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial w_2}\frac{\partial w_2}{\partial w_1}\frac{\partial w_1}{\partial x} \tag{3.2.5}$$

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 Startwert (eng.: 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 eingegangen, welcher die Un- und Gleichheitsbedingungen definiert. Wie der Name Model Predictive Control schon andeutet, benötigt man eine Modellbeschreibung des zu regelnden Systems. 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 keine wirkenden Kräfte berücksichtigt, sondern nur die geometrischen Beziehungen des Fahrzeug betrachtet, um die Bewegung zu berechnen.
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. 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

$$\dot{x} = v\cos(\psi+\beta) \tag{3.3.1}$$
$$\dot{y} = v\sin(\psi+\beta) \tag{3.3.2}$$
$$\dot{\psi} = \frac{v}{l_r}\sin(\beta) \tag{3.3.3}$$
$$\dot{v} = a \tag{3.3.4}$$
$$\beta = \arctan\left(\frac{l_r}{l_f+l_r}\tan(\delta_f)\right) \tag{3.3.5}$$

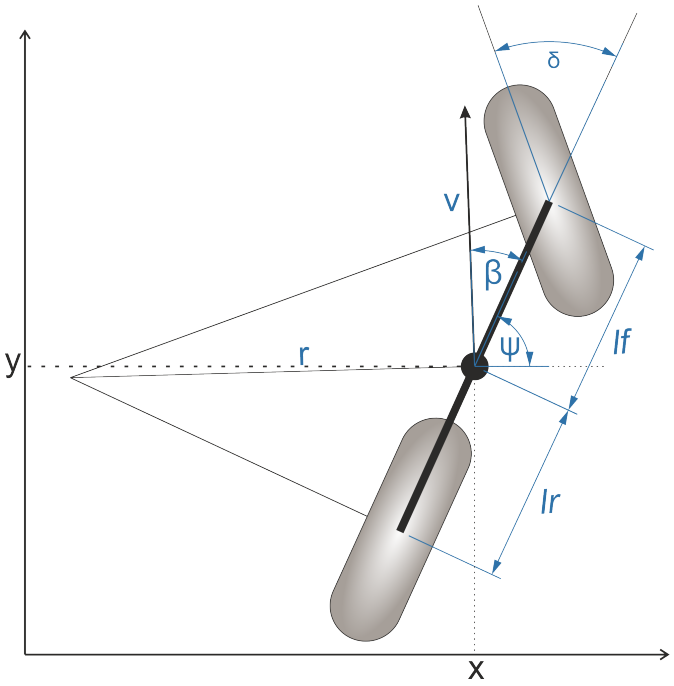

Abbildung 3.7: Kinematisches Modell

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. ψ 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 (β), 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.
Die Parameter lassen sich in zwei Bereiche unterteilen:

- Steuerparameter
  a, δ
- Zustandsgrößen
  x, y, v, ψ

Die Annahme eines kräftefreien Modells, bei dem das Vorderrad genau in die Richtung rollt, in die es zeigt, ist nur bis etwa 5 m/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 den besten *Acceleration* Durchgang in FSG-2017 mit der Zeit $t = 4.5s$ und der Endgeschwindigkeit $v = 32.78\frac{m}{s^2}$ berechnet. Für

$$a = \frac{v}{t} \tag{3.3.6}$$

erhält man eine mittlere Beschleunigung von $7.284\frac{m}{s^2}$.

Überprüft man die Werte mit der Formel zum Berechnen der zurück gelegten Entfernung

$$x = \frac{1}{2} * a * t^2 \tag{3.3.7}$$

liegt man mit $73.75m$ nur zwei Prozent neben der genauen Streckenlänge von $75m$.

Begrenzung des Kurvenradius

Da beim kinematischen Modell die Geschwindigkeit in Kurven keine Rolle spielt, kann das Fahrzeug mit jedem v den gleichen Kurvenradius durchfahren. Um auch in höheren Geschwindigkeiten eine akzeptable Regelung zu erhalten, wird eine Beschränkung eingefügt, welche die maximale Querbeschleunigung beschränkt. Der Maximalwert entspricht dem höchsten g-Wert, den das Fahrzeug des High Octane Motorsports mit Aerodynamik in einer Kurve erreichen kann: $a_{max} \sim 2.0g$. Mit der Gleichung

$$|\beta| \leq \arctan(\frac{1}{2}\frac{\frac{l}{2}a_{max}}{v^2}) \tag{3.3.8}$$

wird dies über die Einschränkung des Schwimmwinkels, welcher 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, welches die Fahrleistungen des Rennautos am stärksten beeinflusst.

Die Kräfte, welche ein Reifen auf die Straße übertragen kann, sind in longitudinal und lateral unterteilt. 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 bezeichnet man den von der Radmittelebene δ (Lenkwinkel) und der Bewegungsrichtung θ_{vf} des Reifens eingeschlossenen Winkel (siehe Abbildung 3.8)

$$\alpha_f = \delta - \theta_{vf} \tag{3.3.9}$$

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

$$\alpha_r = \theta_{vr} \tag{3.3.10}$$

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

$$F_{yf} = C_\alpha \alpha_f \tag{3.3.11}$$

$$F_{yr} = C_\alpha \alpha_r \tag{3.3.12}$$

Am Schaubild 3.10 lässt sich dieser Bereich zwischen -1.5° und 1.5° sehr gut erkennen. Für größere Schräglaufwinkel kann kein linearer Verlauf mehr angenommen werden

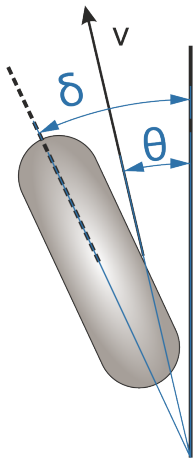

Abbildung 3.8: Der Schrälaufwinkel ergibt sich aus Lenkwinkel δ und Bewegungsrichtung θ_v

und es muss eine genauere Approximation der Kräfte gewählt werden. Hierfür wird die sogenannte *Magic Formula* [PB92] verwendet. Dabei handelt es sich um eine mathematische Gleichung, die sehr gut Messkurven approximiert, welche 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. Bei Eingabe des Schräglaufwinkels α erhält man die lateral auf die Straße wirkende Kraft F_y.

$$F_y = D\sin(C\arctan(B\alpha - E(B\alpha - \arctan(B\alpha)))) \tag{3.3.13}$$

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

$$C = \quad 1 + (1 - (\tfrac{2}{\pi}\arcsin(\tfrac{y}{D}))) \tag{3.3.14}$$

$$B = \quad \frac{\tan(beta)}{C * D} \tag{3.3.15}$$

$$E = \quad \frac{B * x_m - \tan(\frac{pi}{2C})}{B * x_m - \arctan(B * x_m)} \tag{3.3.16}$$

Diese werden abhängig von den folgenden Parametern berechnet.

- D
  Die maximale Kraft welche der Reifen im höchsten Punkt übertragen kann.
- $beta$
  Die Steigung im linearen Bereich.

- x_m
  Der Punkt in rad, an dem der die maximalen Kraft übertragen wird.

- y_a
  Die maximale Kraft, gegen welche die Kurve für große Schräglaufwinkel tendiert.

Diese ändern sich mit der Normalkraft, welche das Fahrzeug auf die Straße presst. Ist diese konstant wie in unserem Fall, müssen die Werte nur einmal berechnet werden und bleiben danach gleich. Sie unterscheiden sich jedoch für das vordere $(.)_r$ und hintere Rad $(.)_r$. Werden in einem komplexeren Modell auch die aerodynamischen Abtriebskräfte und die Steigung der Straße berücksichtigt, ändern sich D, $beta$, xm 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 dargestellt, welchen Einfluss die einzelnen Werte auf die Kurve der Reifenkräfte haben.

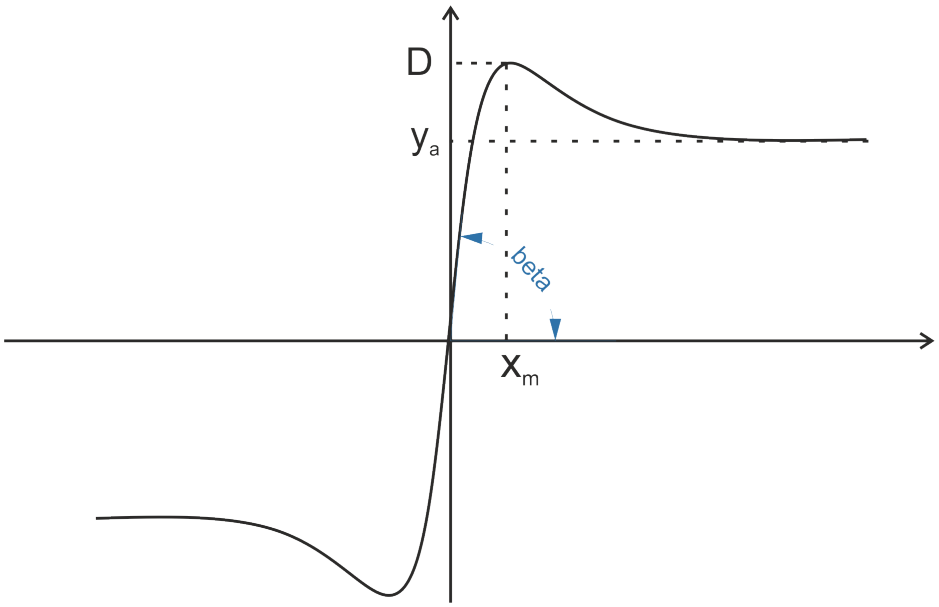

Abbildung 3.9: Parameter im Reifenmodell

Zusätzlich zu dem ursprünglichen Reifenmodell gibt es zwei Vereinfachungen, welche jeweils weniger Rechenaufwand erzeugen, dafür jedoch in ihrer Genauigkeit eingeschränkt sind. Die Modelle basieren weiterhin auf den oben beschriebenen Parametern B,C und D. Die Gleichungen sind

$$F_y = D\sin(C\arctan(Bx)) \tag{3.3.17}$$

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
$beta_f$	$\frac{\pi}{1.9}$	rad
$beta_r$	$\frac{\pi}{1.9}$	rad

Tabelle 3.1: Magic Formula Parameter

welche aus der Arbeit [AAM] entnommen wurde und eine lineare Version

$$F_y = Cx \tag{3.3.18}$$

welche mit $|F_y| <= F_{max}$ die Maximalkraft beschränkt. Nicht weiter spezifiziert, wird im Folgenden davon ausgegangen, dass das genaueste Modell Verwendung findet. Die Parameter, welche für die *Magic Formula* benötigt werden, wurden vom High Octane Motorsports e.V. zur Verfügung gestellt.

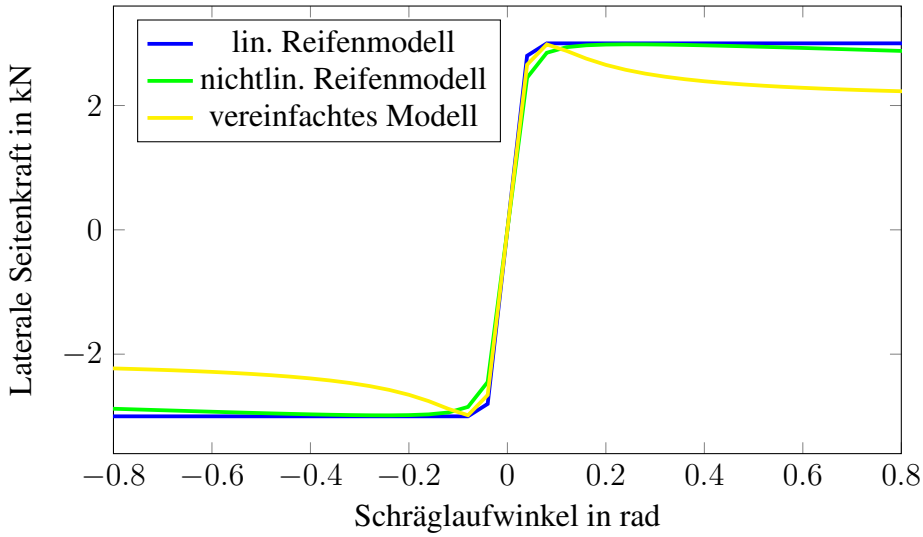

Abbildung 3.10: Reifenmodell

Kammscher Kreis

Ähnlich wie bei F_y wird auch die Kraft, welche das Fahrzeug in Längsrichtung beschleunigt F_x 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.

$$F_x \leq F_{max} \tag{3.3.19}$$

Der Zusammenhang zwischen lateraler und longitudinaler Kraft wird über den Kammscher Kreis (K.K.) modelliert (siehe Schaubild 3.11). Dieser schränkt die wirkenden Kräfte so ein, dass die Hypotenuse aus F_x und F_y sich maximal auf einem Einheitskreis bewegen kann. Der Radius entspricht der maximalen Kraft, welche die Reifen übertragen können (F_{max}).

$$F \leq F_{max} \tag{3.3.20}$$
$$F_{ll} = \sqrt{F_x^2 + F_y^2} \tag{3.3.21}$$
$$\tag{3.3.22}$$

Falls die Kraft F_{ll} größer als F_{max} ist, wird das Verhältnis der Kräfte berechnet und auf F_{max} herunterskaliert.

$$\alpha = \arctan(F_x, F_y) \tag{3.3.23}$$
$$F_x = F_{max} \sin(\alpha) \tag{3.3.24}$$
$$F_y = F_{max} \cos(\alpha) \tag{3.3.25}$$

Der aufmerksame Leser wird bemerkt haben, dass bei einem gleichen Anteil an longitudinaler und lateraler Kraft die Reifen das größte Moment auf die Straße übertragen können. Dieser Punkt wird von Rennfahrern versucht, in Kurven zu treffen, um das gesamte Potenzial der Reifen auszunutzen.

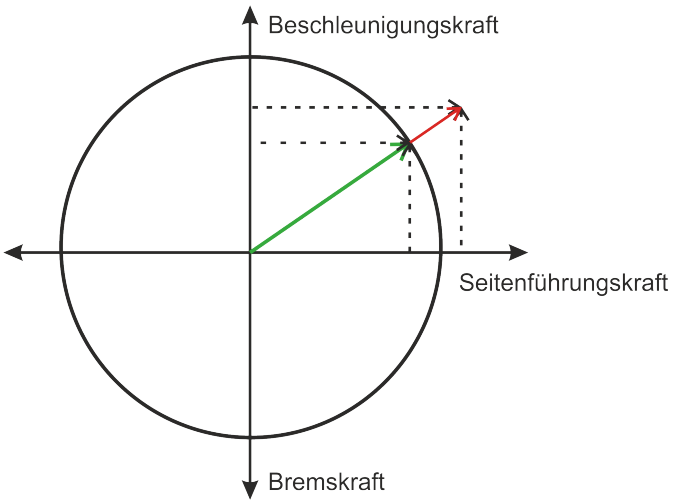

Abbildung 3.11: Kammscher Kreis: Zusammenhang von longitudinaler, lateraler und maximaler Reifenkraft

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.

$$ma_y = F_{yf} + F_{yr} \tag{3.3.26}$$

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

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

$$I_z\ddot{\psi} = l_f F_{yf} - l_r F_{yr} \tag{3.3.27}$$

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

$$m\ddot{x} = m\dot{y}\dot{\psi} + F_x \tag{3.3.28}$$

$$m\ddot{y} = -m\dot{x}\dot{\psi} + F_y \tag{3.3.29}$$

$$I(\ddot{\psi}) = l_f F_{yf} - l_r F_{yr} \tag{3.3.30}$$

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

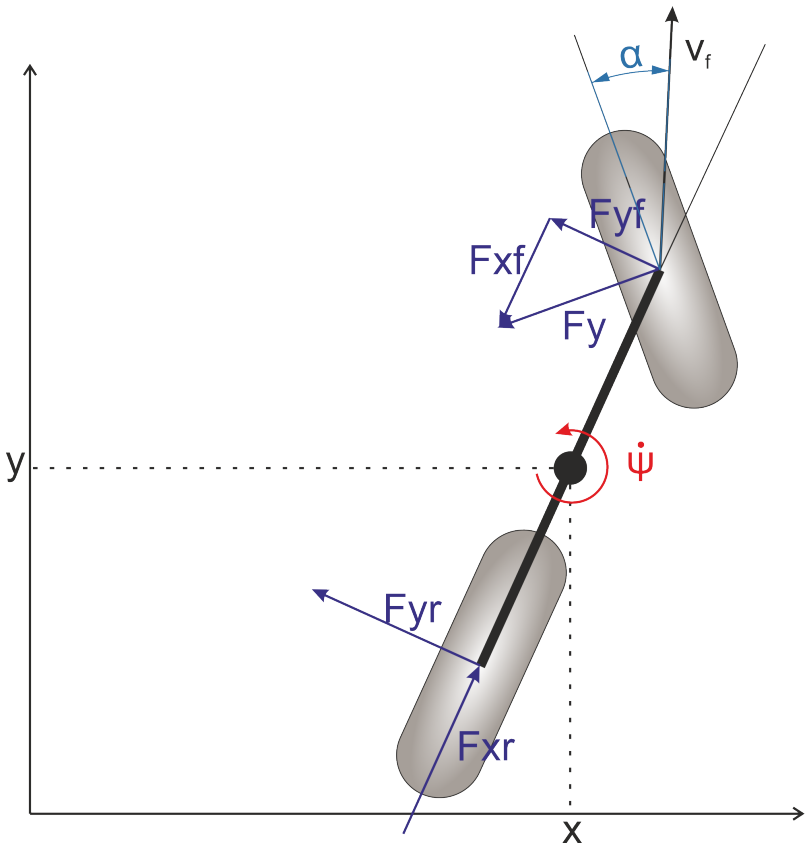

Abbildung 3.12: Dynamic Vehicle Model

$$F_x = F_{xf} + F_{xr} \tag{3.3.31}$$
$$F_y = F_{yf} + F_{yr} \tag{3.3.32}$$

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

$$F_{xf} = -2F_{Cf}\sin(\delta_f) \tag{3.3.33}$$
$$F_{yf} = 2F_{Cf}\cos(\delta_f) \tag{3.3.34}$$
$$F_{xr} = F_{lr} \tag{3.3.35}$$
$$F_{yr} = 2F_{Cr} \tag{3.3.36}$$

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 *magic formula* wie im letzten Abschnitt 3.3.2 berechnet. Die dafür benötigten Schräglaufwinkel werden durch folgende Formeln bestimmt:

$$\alpha_f = \delta_f - \arctan(\frac{\dot{y} + l_f\dot{\psi}}{\dot{x}}) \tag{3.3.37}$$
$$\alpha_r = -\arctan(\frac{\dot{y} - l_r\dot{\psi}}{\dot{x}}) \tag{3.3.38}$$

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. Dies wird realisiert über die Motorleistung und F_{max}.

$$F_{lr_{acc}} = \frac{P_{engine}}{|\dot{x}|} \tag{3.3.39}$$

$$F_{lr_{acc}} \leq 2F_{max} \tag{3.3.40}$$

$$F_{lr_{dec}} = -2F_{max}break \tag{3.3.41}$$

Die Motorleistung berechnet sich aus der maximalen Motorleistung und der Gaspedalstellung tp.

$$P_{engine} = P_{engine_{max}}tp \tag{3.3.42}$$

Die Berechnung der Reifenkräfte ist für Beschleunigung und Bremsvorgang unterschiedlich.

Beschleunigt das Fahrzeug, wirkt nicht nur die Rollreibung der longitudinalen Kraft des Motors entgegen, sondern auch der Luftwiderstand

$$F_{reib} = m\mu g \tag{3.3.43}$$

$$F_{aero} = \frac{1}{2}\rho C_d A_f \dot{x}^2 \tag{3.3.44}$$

Diese Gleichungen sind nur zulässig, solange von einem Rennkurs ausgegangen wird, der keine Steigung besitzt. Die resultierende longitudinale Kraft

$$F_l r = F_{lr} - F_{reib} - F_{aero} \tag{3.3.45}$$

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

$$\dot{x} = \dot{x}\cos(\psi) - \dot{y}\sin(\psi) \tag{3.3.46}$$

$$\dot{y} = \dot{x}\sin(\psi) - \dot{y}\cos(\psi) \tag{3.3.47}$$

$$m\ddot{x} = m\dot{y}\dot{\psi} + F_{xf} + F_{xr} \tag{3.3.48}$$

$$m\ddot{y} = -m\dot{x}\dot{\psi} + F_{yf} + F_{yr} \tag{3.3.49}$$

$$I\ddot{\psi} = l_f F_{yf} - l_r F_{yr} \tag{3.3.50}$$

Formelzeichen	Wert	Einheit
l_f	1.09	m
l_r	0.9	m
l_b	1.99	m
r	0.2	m
m	163	kg
I	1700	kgm^2
A_f	1.5	m^2
P_{engine}	40,5	kW
C_d	1.5	-
ρ	1.225	kg / m^3
F_{max}	3	kN

Tabelle 3.2: Vehicle Parameter

Das Gleichungssystem wird hier in diskreter Form dargestellt

$$x_{k+1} = X_k + \Delta t(\dot{x}_k \cos(\Psi_k) - \dot{y}_k \sin(\Psi_k)) \tag{3.3.51}$$
$$y_{k+1} = Y_k + \Delta t(\dot{x}_k \sin(\Psi_k) - \dot{y}_k \cos(\Psi_k)) \tag{3.3.52}$$
$$\Psi_{k+1} = \Psi_k + \Delta t\dot{\psi}_k \tag{3.3.53}$$
$$\dot{x}_{k+1} = \dot{x}_k + \Delta t(\frac{F_{xf_k}+F_{xr_k}-F_a}{m} + \dot{y}_k\dot{\psi}_k) \tag{3.3.54}$$
$$\dot{y}_{k+1} = \dot{y}_k + \Delta t(\frac{F_{yf_k}+F_{yr_k}}{m} - \dot{x}_k\dot{\psi}_k) \tag{3.3.55}$$
$$\dot{\psi}_{k+1} = \dot{\psi}_k + \Delta t\frac{l_f F_{yf} - l_r F_{yr}}{I} \tag{3.3.56}$$
$$\tag{3.3.57}$$

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

Die zur Berechnung verwendeten Fahrzeugparameter wurden für das zum Verfassen dieser Arbeit aktuellste Fahrzeug erfasst. Die Informationen hierfür sind aus dem CAD-Modell oder im Feld erfassten Testdaten entnommen worden (siehe Anhang).

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, möglichst viel Streckenlänge in der kleinstmöglichen 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.

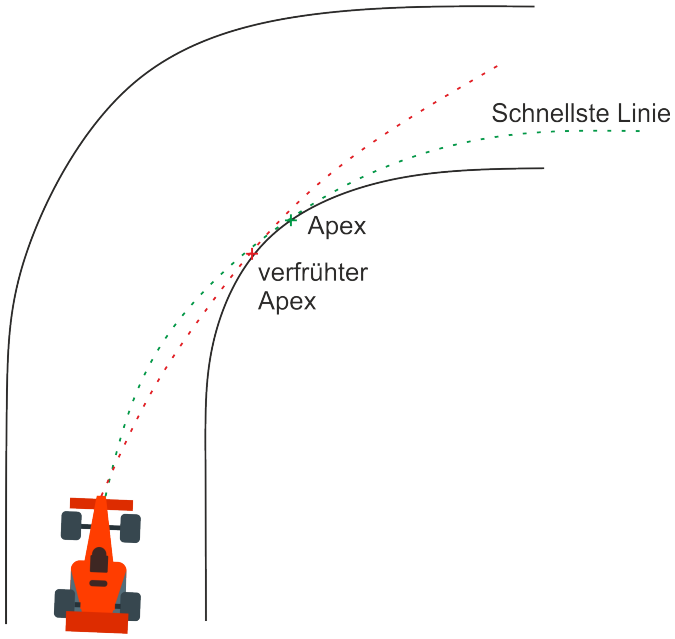

Abbildung 4.1: Ideale Trajektorie für eine 90 Grad Kurve

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 entwickeln.

Die Vorteile werden aber mit dem Nachteil begleitet, dass bei einem Rennauto die longitudinale und laterale Bewegung des Fahrzeugs ganz signifikant miteinander verbunden ist. 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 hat man 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 der Abbildung 4.2 ist grafisch dargestellt, wie man sich die Prädiktion für $N = 3$ vorstellen kann. Zwischen jedem der einzelnen Schritte wird ein Δt angenommen, welches der Abtastrate der Positionsschätzung entspricht, also $\frac{1}{20}s$. Obwohl immer nur der erste Steuerbefehl des berechneten Steuervektors 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 geht solange die Zeitverzögerung kleiner als die gesamte Zukunftsprädiktion ist. Obwohl man diese Werte zum Regeln nutzen kann, nimmt aufgrund von Modellfehlern die Güte der berechneten Werte mit n ab. Man ist also bestrebt, den Algorithmus mit der gleichen Rate wie die Positionsschätzung laufen zu lassen.

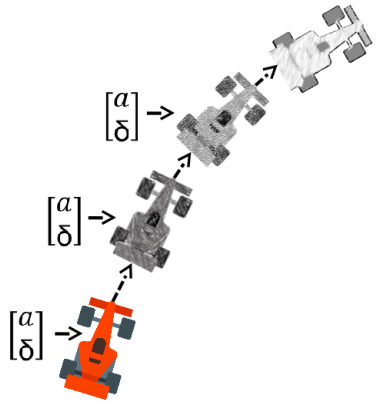

Abbildung 4.2: Grafische Visualisierung der Prädiktion abhängig von den Steuerparametern für drei Schritte in die Zukunft

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 Ungleicheitsbedingungen in das zu optimierende nichtlineare Problem integriert. Dazu wird die diskretisierte Form des Systemmodells 3.3.1 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 auch 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

In dieser Funktion wird die Summe aller Geschwindigkeitswerte der einzelnen Prädiktionsschritte addiert und als Kosten versucht zu maximieren.
$f(\vec{X}) = \sum_1^N v_i$
Die Berechnung startet nicht beim 0ten Schritt, da dieser dem aktuellen Zustand entspricht und der Optimierer keinen Einfluss mehr auf diesen hat. Die Idee hinter dieser Funktion ist, dass sie einfach zu berechnen ist und im Vorhersagehorizont eine möglichst schnelle Trajektorie entsteht.

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 versuchen, Steuerparameter zu finden, die ihn möglichst schnell auf dieses Ziel zufahren lassen. Das virtuelle Ziel wird nach jedem Update wieder so neu positioniert, dass das Rennauto den Punkt niemals erreichen kann und damit konstant schnell weiter fährt. Die Distanz, in der das virtuelle Ziel platziert wird, wird durch die maximale Geschwindigkeit und die Anzahl der Prädiktionsschritte berechnet.

$$d_{goal} = v_0 + N a_{max} \Delta t \tag{4.2.1}$$

Eine bildliche Darstellung der Kostenfunktion ist in Abbildung 4.5 zu sehen.

Maximieren der Streckendistanz

Um eine Kostenfunktion zu finden, welche einer Idealtrajektorie am nächsten kommt, muss überlegt werden wie man diese messen kann. Die bestmögliche Trajektorie ist diejenige, welche für ein Δt die zurückgelegte Strecke, nicht bezogen auf das Fahrzeug, sondern auf den Rennkurs, maximiert. Dies lässt sich zu einer Minimierung der nach einem Zeitschritt übrig bleibenden Reststrecke umformen. Die Restrecke wird einfachheitshalber auf einen Punkt bezogen, welcher 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

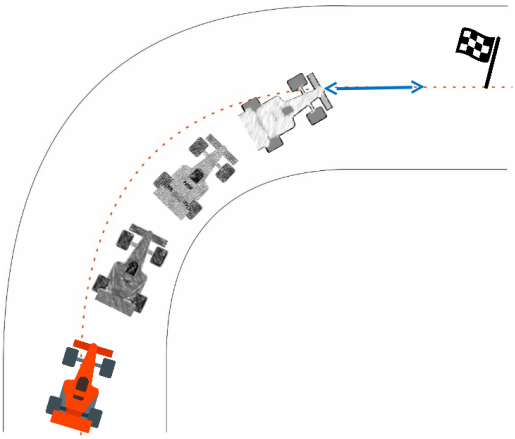

Abbildung 4.3: Zielpunkt Distanzminimierung: Das virtuelle Ziel wird immer weit genug vor dem Fahrzeug her geführt, so dass es nie erreicht werden kann.

Rennkursmitte bezieht. Realisiert wird dies durch eine *vector rejection* $d = |a - \frac{ab}{bb}b|$. Dies ist in der Abbildung 4.4 bildlich veranschaulicht.

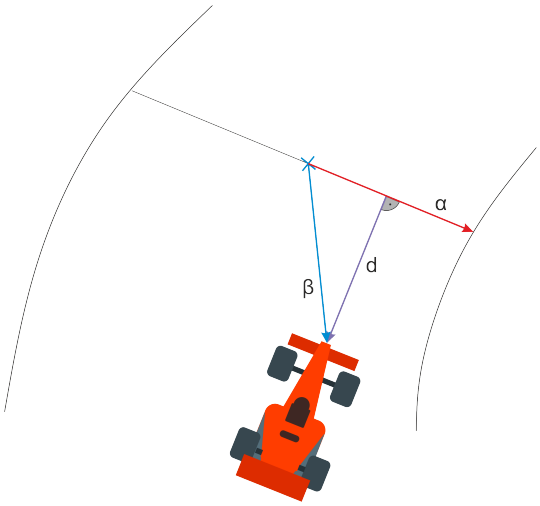

Abbildung 4.4: Maximieren der Streckendistanz (Dummy)

4.3 Strecken-, und Positionsbeschränkung

Die Kostenfunktionen führen dazu, dass der Optimierer die Steuerparameter so anpasst, dass die Kosten minimal werden. Dies würde auf einem Rennkurs zum Abkürzen bei Kurven führen, da in diesem Fall die Kosten geringer werden. Um dies zu verhindern,

wurde eine zusätzliche Streckenbeschränkung eingeführt. Diese basiert auf zwei Tangenten, die für jedem Prädiktionsschritt an die Fahrbahnaußen- und Innenseite projiziert werden. Solange sich das Rennauto innerhalb dieser Tangenten bewegt, ist die Beschränkung erfüllt. Während des Optimierungsvorgangs wird die Position der Tangenten nicht verändert. Um für den Prädiktionsschritt n den passenden Streckenabschnitt zu suchen, für den die Beschränkung ausgelegt wird, kann auf die Ergebnisse der vorausgehenden Optimierung zurückgegriffen werden. Die Position des $n+1$-Schritt 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.

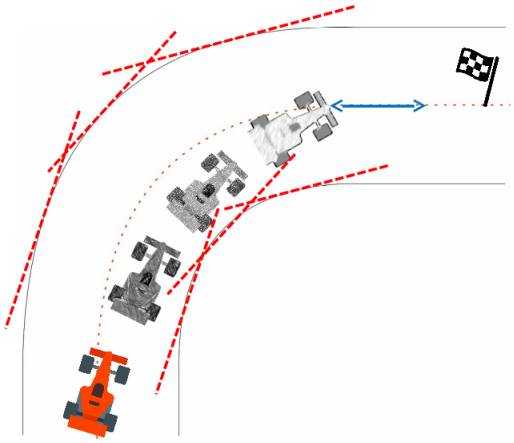

Abbildung 4.5: Tangentiale Begrenzung hält das Fahrzeug auf der Fahrspur

Dieses Verfahren funktioniert nur, da die Beschleunigung des Rennautos physikalischen Grenzen unterliegt und sich daher immer in einem bestimmten Bereich um den Punkt herum bewegt, welcher bei gleichbleibender Geschwindigkeit Δtv Meter entfernt von seinem Vorgängerzustand liegt. Wenn sich dieser Punkt in einer Kurve direkt an der Streckenmarkierung befindet, könnte das Fahrzeug sich, trotz Erfüllen der Tangentialbeschränkung, vom Kurs herunter bewegen. 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 $d = \frac{a*b}{|b|}$ realisieren. A entspricht dem Vektor vom Mittelpunkt der Strecke zum Massenschwerpunkt des Rennautos und b dem Vektor vom Mittelpunkt zum Rand des Rennkurses.

Für diese Beschränkung können sich die Räder des Fahrzeuges über den Rand hinaus bewegen. Soll dies verhindert werden, muss diese Beschränkung auf jedes der einzelnen Räder erweitert werden.

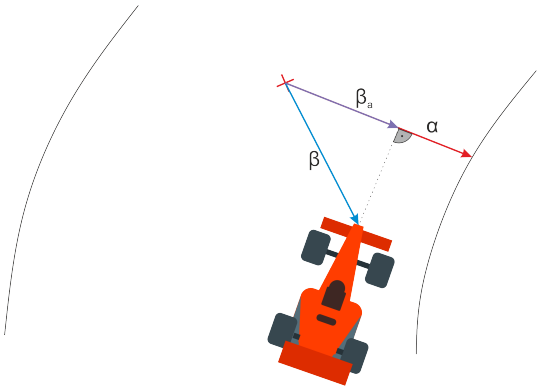

Abbildung 4.6: Vektor Projektion zum berechnen der tangentialen Beschränkung

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

Suchbereichsbeschränkung

Während dem Testen der Kostenfunktion zum Maximieren der Geschwindigkeit ist das Phänomen aufgetreten, dass der Optimierer augenscheinlich unmögliche Lösungen gefunden hat, bei denen sich das Rennauto links aus einer Rechtskurve und andersherum herausbewegt. Dies passiert jedoch nur in Kurven, die mehr als 90ř 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, welche eine höhere Geschwindigkeit ermöglicht. 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. Es entsteht also ein Dreierzyklus, welcher im schlechtesten Fall solange durchlaufen wird, bis das Fahrzeug eine Tangente trifft und die Optimierung beendet wird, da keine weitere mögliche Lösung gefunden werden kann.
Um dieses Verhalten zu verhindern, wird für diese Kostenfunktion eine zusätzliche Suchbereichsbeschränkung eingeführt. Diese legt um die Prädiktionsschritte der letzten Optimierung einen Kreis, welcher den Suchbereich so einschränkt, dass keine neue

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

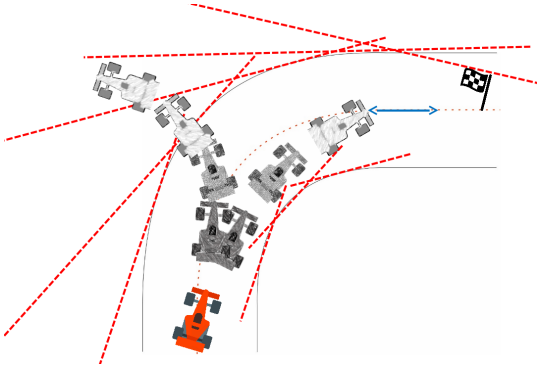

Abbildung 4.7: Fehlerhaftes Verhalten in Spitzkurven

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

Elastische Distanzminimierung

Der Nachteil beider oben aufgeführten Kostenfunktionen ist, dass sie keinen Spielraum beim Erfüllen der Streckenbeschränkung haben. In der Simulationsumgebung kann, wenn sowohl das im Simulator als auch im MPC hinterlegte Fahrzeugmodell exakt gleich sind, das virtuelle Rennauto bis an die Streckenbegrenzung heranfahren, ohne dass die Optimierung unlösbar wird. In der Realität würde eine solche Kostenfunktion das Fahrzeug jedoch zu nah an die Pylonen heranführen. Bei nur den kleinsten Regelfehlern, Rutschen, Abweichungen vom Fahrzeugmodell etc. würde sich der initiale Fahrzeugzustand $\vec{x}_0$ bereits außerhalb der Beschränkungen befinden und damit keine Lösung mehr für das Optimierungsproblem möglich sein. Um dieses Problem zu umgehen, wird eine neue Art der Kostenfunktion eingeführt, welche 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 beiden Kostenfunktionen wird über die Gewichtungsvariablen a und b gesteuert.
$f(\vec{x}) = a(\sum_1^{N+1} v_i) + b(\sum_1^{N+1} g(h(\vec{x}_i)))$
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:

$$g(d) = \alpha d^2 \tag{4.3.1}$$

$$g(d) = e^{\alpha(k_1+d)} + e^{-\alpha(k_2+d)} \tag{4.3.2}$$

$$g(d) = \left|\frac{\alpha}{k_1-d} + \frac{\alpha}{k_2-d}\right| \tag{4.3.3}$$

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

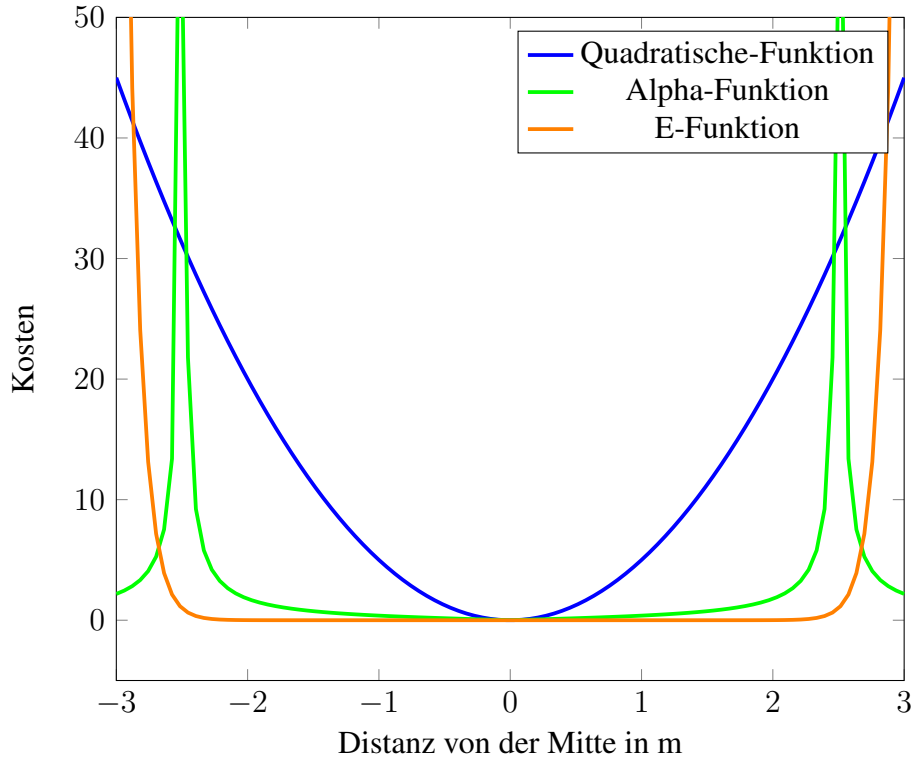

Abbildung 4.8: Verschiedene Distanzmaße um das Verhalten der Trajektorienplanung zu beeinflussen.

Die Idee hinter den Funktionen 4.3.2 und 4.3.3 ist, dass das Rennauto sich uneingeschränkt auf der ganzen Breite der Strecke bewegen kann, ohne dass die Distanzfunktion eine größere Rolle spielt. Erst beim Erreichen des Fahrbahnrandes, werden die Kosten sehr

schnell extrem groß. Idealerweise stellen also diese Kostenfunktionen die tangentialen Begrenzungen ideal nach. Die Funktion, welche quadratisch wächst, soll ebenfalls betrachtet werden, um zu untersuchen, ob mit ihr eine erfolgreiche Regelung möglich wäre, bei gesenktem Rechenaufwand.

5 Simulationsumgebung

Um die Funktion des MPC-Algorithmus verifizieren zu können wurde eine Simulationsumgebung entwickelt. Der Aufbau ist dreigeteilt: Eine Fahrzeugsimulation welche mit austauschbaren Fahrzeugmodellen die Bewegung des Rennautos abhängig von Steuereingaben und einem Δt berechnet. Der Regelanteil in Form des MPC und eine Visualisierung auf Basis einer Spieleengine 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 SQuarez Programming (SLSQP)-Algorithmus verwendet. Dieser ist direkt in die SciPy-Bibliothek [Sci] integriert und unterstützt Eingangs-, Ausgangs- und Zustandsbeschränkungen ohne die es nicht möglich ist, den MPC-Algorithmus für die Fahrzeugregelung auszulegen. Solange die Anzahl der Prädiktionsschritte klein bleibt und zehn nicht überschreitet, ist die Ausführungszeit gering genug, um eine Aktualisierungsrate von 20 Hz zu erreichen.

Bei größeren Prädiktionsvektoren fällt die Geschwindigkeit jedoch sehr schnell ab. 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. Bessere Skalierung erhält man bei der Verwendung eines *Solvers*, der das Innere-Punkte-Verfahren nutzt. Dieser Algorithmus setzt jedoch, um gute Performance zu erreichen, wie in Abschnitt 3.2.2 beleuchtet, die Kenntnis über die Ableitung der Systemfunktion voraus. Diese kann mit Frameworks wie: CasADi, PyADOL-C, PyCppAD 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.

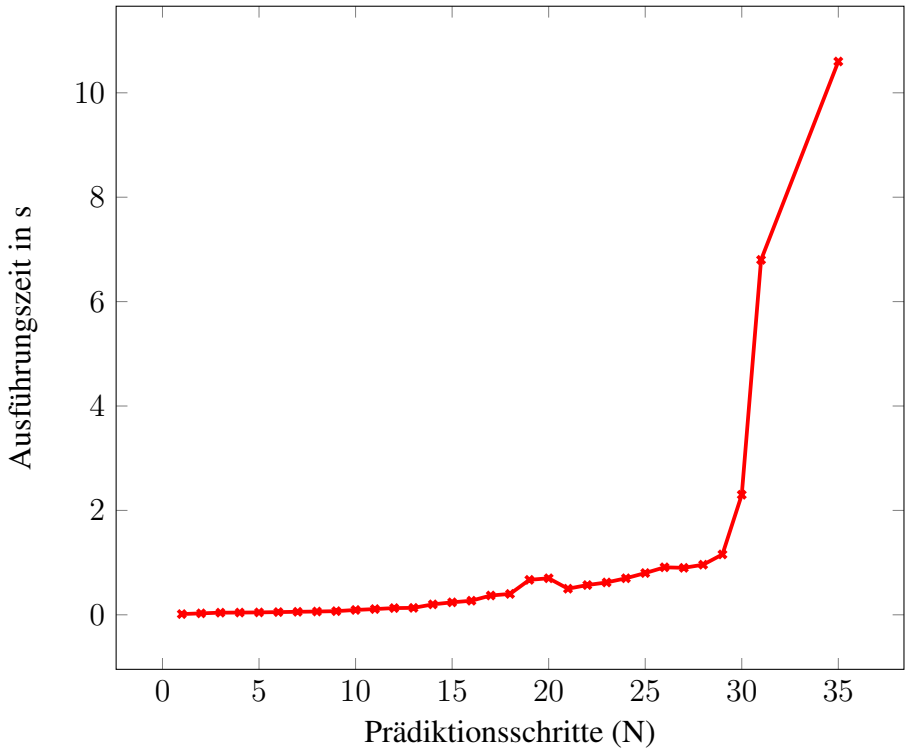

Abbildung 5.1: Ausführungszeiten für verschieden große Horizontlängen

5.2 Julia

Julia ist die Programmiersprache, die die Umsetzung des Model Predictive Control-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 Analyse 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 Feautres 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

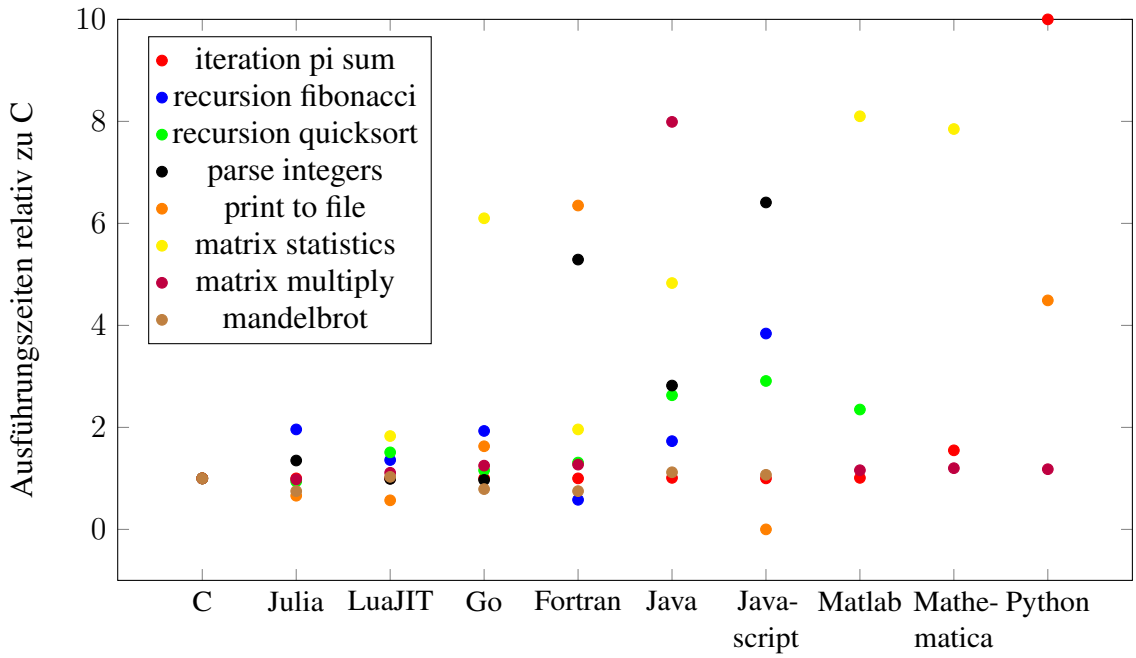

Abbildung 5.2: Ausführungszeiten für verschiedene Sprachen

wird die am besten passende Methodendefinition aufgerufen.

- 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 ist, dass während der Ausführung kontinuierlich Analysen durchgeführt werden, um Bereiche ausfindig zu machen, die durch ein Neukompilieren eine signifikante Verbesserung bei der Ausführungszeit erfahren würden.

Obwohl es möglich wäre, die einzelnen Bestandteile, welche nötig sind, um den Model Predictice Control - Algorithmus zu berechnen, in Julia zu implementieren, gibt es eine

umfangreiche Erweiterung, welche eine passende Schnittstelle für solche Probleme zur Verfügung stellt.

Jump

Jump ist eine Erweiterung für Julia mit der sich mathematische Probleme modellieren und lösen kann. 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 5.4). Ebenfalls ein großer Vorteil ist, dass Jump direkt die automatische Ableitung berechnet und so dem Programmierer sehr viel Arbeit erspart.

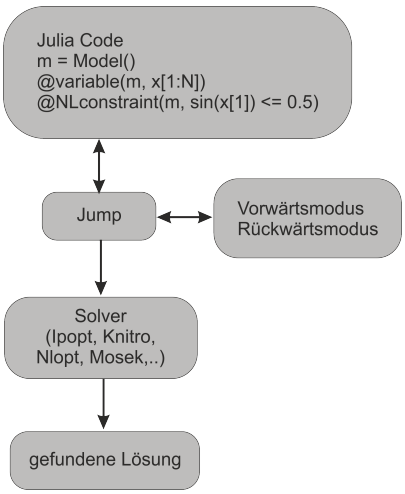

Abbildung 5.3: Blockdiagramm von Jump

Die Syntax, welche Jump für die Definition der Probleme vorsieht, ähnelt der standardmäßigen mathematischen Definition, was die Anwendung intuitiv gestaltet. existiert eine ausführliche Dokumentation. Die Verwendung von Jump und Ipopt führe zu einer deutlich besseren Skalierung und Laufzeit (siehe 7.2.2) als der SLSQP-Ansatz in Python. Die Abbruchbedingung in Ipopt wurde auf $1e-1$ festgelegt.

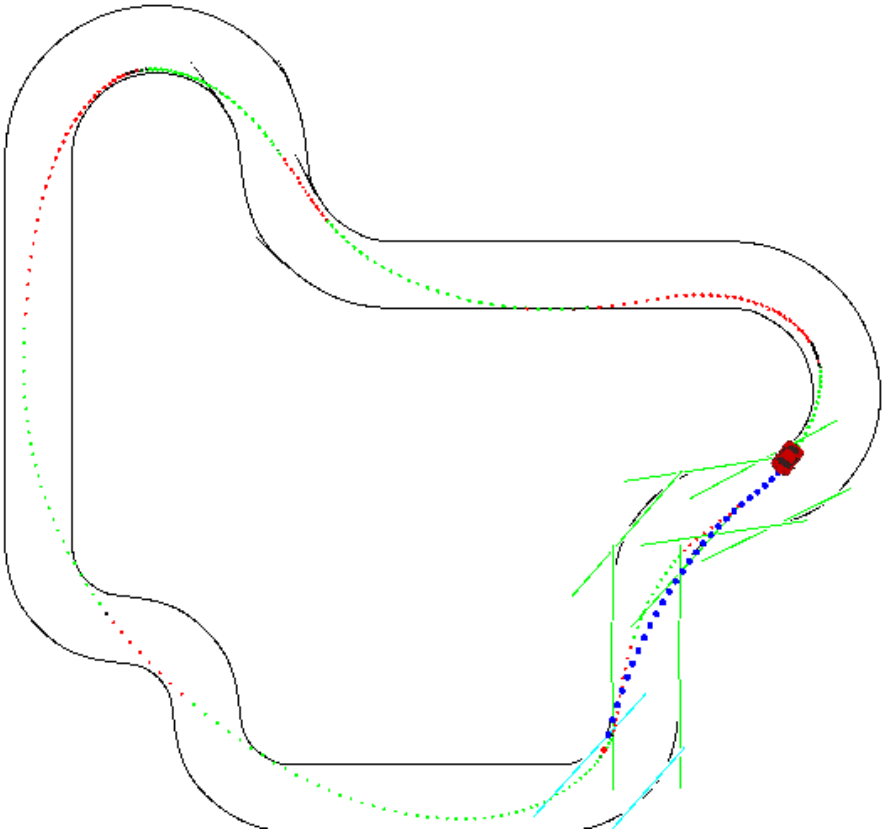

Abbildung 5.4: Grafische Darstellung des MPC - Algorithmus in der Simulationsumgebung

5.3 Simple and Fast Multimedia Library

Die Basis für die Simulation bildet die *Simple and Fast Multimedia Library (SFML)* welche für die grafischen Darstellungen des Kurses, Rennauto, Prädiktionsschritte und Fahrzeuginformationen genutzt wird. Zusätzlich zur Visualisierung, stellt die Spieleengine auch sicher, dass die zeitlichen Abläufe eingehalten werden. Das Grundprinzip ist eine einzige unendlich laufende Schleife die mit einer vorher festgelegten Häufigkeit pro Sekunde (fps) ausgeführt wird. Benötigen die Berechnungen innerhalb dieser Schleife länger als das angegebene $\Delta t = \frac{1}{fps}$
sinkt die Ausführungsrate, überschritten wird sie jedoch nie. Die Schritte, die innerhalb dieser Schleife abgearbeitet werden sind zuerst das Abfragen möglicher Eingaben des Nutzers oder *events* der einzelnen Objekte, z.b. eine Kollision. Im zweiten Schritt werden alle Berechnungen der eigentlichen Fahrzeugsimulation und MPC ausgeführt und im letzten Schritt werden die grafische Elemente erstellt und angezeigt. Wie für eine

Spieleengine 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, welche die Fenstergröße in Pixeln festlegen (in x- und y-Richtung) und zusätzlich eine Angabe wie viel Metern dieser jweils in Pixeln entspricht. Die daraus resultierenden Verhältnisse

$$scaleX = \frac{windowSizeXinPixel}{windowSizeXinM} \tag{5.3.1}$$

$$scaleY = -\frac{windowSizeYinPixel}{windowSizeYinM} \tag{5.3.2}$$

werden verwendet, um alle Größenverhältnisse einheitlich in der Simulation zu halten und eine realistische Visualisierung zu gewährleisten. Durch die Parameter kann nun bequem die Größe des Bereichs, in dem der Rennkurs abgesteckt wird und die Fenstergröße, zur Darstellung der Simulation, angepasst werden. Zusätzlich wurde ein Offset eingeführt welcher die Nullpunkt 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 und setzt 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-Alogrithmus 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 Δt von $\frac{1}{20}s$ für das Rennauto berechnen zu können. Zum Testen der Algorithmen wurde ein beliebiger Kurs definiert, welcher sich an die Vorgaben des Regelwerkes hält und damit einen minimalen Kurvenradius von 9 m, Maximallänge einer Geraden von 80m

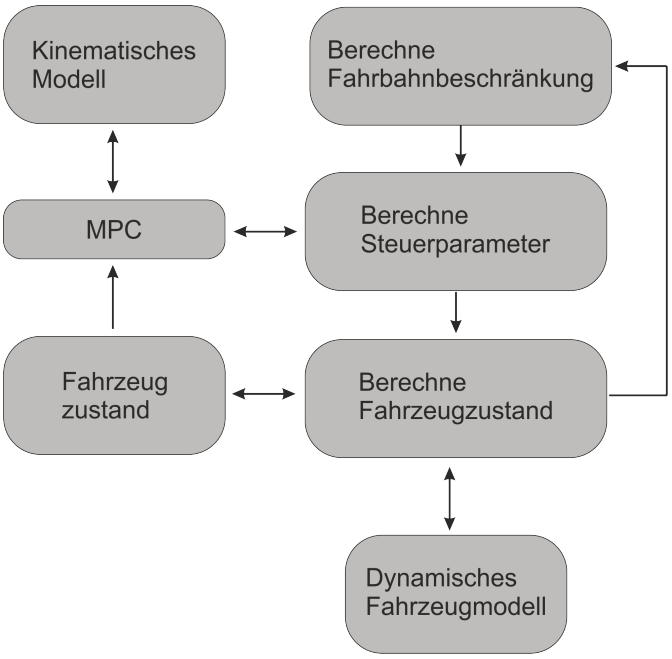

Abbildung 5.5: Blockdiagramm der Simulation mit dem MPC-Algorithmus

und maximal 180° Spitzkurven besitzt. 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 Modellfehler

Ohne eine genaue 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 Motorspors als Vergleichsbasis genutzt.

6.1 Acceleration

In dieser Disziplin ist es das Ziel der Teams eine 75m 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 aus den Fahrzeugdaten, wie in 3.3.6 beschrieben, errechnet wird. Das zweite untersuchte Modell basiert auf der Leistung des Motors und der maximal übertragbaren Kraft der Reifen (3.3.39). Das dritte Modell betrachtet zusätzlich zur Motorleistung noch Reibung und 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 118 kmh, welche durch die Übersetzung im Getriebe limitiert wird. Das kinematische Modell trifft mit seiner durchschnittlichen Beschleunigung exakt die gefahrenen Testwerte. Da aber die dynamischen Modelle eine größere Anfangsbeschleunigung haben und vom kinematischen Modell, hinterlegt im MPC, geregelt werden, wird die Beschleunigung im kinematischen Modell so angepasst, dass es die gleiche Anfangssteigung besitzt. Die Steigung beträgt 24 $\frac{m}{s^2}$. Als nächstes wird die Zeit über die Strecke von 75 m gemessen. Das Ergebnis ist für das kinematische Modell mit angepasster Beschleunigungskurve 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 und Reifentemperatur. Das dynamische Modell welches keinerlei Reibung berücksichtigt, ist ebenfalls mit 3.45 Sekunden zu schnell. Mit Reibung entsteht ein sehr realistischer

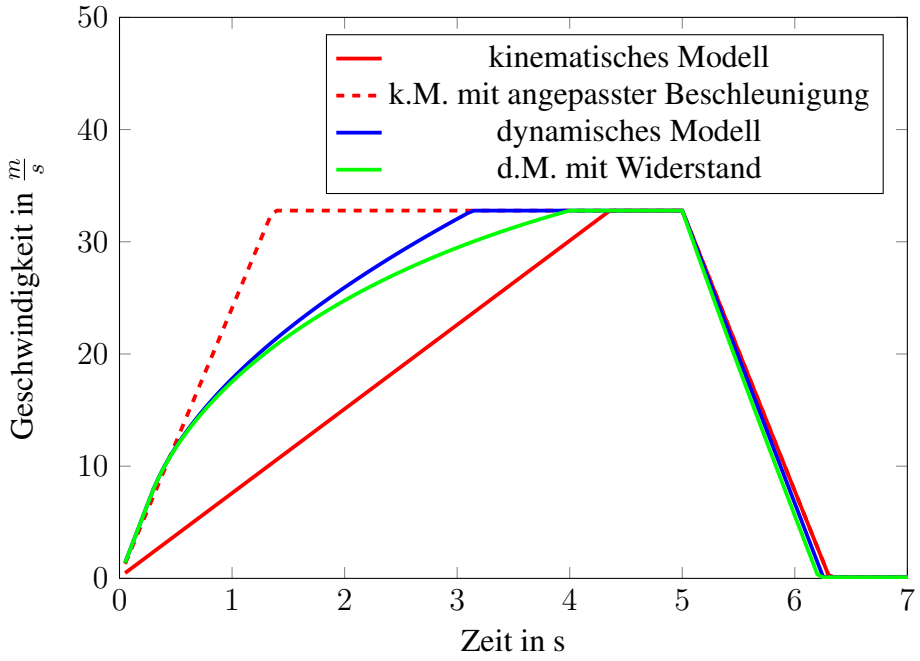

Abbildung 6.1: Beschleunigung für verschiedene longitudinale Modelle

Wert von 4.3 Sekunden was typischen Messzeiten für das Fahrzeug des High Octane Motorspors e.V. entspricht. Für ein gutes longitudinales Modell des Fahrzeugs muss also der Luftwiderstand, der einen deutlich größeren Anteil auf die Gesammtreibung hat als der Rollwiderstand, berücksichtigt werden. Im Folgenden wird daher immer das um Reibung erweiterten longitudinalen Modell verwendet.

6.2 Skidpad

In dieser Disziplin fährt das Fahrzeug eine liegende 8. Der limitierende Faktor ist hier also die maximale Querbeschleunigung, welche das Rennauto noch auf dem Kurs hält. 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 3.3.1 mit der Simulation übereinstimmt, kann durch die wirekende Zentripetalkraft überprüft werden.

$$\omega = 2\pi f \tag{6.2.1}$$

$$v = r * \omega \tag{6.2.2}$$

$$a = \frac{v^2}{r} \tag{6.2.3}$$

$$\tag{6.2.4}$$

Für das in der Simulationsumgebung hinterlegte kinematische Modell ist die gemessene Rundenzeit 4.65 s. Daraus lässt sich eine Durchschnittsgeschwindigkeit $v = 13.9\frac{m}{s}$ berechnen, was einer Querbeschleunigung von $a = 19.37\frac{m}{s^2}$ entspricht. Das im Fahrzeugmodell hinterlegte a_{max} ist mit 19.62 $\frac{m}{s^2}$ also sehr nah an dem in der Simulation erfahrenen werten. Die Differenz zu der schnellsten gemessenen Runde mit dem echten Fahrzeug (4.97 s) lässt sich dadurch erklären, dass der MPC-Algorithmus die ideale Stellgrößen berechnet und damit auch einen idealen Rundkurs abfährt. Zudem wurde die maximal gemessene Querbeschleunigung des Fahrzeugs als a_{max} gewählt, dies kann in der Realität nicht durchgängig gehalten 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 Models.

TODO Liegt an der Radlastverteilung in der Kurve dass beide Reifen zusammen nicht so viel Kraft wirken können!!

Um nachzuvollziehen, wodurch diese niedrige Rundenzeit entsteht, wurde überprüft, wie sich das dynamische Modell in hintereinander folgenden Kurven verhält. Dieses deutlich dynamischere Szenario hängt nicht nur von den, im Skidpad konstanten, lateralen Kräften ab. Ganz entscheidend ist für dies Fahraufgabe das Trägheitsmoment.

6.3 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 $\frac{m}{s}$ ohne jegliche longitudinale Beschleunigung getestet. Dies soll einen guten Überblick über den gesamten Geschwindigkeitsbereich ermöglichen.

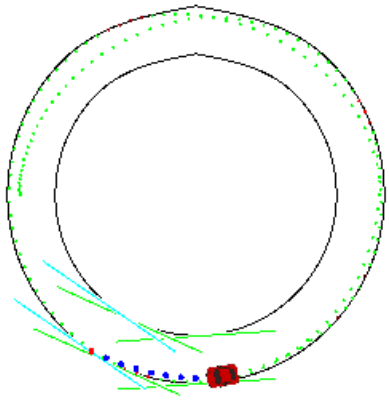

Abbildung 6.2: Rundkurs zum messen der maximalen Querbeschleunigung

Zuerst wird das Verhalten des kinematischen Modells betrachtet. Bei der geringsten Geschwindigkeit von 7 $\frac{m}{s}$ 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 $\frac{m}{s}$ ist dies deutlich zu erkennen, der Anfangspunkt der Kurve ist noch leicht abgerundet. Dies ist bei 21 und 28 $\frac{m}{s}$ nicht mehr der Fall. Ab der Geschwindigkeit von 14 $\frac{m}{s}$ 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 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. Aus dem Verlauf der Diagramme wird jedoch deutlich, dass das Trägheitsmoment dazu führt, dass die maximale Orientierung deutlich dem kinematischen Modell und damit dem Lenkwinkel nacheilt. Ebenfalls ersichtlich ist, wie das Fahrzeugmodell bei höheren Geschwindigkeiten immer mehr zum Übersteuern neigt und die maximale Orientierung immer weiter steigt.

Wird nun das gleiche Szenario aber für eine zusätzliche longitudinale Beschleunigung betrachtet, ergibt sich ein gänzlich neues Bild (s.Abb. 6.4). Zuerst lohnt sich nochmals der Blick auf das kinematische Modell. Hier ist nun sehr gut zu sehen, wie mit

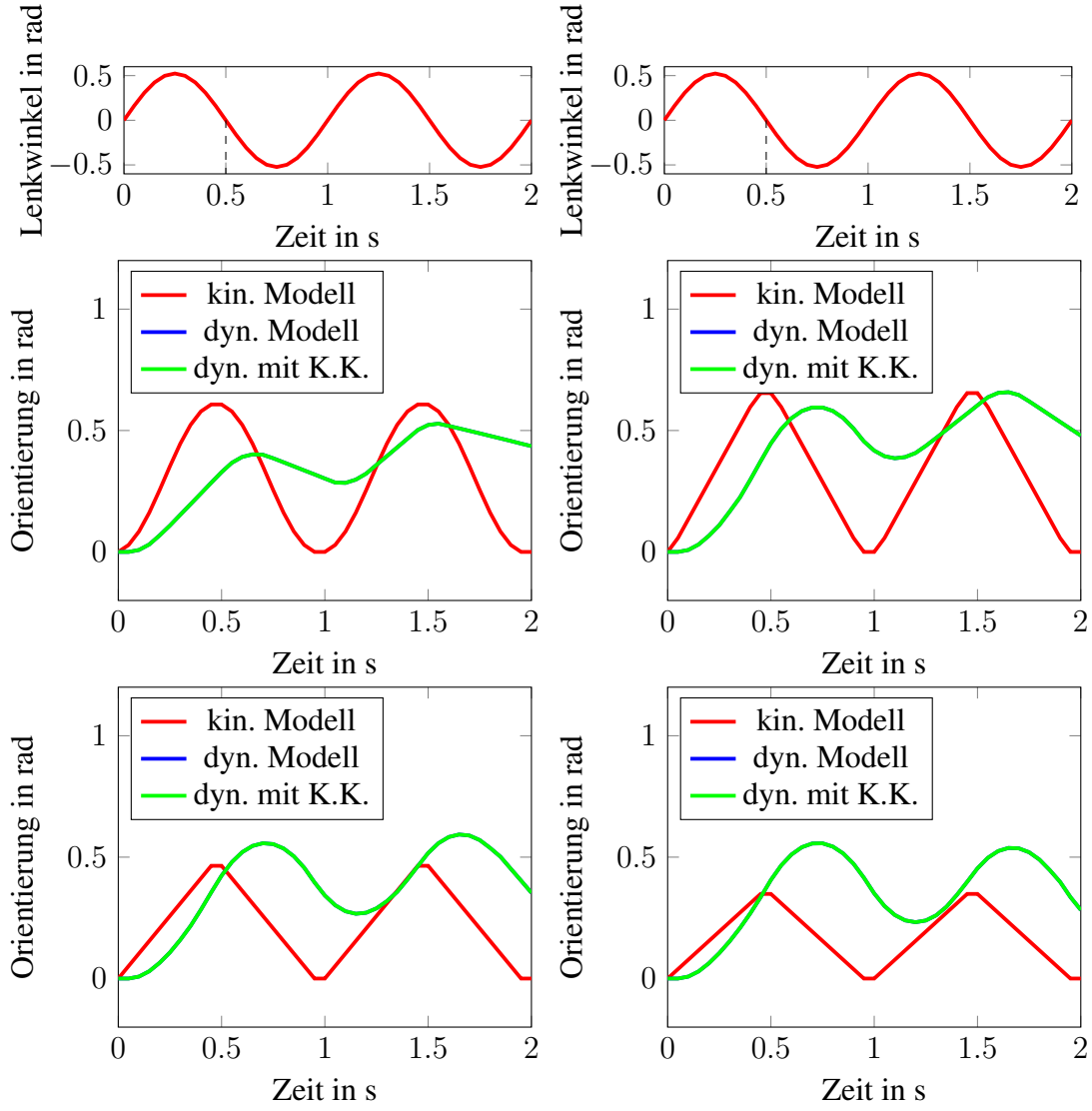

Abbildung 6.3: Orientierung des Rennautos für eine Sinuskurve als Eingangswert des Lenkwinkels. Es werden die Geschwindigkeiten 7,14,21 und 28 $\frac{m}{s}$ betrachtet.

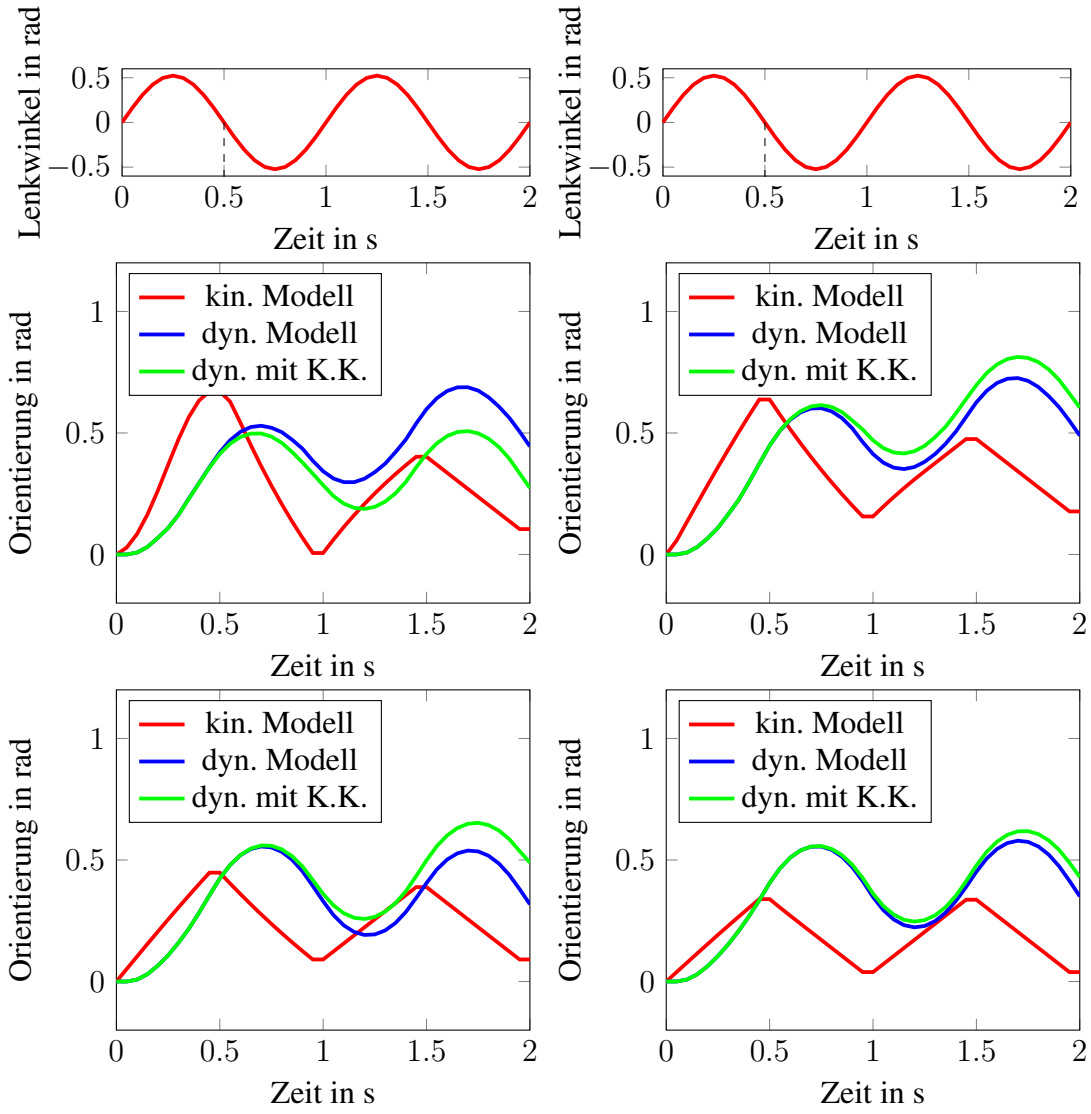

Abbildung 6.4: Die gleiche Betrachtung wie in 6.4 mit maximaler longitudinaler Beschleunigung.

zunehmender Geschwindigkeit die Kurve immer weniger rund ist und durch die maximale Querbeschleunigung begrenzt wird. Das Verhalten der beiden dynamischen Modelle ist in diesem Szenario nicht mehr kongruent. Für 7 $\frac{m}{s}$ ist die große Differenz klar durch die eingeschränkten lateralen Kräfte durch den K.K. ersichtlich. Dass die Differenz bei dem letzten Diagramm mit 28 $\frac{m}{s}$ nahezu verschwindet, ist darauf zurück zu führen, dass F_x welches mit der Gleichung 3.3.39 $F_{lr_{acc}} = \frac{P_{engine}}{|\dot{x}|}$
berechnet wird, mit zunehmender Geschwindigkeit immer kleiner wird.

6.4 Reifenmodelle

Wie bereits im Kapitel 3.3.2 wird für die Simulation standardmäßig das genauste Berechnungsmodell genutzt. Im Folgenden soll nun kurz untersucht werden, was für einen Auswirkung die drei verschiedenen Modelle auf das Fahrzeugverhalten haben. 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 sehr großen Einfluss auf das Verhalten des Fahrzeugs besitzt. Der Unterschied beträgt im Maximum 0.2 rad zwischen dem vereinfachten und dem genauesten Modell. Ob die Differenz auch einen Einfluss auf die Regelung des dynamischen Modells hat, wird im Abschnitt 8 näher beleuchtet.
Ausgehend von der Messung sollte, solange das Einsparen von Rechenzeit nicht das oberste Ziel ist, immer das genaueste 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 zum vereinfachten *magic model*. Es liefert eine gute Annäherung und 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.

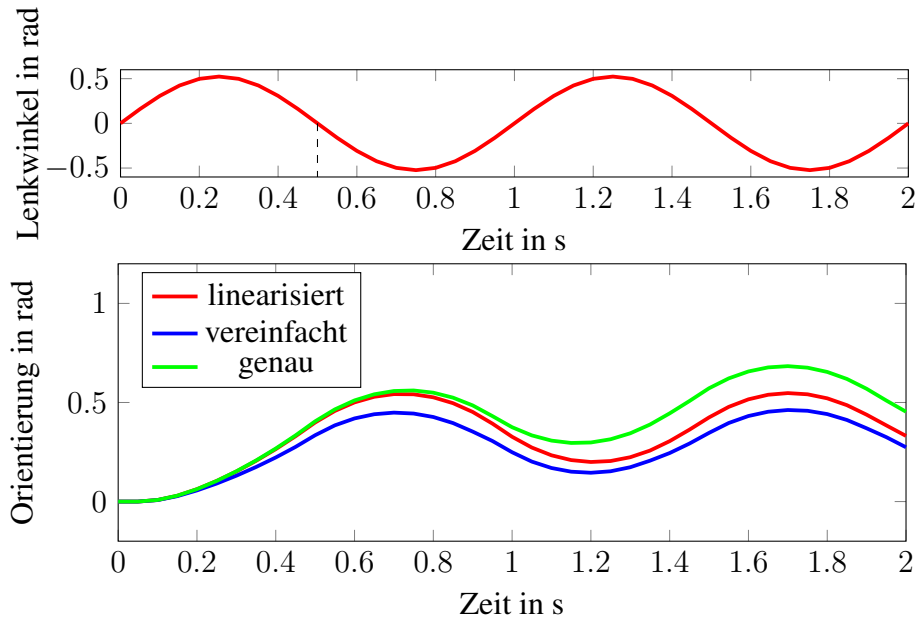

Abbildung 6.5: Auswirkung der drei Reifenmodelle auf die Orientierung des Fahrzeugs.

7 Regelung des kinematischen Modells

In diesem Kapitel wird die Regelung des im Simulator hinterlegten kinematischen Modells mit Hilfe des MPC-Algorithmus untersucht. Zuerst wird untersucht werden 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.

7.1 Prädiktionshorizont

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, da nicht davon auszugehen ist, 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 soll der minimale Prädiktionshorizont gesucht werden, der nötig ist um das Fahrzeug sicher um den Kurs zu fahren. Um diesen Wert zu finden, wurde eine Teilstrecke definiert die aus einer 80 Meter langen Gerade direkt in zwei aneinander gehängte $180\check{r}$ Kurven mit 9 m Innenradius übergeht. Diese Konstellation entspricht der Fahrsituation die nach Regelwerk zulässig ist und das Fahrzeug sowohl longitudinal (bremsen) und 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 118km/h auf der Graden und eine maximalen Kurvengeschwindigkeit von 32km/h. Für die maximiere Geschwindigkeit (m.G.)-Kostenfunkt wird ein anderen Horizont benötigt als für die Kostenfunktionen welche auf ein virtuelles Ziel zu fahren. Der Algorithmus lenkt in Kurven nicht direkt so ein, dass das Fahrzeug am inneren der Kurve entlangfährt sondern fährt möglichst lange geradeaus um die maximale Geschwindigkeit zu erreichen.

Der minimale Horizont ist daher im Vergleich zu den beiden Funktionen 8 Zeitschritte länger und damit mindestens 20.

7.2 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.

7.2.1 Verifizierung der Funktionen

Bevor die Messungen für die einzelnen Kostenfunktionen erstellt werden, soll überprüft werden, dass die Funktionen auch fehlerfrei funktionieren. Auf einer im Simulator definierten 75m Geraden wird überprüft, ob die Zeit zum Abfahren dieser Strecke für alle Funktionen gleich ist. Abweichungen könnten entstehen, wenn die Zielposition 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 Kostenfunktion welche in jedem Zeitschritt versucht die Geschwindigkeit zu maximieren. Ebenfalls wurde überprüft 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 ideal Beschleunigt und den 75 m Kurs in der gleichen Zeit absolviert.

7.2.2 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 Prädiktionshorizont abhängt. Die Ergebnisse der Messung sind in der Abbildung 7.1 visualisiert. Bei einigen Kostenfunktionen dauert die Berechnung des initialen Optimierungsvektor, im Vergleich zu allen darauf folgenden Schritten, extrem lange. Dies liegt an dem sogenannten *hot start*. 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 kann ist analog zu einem Schieberegister, in dem die Lösung einen Schritt nach links

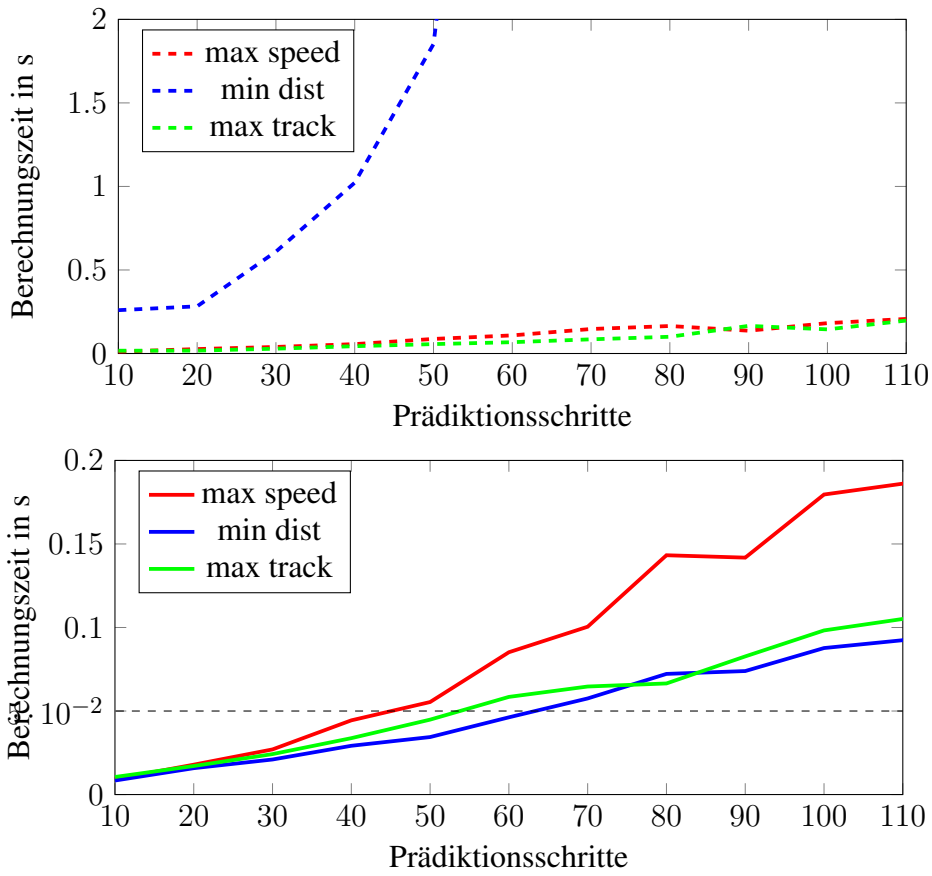

Abbildung 7.1

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 *hot start* durchgeführt wird. Da sich die ersten 5-10 Schritte die geplante Trajektorie Die Berechnungszeit der maximiere Distanz (m.D.)-Funktion ist den anderen beiden überlegen. Die ersten Schritte werden nahezu mit der gleichen Geschwindigkeit berechnet wie der Rest, sie besitzt zudem zusammen mit der min_dist Kostenfunktion die beste Laufzeit und skaliert auch besser mit den Prädiktionsschritten als die max_speed Funktion.

Im späteren Verlauf wird die Verwendung der elastischen Begrenzungen nötig. Um eine Vorauswahl zu treffen, welche Kostenfunktionen hierfür in Betracht gezogen werden sollen, wird im Folgenden die gleiche Messung wie für die normalen Kostenfunktionen durchgeführt. Die Messungen werden mit dem kinematischen Modell im Simulator vorgenommen, um möglichst gute Vergleichbarkeit zu erhalten.

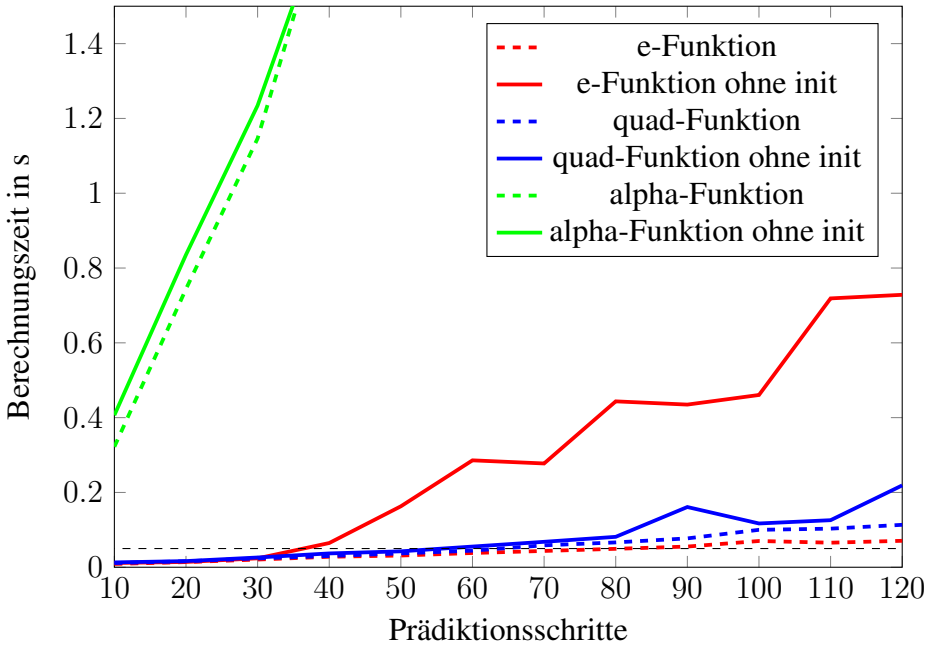

Abbildung 7.2: Berechnungszeit für verschiedene Kostenfunktionen mit elastischer Beschränkung.

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

Die deutlich höhere Ausführungszeit ist bei der alpha-Funktion darauf zurückzuführen, dass jeweils auf der Rückseite der Erhöhung die Kosten wieder gegen null fallen. Der Optimierer muss also einen sehr großes Suchraum betrachten. Es werden auch mögliche Trajektorien evaluiert, welche 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.

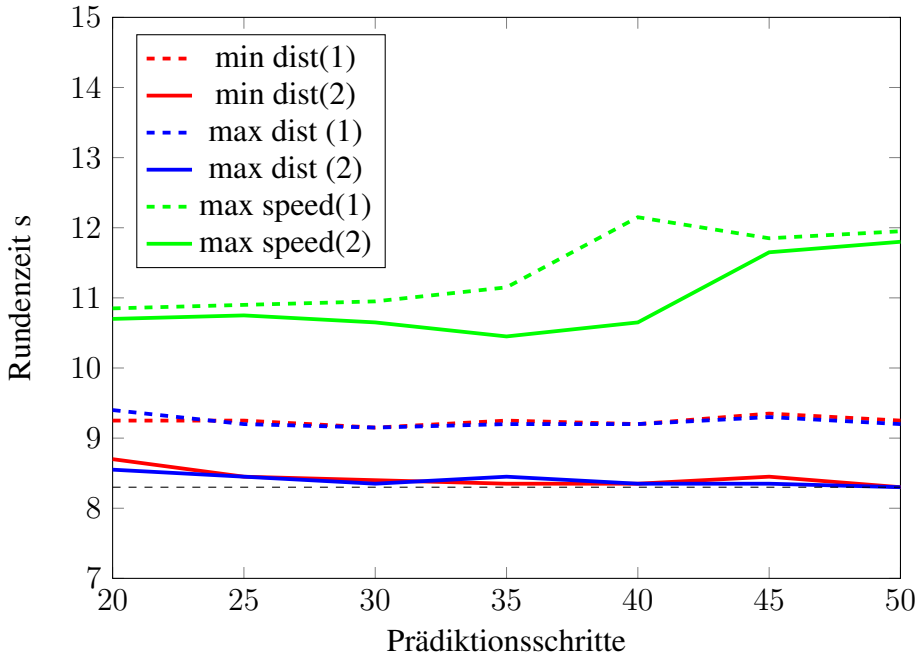

Abbildung 7.3

7.2.3 Rundenzeit

Um die Qualität der Kostenfunktionen einschätzen zu können, wird im folgenden Untersucht, wie schnell die einzelnen Rundenzeiten, abhängig von den Prädiktionsschritten, für die einzelnen Funktionen sind. Gemessen wird immer die zweite Runde, 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 7.3 visualisiert.

Das Verhalten, der Kostenfunktion welche die Maximalgeschwindigkeit in jedem Schritt maximiert, mit längerer Vorwärtsprädiktion immer langsamer zu werden, lässt sich mit dem Wissen über den Kurs 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 leider 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. Betrachtet man zusätzlich die Berechnungszeit setzt sich die Kostenfunktion zur

Maximierung der Streckendistanz 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.

8 Regelung des dynamischen Modells

In der Realität ist das kinematische Modell nur für geringe Geschwindigkeiten genau genug 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 zu evaluieren, wie gut sich das deutlich komplexere System mit dem weiterhin im MPC-Algorithmus hinterlegte, sehr einfache kinematischen Modell, regeln lässt. Die deutlichen Unterschiede der Modelle wurden bereits im Kapitel 6 evaluiert. Für diesen Abschnitt, wird nun von dem Fahrzeugmodell ausgegangen, welches Reibung und Luftwiderstand berücksichtigt, durch den K.K. erweitert wird und die Reifenkräfte mit dem genausten *magic model* errechnet.

8.1 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 Beschräunkung basierend auf der e-Funktion gewählt und ein sehr hoher Kostenfaktor eingestellt um sicherzustellen, dass das Fahrzeug nicht den Kurs verlässt um die Strecke abzukürzen. Erst wenn der Regler die Kontrolle verliert und das Fahrzeug nicht mehr auf dem Rennkurs halten kann ist die maximal mögliche Querbeschleunigung erreicht.

8.1.1 Dynamisches Modell

Für dieses Anwendugnszenario wird zuerst das dynamische Modell evaluiert um eine Vergleich zum Modell, welches durch den K.K. erweitert wurde anstellen zu können. Der in Abbildung **??** dargestellte Graph zeigt den Zusammenhang aus maximaler Querbeschleunigung und Prädiktionsschritten an. Die beste erreichte Zeit 11.6 s ist entgegen der Erwartung für einen sehr 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 vorbei führt. Dadurch sind

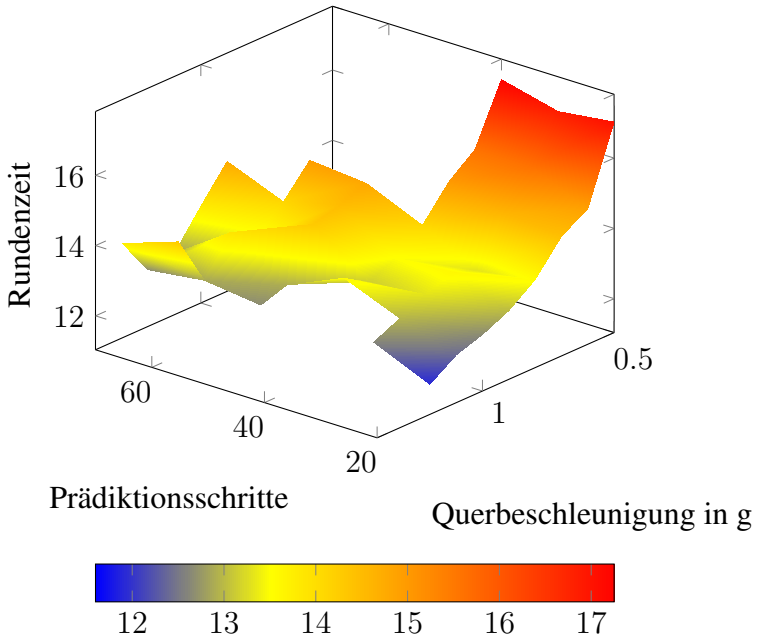

Abbildung 8.1

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 $1.4g$.

Die Rundenzeiten wurden jeweils für die zweite Runde gemessen.

Nachdem die erste Messung nur recht grob durchgeführt wurde, ist in der Grafik 8.2 der Bereich von 20-30 Prädiktionsschritten genauer untersucht worden.

8.1.2 Kammsches Modell

Die beste Rundenzeit welche für das dynamische Modell mit hinterlegtem K.K. erreicht wurde, liegt bei 11.65 Sekunden. Dieser Wert wurde für einen Prädiktionshorizont von 21 Schritten bei der maximalen Querbeschleunigung von 1.2g erreicht. Dieses Ergebnis überrascht im Hinblick auf die deutlichen Unterschiede der Modelle. Das dynamische Modell welches in Situationen, in denen stark beschleunigt und gebremst wird deutlich höhere Kräfte übertragen kann, kann sich in der schnellsten Runde nur 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

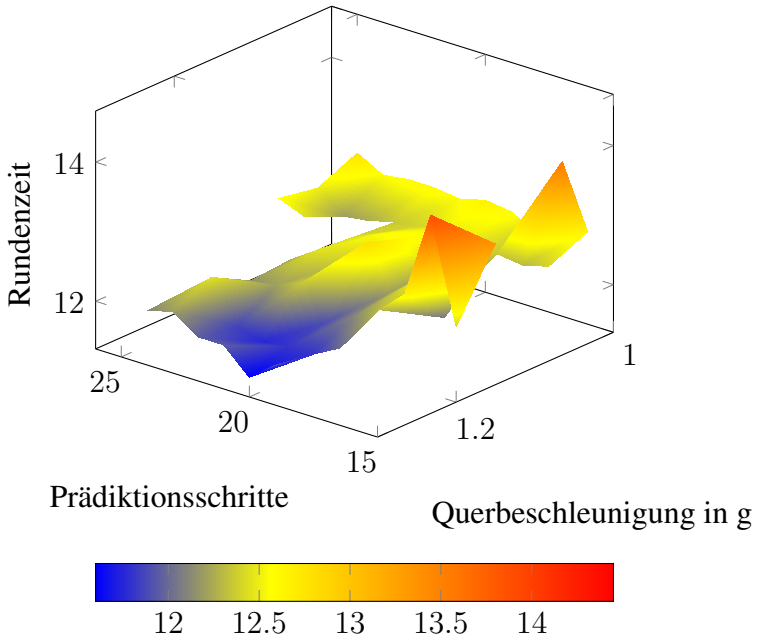

Abbildung 8.2

anfängt, um in der Kurve möglichst nah an der inneren Kurve zu bleiben. Die wirkenden Kräfte sind dadurch in der Grenzsituation beim Einfahren in die Kurve so klein, dass die Maximalkräfte welche durch den K.K. beschränkt werden, nur geringfügig auftreten.

8.2 Einfluss des Reifenmodells

Der Einfluss des Reifenmodells soll auch kurz untersucht werden. Hierfür wird die Grenzsituation, welche im vorherigen Abschnitt gefunden wurde, gewählt, um die Auswirkung der verschiedenen Reifenkräfte zu überprüfen.

Das Ergebnis der Messung entspricht dem erwarteten Ergebnis. Die Gleichung welche das Ursprüngliche *magic model* vereinfacht, fällt nach dem Erreichen seines Maximalwertes für die lateralen Kräfte deutlich stärker ab als die beiden anderen Modelle (siehe 3.10). Dieser Umstand macht sich auch bei den Rundenzeiten bemerkbar, das Fahrzeugmodell welches auf das vereinfachte Reifenmodell zurückgreift ist durchgängig langsamer als die anderen beiden. Da nur in dem Fall der Rennreifen die linearisierte Version gute Ergebnisse liefert, ist davon abzuraten nicht das genauste Reifenmodell welches zur Verfügung steht zu nutzen.

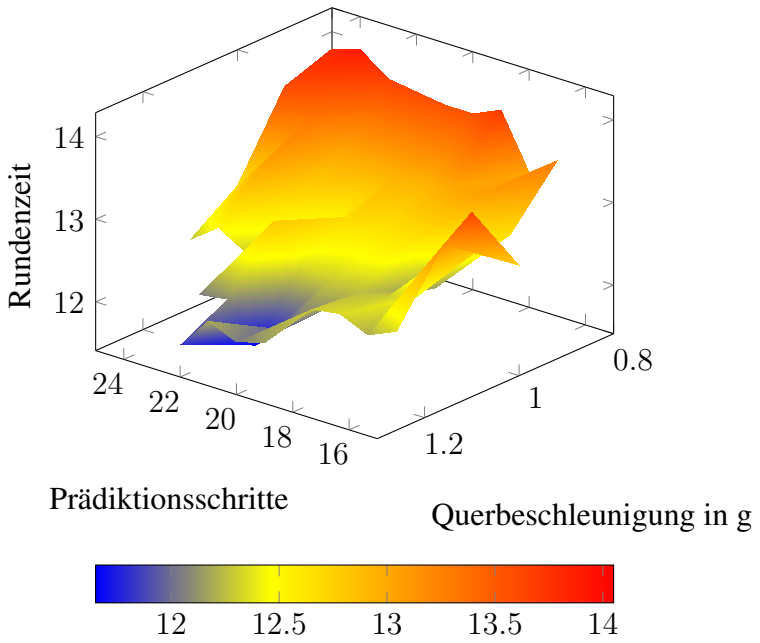

Abbildung 8.3

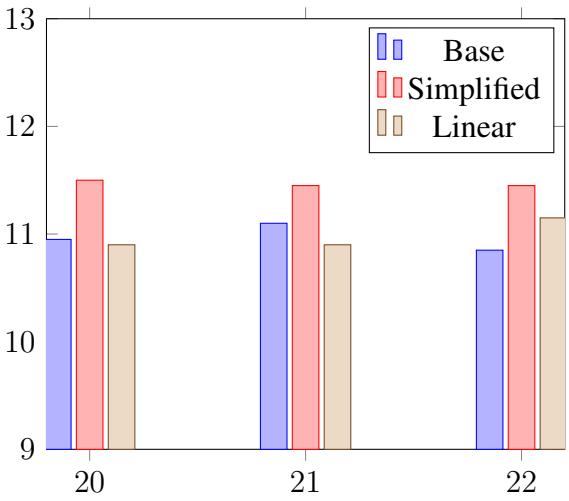

Abbildung 8.4

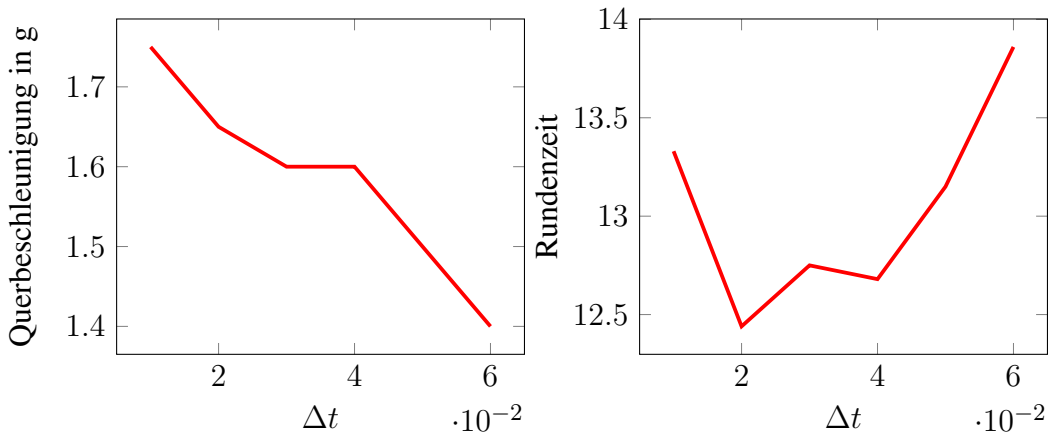

Abbildung 8.5

8.3 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 auszubrechen. Bei Abweichungen des Fahrzeugs von der erwarteten Trajektorie sollte mit einer höheren Aktualisierungsrate weniger Zeit bis zu einer neu geplanten Trajektorie vergehen. Unabhängig davon, kann ein Fahrzeug, welches zu schnell in die Kurve fährt und nicht genug laterale Kräfte übertragen kann um auf der Kreisbahn zu bleiben, auch durch die schneller geplante Trajektorie nicht auf dem Kurs gehalten werden. Um eine gute Vergleichbarkeit zu erzielen, wird für jedes Δt 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 Δt besitzt, wurde der nächst mögliche Wert gewählt.

8.4 Quadratische Funktion

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 α und das Verhältnis aus a und b der Regler die besten Rundenzeiten liefert ohne den Kurs zu verlassen. Für N wurde ein konstanter Wert von 25 Schritten mit einem Δt von 0.05s 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 Berechnugnszeit, mit dieser Kostenfunktion nicht zu empfehlen. Die problematische Stelle im Kurs ist schematisch für die zwei verschiedenen Verhalten für die quadratische und die exponentielle Kostenfunktion dargestellt.

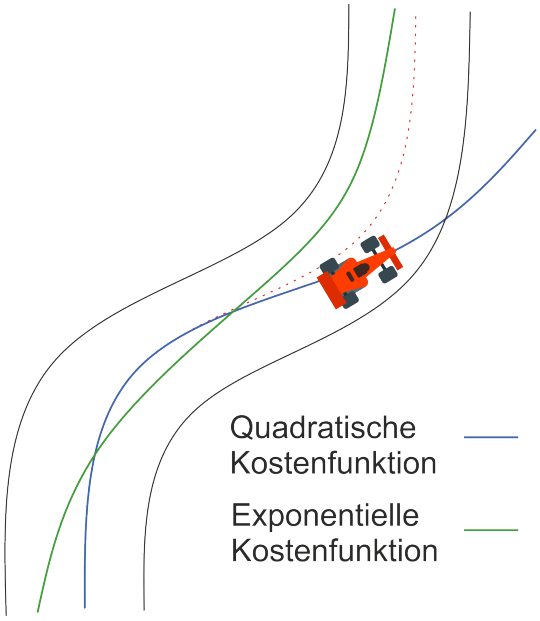

Abbildung 8.6: Platzhalter

9 Zusammenfassung und Ausblick

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

9.1 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 Fahrzeug bilden. Es handelt sich dabei um ein hoch performantes Rennauto, welches für niedrige Geschwindigkeiten (bis 118 kmh) und hoche 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 Modell, ein dynamisches Fahrzeugmodell welches die Reifenkräfte mit betrachtet, 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.

9.2 Ausblick

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

9.2.1 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 alle vier Räder geändert wird. 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 werden für die Übergange zwischen der Geraden und dem Kreis Klothoiden genutzt. Das Ergebnis sind Kurven die ruckfrei bzw. ihre Krümmung eine steige Funktion der Länge sind. Eine Vereinfachung welche in der Realität auch eine entscheidende Rolle spielt ist die Änderungsrate des Lenkwinkels. Im Simulator kann das Fahrzeug beliebig schnell zwischen Volleinschlag links und Volleinschlag rechts wechseln. Dies ist natürlich im echten Rennauto nicht möglich. Ebenfalls vorstellbar wäre die Berücksichtigung der Bewegung des Fahrzeugs während die Regelparameter vom MPC-Algorithmus berechnet werden. Die Stellgrößen beziehen sich auf eine Position welche nicht mehr der echten entspricht wenn die Berechnung abgeschlossen ist.

9.2.2 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 würden 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 Horizont bedeutet, da die

Komplexität der Berechnung steigt. Die Messungen in Kapitel 7.2.3 haben jedoch ergeben, dass der größere Prädiktionshorizont weniger Einfluss auf die Rundenzeit als das Modell hat.

9.2.3 Zweispurmodell

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

9.2.4 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 noch kein Framework wie Jump für C++, welches die Verwendung des Optimierers und die Automatische Ableitung kombiniert. Diese zwei Bestandteile müssen also manuell integriert 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 implementiere. Wie der Benchmark 5.2 in Kapitel 5.2 jedoch gezeigt hat ist eine höhere Performance von einer C-Nahen Implementierung zu erwarten.

9.2.5 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 werden 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

Elektronischer Anhang

Fahrzeugdaten

Abbildung A1: Engine Power

Engine speed	450SXF_2_Restriktor.s	450SXF_2_Restriktor.sum
rpm	hp	N*m
2000,0030517578	8,3819561005	29,8436508179
2500,001953125	10,3697595596	29,5369606018
3000,001953125	13,8593196869	32,8971099854
3500,00390625	16,6403408051	33,8556404114
3999,9990234375	18,1146297455	32,2482910156
4499,994140625	18,2581005096	28,8922195435
4999,998046875	24,487859726	34,8753318787
5500,001953125	26,7614707947	34,6484909058
6000,0009765625	32,7806510925	38,9048194885
6500,001953125	33,4750900269	36,6729202271
7000	36,7797889709	37,4152297974
7500	43,285118103	41,0974311829
8000	46,9468193054	41,7881698608
8499,9990234375	49,3668899536	41,3574790955
9000	51,845741272	41,0211410522
9499,9990234375	53,4694099426	40,0792007446
9999,9990234375	54,3385696411	38,6941604614
10500	55,5974998474	37,7053718567
11000	57,0893096924	36,9572181702
11500	57,8139610291	35,7990989685
12000	57,2001113892	33,9431991577
12500	54,5902709961	31,0987205505
13000	51,3287887573	28,1160907745

Einbindung Grafik im Anhang

A.0.1 Code

```
[caption={ein paar Zeilen code}\label{lst:vehicleModel},captionpos=t]
    for i in 0: N-1

    @NLexpression(m, x_dd, (VehicleModel.max_long_acc * x[8*i + 7]/10.0

    @NLconstraints(m, begin
    x[(i + 1)*8 + 1] - (x[i * 8 + 1] + x[8*i + 3]*
    dt*cos(x[i*8 + 4] + atan(lr/(lf + lr) * tan(x[i*8 + 8])))) == 0
    x[(i + 1)*8 + 2] - (x[i * 8 + 2] + x[8*i + 3]*
    dt*sin(x[i*8 + 4] + atan(lr/(lf + lr) * tan(x[i*8 + 8])))) == 0
    x[(i + 1)*8 + 3] - (x[i * 8 + 3] + x_dd*dt) == 0
    x[(i + 1)*8 + 4] - (x[i * 8 + 4] + x[8*i + 3]*
    dt / lr*sin(atan(lr/(lf + lr) * tan(x[i*8 + 8])))) == 0
    atan(0.5 * (lf + lr) * mpc_struct.beta_max / x[i*8 + 3]^2)
```

Abbildung A2: Max Tire Force

Engine speed	450SXF_2_Restriktor.s	450SXF_2_Restriktor.sum
rpm	hp	N*m
2000,0030517578	8,3819561005	29,8436508179
2500,001953125	10,3697595596	29,5369606018
3000,001953125	13,8593196869	32,8971099854
3500,00390625	16,6403408051	33,8556404114
3999,9990234375	18,1146297455	32,2482910156
4499,994140625	18,2581005096	28,8922195435
4999,998046875	24,487859726	34,8753318787
5500,001953125	26,7614707947	34,6484909058
6000,0009765625	32,7806510925	38,9048194885
6500,001953125	33,4750900269	36,6729202271
7000	36,7797889709	37,4152297974
7500	43,285118103	41,0974311829
8000	46,9468193054	41,7881698608
8499,9990234375	49,3668899536	41,3574790955
9000	51,845741272	41,0211410522
9499,9990234375	53,4694099426	40,0792007446
9999,9990234375	54,3385696411	38,6941604614
10500	55,5974998474	37,7053718567
11000	57,0893096924	36,9572181702
11500	57,8139610291	35,7990989685
12000	57,2001113892	33,9431991577
12500	54,5902709961	31,0987205505
13000	51,3287887573	28,1160907745

Abbildung A3: Unterschrift Bild x Die auf die Rotationsfrequenz des Innenzylinders normierten Eigenfrequenzen der gefun-denen Grundmoden der Taylor-Strömung für h(Die azimutale Wellenzahl ist mit m bezeichnet.)

```
    - atan(lr/(lf + lf) * tan(x[i*8 + 8])) >= 0
#max_beta - beta
    atan(0.5 * (lf + lr) * mpc_struct.beta_max / x[i*8 + 3]^2)
    + atan(lr/(lf + lf) * tan(x[i*8 + 8])) >= 0
#max_beta + beta
    end)
    end
```

Abkürzungsverzeichnis

ROS	Robot Operating System
m.G.	maximiere Geschwindigkeit
m.D.	maximiere Distanz
K.K.	Kammscher Kreis
fps	Häufigkeit pro Sekunde
MPC	Model Predictive Control
FS	Formula Student
JIT	just-in-time
SLSQP	Sequential Least SQuarez Programming
SFML	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.

[BAS] Rechtsfolgen zunehmender fahrzeugautomatisierung. `https://www.bast.de/BASt_2017/DE/Publikationen/Foko/2013-2012/2012-11.html`. Accessed on 2018-06-22.

[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.

[HB15] Grischa Gottschalg Lukas Ortmann Henrik Bey, Lukas Brunke. MPC for Trajectory Planning of Race Cars with Obstacle Avoidance. 2015.

[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.

[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/`. Accessed on 2018-05-15.

[WDG+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.

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