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:

Am Lehrstuhl für Informationstechnik mit dem Schwerpunkt Kommuniktaionselektronik (LIKE)

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) [williams2016aggressive].

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.

- 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 Trajektionsplanung für ein eben solches Driverless-Racecar. Die Basis hierfür ist ein Model Predictive Control (MPC) Algorithmus. Er vereint die Regelung und Trajektionsplanung und ist sehr 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 kann die Regelung an die Grenzen des realen Fahrzeuges gehen. Neben der Auslegung für das aktuellste FS-Fahrzeug des High Octane Motorsports für die Driverless Umrüstung, wird untersucht ab welchem Punkt ein kinematisches Modell nicht mehr ausreicht um das Fahrzeug sicher auf dem Rennkurs zu führen.

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 bewegt. Das Jahr 2018 markiert dabei einen ganz besonderen Meilenstein. Ende des Jahres wird die Alphabet Tochter Waymo die ersten voll autonomen 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 den Fahrzeugen hinterlegt. Sollen die Fahrzeuge in unbekannten Umgebungen agieren, müssen sowohl die Sensorsysteme, wie auch die Algorithmen noch deutlich weiterentwickelt werden, bevor autonome Autos auf Stufe 5 frei verfügbar sind. Die Unterteilung dieser Level wurde von der Bundesanstalt für Straßenwesen zur Klassifizierung von selbst fahrenden Autos erstellt. Die Spanne geht von Stufe 0, keinerlei Assistenz beim fahren, bis Stufe 5, Vollautomatisierung. In diesen Fahrzeugen ist dann auch kein Lenkrad mehr vorgesehen. Die Forschung um dieses Ziel zu erreichen, wird nicht nur in den großen Automobilkonzernen betrieben. 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 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 verlangt. 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 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 hat. Die Messungen und Vergleiche beziehen sich damit auch immer auf einen Kurs, der so in einem FS-Event für die Driverless Fahrzeuge vorkommen könnte.

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 Trajektionsplanung und Regelung eines Formula Student 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 benötigt es hierfür ein Modell des zu kontrollierenden Systems 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.

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 kritische 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 (*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, benötigt es ein Modell, 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. Da die Modelle die Fahreigenschaften nicht nur möglichst genau abbilden müssen, sondern auch schnell berechenbar sein müssen, werden verschiedene Vereinfachungen angewandt. Die Ansätze variieren nach Komplexität und zu bewältigenden Fahrsituationen:

- Ein einfaches kinematisches Modell [[HB15]], welches sehr schnell berechenbar ist, aber keine dynamischen Effekte berücksichtigt.
- Die Verwendung einer dynamischen Fahrzeugbeschreibung mit linearem oder nichtlinearem Reifenmodell [[AAM], [PER16], [CUR18]].

- Ein anderer Ansatz ist es, nicht vom Schwerpunkt des Autos aus die Fahrzeugdynamik zu berechnen, sondern den *point of percussion* zu wählen. In diesem Punkt heben sich die rotatorischen und translatorischen Kräfte, hervorgerufen durch die Hinterreifen, gegenseitig auf [KG12].

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 verwirklicht werden [WDG+16]. Die Berechnungszeit für neue Steuerparameter sollte im Anwendungsfall eines autonomen Rennautos 40 bis 50 ms nicht überschreiten und bewegt sich in der Regel zwischen 50 und 100 Hz [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 Model Predictive Control (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, Trennung von longitudinalem und lateralem Regler [PER16], [CUR18] oder kombinierter Betrachtung [AAM].

3 Grundlagen

Um für das Fahrzeug ideale Trajektorien zu berechnen und gleichzeitig das Rennauto in Echtzeit zu regeln wurde der Model Predictive Control Ansatz gewählt. Dieses Verfahren basiert auf der Optimierung eines nicht linearen Programms.

3.1 Model Predictive Control

Bei MPC handelt es sich um einen Algorithmus, mit dem sich sehr komplexe, multivariable Regelungsprobleme lösen lassen. Er besitzt seine Stärken vor allem dort, wo Prozesse mit mehreren Ein-, und Ausgängen kontrolliert werden sollen und dabei verschiedene Beschränkungen eingehalten werden müssen. Vorausgesetzt es ist ein ausreichend genaues Modell des Prozesses vorhanden, können unter ausnutzen von Messungen und dem Modell 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.
- Einschränkungen der Eingangs- und Ausgangsgrößen werden berücksichtigt.
- Die Vorhersage des Fahrzeugzustands kann genutzt werden um mögliche Probleme frühzeitig zu detektieren.

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 der Ölraffinerie, zu steuern. Für diese Aufgabengebiete war Model Predictive Control 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.

Funktion

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

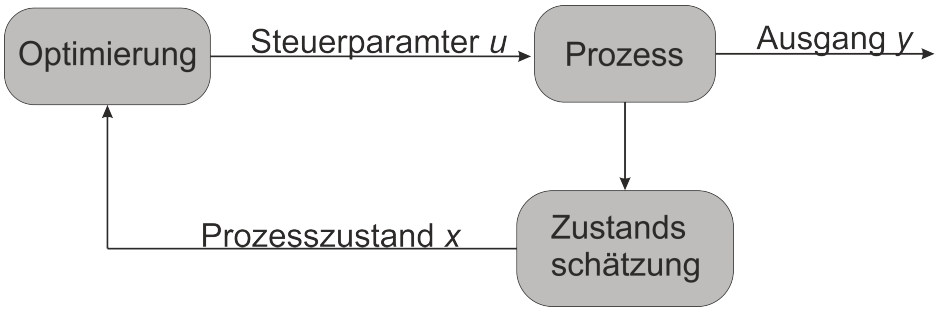

Abbildung 3.1: Block Diagramm zu MPC

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, Kostenreduktion, Temperatursteuerung, etc.
- Steuerung
  Die berechneten Steuerparameter werden auf dem realen System angewandt. Danach wird wieder beim ersten Schritt fortgefahren.

Die Mächtigkeit von MPC basiert auf der Prädiktion des Systemverhaltens. Ein präzises Vorhersagen setzt daher eine möglichst gute Systembeschreibung voraus. Je genauer diese an die Realität heranreicht, desto weiter in die Zukunft können die Systemzustände berechnet werden. Zudem kann das System näher an seine Systemgrenzen geführt werden. Bei Model Predictive Control handelt es sich um ein sogenanntes *finite-horizon* Verfahren, das heißt, die Optimierung wird immer für einen bestimmten Zeitraum in die Zukunft durchgeführt $[t_0, t_0 + T]$. Dieser Bereich T wird dann in k Schritte unterteilt, für die jeweils ein Prädiktionsschritt berechnet wird. Dieser geht aus dem vorherigen Zustand $k-1$ 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 Vorhersage Horizont wird der Steuerparameter berechnet, welcher den Systemzustand möglichst schnell der vorgegebenen Trajektorie annähert. Das wissen über das zu steuernde System hilft dafür die passenden Steuerparameter zu finden und frühzeitig genug die Amplitude so anzupassen dass kein Übersteuern entsteht. Es wird also deutlich, dass ein MPC-Algorithmus sehr gut an die Systemgrenzen regeln kann.

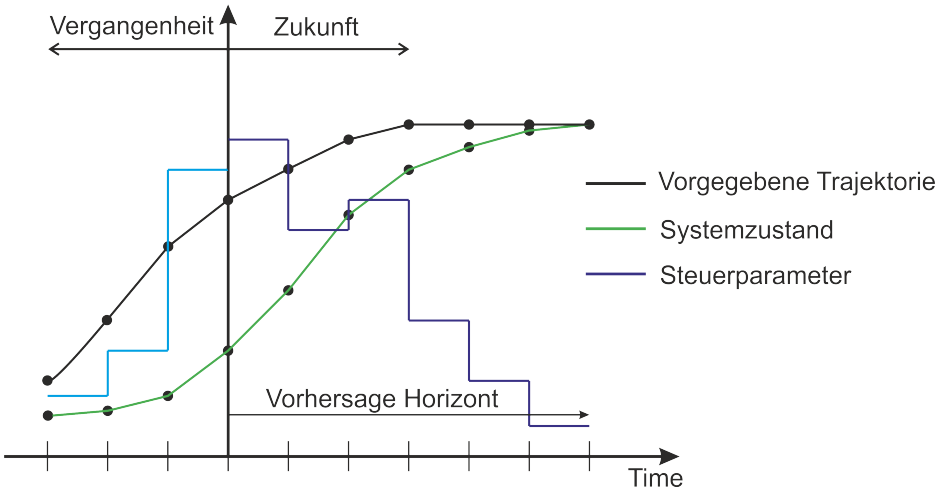

Abbildung 3.2: Funktionsprinzip von MPC: Die Kostenfunktion ist die Reduktion der Distanz zur Referenztrajektorie. Durch das Modell kann das System so geregelt werden dass es nicht über die Zieltrajektorie hinaus schießt.

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 Änderungsrate der Querbeschleunigung, welche ein direkter Indikator für den Komfort eines autonomen Fahrzeugs während der Fahrt darstellt.

Um den *Model Predictive Control* Algorithmus für den Anwendungsfall eines Rennautos zu implementieren, braucht man Folglich ein Fahrzeugmodell und einen Optimierungsalgorithmus. Auf beides wird in den nächsten Kapiteln eingegangen.

3.2 Optimierung

Das Konzept 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 / Verlust in einem Betrieb, Geschwindigkeit oder Distanz in einem physikalischen Problem oder der erwartete *return of investment* für eine Geldanlage. Die Bezeichnung lineare Programmierung bezieht sich auf die Lösung eines Optimierungsproblems und hat nichts mit dem eigentlichen Programm zu tun.

Die allgemeine mathematische Definition eines Optimierungsproblems ist

$$\begin{array}{ll} minimize & f(\vec{x}) \\ subject\ to & g_i(\vec{x}) \leq 0,\ i = 1,2,...,p \\ & h_j(\vec{x}) = 0,\ j = 1,2,...,m \end{array}$$

Der Eingabeparameter $\vec{x}$ sei aus $\Re^n$, das heißt, das Problem hängt von n Einflussparameter ab, die im Vektor $\vec{x}$ eingelagert sind. Die Zielfunktion $f : D \to \Re$ sei einmal stetig differenzierbar. Weiterhin sind die Nebenbedingungen in Ungleichheitsform $g_i : D \to \Re$ mit $1 \leq i \leq p$ und in Gleichheitsform $h_j : D \to \Re$ mit $1 \leq j \leq m$ gegeben. Wird $f(\vec{x})$ durch $-f(\vec{x})$ ersetzt, wird aus dem Mini-, ein Maximierungsproblem.

Ein Optimierungsproblem hat nicht immer eine Lösung. Die Ungleichungen, welche den Suchraum einschränken, können sich widersprechen (z.b. $x \leq 0$ und $x > 0$). In diesem Fall gibt es keine Lösung. Außerdem kann das Problem unbeschränkt sein, was unendlich viele zulässige Lösungen zur Folge hätte und damit auch als nicht lösbar eingestuft wird.

Ein lineares Programm (alle Funktionen sind linear) lässt sich in Matrixschreibweise darstellen. Es besteht aus $A \in \Re^{m,n}$ und zwei Vektoren $b \in \Re^{m,1}$ und $c \in \Re^{1,n}$.

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

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:
$min\{c^{\mathrm{T}}x|Ax \leq b, x \geq 0\}$

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

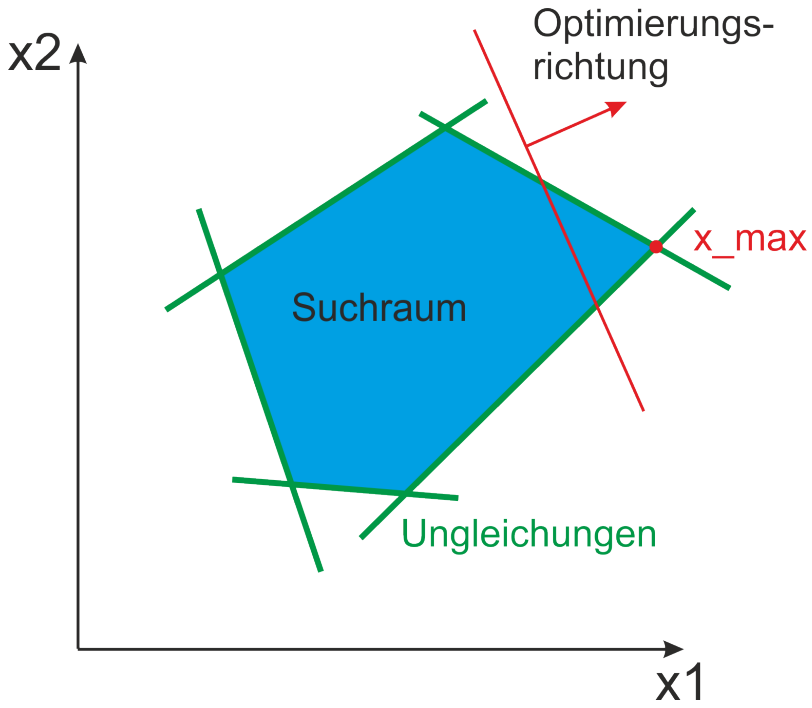

Abbildung 3.3: Lineare Optimierung

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

Lösungsverfahren

3.2.1 Simplex - Verfahren

Bei der in dieser Arbeit zu lösenden Optimierung, handelt es sich um ein nichtlineares Problem. Um ein besseres Verständnis für das Lösungsverfahren zu bekommen, welches hierfür genutzt wird, soll zuerst das Simplex-Verfahren kurz vorgestellt werden. Das Verfahren wurde 1947 von Dantzig [Dan51] entwickelt und nutzt die Eigenschaft, dass ein *well behaved* (wenn das aufgestellte Problem der Standardform entspricht und lösbar ist) lineares Programm immer einen Knoten besitzt an dem sich Kanten verschiedener Ungleichheitsbedingungen treffen und die Kostenfunktion minimieren. Das Verfahren nutzt diesen Zustand aus, in dem es sich iterativ von einem Eckpunkt zum nächsten bewegt.

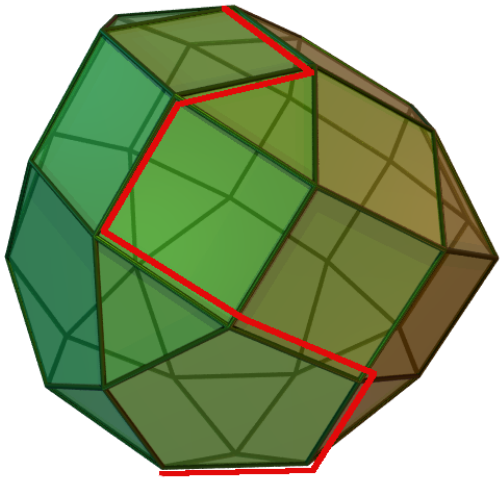

Abbildung 3.4: Simplex Methode: Ablaufen des Suchraums entlang der Kanten der Ungleichheitsbedingungen von Knoten zu Knoten

Die Richtung, welche gewählt wird, hängt von der Flanke (Ungleichung) ab, welche die Kostenfunktion weiter minimiert. Kann keine weitere Flanke gefunden werden, welche die Kosten weiter reduziert, endet das Verfahren im Optimum. Da im schlechtesten Fall der Simplex-Algorithmus jeden Knotenpunkt einmal besuchen muss, hat das Verfahren ein exponentielles Wachstum $\mathcal{O}(c^{\mathrm{n}})$, abhängig von der Dimension des Optimierungsproblems [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 [sim].
Die Suche nach noch besseren Verfahren, dessen 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.

3.2.2 Innere-Punkte-Verfahren

Im Folgenden wir das Innere-Punkte-Lösungsverfahren für lineare, quadratische und nichtlineare Optimierungsprobleme kurz vorgestellt. Die Idee hinter dem Verfahren ist es, sich nicht mehr auf den Kanten zu bewegen, sondern immer im Inneren des erlaubten Bereiches. Diese Bewegung durch das Innere des Polytop ist in Abbildung 3.5 veranschaulicht.

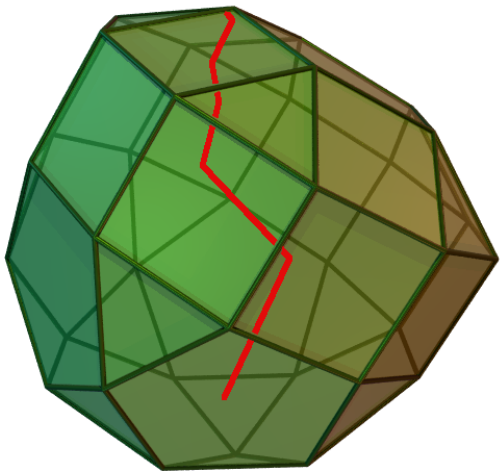

Abbildung 3.5: Beim Innere-Punkte-Verfahren bewegt sich der Optimierer durch das Innere des Polytop auf der Suche nach dem Mimimum.

Um dieses Verfahren umzusetzen wird die Positivitätsbedingungen $x \geq 0$ der Standardform durch einen logarithmische Strafterm $-\mu \ln x_i$ ersetzt und in die Kostenfunktion integriert. Der Faktor μ wird genutzt um die Gewichtung des Strafterms zu verändern. Die daraus resultierende Funktion wird als Barrierefunktion bezeichnet und ist definiert als $B(x,\mu) = c^{\mathrm{T}}x - \mu \sum_i^n \ln x_i$, mit $\mu > 0$.

Das ursprüngliche Optimierungsproblem wird damit ersetzt durch $min\{B(x,\mu)|Ax \leq b\}$. Wird μ sehr klein (tendiert gegen 0) ist die Lösung des Barriereproblems die gleiche wie unser ursprüngliches Optimierungsproblem. Dies nutzt man aus, um von einem großen μ als Startpunkt iterativ $B(x,\mu)$ zu optimieren und dann vor der nächsten Iteration μ zu verkleinern.

Für kleine Werte von x wird $-\ln x$ sehr groß. Dies nutzt man aus, um kleine Werte von x zu bestrafen und damit die Suche der Lösung immer innerhalb des zulässigen Bereiches zu führen (verdeutlicht in Abbildung 3.5). Während der Suche wird μ sukzessive verkleinert, bis im Grenzfall $\mu \to 0$ das Barriereproblem gegen die Lösung des Optimierungsproblems konvergiert (siehe Abbildung 3.6).

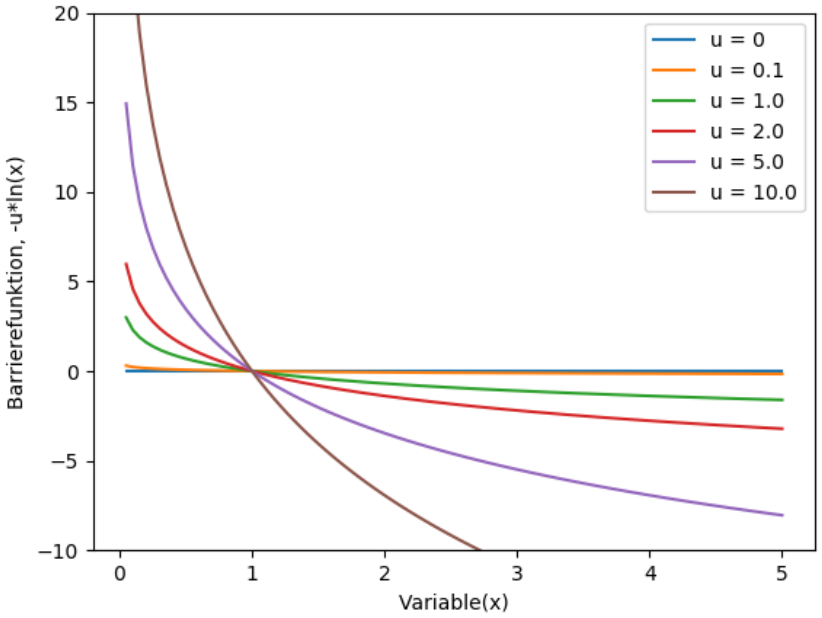

Abbildung 3.6: Logarithmische Barrierefunktionen

Um in jedem der iterativen Schritte möglichst schnell zu einer Lösung zu kommen, bedient man sich dem Newton-Rapshon-Verfahren. Der Algorithmus benötigt die Ableitungsmatrix 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 der Berechnung und die Geschwindigkeit sind jedoch oftmals problematisch.

Ein möglicher Ansatz die Ableitung $\Delta f(x) = (\frac{\partial f}{\partial x_1}(x), ..., \frac{\partial f}{\partial x_n})$ der Funktion $f : \Re^n \to \Re$ zu berechnen ist
$f'(x) \approx \frac{f(x+h)-f(h)}{h}$ mit $h \to 0$.
Dieser Ansatz ist jedoch nicht nur sehr ineffizient $\mathcal{O}(n)$, sondern besitzt zudem noch 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 (*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 es 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.

Vorwärtsmodus

Ein Verfahren der Umsetzung ist der Vorwärtsmodus. Er basiert auf der Kettenregel $(f \circ g)' = (f' \circ g)g'$.
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.

$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial w_2}\frac{\partial w_2}{\partial w_1}\frac{\partial w_1}{\partial x}$.
Zum besseren Verständnis wird die Ableitung für folgende Funktion $y = f(x_1, x_2) = x_1x_2 + \sin(x_1)$ berechnet.
Dazu werden wie oben bereits beschrieben zuerst die Bestandteile substituiert:
$y = w_1w_2 + \sin(w_1)$
$y = w_3 + w_4$
$y = w_5$

Danach muss der sogenannte *seed* errechnet werden. Er kodiert über welche Variable 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. Nachdem die Grundlagen zur Optimierung gelegt wurden, wird nun auf den Teil im Model Predictive Control eingegangen, welcher die Un- und Gleichheitsbedienungen definiert.

3.3 Fahrzeugmodelle

Wie der Name Model Predictive Control schon verdeutlicht benötigt man eine Systembeschreibung des zu regelnden Modells. Diese wird genutzt, um zukünftige Zustände zu berechnen und bildet damit einen wichtigen Bestandteil. Desto 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, welche weiter unten beschrieben werden, kann ein kinematisches Modell die laterale und longitudinale Bewegung eines Fahrzeuges mathematisch beschreiben. In diesem sehr stark vereinfachten Modell werden keine wirkenden Kräfte berücksichtigt, sondern nur die geometrischen Beziehungen des Fahrzeuges genutzt, um die Bewegung zu berechnen.
Im ersten Schritt werden die jeweils an einer Achse verbundenen Räder zu einem einzigen zusammengefasst. Dies wird als Bicycle Modell bezeichnet und vereinfacht die Berechnungen erheblich [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, zeit kontinuierlichen Gleichungen basieren auf [Raj11, KPSB15] und beschreiben das kinematische Modell bezüglich eines Inertialsystems (siehe Abbildung 3.7),

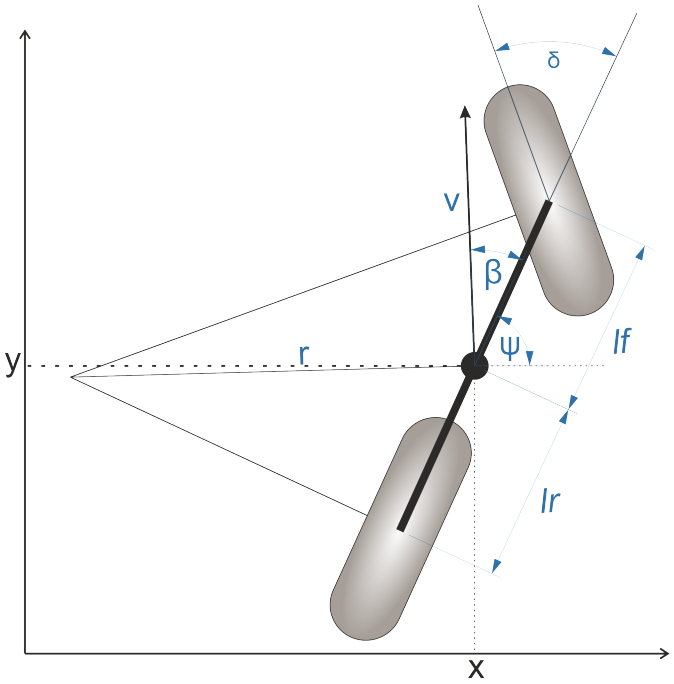

Abbildung 3.7: Kinematisches Modell:

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

in dem x und y die Koordinaten des Schwerpunktes im Inertialsystem darstellen. φ 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 gleiche Richtung wie auch die Geschwindigkeit. 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 mit betrachtet 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 aus dem Jahr 2017 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: $a_{max} = 2.0g$. 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 Schwimmwinkel (und damit gleichzeitig des Lenkwinkels) erreicht.

3.3.2 Reifenmodell

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

Die Kräfte, welche ein Reifen auf die Straße übertragen kann, hängen von dem Schräglaufwinkel, Schlupf und der Radlast ab. Die Radlast F_z berechnet sich aus der Normalkraft und der Radlastverteilung und wird im folgenden als konstant angesehen. Die Seitenführungskraft F_y wirkt bei einer Kurvenfahrt der Fliehkraft entgegen und hält das Fahrzeug auf der Spur solange ein Kräftegleichgewicht besteht. Als Schräglaufwinkel bezeichnet man den von der Radmittelebene δ (Lenkwinkel) und der Bewegungsrichtung θ_{vf} des Fahrzeugs eingeschlossenen Winkel (siehe Abbildung 3.8). Dieser ist notwendig, damit der Reifen eine Seitenkraft aufbauen kann.

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

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.

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

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

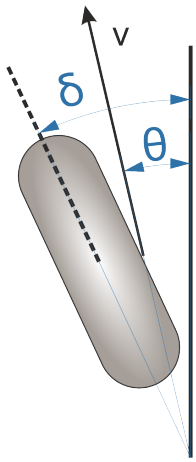

Abbildung 3.8: Schräglaufwinkel

Parameter	Wert
B	0.71
C	1.41
D	1.00
E	-0.20

Tabelle 3.1: Magic Formula Parameter

Am Schaubild 3.9 lässt sich dieser Bereich zwischen -1.5° und 1.5° sehr gut erkennen. Für größere Schräglaufwinkel kann kein linearer Verlauf mehr angenommen werden und es muss eine genauere Approximation des 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. Es 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 in x erhält man die lateral auf die Straße wirkende Kraft F_y.

$$F_y = D\sin[C\arctan Bx - E(Bx - \arctan(Bx))] \tag{3.3.13}$$

Die Parameter, welche für die *Magic Formula* benötigt werden, wurden vom High Octane Motorsports e.V. zur Verfügung gestellt.

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

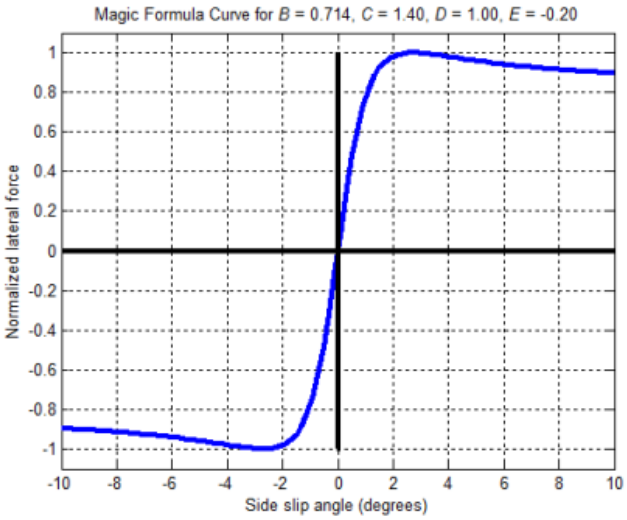

Abbildung 3.9: Tire Model

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.

$$F_x <= F_{max} \tag{3.3.14}$$

Der Zusammenhang zwischen lateraler und longitudinaler Kraft wird über den *Kammschen Kreis* modelliert (siehe Schaubild 3.10). 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 maximalen Kraft welche die Reifen übertragen können (F_{max}).

$$F <= F_{max} \tag{3.3.15}$$

$$F_{ll} = \sqrt{F_x^2 + F_y^2} \tag{3.3.16}$$

$$\tag{3.3.17}$$

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

$$\alpha = \arctan(F_x, F_y) \tag{3.3.18}$$

$$F_x = F_{max} \sin(\alpha) \tag{3.3.19}$$

$$F_y = F_{max} \cos(\alpha) \tag{3.3.20}$$

$$\tag{3.3.21}$$

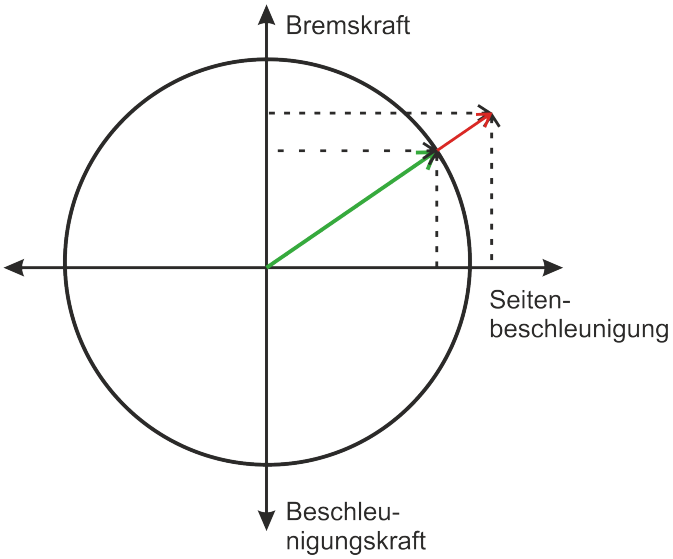

Abbildung 3.10: Kammscher Kreis: Zusammenhang von longitudinaler, lateraler und maximaler Reifenkraft

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.

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.

$$ma_y = F_{yf} + F_{yr} \tag{3.3.22}$$

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 3.11).

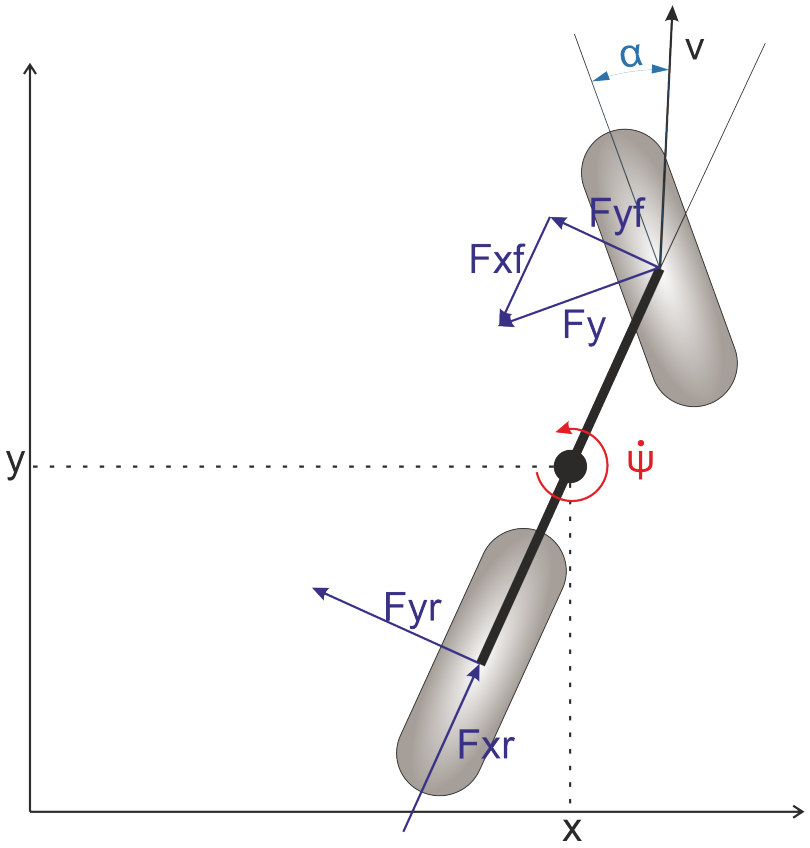

Abbildung 3.11: Dynamic Vehicle Model

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

$$I_z\ddot{\psi} = l_f F_{yf} - l_r F_{yr} \tag{3.3.23}$$

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

$$m\ddot{x} = m\dot{y}\dot{\psi} + F_x \tag{3.3.24}$$
$$m\ddot{y} = -m\dot{x}\dot{\psi} + F_y \tag{3.3.25}$$
$$I(\ddot{\psi}) = l_f F_{yf} - l_r F_{yr} \tag{3.3.26}$$

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

$$F_x = F_{xf} + F_{xr} \tag{3.3.27}$$
$$F_y = F_{yf} + F_{yr} \tag{3.3.28}$$

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.

$$F_{xf} = -2F_{Cf}\sin(\delta_f) \tag{3.3.29}$$
$$F_{yf} = 2F_{Cf}\cos(\delta_f) \tag{3.3.30}$$
$$F_{xr} = F_{lr} \tag{3.3.31}$$
$$F_{yr} = 2F_{Cr} \tag{3.3.32}$$

Es ist zu beachten, dass das Fahrzeug in der Realität vier Reifen besitzt und daher die Kräfte mit zwei multipliziert 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:

$$\alpha_f = \delta_f - \arctan(\frac{\dot{y} + l_f\dot{\psi}}{\dot{x}}) \tag{3.3.33}$$

$$\alpha_r = -\arctan(\frac{\dot{y} - l_r\dot{\psi}}{\dot{x}}) \tag{3.3.34}$$

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 maximale Motorleistung und F_{max}.

$$F_{lr_{acc}} = \frac{P_{engine} * throttle}{|\dot{x}|} \tag{3.3.35}$$

$$F_{lr_{acc}} <= 2F_{max} \tag{3.3.36}$$

$$F_{lr_{dec}} = -2F_{max} * break \tag{3.3.37}$$

$$\tag{3.3.38}$$

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. Es wird von einem Rennkurs ausgegangen der keine Steigung besitzt.

$$F_{reib} = m\mu g \tag{3.3.39}$$

$$F_{aero} = \frac{1}{2}\rho C_d A_f \dot{x}^2 \tag{3.3.40}$$

Die resultierende longitudinale Kraft (getrennt nach Beschleunigung und Bremsvorgang)

$$F_lr = F_{lr_{acc}} - F_{reib} - F_{aero} \tag{3.3.41}$$

$$F_lr = F_{lr_{dec}} - F_{reib} - F_{aero} \tag{3.3.42}$$

$$\tag{3.3.43}$$

wird mit den lateralen Kraft im Kammschen-Kreis verrechnet und in der finalen Bewegungsgleichung verwendet.

$$\dot{X} = \dot{x}\cos(\psi) - \dot{y}\sin(\psi) \tag{3.3.44}$$
$$\dot{Y} = \dot{x}\sin(\psi) - \dot{y}\cos(\psi) \tag{3.3.45}$$
$$m\ddot{x} = m\dot{y}\dot{\psi} + F_{xf} + F_{xr} \tag{3.3.46}$$
$$m\ddot{y} = -m\dot{x}\dot{\psi} + F_{yf} + F_{yr} \tag{3.3.47}$$
$$I\ddot{\psi} = l_f F_{yf} - l_r F_{yr} \tag{3.3.48}$$
$$\tag{3.3.49}$$

Das Gleichungssystem kann auch in diskreter Form aufgestellt werden.

$$X_{k+1} = X_k + \Delta t(\dot{x}_k\cos(\Psi_k) - \dot{y}_k\sin(\Psi_k)) \tag{3.3.50}$$
$$Y_{k+1} = Y_k + \Delta t(\dot{x}_k\sin(\Psi_k) - \dot{y}_k\cos(\Psi_k)) \tag{3.3.51}$$
$$\Psi_{k+1} = \Psi_k + \Delta t\dot{\psi}_k \tag{3.3.52}$$
$$\dot{x}_{k+1} = \dot{x}_k + \Delta t(\frac{F_{xf_k}+F_{xr_k}-F_a}{m} + \dot{y}_k\dot{\psi}_k) \tag{3.3.53}$$
$$\dot{y}_{k+1} = \dot{y}_k + \Delta t(\frac{F_{yf_k}+F_{yr_k}}{m} - \dot{x}_k\dot{\psi}_k) \tag{3.3.54}$$
$$\dot{\psi}_{k+1} = \dot{\psi}_k + \Delta t\frac{l_f F_{yf}-l_r F_{yr}}{I} \tag{3.3.55}$$
$$\tag{3.3.56}$$

Im dynamische Fahrzeugmodell ändert sich damit auch der Zustandsvektor:
X, Y, x_d, y_d, Ψ, $\dot{\psi}$
Der Steuervektor bleibt gleich.

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

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	1000	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

3.4 Programmiersprachen

3.4.1 Python

Zu Beginn der Arbeit wurde zuerst die Programmiersprache Python verwendet um den MPC-Ansatz zu programmieren. Als Optimierer wurde ein *Sequential Least SQuarez Programming (SLSQP)*-Algorithmus verwendet. Dieser ist direkt in die SciPy-Bibliothek integriert und unterstützt Eingangs-, Ausgangs- und Zustandsbeschränkungen, ohne die es nicht möglich ist den MPC-Algorithmus für die Fahrzeugregelung auszulegen. So lange die Anzahl der Prädiktionsschritte klein bleibt und 10 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 3.12 ist für das fertige Optimierungsproblem gemessen worden. Dies liegt daran, dass der SLSQ- 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 nach einer Alternativlösung gesucht. Das Ergebnis dieser Suche führte zur Verwendung einer neuen Programmiersprache.

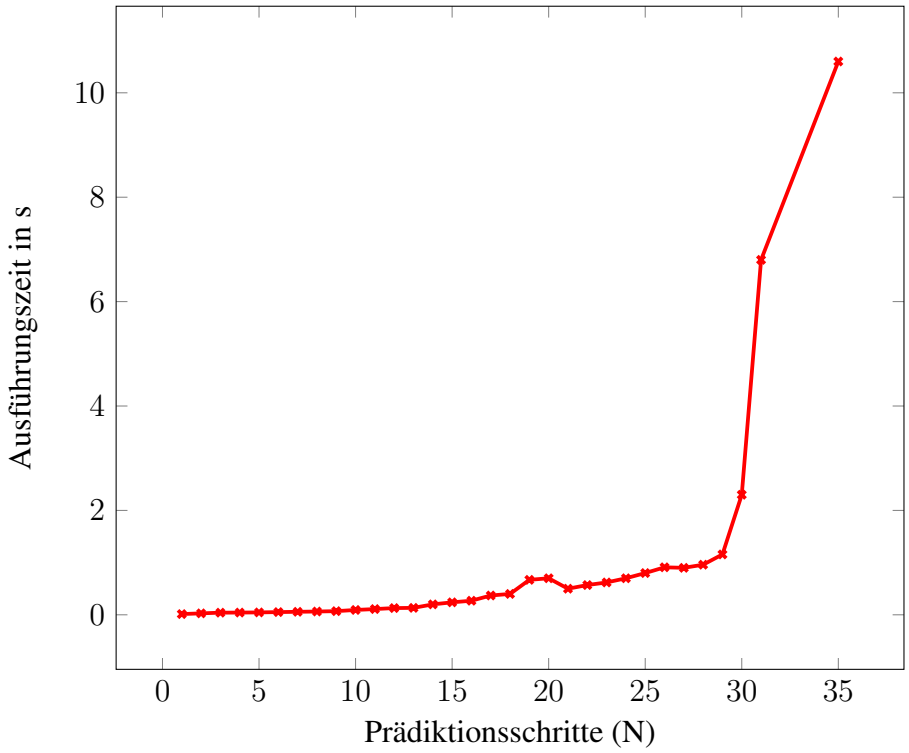

Abbildung 3.12: Ausführungszeiten für verschiedene große Horizontlängen

3.4.2 Julia

Julia ist die Programmiersprache, die die Umsetzung des Model Predictive Control-Algorithmus effizient und performant ermöglichte. Sie wurde im Jahr 2009 von Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and 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 dem Benchmark 3.13 zu sehen.

Die wichtigsten Feautres von Julia sind:

- multiple dispatch
  Wird genutzt um die gleiche Methode für verschiedene Kombinationen an

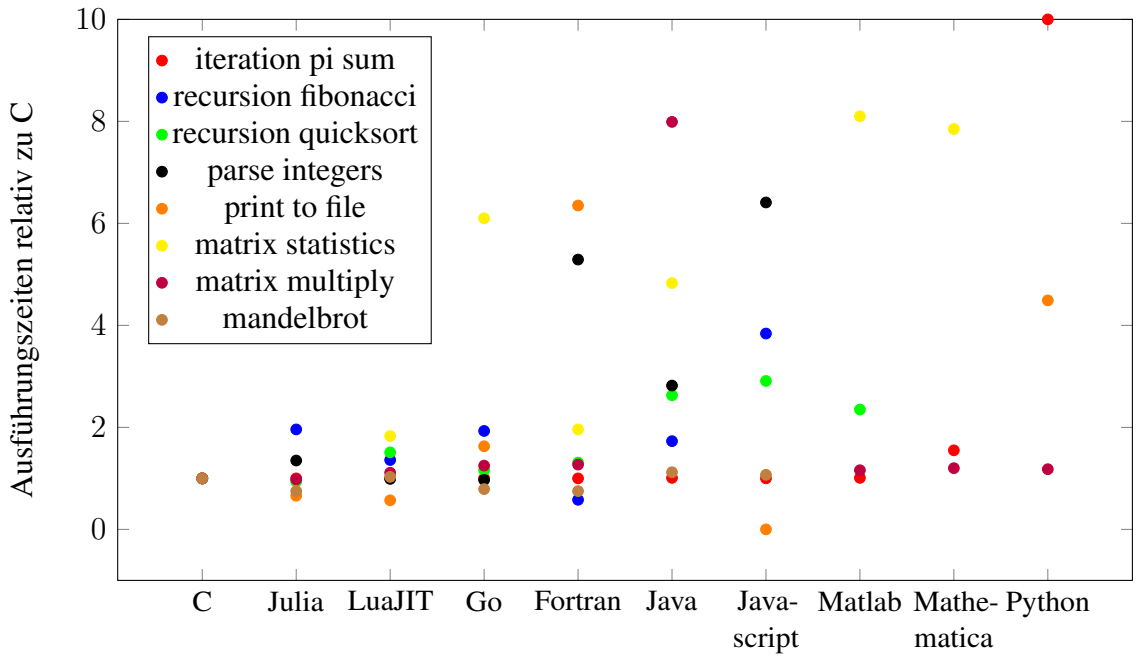

Abbildung 3.13: Ausführungszeiten für verschiedene Sprachen

Eingabeparametern oder Variablentypen zu überladen. Bei dem Funktionsaufruf wird die am besten passende Methodendefinition aufgerufen.

- type inference
  Das automatische detektieren des Datentyps. Dies befreit den Programmierer vom festlegen des Typs und ermöglicht trotzdem Typprüfung.

- just-in-time (JIT) Kompilierung
  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 sich zum Beispiel lineare oder nichtlineare Probleme lösen lassen. 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 5.1). Ebenfalls ersichtlich ist das Jump sich direkt um die automatische Ableitung kümmert und so dem Programmierer sehr viel Arbeit erspart.

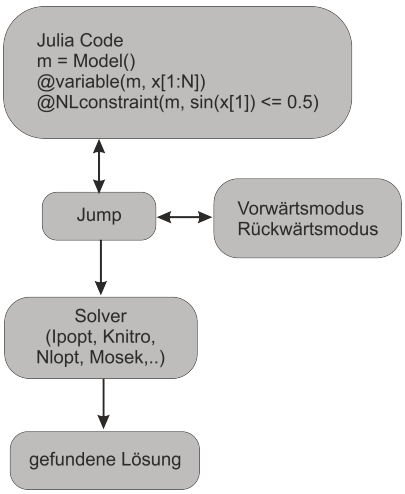

Abbildung 3.14: Blockdiagramm von Jump

Die Syntax welche Jump für die Definition der Probleme vorsieht ähnelt der standardmäßigen mathematischen Definition, was die Anwendung intuitiv gestaltet. Zudem ist die Dokumentation sehr ausführlich.

4 MPC zur Trajektionsplanung 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. Außerdem kann er sich dynamisch an Änderungen der Streckeverhältnisse 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, wie in Abbildung 4.1 dargestellt, leicht zu berechnen. Der so genannte Scheitelpunkt markiert hierbei den Punkt des geringste 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.

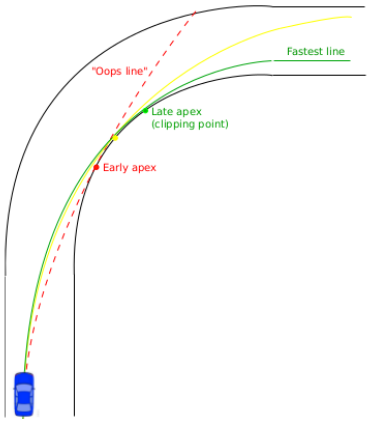

Abbildung 4.1: Ideale Trajektorie für eine 90 Grad Kurve (Dummy Bild)

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 wurde sich gegen den zweigeteilten Ansatz entschieden.

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

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

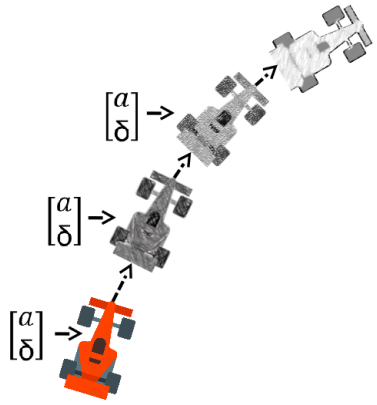

Abbildung 4.2: Grafische Visualisierung der Prädiktion abhängig von den Steuerparametern für drei Schritte in die Zukunft

4.1 Fahrzeugmodell

Zu Beginn haben die Teilvektoren $\vec{x}$ keinen Zusammenhang untereinander. Da jedoch die einzelnen Prädiktionsschritte voneinander über den Fahrzeugzustand, Steuerparameter und das Fahrzeugmodell zusammenhängen, werden im zweiten Schritt die Beschränkungen hierfür integriert. Dazu wird die diskretisierte Form des Systemmodells 3.3.1 genutzt, um immer zwei aufeinander folgende Schritte miteinander zu verknüpfen. Die entstehenden Gleichheitsbedingungen sind im Anhang unter **??**ufgeführt.

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 zwei verschiedene Kostenfunktionen implementiert:

Maximalgeschwindigkeit

In dieser Funktion wird die Summe aller Geschwindigkeitswerte der einzelnen Prädiktionsschritte addiert und als Kosten versucht zu maximieren.
$f(\vec{X}) = \sum_1^{N+1} v_i$
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 den 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 zu fahren 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. Eine bildliche Darstellung der Kostenfunktion ist in Abbildung 4.5 zu sehen.

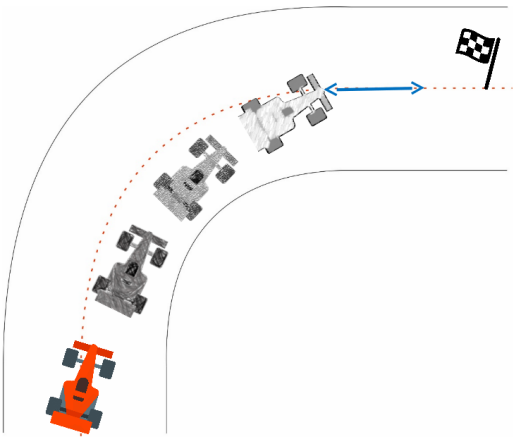

Abbildung 4.3: Zielpunkt Distanzminimierung: Das virtuelle Ziel wird immer weit genug vor dem Fahrzeug her geführt, so dass es nie erreicht werden kann.

Maximieren der Streckendistanz

Um eine Kostenfunktion zu finden welche einer Idealtrajektorie am nächsten kommt, muss überlegt werden wie man diese Messbar macht. 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 durch eine Umformung zu einer Minimierung der nach dem Zeitschritt übrig bleibenden Reststrecke umformen. Mit der Vereinfachung, dass das Ziel und damit die Reststrecke in einer Distanz vorm dem letzten Prädiktionsschritt sich befindet welchen das Fahrzeug nicht erreichen kann. Der Unterschied zur Kostenfunktion welche die Distanz minimiert ist, dass sich die Differenz nicht auf einen Punkt sondern auf eine Gerade senkrecht zur Rennkursmitte bezieht. Realisiert wird dies durch eine *vector rejection* $d = |a - \frac{ab}{bb}b|$. Dies ist in der Abbildung 4.4 bildlich veranschaulicht.

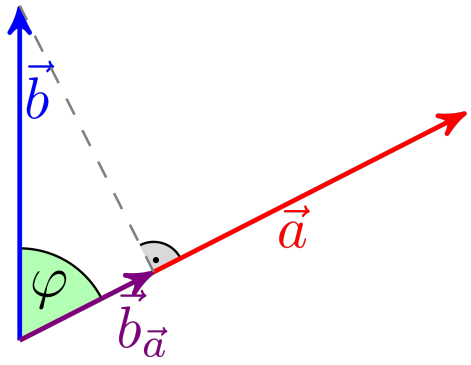

Abbildung 4.4: Maximieren der Streckendistanz (Dummy)

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 Fahrbahn Außen-, und Innenseite projiziert werden. So lange sich das Rennauto innerhalb dieser Tangenten bewegt ist die Beschränkung erfüllt. Während des Optimierungsvorgangs wird die Position der Tangenten nicht verändert, sondern vor der Optimierung fix gesetzt. 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 als Ausgangssituation gewählt. Für den letzten Schritt N ist dies nicht möglich, hier wird mithilfe der Geschwindigkeit und Orientierung ein virtueller Punkt projiziert.

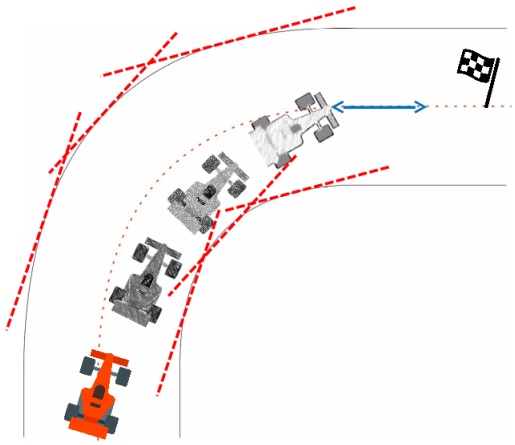

Abbildung 4.5: Tangentiale Begrenzung hält das Fahrzeug auf der Fahrspur

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 $\Delta t * v$ Meter entfernt von seinem Vorgängerzustand liegt. Wenn sich dieser Punkt in einer Kurve direkt an der Streckenmarkierung befindet, könnte das Fahrzeug sich trotzdem, 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. Dabei 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.

Im letzten Schritt, bevor der MPC-Algorithmus funktionsbereit ist, muss der Fahrzeugzustand des aktuellen Zeitschritts t_0 als Beschränkung verankert werden. Dies

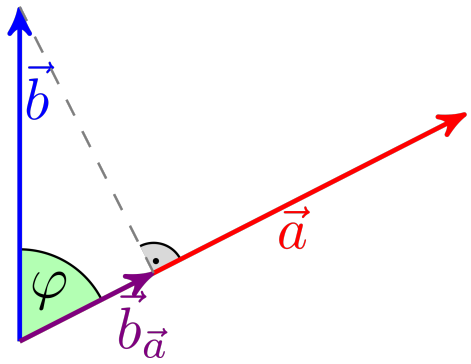

Abbildung 4.6: Vektor Projektion zum berechnen der tangentialen Beschränkung (platzhalter für richtiges Bild)

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 $90\check{r}$ aufweisen. Der Grund hierfür liegt in den tangentialen Beschränkungen und 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 Tangen 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 3er Zyklus, 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.

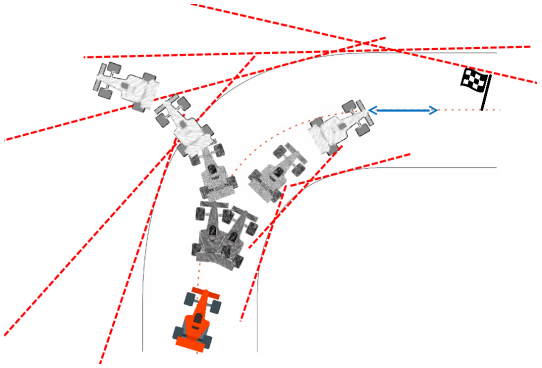

Abbildung 4.7: Fehlerhaftes Verhalten in Spitzkurven

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 wie auch im MPC hinterlegte Fahrzeugmodell exakt gleich ist, 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 dem kleinsten Regelfehler, Rutschen, Abweichungen vom Fahrzeugmodell etc. würde sich der initiale Fahrzeugzustand $\vec{x}_0$ bereits außerhalb der Beschränkungen befindet 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. Die elastische Kostenfunktion. Hier 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 hier 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:

$$g(d) = \alpha d^2 \tag{4.3.1}$$
$$g(d) = \alpha |d| \tag{4.3.2}$$
$$g(d) = e^{\alpha(k_1+d)} + e^{-\alpha(k_2+d)} \tag{4.3.3}$$
$$g(d) = |\tfrac{\alpha}{k_1-d} + \tfrac{\alpha}{k_2-d}| \tag{4.3.4}$$

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

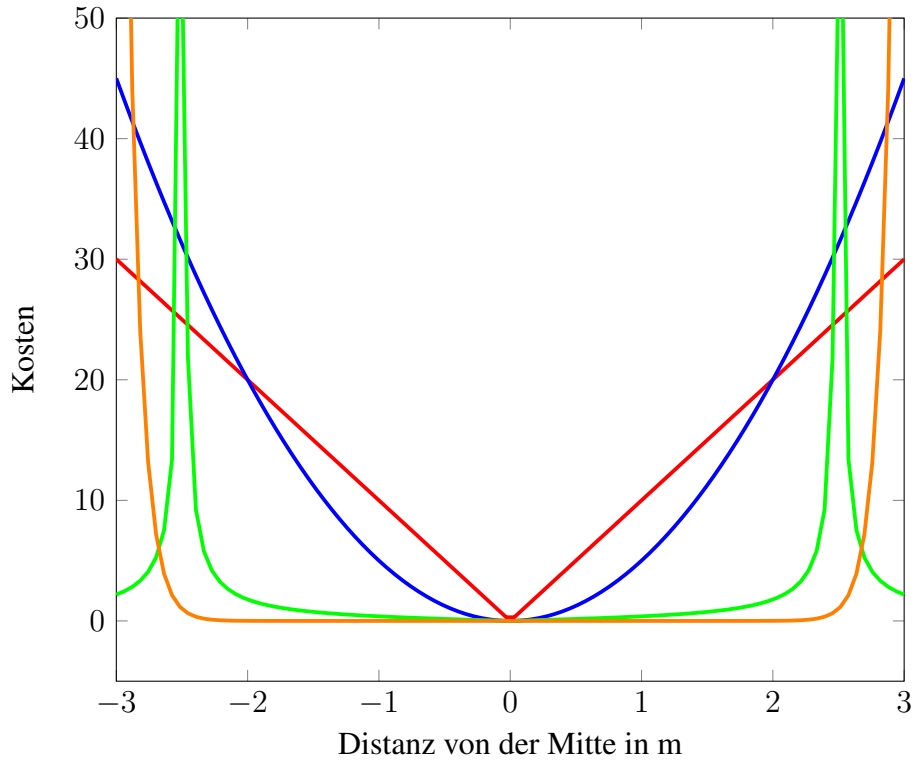

Abbildung 4.8: Verschiedene Distanzmaße um das Verhalten der Trajektionsplanung zu beeinflussen.

Die Idee hinter den Funktionen 4.3.3 und 4.3.4 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.

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.

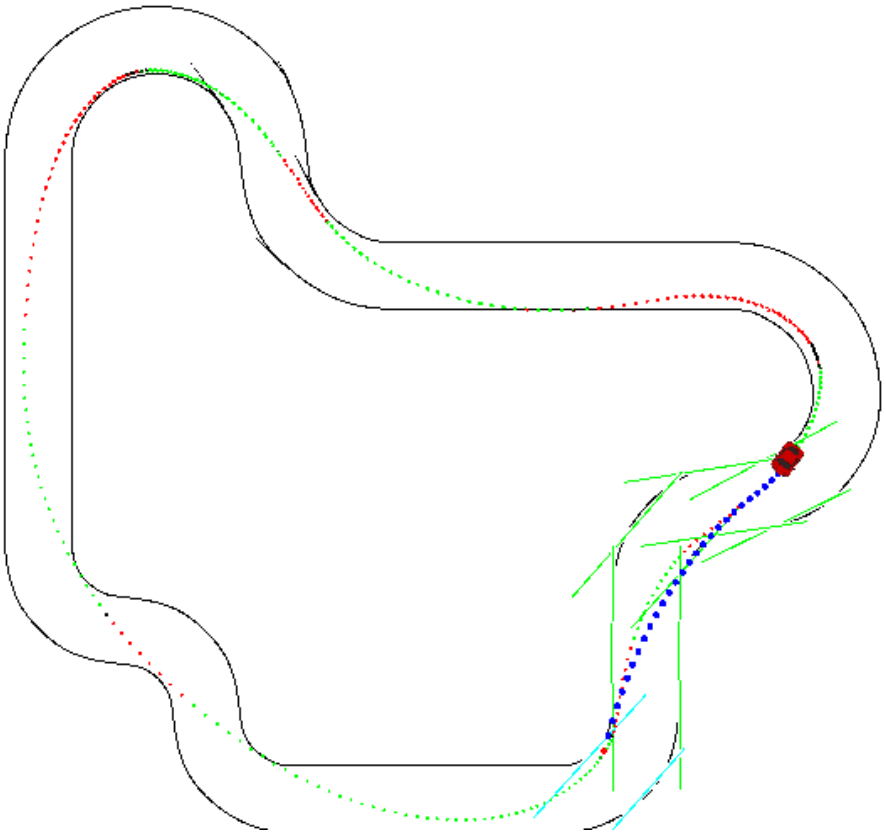

Abbildung 5.1: Grafische Darstellung des MPC - Algorithmus in der Simulationsumgebung

5.1 Simple and Fast Multimedia Library

Die Basis für die Simulation bildet die *Simple and Fast Multimedia Library (SFML)* welche für die grafische Darstellung des Kurses, Rennauto, Prädiktionsschritte und Fahrzeuginformationen genutzt wird. Gleichzeitig 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 (frames per second fps) ausgeführt wird. Benötigen die Berechnungen innerhalb dieser Schleife länger als das angegebenen $\Delta t = \frac{1}{fps}$ 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 2 Parameter eingeführt welche die Fenstergröße in Pixeln festlegen (in x- und y-Richtung) und zusätzlich eine Angabe wie viel Metern dieser Pixelbereich jeweils entspricht. Die daraus resultierenden Verhältnisse
$scaleX = \frac{windowSizeXinPixel}{windowSizeXinM}$
$scaleY = -\frac{windowSizeYinPixel}{windowSizeYinM}$
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 Null-Position 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 5.2 nochmals zusammengefasst.

5.2 Trackdrive

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

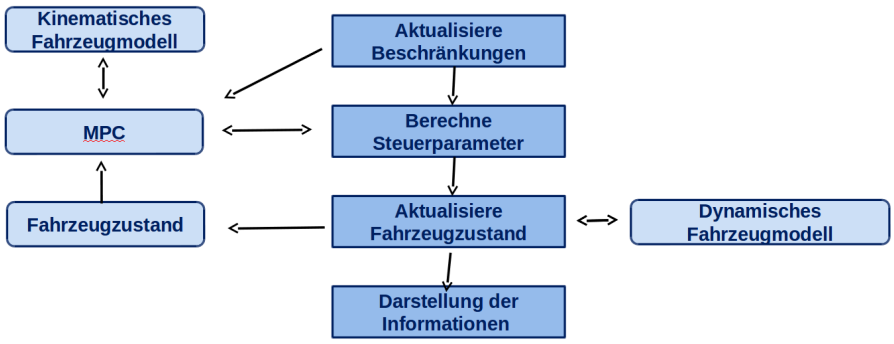

Abbildung 5.2: Blockdiagramm der Simulation ohne MPC

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 und maximal 180° Spitzkurven besitzt.

6 Evaluierung

Im folgenden Abschnitt werden zuerst die Fahrzeugmodelle mit Fahrsituationen evaluiert, für welche genaue Testdaten des High Octane Motorsports Rennautos aus dem Jahr 2017 bekannt sind. Nachdem sichergestellt wurde dass die Modelle nahe genug der Realität entsprechen wurden die verschiedenen Kostenfunktionen untersucht. Dieser Teil der Arbeit geht fließend in die Untersuchung der Regelbarkeit des Rennautos über, wenn im MPC-Algorithmus nur ein kinematisches Modell verwendet wird.

6.1 Verifizierung der Fahrzeugmodelle

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 Events 2017 in Hockenheim als Basis gewählt. Der MPC-Algorithmus wurde mit 10 Prädiktionsschritten und der Kostenfunktion für maximale Geschwindigkeit durchgeführt, da hierfür

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 (3.3.35). Das genaueste Modell betrachtet zusätzlich noch Reibung und Luftwiderstand.

In der Grafik 6.1 wird deutlich, wie stark sich die einzelnen Modelle, obwohl sie sich auf das gleiche Fahrzeug beziehen voneinander unterscheiden. Für alle Modelle wird 7.5 s lang maximal beschleunigt und danach 2.5 s mit maximaler Kraft gebremst. Das Fahrzeug besitzt eine Maximalgeschwindigkeit von 118 kmh, welche durch die Übersetzung im Getriebe herrührt. Das kinematische Modell trifft mit seiner durchschnittlichen Beschleunigung exakt die gefahrenen Testwerte. Da aber die dynamischen Modelle eine

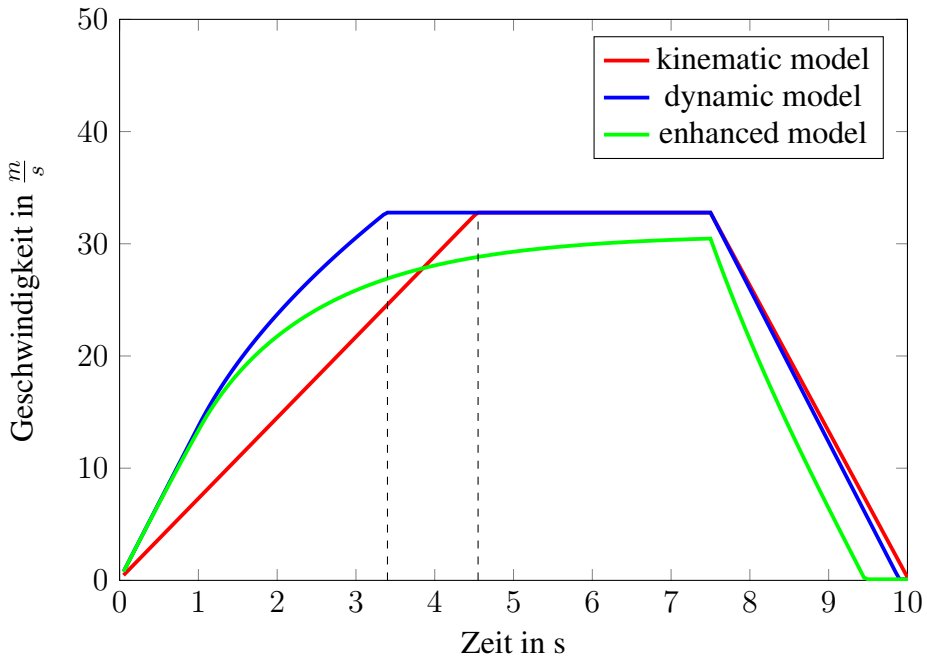

Abbildung 6.1: Beschleunigung für verschiedene longitudinale Modelle

größere Anfangsbeschleunigung haben und vom kinematischen Model, hinterlegt im MPC, geregelt werden, wird die Beschleunigung im kinematischen Modell so angepasst, dass es die gleiche Anfangssteigung besitzt.

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 Zentripetalkraft, welche wirkt, überprüft werden.

$\omega = 2\pi f$
$v = r * \omega$
$a = \frac{v^2}{r}$

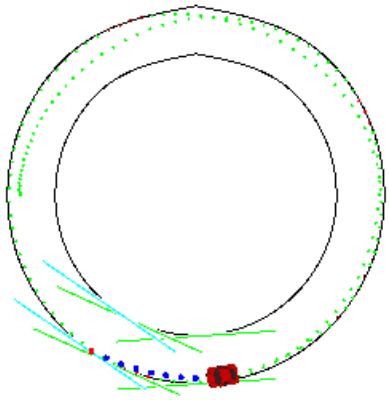

Abbildung 6.2: Rundkurs zum messen der maximalen Querbeschleunigung

Für das in der Simulationsumgebung hinterlegtes kinematisches Modell ist die gemessene Rundenzeit 4.65 s. Daraus lässt sich eine Durchschnittsgeschwindigkeit $v = 13.9\frac{m}{s}$ berechnen, was einer Querbeschleunigung von $a = 19.37\frac{m}{s}$ entspricht. Das im Fahrzeugmodell hinterlegte a_{max} ist mit 19.62 also sehr nah an dem in der Simulation erfahrenen werten. Die Differenz zu der schnellsten gemessenen Runde mit dem echten Fahrzeug (7.5 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 kann. Für das dynamische Fahrzeugmodell wurde eine minimale Rundenzeit 4.45 s gemessen. Die damit erhaltene maximale Querbeschleunigung ist nur 5 größer als die des kinematischen Models.

6.2 Regelung des kinematischen Modells im Rennkurs

In diesem Kapitel wird untersucht mit welcher Kostenfunktion die besten Rundenzeiten erreicht werden können. Zudem wir erprobt, wie groß der Einfluss des Vorhersagehorizontes auf die gefahrene Zeit ist. Um dies tun zu können, muss zuerst die mindestens erforderliche länge des Prädiktionshorizontes gefunden werden. In einem Szenario im öffentlichen Verkehr muss der Horizont immer mindestens so lang sein, dass eine Vollbremsung von der aktuellen Geschwindgkeit durchgeführt werden kann, bevor das Fahrzeug das Ende des Horizont erreicht hat. Auf dem Rennkurs ist dies nicht nötig, da hier 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 das gleiche Fahrzeugmodell hinterlegt ist, wird die tangentiale Beschränkung genutzt um das Rennauto daran zu hindern die Strecke zu verlassen.

6.2.1 Prädiktionshorizont

Um die Mindestlänge zu finden, wurde die Fahrsituation nachgestellt, welche nach Regelwerk den stärksten Bremsvorgang und Querbeschleunigung erzeugt. Eine 80 m lange Gerade mit zwei angehängten 180 Grad Kurven mit 9 m Innenradius. Das Ergebnis ist ein Prädiktionsvektor von 17 Schritten für den die geringste Geschwindigkeit in der kurve 36.72 kmh gemessen wurde. Die höchste auf der geraden erreichte Geschwindigkeit beträgt 85.32 kmh was einer Differenz von 48.6 kmh entspricht. Mit der im kinematischen Model hinterlegten maximalen Verzögerung kann das Fahrzeug in einer Sekunde 47.2 kmh abbremsen. Die 17 Zeitschritte entsprechen 0.85 s. Der Zusammenhang aus Bremsleistung und maximaler Kurvengeschwindigkeit ist also klar ersichtlich.

6.2.2 Kostenfunktionen

Mit der Information wie groß der geringsten Prädiktionshorizont 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.

Laufzeit

Zuerst wird die Berechnungszeit für alle drei Kostenfunktionen verglichen. Die Geschwindigkeit wurde auf 7 $\frac{m}{s}$ limitiert um auch kleinere Horizonte testen zu können. Die Ergebnisse der Messung sind im Benchmark 6.3 visualisiert.

Die Berechnung des initialen Optimierungsvektor dauert im Vergleich zu allen darauf folgenden Schritten extrem lange. Dies liegt an dem so genannten *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 man sich wie in einem shift Register vorstellen, in dem

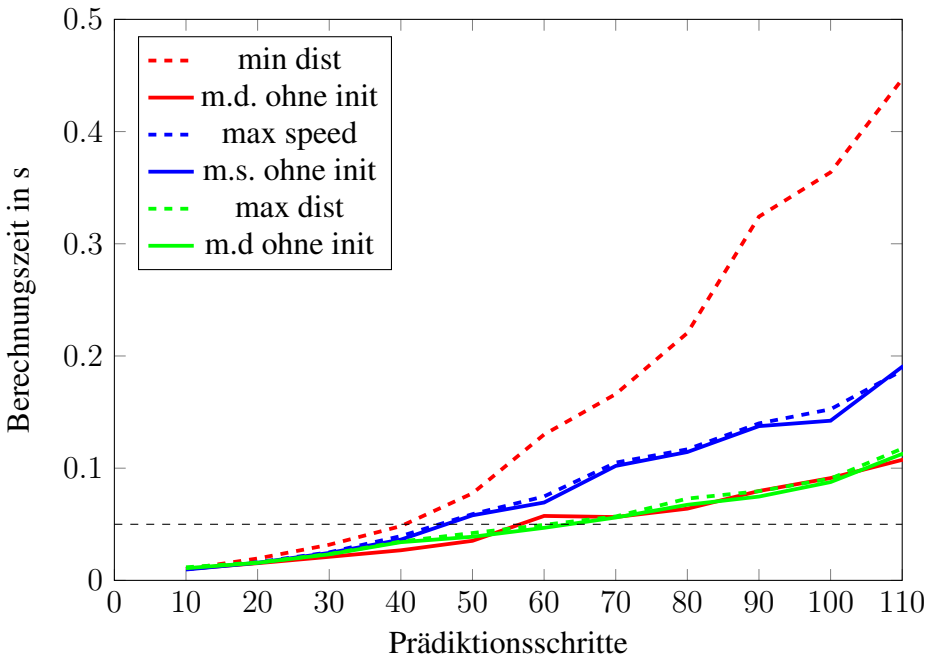

Abbildung 6.3: Berechnungszeit für verschiedene Kostenfunktionen mit und ohne die ersten 40 von 240 Berechnungen

die Lösung nur einen Schritt nach links verschoben wird. Bei einem Vorhersagehorizont von 110 Schritten brauchen zum Beispiel die ersten 10 Schritte über 6 s zum Berechnen, danach nur noch im Schnitt 0.1 s

Rundenzeit

Um die Qualität der Kostenfunktionen einschätzen zu können wird im folgenden Untersucht, wie schnell die einzelnen Rundenzeiten abhängig von den Prädiktionsschritten für jede einzelne Funktion ist. Gemessen wurde sowohl die erste und 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 zweiten.

Das Verhalten der Kostenfunktion, welche die Geschwindigkeit maximiert, mit längerer vorrausprädiktion 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

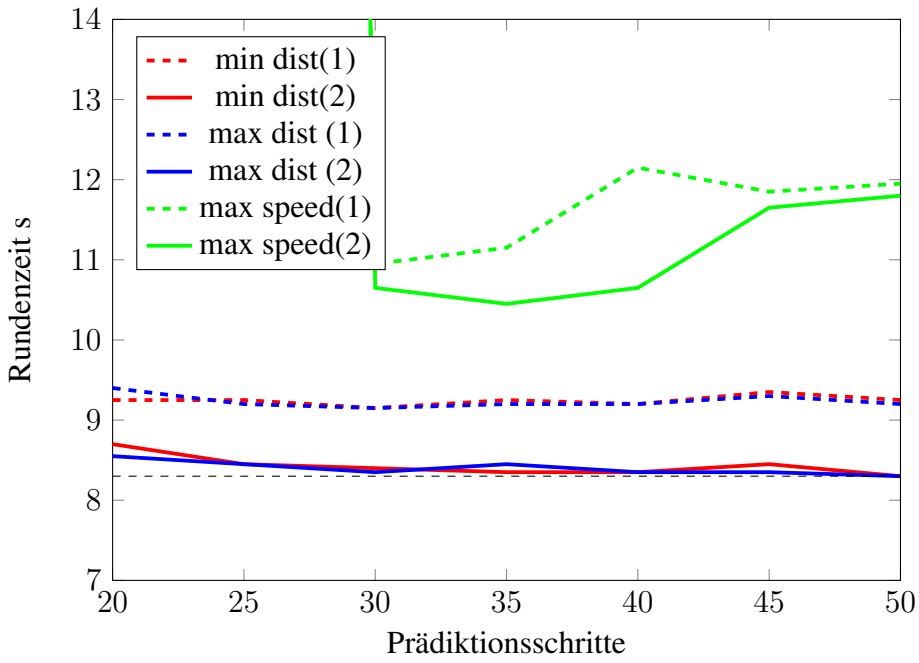

Abbildung 6.4

zwar die Geschwindigkeit erhöht werden, die Rundenzeit leider allerdings deutlich darunter. Die Differenz zwischen den beiden Kostenfunktionen, welche die Distanz zu einem Punkt bzw die gefahrene Streckendistanz maximieren, ist sehr klein und auch bei geringeren Horizontlängen bereits sehr gut. Betrachtet man jedoch zusätzlich die Berechnungszeit setzt sich die Kostenfunktion zur maximierung der Streckendistanz deutlich als die Beste ab.

6.3 Regelung des dynamischen Modells im Rennkurs

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 ein dynamische Fahrzeugmodell mit berechneten Reifenkräften ersetzt und evaluiert, wie gut sich das deutlich komplexere System mit dem weiterhin sehr einfachen kinematischen Modell im MPC-Algorithmus regeln lässt. Zuerst wird untersucht, ab welchen Geschwindigkeiten die Abweichung zwischen den Modellen signifikant wird. Danach werden die einzelnen elastischen Begrenzung auf ihre Fähigkeit getestet das dynamische Modell auf dem Kurs zu halten und welche Rundenzeiten sie ermöglichen. Im letzten Schritt wird zusätzlich noch untersucht welchen Einfluss β_{max}

auf die Regelfähigkeit des Algorithmus hat und ob sie sich durch verändern der Werte verbessern lässt.

6.3.1 Differenz der Modelle

6.3.2 Elastische Beschränkung

6.3.3 Einfluss der Geschwindigkeit und Beta Max

Die Idee hinter dem verändern von β_{max} ist die geringere Kurvengeschwindigkeit. Da die Querbeschleunigung von Kurvenradius und Geschwindigkeit abhängt, wird der MPC-Algorithmus in engen Kurven, für ein kleineres β_{max}, vorher stärker abbremsen. Die geringere Geschwindigkeit in der Kurve verkleinert die Differenz der zwei Modelle und sollte daher die Regelbarkeit erhöhen. Der Unterschied zu einer dauerhaften Verringerung der Geschwindigkeit ist, dass bei geraden Strecken wenig laterale Kräfte wirken und das Modell sich daher schneller bewegen kann ohne unkontrollierbar zu werden.

7 Zusammenfassung und Ausblick

7.1 Dynamisches Fahrzeugmodell im MPC

7.2 Trajektionsregelung

7.3 Zweispurmodell

7.4 Zusammenfassung

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

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

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

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.)

Abkürzungsverzeichnis

MPC Model Predictive Control

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.

[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.

[KG12] K. Kritayakirana and J. C. Gerdes. Using the centre of percussion to design a steering controller for an autonomous race car. *Vehicle System Dynamics, International Journal of Vehicle Mechanics and Mobility*, 50:33–51, January 2012.

[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.

[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