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	08. 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 vor dem 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 Autonomes Fahren

Das Jahr 2018 entwickelt sich immer mehr zum wichtigsten Jahr für autonome Fahrzeuge seit dem die Darpa im Jahr 2004 den Hype mit der *Grand Challenge* ins Rollen brachte.

1.1 Stand der Technik

Gegen Ende des Jahres 2018, also nur wenigen Monaten nachdem diese Zeilen verfasst werden, wird Alphabets Tochterunternehmen Waymo die ersten voll autonomen (Level 5) Fahrzeuge als Taxis in Phoenix in Betrieb nehmen [**?**]. Dieser historische Moment wird eine Flut an neuen Assistenzsystem nach sich ziehen die in wenigen Jahren das Straßenbild verändern werden.
Während sich die traditionellen Automobilhersteller aktuell auf Assistenzsysteme im Bereich von Level zwei und drei konzentrieren, wollen Unternehmen wie Google, GM und Uber alle Zwischenschritte überspringen und direkt voll autonom fahren um fahrerlose Taxiflotten aufzubauen. Dies ist nach aktuellem Stand der Technik jedoch nur kostengünstig und mit Rechnerplattformen welche in bestehende Fahrzeugkonzepte integrierbar sind realisierbar wenn die Fahrzeuge auf eine genaue Umgebungskarte zurückgreifen können. Erst mit Echtzeitupdates und bei einer Genauigkeit von +-10 cm des Kartenmaterials können Fahrzeug auch im innerstädtischen Verkehr autonom fahren [SH16]. Nicht nur im normalen Straßenverkehr ist ein möglichst genaues Wissen der Umgebung von Vorteil, auch bei autonomen Rennserien spielt die Kenntnis über den Rennkurs eine große Rolle. Kann das Fahrzeug nur auf Sicht fahren muss die Geschwindigkeit so gewählt werden, dass innerhalb der Sichtweite angehalten werden kann. Außerdem ist das berechnen einer idealen Trajektorie nur möglich wenn die gesamte Strecke bekannt ist.

1.2 Autonomes Rennfahren

Auch für normale Konsumer kann die Technik interessante Vorteile bieten. So ist es vorstellbar dass für Wochenendausflüge auf Rennstrecken die Fahrzeuge den Fahrern optimale Trajektorien auf dem Kurs direkt in der realen Situation vorfahren können. Ebenfalls ein wichtiges Einsatzkriterium ist das Vorausberechnen ob ein Fahrer die

eigenen Fähigkeiten und/oder die des Autos in dem er fährt überschätzt. In diesem Fall kann das Fahrzeug eingreifen und schlimmeres Verhindern. Zudem ist es deutlich günstiger Grenzerfahrungen in einem autonom gesteuerten Fahrzeug zu erleben als einen professionellen Fahrer zu bezahlen der viele Jahre Erfahrung am Steuer gesammelt haben muss.

1.3 Formula Student Driverless

Die Formula Student ist ein Ingenieurswettbewerb für Studenten. Er hat seine Wurzeln in den USA im Jahre 1981 und wurde ab 1998 auch in Europa ausgetragen. 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 um ein Sensorsystem sowie Aktoren so erweitert, dass das Rennauto die Kurse autonom bestreiten kann. Der Wettbewerb ist unterteilt in Dynamische und Statische Events. In letzteren werden verschiedene Präsentationen von den Teams verlangt. Diese beziehen sich auf die technische Realisierung, Softwaredesign, Kostenaufstellung und ein Businessplan in dem die Teams ein Konzept erstellen müssen wie man das gebaute Fahrzeug über eine kleine Massenfertigung gewinnbringend verkaufen kann.

Die dynamischen Disziplinen, in denen das Fahrzeug selbstständig fährt, sind hierbei:

- Acceleration
  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 2 rechte und 2 linke Runden gefahren werden. Die zweite Runde wird jeweils gezählt. 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 9 Metern. Es werden 11 Runden gefahren und die Teams erhalten im Vornherein keine Möglichkeit Messungen am Kurs vorzunehmen.

Für diese Arbeit ist vorallem 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 (wenn auch nicht so kurz) in einem FS-Event für die Driverless Fahrzeuge vorkommen könnte.

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

1.4 Ziel der Arbeit

2 Trajektionsplanung und Fahrzeugregelung

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

2.1 Model Predictive Control

Bei Model Predictive Control (MPC) handelt es sich um einen fortschrittlichen Algorithmus, mit dem sehr komplexe, multivariable Regelungsprobleme lösen lassen. Soll ein Prozess kontrolliert werden, welcher mehrere Ein- und Ausgänge besitzt und gleichzeitig Ungleich- und Gleichheitsbedingungen erfüllen werden müssen, ist MPC ein mächtiges Werkzeug. Vorausgesetzt 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 Verhalten, für zukünftige Steuerparameter, zu Berechnen [SMED10].
Einige wichtige Vorteile von MPC sind:
(1) das sehr gute annähern an Prozessgrenzen und damit ein hoher Durchsatz/Effizienz, (2) Einschränkungen der Eingangs und Ausgangsgrößen werden berücksichtigt, (3) die Vorhersage des Fahrzeugzustands kann genutzt werden um mögliche Probleme frühzeitig zu detektieren. Seit den späten 70ern bis in die frühen 2000er wurde MPC vor allem in der Chemiebranche genutzt um die komplexen multivariablen Regelungsprozesse bei zum Beispiel 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 Quadcopter [GD17] oder Fahrzeugen [FB05] ist inzwischen möglich.

2.1.1 Funktion

Der systematische Ablauf eines MPC- Algorithmus ist in Abbildung 2.1 aufgezeigt.

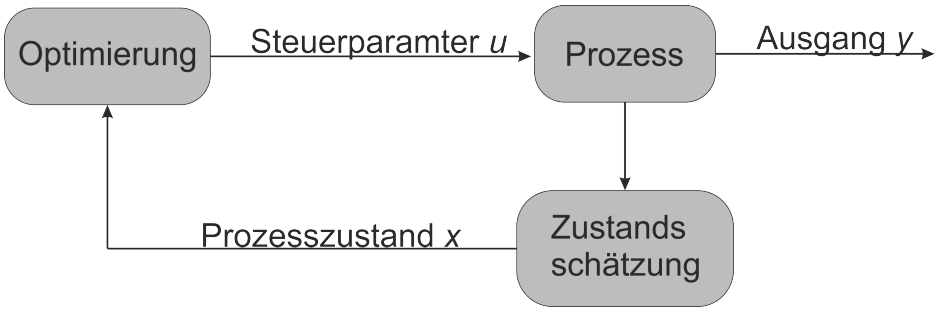

Abbildung 2.1: Block Diagramm zu MPC

Die Funktion lässt sich in mehrere Schritte unterteilen:

- Zustandsschätzung
  Im ersten Schritt wird der aktuelle Zustand des System erfasst und als Ausgangspunkt für die Berechnung des Optimierungsproblems festgelegt.
- Optimierung
  In diesem Schritt werden die zukünftigen Zustände des System berechnet und welche Steuerparameter von Nöten 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 geht es wieder beim ersten Schritt los.

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 heran reicht, desto weiter in die Zukunft können die Systemzustände berechnet werden. Außerdem 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. Nach dem der Optimierer die für die Kostenfunktion idealen Steuergrößen für den gesammten 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 2.2 verdeutlicht.

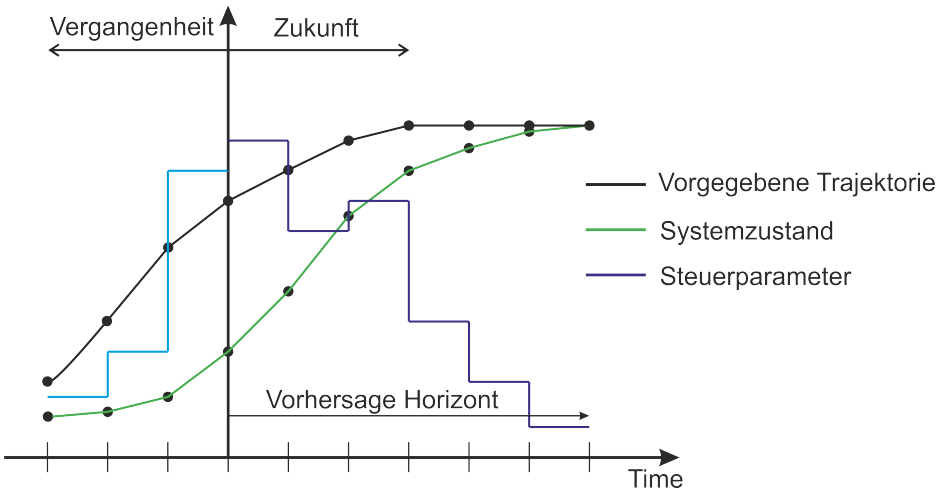

Abbildung 2.2: Funktionsprinzip von MPC: Die Kostenfunktion ist die Reduktion der Distanz zur Referenztrajektorie. Durch das Modell kann das System so geregelt werden dass es nicht über die Zieltrajektorie hinaus schießt.

Wie bereits erwähnt 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 einer autonomen Fahrzeugs während der Fahrt darstellt.

Um den *Model Predictive Control* Algorithmus für den Anwendungsfall eines Rennautos zu implementieren braucht man Folglich:

- ein Fahrzeugmodell
- einen Optimierungsalgorithmus

Auf beides wird in den nächsten Kapiteln eingegangen.

2.2 Optimierung

Das Konzept der Optimierung ist es, eine gegebene Funktion $f(\vec{x})$, auch als Kostenfunktion bezeichnet, zu mini- oder maximieren. Dies wird in einer Vielzahl von Anwendungsgebieten genutzt. Zum Beispiel zur Berechnung von Profit / Verlust in einem Betrieb, Geschwindigkeit oder Distanz in einem physikalischen Problem oder der erwartete *return of investment* für ein 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. In dem $f(\vec{x})$ durch $-f(\vec{x})$ ersetzt wird, kann aus dem Minimierungs, ein Maximierungsproblem gemacht werden.

Eine 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 2.3.

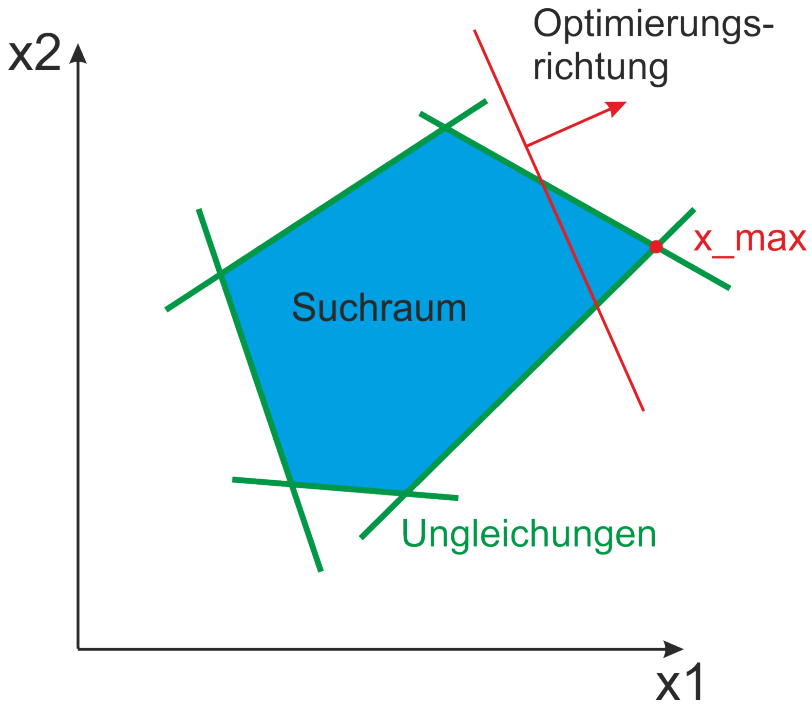

Abbildung 2.3: Lineare Optimierung

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, also die Menge aller Punkte, die für jede Ungleichung in der jeweiligen zulässigen Hälfte des Raumes liegt. Der Punkt, der die Kostenfunktion $c : x \rightarrow c^{\mathrm{T}}x$ 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.

Lösungsverfahren

2.2.1 Simplex - Verfahren

Obwohl zur Lösung der nichtlinearen Probleme die in dieser Arbeit berechnet werden müssen das Innere-Punkte-Verfahren genutzt wird, soll ein kurzer Überblick über das prominenteste Verfahren zum lösen von linearen Programmen gegeben werden. Dies dient der Veranschaulichung im späteren Verlauf. 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.

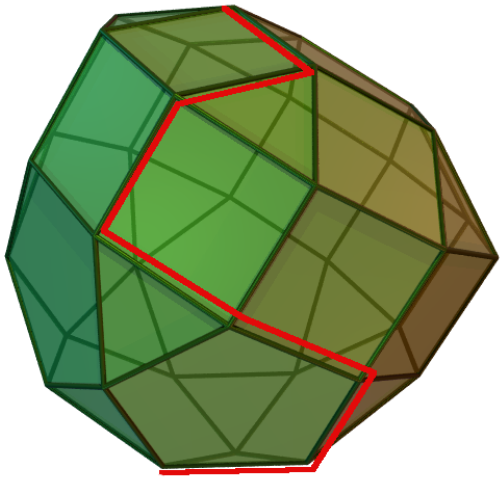

Abbildung 2.4: Simplex Methode: Ablaufen des Suchraums entlang der Kanten der Ungleichheitsbedingungen von Knoten zu Knoten

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 ein mal besuchen muss, hat das Verfahren ein exponentielles Wachstum 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 so genannten 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.

2.2.2 Innere-Punkte-Verfahren

Im Folgenden wir das Innere-Punkte-Lösungsverfahren für lineare, quadratische und nichtlineare Optierungsprobleme 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 2.5 veranschaulicht.

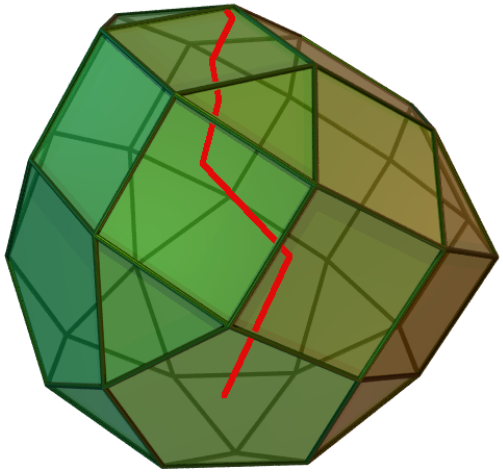

Abbildung 2.5: Innere-Punkte-Verfahren: Bewegt sich durch das Innere des Polytop auf der Suche nach dem mimimum.

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. 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.

Der Strafterm $-\mu \ln x_i$ wird für x-Werte nahe an ihrer Grenze sehr groß was in dem Plot 2.6 aufgezeigt wird. Das Ergebnis ist, dass die Lösungen des Barriereproblemes, solange μ groß genug ist, von den Flächen des Polytop entfernt bleiben. Dadurch nähert man sich nicht von außen entlang der Grenzen der Hyperebene, sondern durch das Innere des zulässigen Bereiches der optimalen Lösung des Optimierungsproblems an.

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 2.5). Während der Suche wird μ sukzessive verkleinert, bis im Grenzfall $\mu \to 0$ das Barriereproblem gegen die Lösung des Optimierungsproblemes konvergiert (siehe Abbildung 2.6).

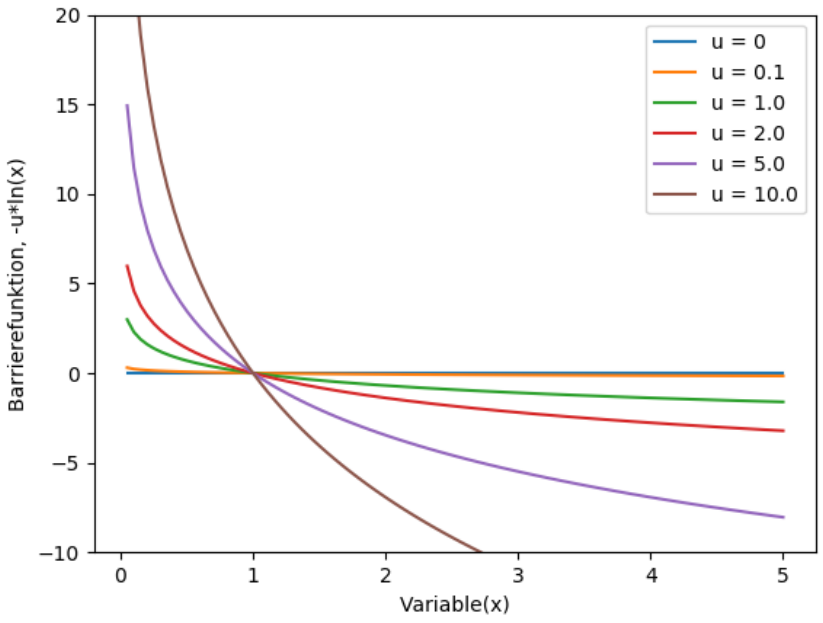

Abbildung 2.6: Logarithmische Barrierefunktionen

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.

2.2.3 Automatische Ableitung

Ableitungen sind eine Grundvoraussetzung für viele numerische Algorithmen. Die Genauigkeit der Berechnung und die Geschwindigkeit ist 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 ist jedoch sehr ineffizient mit einer Laufzeit von $\mathcal{O}(n)$, zudem spielt hier der Rundungsfehler des *float*-Datentyps mit in die Genauigkeit der Lösung [jul].

Eine andere Alternative wäre die Ableitung manuell zu berechnen, was jedoch für komplexere Systeme sehr aufwändig 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* berechnen. Zu der höheren Präzision kommt noch die deutlich bessere Laufzeit. Im Idealfall entspricht die Komplexität $\mathcal{O}(1)$. 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:
$y = w_1 w_2 + \sin(w_1)$
$y = w_3 + w_4$
$y = w_5$

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 für die Un-, und Gleichheitsbedienungen erzeugt.

2.3 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 realen 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.

2.3.1 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 da das Driverless Fahrzeug der Uni Erlangen nur über eine solche verfügt. Die Lenkwinkel welche durch das Bicycle Modell berechnet werden entsprechen nicht den Lenkwinkel 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 basieren auf [Raj11, KPSB15] und beschreiben das kinematische Modell bezüglich eines Inertialsystems (siehe Abbildung 2.7),

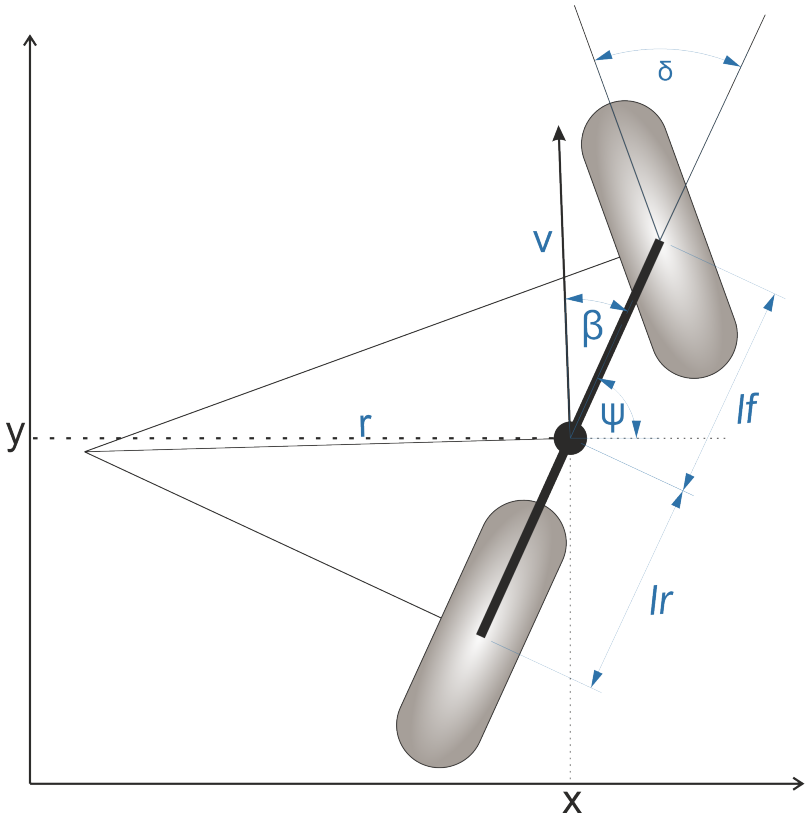

Abbildung 2.7: Kinematisches Modell:

$$\dot{x} = v\cos(\psi+\beta) \tag{2.3.1}$$
$$\dot{y} = v\sin(\psi+\beta) \tag{2.3.2}$$
$$\dot{\psi} = \frac{v}{l_r}\sin(\beta) \tag{2.3.3}$$
$$\dot{v} = a \tag{2.3.4}$$
$$\beta = \arctan(\frac{l_r}{l_f+l_r}\tan(\delta_f)) \tag{2.3.5}$$

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 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 der Uni Erlangen (siehe 2.2). 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

$$a = \frac{v}{t} \tag{2.3.6}$$

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

$$x = \frac{1}{2} * a * t^2 \tag{2.3.7}$$

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

$$|\beta| \leq \arctan(\frac{1}{2}\frac{\frac{l}{2}a_{max}}{v^2}) \tag{2.3.8}$$

wird dies über die Einschränkung des Schwimmwinkel (und damit gleichzeitig des Lenkwinkels) erreicht.

2.3.2 Reifenmodell

Da die Reifen der einzige Kontaktpunkt zwischen Fahrbahn und Fahrzeug sind, beeinflusst 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, dem 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 2.8). Dieser ist notwendig,

damit der Reifen eine Seitenkraft aufbauen kann.

$$\alpha_f = \delta - \theta_{vf} \tag{2.3.9}$$

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

$$\alpha_r = \theta_{vr} \tag{2.3.10}$$

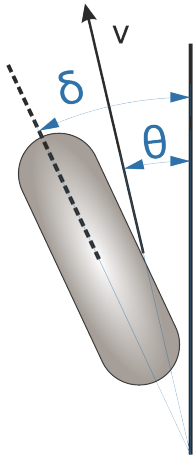

Abbildung 2.8: Schräglaufwinkel

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

$$F_{yf} = C_\alpha \alpha_f \tag{2.3.11}$$

$$F_{yr} = C_\alpha \alpha_r \tag{2.3.12}$$

Am Schaubild 2.9 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 Reifenverhaltens gewählt werden. Hierfür wird die so genannte *Magic Formula* [PB92] verwendet. Dabei handelt es sich um eine mathematische Gleichung die sehr gut Messkurven approximiert welche auf Teststänsden gemessen werden. Es wurde 1993 von Pacejka und Bakker entwickelt, und eignet sich

Parameter	Wert
B	0.71
C	1.41
D	1.00
E	-0.20

Tabelle 2.1: Magic Formula Parameter

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.

$$F_y = D\sin[C\arctan Bx - E(Bx - \arctan(Bx))] \tag{2.3.13}$$

Die Parameter, welche für die *Magic Formula* benötigt werden, wurden vom High Octane Motorsports e.V. zur Verfügung gestellt und beziehen sich auf das Fahrzeug des Jahres 2018.

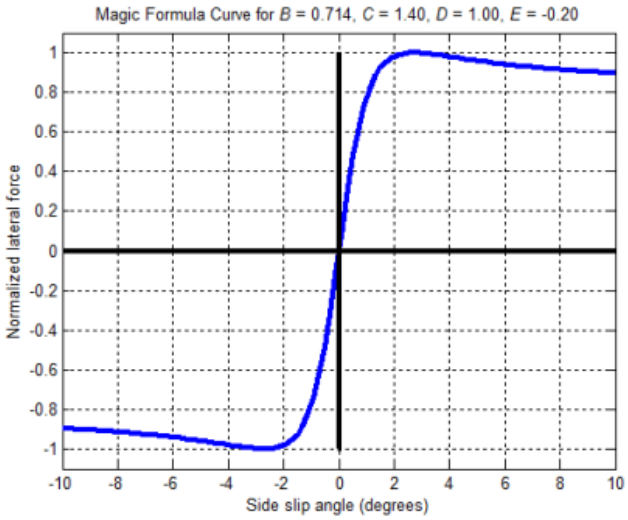

Abbildung 2.9: Tire Model

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 Section 2.3.3) und durch F_{max} begrenzt. F_{max} entspricht der maximalen Kraft, die der

Reifen übertragen kann.

$$F_x <= F_{max} \tag{2.3.14}$$

Der Zusammenhang zwischen lateraler und longitudinaler Kraft wird über den *Kammschen Kreis* Modelliert (siehe Schaubild 2.10). Dieser schränkt die wirkenden Kräfte so ein, dass die Hypotenuse aus F_x und F_y sich maximal auf einem Einheitskreis bewegen kann. Dieser hat den Radius der maximalen Kraft die der Reifen übertragen kann (F_{max}).

$$F <= F_{max} \tag{2.3.15}$$
$$F_{ll} = \sqrt{F_x^2 + F_y^2} \tag{2.3.16}$$
$$\tag{2.3.17}$$

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

$$\alpha = \arctan(F_x, F_y) \tag{2.3.18}$$
$$F_x = F_{max} \sin(\alpha) \tag{2.3.19}$$
$$F_y = F_{max} \cos(\alpha) \tag{2.3.20}$$
$$\tag{2.3.21}$$

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 Potential der Reifen auszunutzen.

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

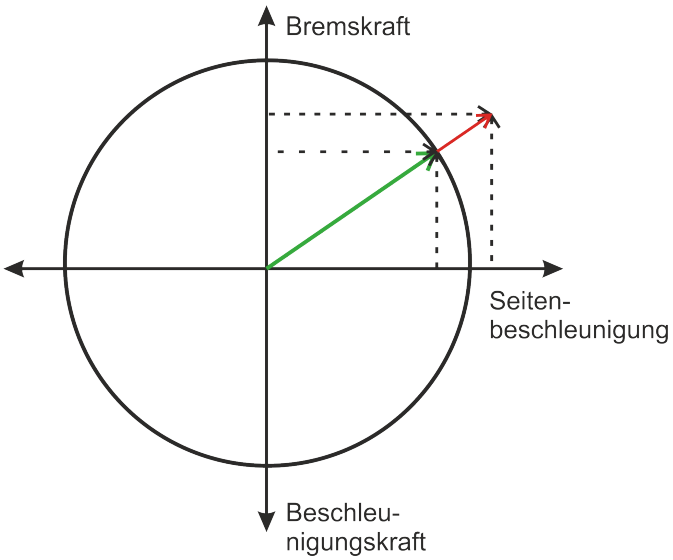

Abbildung 2.10: Kammscher Kreis: Zusammenhang von longitudinaler, lateraler und maximaler Reifenkraft

2.3.3 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.

$$ma_y = F_{yf} + F_{yr} \tag{2.3.22}$$

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 2.11).

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

$$I_z\ddot{\psi} = l_f F_{yf} - l_r F_{yr} \tag{2.3.23}$$

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

$$m\ddot{x} = m\dot{y}\dot{\psi} + F_x \tag{2.3.24}$$
$$m\ddot{y} = -m\dot{x}\dot{\psi} + F_y \tag{2.3.25}$$
$$I(\ddot{\psi}) = l_f F_{yf} - l_r F_{yr} \tag{2.3.26}$$

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

$$F_x = F_{xf} + F_{xr} \tag{2.3.27}$$
$$F_y = F_{yf} + F_{yr} \tag{2.3.28}$$

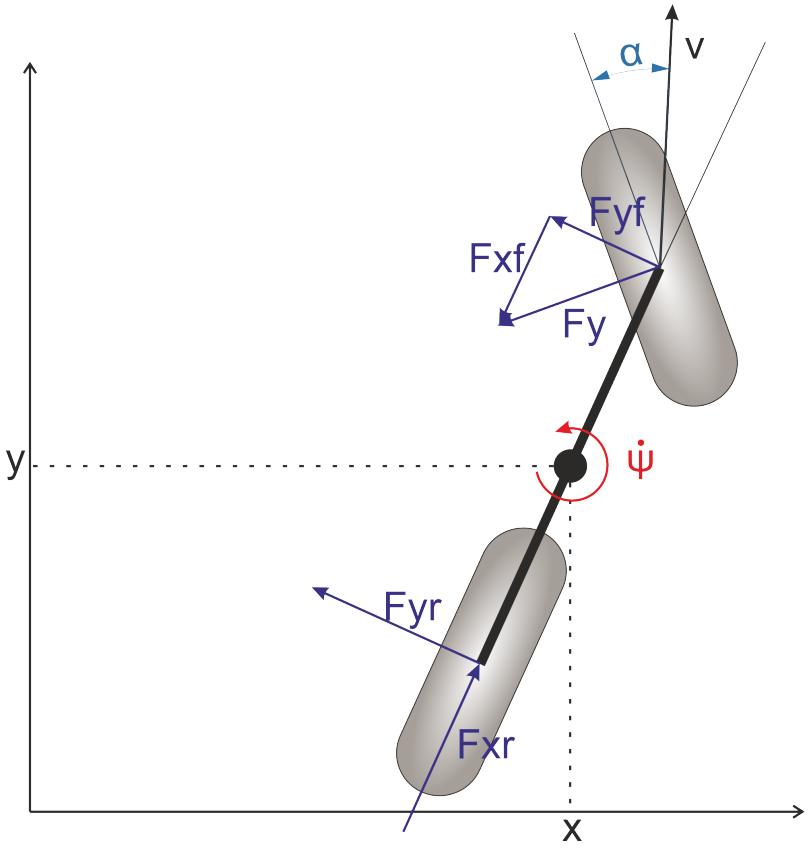

Abbildung 2.11: Dynamic Vehicle Model

Diese hängen ab von den lateralen $(.)_C$ und longitudinalen $(.)_l$ Radkräften und dem Lenkwinkel. Da das Vorderrad nicht angetrieben ist besitzt es keinen longitudinalen Anteil.

$$F_{xf} = -2F_{Cf}\sin(\delta_f) \tag{2.3.29}$$
$$F_{yf} = 2F_{Cf}\cos(\delta_f) \tag{2.3.30}$$
$$F_{xr} = F_{lr} \tag{2.3.31}$$
$$F_{yr} = 2F_{Cr} \tag{2.3.32}$$

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 2.3.2 berechnet. Die dafür benötigten Schräglaufwinkel werden durch folgende Formeln bestimmt:

$$\alpha_f = \delta_f - \arctan(\frac{\dot{y} + l_f\dot{\psi}}{\dot{x}}) \tag{2.3.33}$$
$$\alpha_r = -\arctan(\frac{\dot{y} - l_r\dot{\psi}}{\dot{x}}) \tag{2.3.34}$$

Longitudinale Kräfte

Da keine Simulation des Motors genutzt wird müssen die Kräfte welche die Reifen longitudinal auf die Straße bringen F_{lr} anders als über den Schlupf berechnet werden. Dies realisiert man über die maximale Motorleistung und F_{max}.

$$F_{lr_{acc}} = \frac{P_{engine} * throttle}{|\dot{x}|} \tag{2.3.35}$$
$$F_{lr_{acc}} <= 2F_{max} \tag{2.3.36}$$
$$F_{lr_{dec}} = -2F_{max} * break \tag{2.3.37}$$
$$\tag{2.3.38}$$

Die Berechnung der Reifenkräfte sind 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.

$$F_{reib} = m\mu g \tag{2.3.39}$$
$$F_{aero} = \frac{1}{2}\rho C_d A_f \dot{x}^2 \tag{2.3.40}$$

Die resultierende longitudinale Kraft (getrennt nach Beschleunigung und Bremsvorgang)

$$F_l r = F_{lr_{acc}} - F_{reib} - F_{aero} \tag{2.3.41}$$
$$F_l r = F_{lr_{dec}} - F_{reib} - F_{aero} \tag{2.3.42}$$
$$\tag{2.3.43}$$

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

$$\dot{X} = \dot{x}\cos(\psi) - \dot{y}\sin(\psi) \tag{2.3.44}$$
$$\dot{Y} = \dot{x}\sin(\psi) - \dot{y}\cos(\psi) \tag{2.3.45}$$
$$m\ddot{x} = m\dot{y}\dot{\psi} + F_{xf} + F_{xr} \tag{2.3.46}$$
$$m\ddot{y} = -m\dot{x}\dot{\psi} + F_{yf} + F_{yr} \tag{2.3.47}$$
$$I\ddot{\psi} = l_f F_{yf} - l_r F_{yr} \tag{2.3.48}$$
$$\tag{2.3.49}$$

Das Gleichungssystem kann auch in diskreter Form aufgestellt werden.

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 2.2: Vehicle Parameter

$$X_{k+1} = X_k + \Delta t(\dot{x}_k \cos(\Psi_k) - \dot{y}_k \sin(\Psi_k)) \tag{2.3.50}$$
$$Y_{k+1} = Y_k + \Delta t(\dot{x}_k \sin(\Psi_k) - \dot{y}_k \cos(\Psi_k)) \tag{2.3.51}$$
$$\Psi_{k+1} = \Psi_k + \Delta t\dot{\psi}_k \tag{2.3.52}$$
$$\dot{x}_{k+1} = \dot{x}_k + \Delta t(\frac{F_{xf_k}+F_{xr_k}-F_a}{m} + \dot{y}_k\dot{\psi}_k) \tag{2.3.53}$$
$$\dot{y}_{k+1} = \dot{y}_k + \Delta t(\frac{F_{yf_k}+F_{yr_k}}{m} - \dot{x}_k\dot{\psi}_k) \tag{2.3.54}$$
$$\dot{\psi}_{k+1} = \dot{\psi}_k + \Delta t\frac{l_f F_{yf} - l_r F_{yr}}{I} \tag{2.3.55}$$
$$\tag{2.3.56}$$

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).

2.4 Programmiersprachen

2.4.1 Python

Zu Beginn der Masterarbeit wurde zuerst die Programmiersprache Python verwendet um das MPC 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 das MPC für die Fahrzeugregelung auszulegen. So lange die Anzahl der Prädiktionsschritte klein bleibt und 10 nicht überschreitet ist die Ausführungszeit gering genug um eine Updaterate von 20Hz zu erreichen.

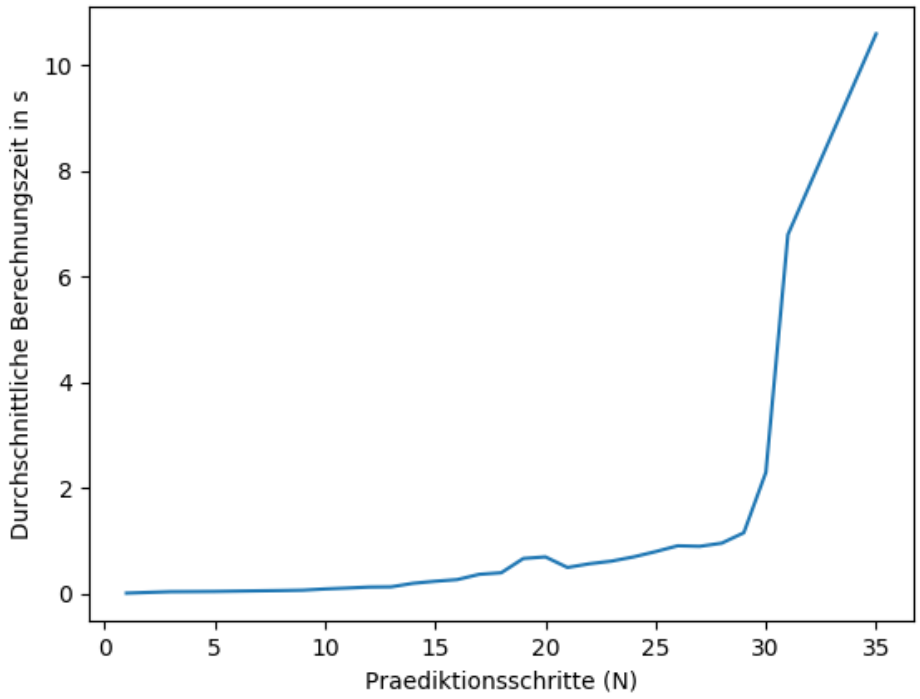

Abbildung 2.12: Ausführungszeiten für verschiedene große Horizontlängen

Bei größeren Prädiktionsvektoren fällt die Geschwindigkeit jedoch sehr schnell ab. 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 2.2.2 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.

2.4.2 Julia

Die Sprache welche die Umsetzung des Model Predictive Control-Algorithmus effizient und performant ermöglichte ist die Programmiersprache Julia. 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 genutzten Funktionen im wissenschaftlichen Umfeld ist in dem Benchmark 2.13 zu sehen.

Die wichtigsten Feautres von Julia sind:

- multiple dispatch: Wird genutzt um die gleiche Methode für verschiedene Kombinationen an 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 kontinuierliche Analysen durchgeführt werden um Bereiche ausfindig zu machen die durch ein Neukompilieren eine signifikante Verbesserung bei der Ausführungszeit erfahren würden.

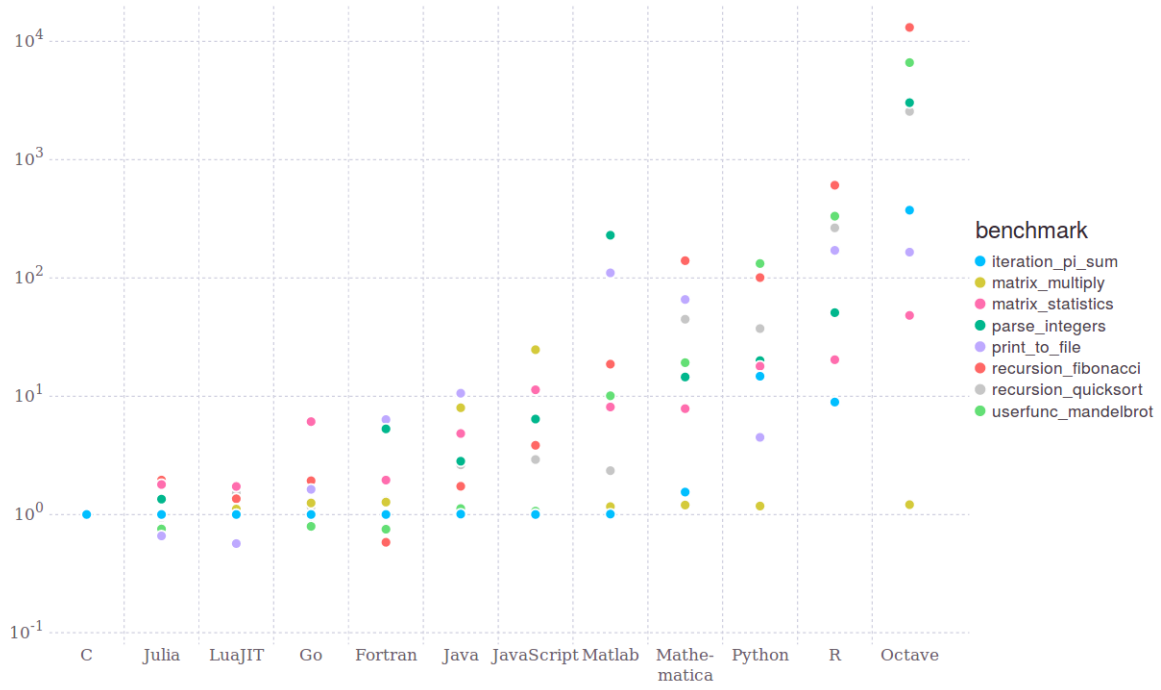

Abbildung 2.13: Ausführungszeiten für verschiedene Sprachen

Jump

An diesem Punkt kommt Jump ins Spiel. Jump ist eine Erweiterung für Julia mit der man 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 4.1). Ebenfalls ersichtlich ist dass Jump sich direkt um die automatische Ableitung kümmert und so dem Programmierer sehr viel Arbeit erspart.

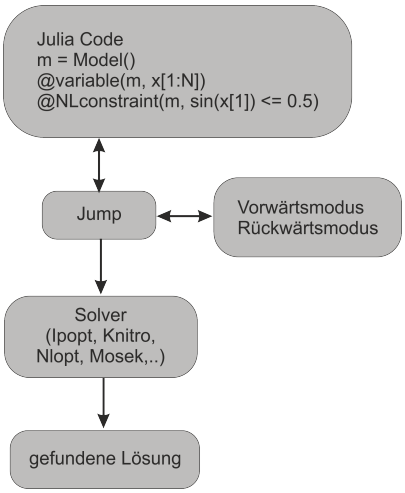

Abbildung 2.14: Blockdiagramm von Jump

3 MPC zur Trajektionsplanung und Regelung

Im Folgenden werden die Bestandteile des Model Predictive Control-Algorithmus zur Trajektionsplanung und Regelung für das Rennauto aufgeschlüsselt und Veranschaulicht.

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

- Einfachere Anpassung der Parameter für reales Fahrzeug.
- Robustheit. Der Ausfall eines Reglers würde zumindest noch eine eingeschränkter Kontrolle ermöglichen.
- Zwei einfacher zu berechnende Probleme. Da die Berechnungszeit nicht linear mit der Komplexität steigt wären hier möglicherweise deutliche Geschwindigkeitssteigerungen möglich.
- Weniger Komplexes MPC 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 sind. Durch die Trennung kann keine optimale Lösung mehr gefunden werden. Zudem ist die Integration komplexer da beide Regler gleichzeitig laufen müssen und ihre Berechnungen miteinander Austauschen müssen. Auf Grund 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 2.3.1). 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 ist, besteht der Vektor also aus $N + 1$ mal $\vec{x}$. In der Abbildung 3.1 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 $\frac{1}{20}s$. Der Vorteil der hieraus entsteht ist dass bei einer Verzögerung der Positionsschätzung der nächsten im Prädiktionshorizont

berechneten Steuerbefehl genutzt werden kann. Dies geht solange bis die Zeitverzögerung genauso groß ist wie die gesamte Zukunftsprädiktion. Obwohl man diese Werte zum Regeln nutzen kann, nimmt aufgrund von Modellfehlern die Güte der berechneten Werte mit n zu. Man ist also bestrebt den Algorithmus mit der gleichen Rate wie die Positionsschätzung laufen zu lassen.

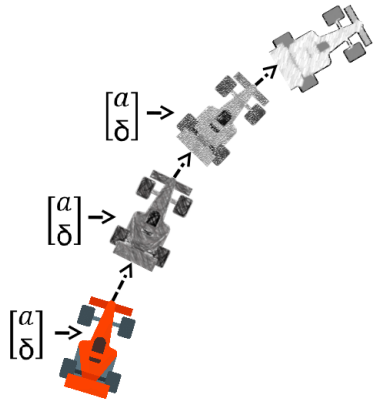

Abbildung 3.1: Grafische Visualisierung der Prädiktion abhängig von den Steuerparametern für drei Schritte in die Zukunft

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

3.1 Fahrzeugmodell

Zu Beginn haben die Teilvektoren $\vec{x}$ keinen Zusammenhang untereinander, da aber 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 2.3.1 genutzt um immer zwei aufeinander folgende Schritte miteinander zu verknüpfen. Die entstehenden Gleichheitsbedingungen sind im Anhang unter **??**ufgeführt.

Zusätzlich zu der Systembeschreibung fehlen aber 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 dazu hat das Rennauto zu beschleunigen.

3.2 Kostenfunktionen

Erst eine geeignete Kostenfunktion führt zu der Planung einer Trajektorie die das Rennauto um den Kurs führt. Es wurden mehrere 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 bei einem 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 das 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 3.3 zu sehen.

3.3 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 hierfür die Kosten schneller minimal werden. Um dies zu verhindern wurde eine zusätzliche Streckenbeschränkung eingeführt. Diese basiert auf zwei Tangenten die in jedem Prädiktionsschritt an die Fahrbahn Außen-, und Innenseite projiziert werden und

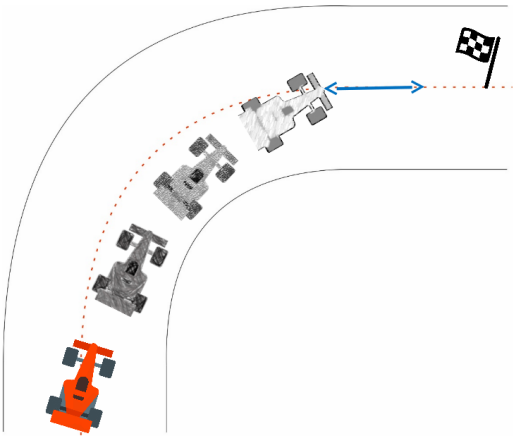

Abbildung 3.2: Zielpunkt Distanzminimierung: Das virtuelle Ziel wird immer weit genug vor dem Fahrzeug her geführt, so dass es nie erreicht werden kann.

einer Bedingung die nur erfüllt ist, solange sich das Fahrzeug innerhalb dieser Tangenten bewegt.

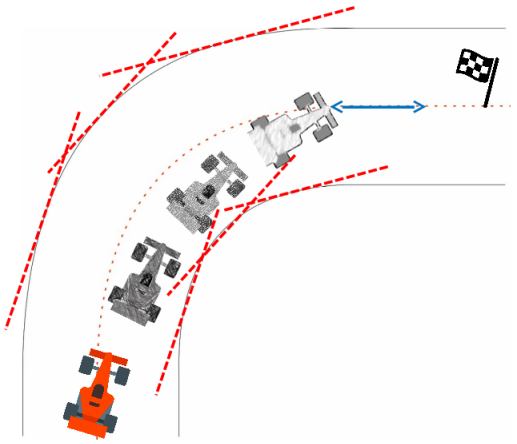

Abbildung 3.3: Tangentiale Begrenzung hält das Fahrzeug auf der Fahrspur

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 gleichbleibenden 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 realisieren $d = \frac{a*b}{|b|}$ mit a als Vektor vom Mittelpunkt der Strecke zum Massenschwerpunkt des Rennautos und b als Vektor vom Mittelpunkt zum Rand des Rennkurses. In diesem Fall können sich die Räder des Fahrzeuges über den Rand hinaus bewegen. Soll das 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 ist nötig um zu verhindern, dass der Optimierer einfach die x-, und y-Position auf einen Punkt legt welcher die Kostenfunktion ideal minimiert.

Elastische Distanzminimierung

Der Nachteil beider oben aufgeführten Kostenfunktionen ist, dass sie kein Spielraum beim erfüllen der Streckenbeschränkung haben. In der Simulationsumgebung kann, wenn sowohl das im Simulator wie auch im MPC hinterlegte Fahrzeugmodell das gleiche 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 was bei nur dem kleinsten Regelfehler, Rutschen, abweichungen vom Fahrzeugmodell etc, dazu führen würde dass sich der inertiale Fahrzeugzustand $\vec{x}_0$ bereits außerhalb der Beschränkungen befindet und damit keine Lösung mehr für das Optimierungsproblem gefunden werden kann. 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 erweitert durch ein Kostenanteil 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.
$f(\vec{x}) = a(\sum_1^{N+1} v_i) + b(\sum_1^{N+1} g(\vec{d_i}))$
Die Funktion $g(\vec{x}_i)$ dient hier als Platzhalter für verschiedene Gewichtungsfunktionen. Die Distanz d vom Mittelpunkt der Strecke wird genauso wie bei der tangentialen Beschränkung durch eine Vektorprojektion berechnet.

$$\alpha d^2 \tag{3.3.1}$$

$$\alpha |d| \tag{3.3.2}$$

$$e^{\alpha(k_1+d)} + e^{-\alpha(k_2+d)} \tag{3.3.3}$$

$$\left|\frac{\alpha}{k_1-d} + \frac{\alpha}{k_2-d}\right| \tag{3.3.4}$$

Die daraus resultierenden Graphen sind zu Veranschaulichung in Abbildung 3.4 dargestellt.

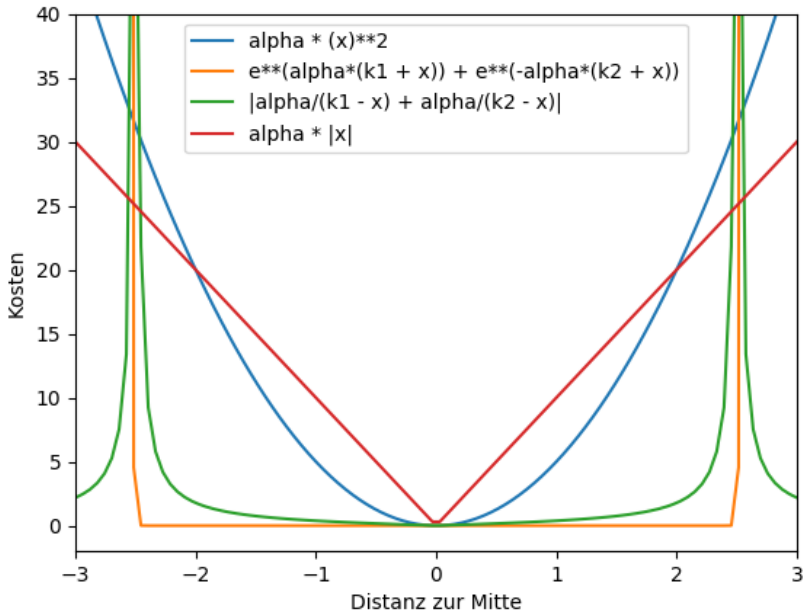

Abbildung 3.4: Verschiedene Distanzmaße um das Verhalten der Trajektionsplanung zu beeinflussen.

Die Idee hinter den Funktionen 3.3.3 und 3.3.4 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 also stellen diese Kostenfunktionen die tangentialen Begrenzungen ideal nach.

4 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. Im folgenden wird zuerst auf die Funktionsweise der Engine eingegangen, da sie das Grundgerüst bildet in welches später der MPC-Algorithmus und die Fahrzeugsimulation eingebettet werden.

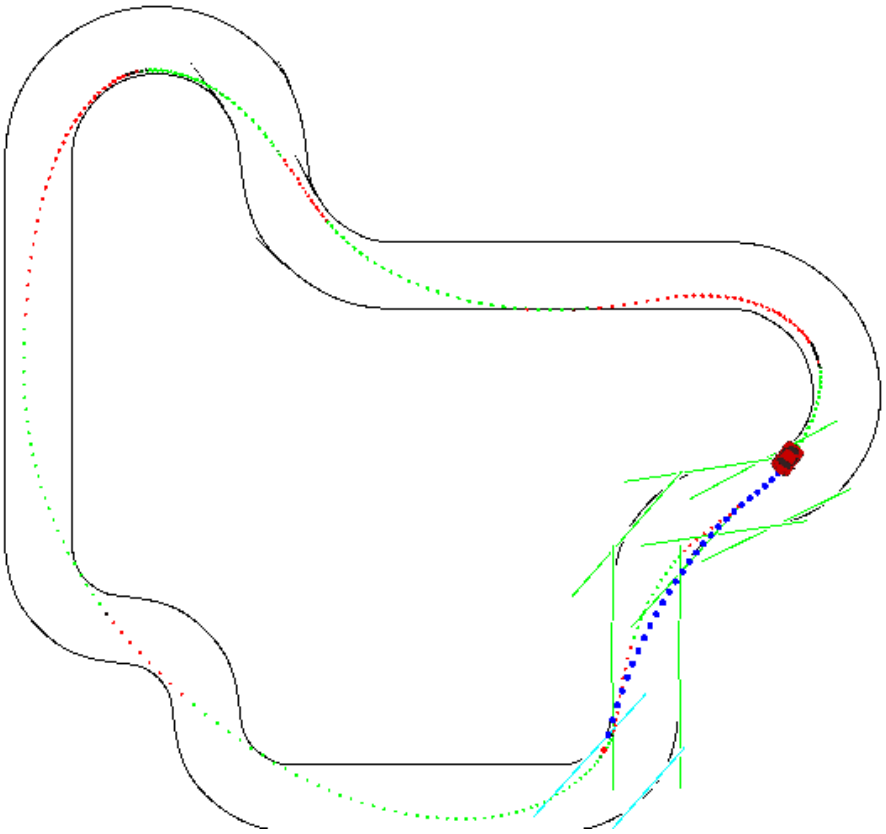

Abbildung 4.1: Grafische Darstellung des MPC - Algorithmus in der Simulationsumgebung

4.1 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 Größenvehältnisse 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 4.2 nochmals zusammengefasst.

4.2 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

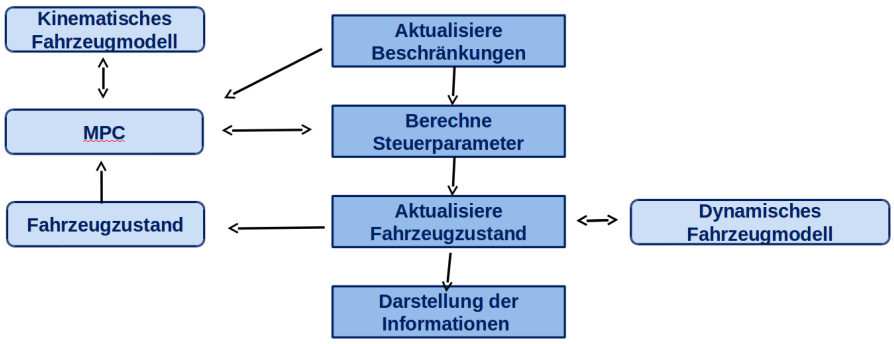

Abbildung 4.2: Blockdiagramm der Simulation ohne MPC

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 20Hz 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 9m, maximallänge einer Geraden von 80m und maximal 180° Spitzkurven besitzt.

4.2.0.1 Suchbereich einschränken (constraint)

4.2.0.2 Implementierung in ROS

4.3 Simulation

5 Evaluierung

5.1 Verifizierung der Fahrzeugmodelle

5.1.1 Kinematisches Modell constraint of beta

5.2 Regelung verschiedener Modelle

5.2.1 Was hat beta für einen Einfluss auf die Regelung?

5.3 Kostenfunktionen

6 Zusammenfassung und Ausblick

6.1 Dynamisches Fahrzeugmodell im MPC

6.2 Trajektionsregelung

6.3 Zweispurmodell

6.4 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

[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.

[jul] Juliadiff. `http://www.juliadiff.org/`. Accessed on 2018-06-01.

[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.

[Raj11] R. Rajamani. *Vehicle Dynamics and Control*. Mechanical Engineering Series. Springer US, 2011.

[SH16] Heiko G. Seif and Xiaolong Hu. Autonomous driving in the icity—hd maps as a key challenge of the automotive industry. *Engineering*, 2(2):159 – 162, 2016.

[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.

[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