# Methoden zur Berechnung optimaler Rennlinien im dynamischen Grenzbereich

Andreas Karl-Phillip Huber

Vollständiger Abdruck der von der Fakultät für Luft- und Raumfahrttechnik der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines

**Doktors der Naturwissenschaften**

genehmigten Dissertation.

Gutachter:

1. Univ. Prof. Dr. rer. nat. Matthias Gerdts
2. apl. Prof. Dr. rer. nat. Kurt Chudej

Die Dissertation wurde am 15. Juni 2020 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Luft- und Raumfahrttechnik am 24. Oktober 2020 angenommen. Die mündliche Prüfung fand am 27. November 2020 statt.

# Zusammenfassung

Im Rahmen dieser Dissertation werden verschiedene Methoden zur Berechnung optimaler Rennlinien im dynamischen Grenzbereich untersucht. Dabei kommen verschiedene Fahrzeugmodelle und Optimierungsalgorithmen zum Einsatz. Zunächst berechnen wir mit den verschiedenen Modellen und Algorithmen offline optimierte Trajektorien. Daraufhin werden ausgewählte Trajektorien mit einem Versuchsfahrzeug getestet und in Zusammenarbeit mit Testfahrern bewertet. Die besten Trajektorien dienen anschließend als Vergleichsrennlinie für die Entwicklung eines vereinfachten Fahrzeugmodells, das sich zur online-Optimierung von Rennlinien auf Rennstrecken eignet.

Für dieses vereinfachte Fahrzeugmodell entwickeln wir ein Optimalsteuerungsproblem, das als Onlinebahnplanung auf einem Versuchsfahrzeug berechnet wird. Zentraler Bestandteil des vereinfachten Fahrzeugmodells ist eine Bogenlängenparametrisierung und eine Entkopplung der Berechnung von Geschwindigkeitsprofil und Trajektorie. Nach den ersten erfolgreichen online-Testfahrten werden auch Rennlinien mit Fahrbahnhindernissen getestet. Die Vorausschau der online berechneten Trajektorien ist durch die Rechenzeiten des verwendeten Optimierers OCPID-DAE1 beschränkt.

Zur Verbesserung der Rechenzeiten wird im Rahmen dieser Arbeit ein neuer Optimierer zur Lösung von Optimalsteuerungsproblemen namens OCPbasic entwickelt. Dieser Optimierer zeichnet sich durch eine geschickte Strukturausnutzung der linearen Gleichungssysteme aus, welche beim Lösen von Optimalsteuerungsproblemen entstehen. Im Vergleich zu namhaften Softwareroutinen zur Lösung von dünn besetzten linearen Gleichungssystemen, wie MA57, verbessern sich die Rechenzeiten durch die Strukturausnutzung der linearen Gleichungssysteme deutlich. Zuletzt wird der Optimierer OCPbasic auf das vereinfachte Fahrzeugmodell der Onlinebahnplanung angewandt. Durch die verbesserte Rechenzeit können wir mit OCPbasic für das vereinfachte Fahrzeugmodell auch Trajektorien mit einer höheren Vorausschau effizient berechnen.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Kapitel 1

# Einführung

## 1.1 Motivation und Forschungsschwerpunkt

Autonomes Fahren ist derzeit, neben der Elektromobilität und anderer regenerativer Antriebstechniken, einer der zentralen Forschungsschwerpunkte in der Automobilindustrie. Das Ziel dieser Forschungen im Automobilbereich ist es, dass Fahrzeuge immer mehr Aufgaben des Fahrers übernehmen. Beim voll autonomen Fahren soll sogar komplett auf Fahrereingriffe verzichtet werden können. Um diese Zielsetzungen zu erreichen, muss eine Vielzahl von technischen Herausforderungen gelöst werden. Diese Herausforderungen beginnen bereits bei einer fehlerfreien Erkennung der Fahrzeugumgebung. Dazu benötigen Fahrzeuge eine Ausstattung mit zahlreichen Sensorsystemen, wie GPS, Radarsensoren, Lidar-Sensoren und Kameras. Die resultierenden Datenmengen müssen während der Fahrt durch Rechnerhardware im Fahrzeug verarbeitet und ausgewertet werden. Das sich daraus ergebende Fahrzeugumfeld wird anschließend an eine Bahnplanereinheit übergeben, welche die Umgebungsdaten bewertet und die geplante Fahrzeugtrajektorie an die Situation anpasst. Neben der Bahnplanereinheit muss auch eine Steuer- und Regelungseinheit entwickelt werden, welche die geplante Trajektorie mit der elektromechanischen Aktuatorik von Motor, Lenkung und Bremse umsetzt. Diese Arbeit wird sich vor allem mit der Bahnplanung befassen.

Eine der stärksten Motivationen für die Entwicklung des autonomen Fahrens ist die Erhöhung der Sicherheit im Straßenverkehr im Bezug auf Fahrfehler und Fehleinschätzungen von Autofahrern. Dafür sollten autonome Fahrzeuge auch in der Lage sein, in extremen Situationen richtig zu handeln. Dementsprechend müssen die Umfelderkennung, Bahnplanung, Steuer- und Regelungsalgorithmen auch im dynamischen Grenzbereich funktionieren. Um an den einzelnen Bereichen für das autonome Fahren forschen zu können, sind

möglichst isolierte Tests der einzelnen Forschungsschwerpunkte durchzuführen. Zur Auslegung der Bahnplanung, Steuer- und Regelungsalgorithmen für den dynamischen Grenzbereich, wird auf Rennstrecken zurückgegriffen. Hier kann ein autonomes Fahrzeug, ohne andere Verkehrsteilnehmer zu gefährden, im Bereich der dynamischen Grenzen fahren. Natürlich kann auch dabei nicht komplett auf Umweltdaten verzichtet werden. Jedoch lassen sich Daten, wie die Fahrzeugposition, auf fest definierten Rennstrecken leichter und genauer bereitstellen, als dies im normalen Straßenverkehr möglich wäre. Bei der Bahnplanung und den Steuer- und Regelungsalgorithmen müssen wir zwischen Echtzeit und online-Algorithmen unterscheiden. Während die Steuer- und Regelungsalgorithmen in Echtzeit benötigt werden, kann die Bahnplanung auch online erfolgen. Der Begriff Echtzeit ist nach der DIN ISO/IEC 2382 (früher DIN 44300) definiert, welche im wesentlichen fordert, dass anfallende Daten sofort verarbeitet werden und die Berechnungsergebnisse nach einer festgelegten Zeitspanne verfügbar sind. Anders lässt sich der Begriff online mit weicheren Anforderungen versehen, die wir benötigen, um für die Bahnplanung auch iterative Algorithmen aus der Optimalsteuerung verwenden zu können. Unter einer Onlinebahnplanung verstehen wir in dieser Arbeit einen Algorithmus, der in den meisten Fällen in einer festgelegten Zeitspanne eine optimale Trajektorie berechnen kann. Zusätzlich muss die Bahnplanung robust genug sein, um noch während des Abfahrens der letzten zulässigen Trajektorie eine neue Trajektorie zu generieren. Im Rahmen der in dieser Arbeit durchgeführten Fahrversuche wurde das Versuchsfahrzeug und entsprechende Steuer- und Regelungsalgorithmen von der Forschungs- und Entwicklungsabteilung des Projektpartners der Volkswagen AG bereitgestellt und betreut.

## 1.2 Stand der Technik

Wie beschrieben sind für diese Arbeit unter den vielen Forschungsbereichen zum autonomen Fahren insbesondere die Bereiche Umfelderkennung, Steuer- und Regelungsalgorithmen und Bahnplanung relevant. Zu allen drei Bereichen gibt es mehrere Forschungsprojekte mit verschiedensten Forschungsergebnissen.
Im Bereich der Umfelderkennung spielen vor allem Sensoren eine wichtige Rolle. Einen Überblick über die verschiedensten Sensoren, welche im Fahrzeugbereich Anwendung finden gibt [26]. Erste Ergebnisse zur Umfelderkennung im Bereich des autonomen Fahrens lassen sich bereits 1986 bis 1989 in [30, 24, 25] finden. Aufbauend auf diese Arbeiten sind zahlreiche weitere Forschungsergebnisse entstanden. Im Bereich der Objektwahrnehmung und Umgebungserfassung können die Arbeiten [56, 64] genannt werden. Weitere Ergebnisse zur Objektbildung aus Sensormessdaten finden sich in [67]. Durch Objekterkennung

gestützt können auch wie in [63] Navigationsstrategien abgeleitet werden.
Eine Arbeit zur Bahnplanung und Regelung eines Elektrofahrzeugs außerhalb des dynamischen Grenzbereichs findet sich in [65]. Die grundlegende Idee von der Kopplung Regler und Bahnplanung ist in [65] ähnlich zum Ansatz dieser Arbeit. Eine besondere Herausforderung im Bezug auf das autonome Fahren auf Rennkursen, ist die Auslegung eines Bahnfolgereglers für den dynamischen Grenzbereich, welche in [53] gezeigt wird.
Aufgrund des großen Forschungsinteresses in der Automobilindustrie gibt es einige Forschungsprojekte und Ergebnisse zur Bahnplanung autonomer Fahrzeuge. Bereits 2009 wurde im Rahmen eines Vorläuferprojekts, eine durch Optimalsteuerung berechnete Bahnplanung [42], durch einen Bahnfolgeregler mit einem realen Versuchsfahrzeug nachgefahren. Die Berechnung der Trajektorie erfolgte dabei allerdings noch offline, somit vor der Testfahrt. Simulierte Ergebnisse zur Bahnplanung durch Optimalsteuerung auf einem Moving Horizon lassen sich bereits in [39, 38] finden. Weitere grundlegende Untersuchungen bezüglich Fahrdynamik und optimaler Steuerung von Fahrzeugen sind in [57, 68] zu finden. In [46] werden separate Algorithmen zur Fahrzeuglängsführung und Fahrzeugquerführung vorgestellt und auch in einem Versuchsfahrzeug getestet. Eine ähnliche Arbeit, die sich mit der Generierung von Trajektorien zu zeitkritischen Verkehrsszenarien befasst ist in [73] zu finden. Auch die Konzepte dieser Arbeit wurden durch Testfahrten mit einem Versuchsfahrzeug validiert. Eine Methode zur schnellen Berechnung von Rennlinien im dynamischen Grenzbereich ist in [54] beschrieben. Diese ist auch mit Fahrzeugtests und einem Vergleich mit Testfahrern validiert. Darauf aufbauend werden auch Algorithmen zur online-Umplanung für Hindernisse in [33] und [66] vorgestellt.
Weitere Forschungsergebnisse bezüglich einer Optimierung der Rundenzeiten auf Rennkursen sind in [55] zu finden. Auch für die Berechnung von Trajektorien zum Überholen auf Rennkursen werden in [17] Methoden vorgestellt.
Ein weiteres Forschungsgebiet, das für diese Arbeit relevant ist, sind Algorithmen aus der Optimalsteuerung. Die Grundlagen zur Optimalsteuerung sind in [2, 18, 13, 40, 4] beschrieben. Interessant sind allerdings insbesondere leistungsstarke, robuste und schnelle Optimierer. Dabei spielen in der heutigen Forschung vor allem Algorithmen für nichtlineare restringierte Optimierungsprobleme [43, 60, 35, 20] eine große Rolle. Die Leistungsstärke und Robustheit von Optimierern ist dabei, insbesondere von einer effizienten Implementierung, wie in [72] und [52] beschrieben abhängig. Für einen besseren Überblick wurde eine Auswahl von aktuellen Softwarepaketen zur Lösung von Optimalsteuerungsproblemen zusammengestellt.

1. FORCES PRO: [27, 28]
   Ein speziell für nichtlineare modellprädiktive Regelung entwickelter kommerzieller

Optimierer, der insbesondere auf online-Lauffähigkeit ausgelegt ist.

2. FALCON.m: [61]
   Eine freie Software zur Diskretisierung von Optimalsteuerungsproblemen in MATLAB, die zur Implementierung, zum Lösen und zur Analyse von Optimalsteuerungsproblemen verwendet werden kann. Zur Lösung von restringierten nichtlinearen diskreten Optimierungsproblemen können verschiedene Optimierer, wie IPOPT [22, 70, 71] oder SNOPT [44] verwendet werden.

3. CasADi: [3]
   Eine freie Software für algorithmische Differenzierungsverfahren, welche zur Diskretisierung und zum Lösen von Optimalsteuerungsproblemen verwendet werden kann. Zur Lösung von restringierten nichtlinearen diskreten Optimierungsproblemen können verschiedene Optimierer, wie IPOPT [22, 70, 71] oder SNOPT [44] verwendet werden.

4. PINS: [7, 5, 6]
   Ein Optimierer zur Lösung von Optimalsteuerungsproblemen durch ein indirektes Penalty-Verfahren. Beutet gezielt die Struktur der linearen Gleichungssysteme eines indirekten Penalty-Verfahrens aus. Gestützt durch ein Softwaretool für Maple kann, durch symbolische Differentialrechnung, für den Optimierer, passender C++ Code, für verschiedenste Optimalsteuerungsprobleme, erzeugt werden.

5. OCPID-DAE1: [36]
   Ein Optimierer des Instituts für Angewandte Mathematik und Wissenschaftliches Rechnen, der Universität der Bundeswehr München von Matthias Gerdts. Implementiert unter Verwendung von BFGS-Updates, Einfach- und Mehrfachschießverfahren und der Lösung von restringierten nichtlinearen diskreten Optimierungsprobleme durch ein SQP-Verfahren mit dicht besetzten Matrizen. Dieser Optimierer wird im Rahmen dieser Arbeit in Kapitel 3 zur Lösung von Optimalsteuerungsproblemen verwendet.

6. OCPBASIC: [51, 48]
   Ein im Rahmen dieser Arbeit entwickelter Optimierer, welcher auf Basis einer eigenständigen Implementierung eines Innere-Punkte-Verfahrens (IPBASIC [50]) entsteht. Dieser Optimierer beutet gezielt die Struktur der linearen Gleichungssysteme von diskretisierten Optimalsteuerungsproblemen aus. Eine ausführliche Beschreibung der Funktionsweise dieses Optimierers kann in Kapitel 4 gefunden werden.

Im Rahmen dieser Arbeit werden die Optimierer OCPID-DAE1 und OCPbasic zur Lösung von Optimalsteuerungsproblemen verwendet. Beide Optimierer wurden am Institut für Angewandte Mathematik und Wissenschaftliches Rechnen der Universität der Bundeswehr München entwickelt. Dabei liegt das Forschungsinteresse im Vordergrund. Da für beide Optimierer der gesamte Quellcode vorliegt, kann die Software gezielt für den Einsatzbereich in der Onlinebahnplanung angepasst werden.

## 1.3 Zielsetzungen und eigene Beiträge

Ziel dieser Arbeit ist es, verschiedene Methoden und Algorithmen der Optimalsteuerung im Hinblick auf Onlinebahnplanung zur Berechnung von Rennlinien zu untersuchen. Dabei sollen unterschiedlich komplexe Fahrzeugmodelle und dynamische Beschränkungen herangezogen werden. Zunächst wollen wir mit Hilfe von offline berechneten Trajektorien die Qualität der Rennlinien von einzelnen Bahnplanern durch ein Nachregeln der Bahnen messen und im Anschluss gemeinsam mit Testfahrern bewerten. Anhand der Bewertung einzelner Methoden und Algorithmen hinsichtlich der Qualität der Rennlinien, der Robustheit des Optimierers und der Länge der Rechenzeiten, wollen wir im Anschluss einen der Ansätze auswählen, um eine Onlinebahnplanung zu realisieren. Diesen Algorithmus wollen wir anschießend für den Online-Betrieb auf einem Versuchsfahrzeug anpassen und auf einer Rennstrecke testen.
Die Forschungsbeiträge, welche im Rahmen dieser Zielsetzungen entstanden sind, umfassen folgende Punkte. Ein erster Beitrag ist die Parametrisierung und Weiterentwicklung eines Optimalsteuerungsproblems für das Einspurmodell zur Berechnung optimaler Rennlinien. Die Weiterentwicklung erfolgt anhand des in [42] entwickelten und beschriebenen Optimalsteuerungsproblems. Ein weiterer Beitrag ist die Entwicklung eines Algorithmus zu Berechnung von Rennlinien durch dynamische Programmierung. Hierzu wurde im Rahmen Arbeit auch eine Publikation [47] veröffentlicht. Über den Umfang von [47] hinaus definieren wir im Rahmen dieser Arbeit auch eine modifizierte Variante des Algorithmus, welche den Speicheraufwand reduziert. Einer der zentralen Forschungsbeiträge ist die Entwicklung eines vereinfachten Fahrzeugmodells und die Entkopplung der Berechnung von Trajektorien und Geschwindigkeitsprofilen. Dieser Algorithmus entspricht einem guten Kompromiss zwischen der Qualität der Rennlinien und den benötigten Rechenzeiten. Der wichtigste Punkt hinsichtlich der Zielsetzung dieser Arbeit ist, dass eine Onlinebahnplanung unter Verwendung des vereinfachten Fahrzeugmodells und der Entkopplung der Berechnung von Trajektorien und Geschwindigkeitsprofilen realisiert werden kann. Diese Onlinebahnplanung wird im Rahmen dieser Arbeit auch auf einem Versuchsfahrzeug

getestet.
Ein Forschungsbeitrag, der Onlinebahnplanung bezüglich der Rechenzeiten weiter verbessern soll ist die Entwicklung eines Verfahrens zur Strukturausnutzung eines Optimalsteuerungsproblems in einem Innere-Punkte-Verfahren. Zur grundlegenden Funktionsweise dieser Strukturausnutzung und des zugehörigen Verfahrens wurde im Rahmen Arbeit auch eine Publikation [48] veröffentlicht. Um die Strukturausnutzung realisieren zu können wird ein Innere-Punkte-Verfahren nach Vorbild zu IPOPT erarbeitet, welches in Form Softwarepakets namens IPBASIC implementiert wird. Durch eine direkte Diskretisierung von Optimalsteuerungsproblemen und die Strukturausnutzung des zu lösenden linearen Gleichungssystems entsteht das Optimierungspaket OCPBASIC. OCPBASIC ist demnach eine eigenständige Optimierungssoftware zur Lösung von Optimalsteuerungsproblemen, welche aus dieser Arbeit hervorgeht. Besonders hervorzuheben ist, dass bezüglich der Rechenzeiten, OCPBASIC mit Optimierern wie IPOPT mithalten beziehungsweise diese auch schlagen kann. Ein weiterer Vergleich erfolgt im Bezug auf das Lösungsverfahren zu Lösung des linearen Gleichungssystems. Hierbei wird die Lösung des linearen Gleichungssystems mit Hilfe der Strukturausnutzung, mit der Softwareroutine MA57 aus [29] verglichen. Die Softwareroutine MA57 dient zur Lösung von dünn besetzten symmetrischen linearen Gleichungssystemen. Gerade bei einer Diskretisierung mit vielen Gitterpunkten kann die Strukturausnutzung die Softwareroutine MA57 bezüglich der Rechenzeiten deutlich schlagen. Zuletzt wird der Optimierer OCPBASIC auch noch zur Lösung des vereinfachten Fahrzeugmodells, welches bei Online-Fahrversuchen verwendet wurde getestet. Durch die verbesserten Rechenzeiten können mit OCPBASIC auch Rennlinien mit einer deutlich längeren Vorausschau in einer online-fähigen Rechenzeit berechnet werden.

# Kapitel 2

# Methoden der Optimierung

## 2.1 Grundlagen der nichtlinearen Optimierung

Verschiedene Klassen von Optimierungsproblemen treten in den unterschiedlichsten Disziplinen, wie den Naturwissenschaften, Ingenieurwissenschaften oder auch den Wirtschaftswissenschaften auf. In dieser Arbeit wollen wir uns hauptsächlich mit nichtlinearen restringierten Optimierungsproblemen befassen, welche sich beispielsweise aus einer direkten Diskretisierung von Optimalsteuerungsproblemen ergeben. Verschiedene Algorithmen, Methoden und Verfahren zur Lösung von nichtlinearen Optimierungsproblemen werden in [41], [43], [60], [35] und [20] vorgestellt.

### 2.1.1 Problemstellung

Zunächst betrachten wir eine allgemeine Problemklasse für endlichdimensionale Optimierungsprobleme. Für diese Problemklasse definieren wir das restringierte nichtlineare Optimierungsproblem:

**Problem 2.1 (Restringiertes nichtlineares Optimierungsproblem (NLP))**

Bestimme den Vektor der Optimierungsvariablen $x \in \mathbb{R}^n$, so dass die Zielfunktion

$$f(x) \tag{2.1}$$

minimal wird, unter den Nebenbedingungen

$$h(x) = 0 \tag{2.2}$$

$$g(x) \leq 0 \tag{2.3}$$

mit $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^p$ und $g : \mathbb{R}^n \to \mathbb{R}^q$.

Das Problem 2.1 beschreibt eine allgemeine nichtlineare Optimierungsaufgabe, welche zum Einen durch die Gleichungsnebenbedingungen (2.2) und zum Anderen durch die Ungleichungsnebenbedingungen (2.3) beschränkt ist.
Mit diesen Nebenbedingungen lässt sich der zulässige Bereich des nichtlinearen restringierten Optimierungsproblems durch

$$\Sigma := \{x \in \mathbb{R}^n \,|g_i(x) \leq 0, i = 1, \ldots, q; h_j(x) = 0, j = 1, \ldots, p\}$$

beschreiben. Für alle zulässigen Optimierungsvariablen $x \in \Sigma$ ist besonders interessant, welche der Ungleichungsnebenbedingungen (2.3) aktiv sind. Deshalb führen wir für zulässige Optimierungsvariablen die Menge der aktiven Ungleichungen

$$\mathcal{A}(x) := \{i \in \{1, \ldots, q\} \,|g_i(x) = 0\} \tag{2.4}$$

ein. In der Menge $\mathcal{A}(x)$ werden somit alle Ungleichungen, welche durch Gleichheit $g_i(x) = 0$ erfüllt sind als aktiv und alle anderen $g_i(x) < 0$ als inaktiv angesehen.
Nach [40] können wir für Problem 2.1 sowohl lokale als auch globale Minima definieren.

**Definition 2.1 (Globales Minimum, vgl. [40])**

Der Vektor $\hat{x} \in \Sigma$ ist genau dann ein globales Minimum, wenn

$$f(\hat{x}) \leq f(x) \qquad \forall x \in \Sigma.$$

Falls für den $\hat{x} \in \Sigma$ zusätzlich die Bedingung

$$f(\hat{x}) < f(x) \qquad \forall x \in \Sigma$$

erfüllt ist, entspricht $\hat{x}$ einem strikten globalen Minimum.

**Definition 2.2 (Lokales Minimum, vgl. [40])**

Der Vektor $\hat{x} \in \Sigma$ ist genau dann ein lokales Minimum, wenn ein $\varepsilon > 0$ mit einer Umgebung $\mathcal{U}_\varepsilon(\hat{x})$ existiert für die gilt

$$f(\hat{x}) \leq f(x) \qquad \forall x \in \Sigma \cap \mathcal{U}_\varepsilon(\hat{x}).$$

Falls für den $\hat{x} \in \Sigma$ zusätzlich die Bedingung

$$f(\hat{x}) < f(x) \qquad \forall x \in \Sigma \cap \mathcal{U}_\varepsilon(\hat{x}),\ x \neq \hat{x}$$

erfüllt ist, entspricht $\hat{x}$ einem strikten lokalen Minimum.

**Bemerkung 2.1.** *Die im Rahmen dieser Arbeit verwendeten Optimierungsalgorithmen, mit Ausnahme der dynamischen Programmierung, befassen sich mit der Berechnung von lokalen Minima.*

### 2.1.2 Notwendige Bedingungen erster Ordnung

Eine allgemeine Form für notwendige Bedingungen erster Ordnung in der nichtlinearen Optimierung beschreiben die Fritz-John-Bedingungen, welche auch ohne Regularitätsbedingung im Allgemeinen für nichtlineare restringierte Optimierungsprobleme gelten. Für die Definition der Fritz-John-Bedingungen benötigen wir zunächst eine allgemeine Form der Lagrangefunktion

$$\mathcal{L}(x, \lambda_g, \lambda_h, \ell_0) = \ell_0 f(x) + \lambda_g^\top g(x) + \lambda_h^\top h(x) \tag{2.5}$$

von Problem 2.1. Die Gradienten $\nabla g(x)$ und $\nabla h(x)$ der Vektorfunktionen $g(x)$ und $h(x)$ entsprechen im Folgenden der transponierten Jacobi-Matrix der jeweiligen Vektorfunktion.

**Satz 2.1 (Fritz-John-Bedingungen, vgl. [43])**

Sei $\hat{x}$ ein lokales Minimum von Problem 2.1. Weiter seien die Funktionen $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^p$ und $g : \mathbb{R}^n \to \mathbb{R}^q$ stetig differenzierbar. Dann existieren Lagrange-Multiplikatoren $\lambda_g \in \mathbb{R}^q$ und $\lambda_h \in \mathbb{R}^p$ und $\ell_0 \geq 0$, wobei mindestens ein Lagrange-Multiplikator $\neq 0$ sein muss, mit denen folgende Bedingungen gelten:

- Stationarität:

$$\nabla \mathcal{L}(x, \lambda_g, \lambda_h, \ell_0) = \ell_0 \nabla f(\hat{x}) + \nabla g(\hat{x}) \lambda_g + \nabla h(\hat{x}) \lambda_h = 0.$$

- Komplementarität:

$$\begin{aligned} \lambda_{g,i} &\geq 0 && \text{für } i = 1, \ldots, q, \\ \lambda_g^\top g(\hat{x}) &= 0. \end{aligned}$$

- Zulässigkeit:

$$\hat{x} \in \Sigma.$$

Für die Fritz-John-Bedingungen ist der Fall $\ell_0 = 0$ besonders interessant, da hierbei die Stationarität unabhängig von der Zielfunktion erfüllt sein kann.

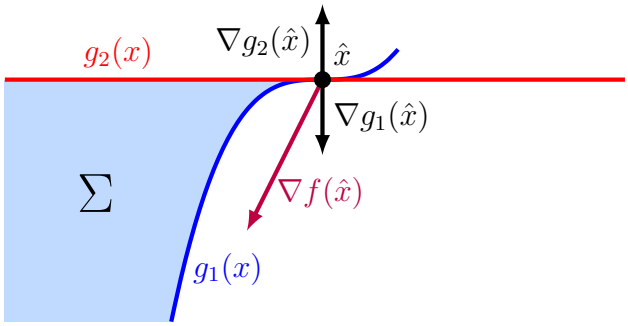


Abbildung 2.1: Skizze zur LICQ

Für den Fall $\ell_0 = 0$ betrachten wir Abbildung 2.1. Die Abbildung charakterisiert mit $x = (x_1, x_2)^\top$ das Optimierungsproblem

**Beispiel 2.1.**

$$\min f(x) = -x_1 - 2x_2$$

*unter den Nebenbedingungen*

$$g_1(x) = x_1^3 - x_2 \leq 0$$
$$g_2(x) = x_2 \leq 0.$$

Der Gradient der Zielfunktion ist durch

$$\nabla f(x) = \begin{pmatrix} -1 \\ -2 \end{pmatrix}$$

und die Gradienten der Nebenbedingungen sind durch

$$\nabla g_1(x) = \begin{pmatrix} 3x_1^2 \\ -1 \end{pmatrix} \qquad \text{und} \qquad \nabla g_2(x) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

gegeben. Wir betrachten den Punkt $\hat{x} = (0,0)^\top$ hinsichtlich der Stationarität der Lagrangefunktion. Im Punkt $\hat{x}$ sind die Gradienten der Nebenbedingungen $\nabla g_1(\hat{x}) = -\nabla g_2(\hat{x}) = (0,-1)^\top$ linear abhängig. Da der Gradient der Zielfunktion $\nabla f(\hat{x})$ linear unabhängig von den Gradienten der Nebenbedingungen ist, ergibt sich in den Fritz-John-Bedingungen der Fall $\ell_0 = 0$ und es gilt

$$(\lambda_1 - \lambda_2) \begin{pmatrix} 0 \\ -1 \end{pmatrix} = 0.$$

Damit ist die Stationarität im Punkt $\hat{x}$ für $\lambda_1 = \lambda_2$ unabhängig von der Zielfunktion erfüllt. Hinzu kommt, dass die Lagrange-Multiplikatoren nicht eindeutig sind.

**Bemerkung 2.2.** *Die Komplementarität $\lambda_g^\top g(\hat{x}) = 0$ ist in $\hat{x}$, durch $g_1(\hat{x}) = 0$ und $g_2(\hat{x}) = 0$ erfüllt.*

Um Situationen mit $\ell_0 = 0$ auszuschließen und damit die Abhängigkeit der Stationarität von der Zielfunktion zu gewährleisten, werden sogenannte Constraint Qualifications benötigt. Zunächst passen wir für den Fall $\ell_0 > 0$ die Lagrangefunktion für das nichtlineare restringierte Optimierungsproblem aus Problem 2.1

$$\mathcal{L}(x, \lambda_g, \lambda_h) = f(x) + \lambda_g^\top g(x) + \lambda_h^\top h(x) \tag{2.6}$$

an.

**Bemerkung 2.3.** *Die Lagrangefunktion aus Formel (2.6) wird zur vereinfachten Schreibweise analog zu Formel (2.5) mit $\mathcal{L}$ bezeichnet. Der Unterschied in der Notation besteht darin, dass wir das Argument $\ell_0$ weglassen.*

Für alle $\ell_0 > 0$ können wir die Stationarität in den Fritz-John-Bedingungen durch $\ell_0$ teilen. Wenn $\ell_0$ entsprechend mit den Lagrange-Multiplikatoren verrechnet wird, erhalten wir auch die Stationarität der Lagrangefunktion. In Formel (2.6) werden die Vektoren $\lambda_g \in \mathbb{R}^q$ und $\lambda_h \in \mathbb{R}^p$ als Lagrange-Multiplikatoren bezeichnet. Hinsichtlich der Stationarität der Lagrangefunktion ergibt sich der Zusammenhang

$$\nabla_x \mathcal{L}(\hat{x}, \lambda_g, \lambda_h) = \nabla f(\hat{x}) + \nabla g(\hat{x})\lambda_g + \nabla h(\hat{x})\lambda_h = 0.$$

Um den Fall $\ell_0 = 0$ auszuschließen, können wir zusätzliche Anforderungen an unser Optimierungsproblem stellen. Diese Anforderungen werden auch als Constraint Qualification bezeichnet. Eine der meist verwendeten ist die Linear Independence Constraint Qualification (LICQ).

**Definition 2.3 (Linear Independence Constraint Qualification (LICQ), vgl. [43])**

Sei $\hat{x}$ ein zulässiger Punkt von Problem 2.1. Die LICQ ist in $\hat{x}$ genau dann erfüllt, wenn die Vektoren

$$\nabla g_i(\hat{x}), i \in \mathcal{A}(\hat{x}) \qquad \text{und} \qquad \nabla h_j(\hat{x}), j = 1, \ldots, p,$$

linear unabhängig sind.

Einer der bedeutendsten Vorteile der LICQ gegenüber anderer Constraint Qualification ist, dass diese auf eindeutig bestimmte Lagrange-Multiplikatoren führt. Unter Forderung einer Einhaltung der Linear Independence Constraint Qualification, können wir die KKT-Bedingungen aufstellen.

**Satz 2.2 (KKT-Bedingungen (Karush-Kuhn-Tucker), vgl. [43])**

Sei $\hat{x}$ ein lokales Minimum von Problem 2.1. Weiter seien die Funktionen $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^p$ und $g : \mathbb{R}^n \to \mathbb{R}^q$ stetig differenzierbar. Ferner erfülle $\hat{x}$ die LICQ. Damit existieren eindeutig bestimmte Lagrange-Multiplikatoren $\lambda_g \in \mathbb{R}^q$ und $\lambda_h \in \mathbb{R}^p$ mit denen folgende Bedingungen gelten:

- Stationarität:

$$\nabla_x \mathcal{L}(\hat{x}, \lambda_g, \lambda_h) = \nabla f(\hat{x}) + \nabla g(\hat{x})\lambda_g + \nabla h(\hat{x})\lambda_h = 0.$$

- Komplementarität:

$$\begin{aligned} \lambda_{g,i} &\geq 0 && \text{für } i = 1, \ldots, q, \\ \lambda_g^\top g(\hat{x}) &= 0. \end{aligned}$$

- Zulässigkeit:

$$\hat{x} \in \Sigma.$$

**Bemerkung 2.4.** *Die KKT-Bedingungen beschreiben die notwendigen Optimalitätsbedingungen erster Ordnung. Sie werden von zahlreichen Optimierungsverfahren als Aus-*

*gangspunkt zur Berechnung stationärer Punkte verwendet. Die in den KKT-Bedingungen definierten notwendigen Voraussetzungen werden im Folgenden immer als erfüllt angenommen.*

### 2.1.3 Notwendige Bedingungen zweiter Ordnung

Stationäre Punkte zeichnen sich durch die notwendigen Bedingungen erster Ordnung aus. Um die Menge der Kandidaten für ein Minimum weiter einzuschränken, können die notwendigen Bedingungen zweiter Ordnung herangezogen werden. Da wir ein restringiertes Optimierungsproblem behandeln, führen wir zunächst den kritischen Kegel

$$\begin{aligned}\mathcal{T}_k(\hat{x}) := \{d \in \mathbb{R}^n | &\nabla g_i(\hat{x})^\top d \leq 0, i \in \mathcal{A}(\hat{x}), \lambda_{g,i} = 0, \\ &\nabla g_i(\hat{x})^\top d = 0, i \in \mathcal{A}(\hat{x}), \lambda_{g,i} > 0, \\ &\nabla h_j(\hat{x})^\top d = 0, j = 1, \ldots, p\}\end{aligned} \tag{2.7}$$

ein. In einem stationären Punkt $\hat{x}$ gilt für alle Richtungen aus dem kritischen Kegel $d \in \mathcal{T}_k(\hat{x})$, dass die Richtungsableitung der Zielfunktion $\nabla f(\hat{x})^\top d = 0$ ist. Daher müssen für diese Richtungen zweite Ableitungen in Form der Hessematrix der Lagrangefunktion berücksichtigt werden.

**Satz 2.3 (Notwendige Bedingungen zweiter Ordnung, vgl. [43])**

Sei $\hat{x}$ ein lokales Minimum von Problem 2.1 und die Funktionen $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^p$ und $g : \mathbb{R}^n \to \mathbb{R}^q$ zweimal stetig differenzierbar. Ferner erfülle das lokale Minimum $\hat{x}$ die LICQ und die KKT-Bedingungen. Dann gilt:

$$d^\top \nabla^2_{xx} \mathcal{L}(\hat{x}, \lambda_g, \lambda_h) d \geq 0 \qquad \forall d \in \mathcal{T}_k(\hat{x}).$$

Das heißt die Hessematrix der Lagrangefunktion ist in einem lokalen Minimum $\hat{x}$ positiv semidefinit, für alle Richtungen $d$ auf dem kritischen Kegel (2.7).

### 2.1.4 Hinreichende Bedingungen

Analog zur Kurvendiskussion in der Analysis muss bei der nichtlinearen restringierten Optimierung zwischen notwendigen und hinreichenden Optimalitätsbedingungen unterschieden werden. Die notwendigen Bedingungen gelten in jedem Optimum, was im Umkehrschluss aber nicht bedeutet, dass ein Punkt, der die notwendigen Bedingungen erfüllt,

auch ein Minimum sein muss. Für die hinreichenden Bedingungen gilt im Gegenzug dazu, dass ein Punkt, welcher diese Bedingung erfüllt, ein Minimum ist. Allerdings muss nicht jedes Minimum die hinreichenden Bedingungen erfüllen.

**Satz 2.4 (Hinreichende Bedingungen zweiter Ordnung, vgl. [43])**

Erfülle der Vektor $\hat{x}$ für Problem 2.1 die LICQ und die KKT-Bedingungen aus Satz 2.2. Weiter seien die Funktionen $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^p$ und $g : \mathbb{R}^n \to \mathbb{R}^q$ zweimal stetig differenzierbar und es gelte

$$d^\top \nabla^2_{xx} \mathcal{L}(\hat{x}, \lambda_g, \lambda_h) d > 0 \qquad \forall d \in \mathcal{T}_k(\hat{x}).$$

Dann ist der Vektor $\hat{x}$ ein lokales Minimum von Problem 2.1.

## 2.2 SQP-Verfahren

Eines der bekanntesten Verfahren zur Lösung von restringierten nichtlinearen Optimierungsproblemen ist die sequentielle quadratische Programmierung, auch SQP-Verfahren genannt. Wir beschreiben die grundlegende Funktionsweise dieses Verfahrens anhand der Ausführungen von [43, 40, 41, 60, 35].

### 2.2.1 Lokales SQP-Verfahren

Beim SQP-Verfahren wird das nichtlineare Optimierungsproblem mit Hilfe der Lösung einer Folge von quadratischen Optimierungsproblemen gelöst. Dazu betrachten wir das restringierte nichtlineare Optimierungsproblem aus Problem 2.1. Für die Lagrangefunktion der KKT-Bedingungen aus (2.6)

$$\mathcal{L}(x, \lambda_g, \lambda_h) = f(x) + \lambda_g^\top g(x) + \lambda_h^\top h(x)$$

definieren wir im Iterationspunkt $(x_k, \lambda_{g,k}, \lambda_{h,k})^\top$ die Hessematrix

$$H_k = \mathcal{L}_{xx}(x_k, \lambda_{g,k}, \lambda_{h,k}).$$

Mit der Hessematrix der Lagrangefunktion können wir ein lokales quadratisches Teilproblem zur Bestimmung einer Suchrichtung $d \in \mathbb{R}^{n_x}$ definieren.

**Problem 2.2 (Quadratisches Optimierungsproblem des SQP-Verfahrens (QP))**

Bestimme die Suchrichtung $d \in \mathbb{R}^{n_x}$ für den Zustand $x_k \in \mathbb{R}^{n_x}$ und die Lagrangemultiplikatoren $\lambda_{g,k} \in \mathbb{R}^q$ und $\lambda_{h,k} \in \mathbb{R}^p$, so dass die die Zielfunktion

$$\min_{d \in \mathbb{R}^{n_x}} \frac{1}{2} d^\top H_k d + \nabla f(x_k)^\top d$$

minimal wird, unter den Nebenbedingungen

$$\begin{aligned} h(x_k) + \nabla h(x_k)^\top d &= 0 \\ g(x_k) + \nabla g(x_k)^\top d &\leq 0, \end{aligned}$$

mit $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^p$ und $g : \mathbb{R}^n \to \mathbb{R}^q$.

Nach [43] handelt es sich bei Problem 2.2 um eine lokale Approximation von Problem 2.1 um den Punkt $x_k \in \mathbb{R}^{n_x}$, $\lambda_{g,k}$ und $\lambda_{h,k}$. Zur Bestimmung einer Suchrichtung des SQP-Verfahrens muss jeweils das quadratische Optimierungsproblem aus Problem 2.2 gelöst werden.

**Lokales SQP-Verfahren**

i Bestimme Startschätzung $(x_0, \lambda_{g,0}, \lambda_{h,0})$ und setze $k = 0$.

ii Falls $(x_k, \lambda_{g,k}, \lambda_{h,k})$ näherungsweise die KKT-Bedingungen aus Satz 2.2 erfüllt: Beende das Verfahren.

iii Berechne die Lösung $(d_k, \lambda_{g,k+1}, \lambda_{h,k+1})$ für das aus $(x_k, \lambda_{g,k}, \lambda_{h,k})$ resultierende quadratische Optimierungsproblem aus Problem 2.2.

iv Setze $x_{k+1} := x_k + d_k$ und $k := k + 1$ und gehe zurück zu ii.

Ein Satz zu den lokalen Konvergenzeigenschaften dieses Verfahrens kann in [43] gefunden werden. Die wesentlichen Aussagen dieses Satzes sind:

- Es existiert eine eindeutige Lösung $(d_k, \lambda_{g,k+1}, \lambda_{h,k+1})$ in jedem Iterationsschritt für jeden Startwert $(x_0, \lambda_{g,0}, \lambda_{h,0}) \in \mathcal{U}$ aus einer lokalen Umgebung eines Minimums.

- Das Verfahren besitzt eine lokal quadratische Konvergenzgeschwindigkeit in der Umgebung eines Minimums.

### 2.2.2 Aktive-Mengen-Verfahren für quadratische Optimierungsprobleme

Zur Lösung des quadratischen Optimierungsproblems aus Schritt iii des lokalen SQP-Verfahrens können verschiedenste Verfahren angewandt werden. Im Rahmen dieser Arbeit verwenden wir ein Aktive-Mengen-Verfahren. Das quadratische Optimierungsproblem aus Problem 2.2 hat im Allgemeinen die Form:

**Problem 2.3 (Quadratisches Optimierungsproblem (QP))**

$$\min_{x \in \mathbb{R}^{n_x}} \frac{1}{2} x^\top W x + c^\top x$$

unter den Nebenbedingungen

$$a_i^\top x - b_i \begin{cases} = 0 & \forall i = 1, \ldots, p \\ \leq 0 & \forall i = p+1, \ldots, p+q \end{cases}$$

mit der symmetrischen und positiv definiten Matrix $W \in \mathbb{R}^{n_x \times n_x}$ und den Vektoren $c \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}^{p+q}$ und $a_i \in \mathbb{R}^{n_x}$ für $i = 1, \ldots, p+q$.

Die Indexmenge aller Gleichungen ist durch

$$\mathcal{I} := \{1, \ldots, p\}$$

definiert. Wäre die Menge der aktiven Ungleichungen

$$\mathcal{A}(\hat{x}) := \left\{ i \in \{p+1, \ldots, p+q\} \,\middle|\, a_i^\top \hat{x} - b_i = 0 \right\}$$

für ein Minimum $\hat{x}$ bekannt, so ist das Optimierungsproblem aus Problem 2.3 äquivalent zu folgender Problemstellung.

**Problem 2.4 (Quadratisches Optimierungsproblem auf aktiver Menge)**

$$\min_{x \in \mathbb{R}^{n_x}} \frac{1}{2} x^\top W x + c^\top x$$

unter den Nebenbedingungen

$$a_i^\top x - b_i = 0 \qquad \forall i \in \mathcal{I} \cup \mathcal{A}(\hat{x})$$

mit der symmetrischen und positiv definiten Matrix $W \in \mathbb{R}^{n_x \times n_x}$ und den Vektoren $c \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}^{p+q}$ und $a_i \in \mathbb{R}^{n_x}$ für $i = 1, \ldots, p+q$.

Die Äquivalenz folgt daraus, dass die inaktiven Ungleichungen keinen Einfluss auf das Minimum $\hat{x}$ haben. Der Vorteil von Problem 2.4 gegenüber des ursprünglichen quadratischen Optimierungsproblems liegt in den affinen Nebenbedingungen. Die Lösung des Optimierungsproblems erfolgt über die KKT-Bedingungen von Problem 2.4. Dafür benötigen wir zunächst die Lagrangefunktion von Problem 2.4

$$\mathcal{L}_{\mathrm{QP}}(x, \lambda) = \frac{1}{2} x^\top W x + c^\top x + \lambda_{\mathcal{I} \cup \mathcal{A}(\hat{x})}^\top \left( A_{\mathcal{I} \cup \mathcal{A}(\hat{x})} x - b \right), \tag{2.8}$$

mit der Matrix $A_{\mathcal{I} \cup \mathcal{A}(\hat{x})} := [a_1, \ldots, a_p, \ldots, a_i, \ldots]^\top \in \mathbb{R}^{(p+q') \times n_x}$ mit allen $i \in \mathcal{A}(\hat{x})$. Die Dimension $q'$ entspricht dabei der Anzahl von aktiven Nebenbedingungen der aktiven Menge $\mathcal{A}(\hat{x})$.
Mit der Lagrangefunktion (2.8) von Problem 2.4 ergeben sich die KKT-Bedingungen

$$\begin{bmatrix} W & A_{\mathcal{I} \cup \mathcal{A}(\hat{x})}^\top \\ A_{\mathcal{I} \cup \mathcal{A}(\hat{x})} & 0 \end{bmatrix} \begin{bmatrix} \hat{x} \\ \lambda_{\mathcal{I} \cup \mathcal{A}(\hat{x})} \end{bmatrix} = \begin{bmatrix} -c \\ b \end{bmatrix} \tag{2.9}$$

in der Form eines linearen Gleichungssystems.

**Bemerkung 2.5.** *Effiziente Lösungsmethoden für die spezielle Struktur des linearen Gleichungssystems aus (2.9) sind in [43] beschrieben.*

Zusammengefasst ergibt sich: Ist die aktive Menge von Ungleichungen $\mathcal{A}(\hat{x})$ bekannt, so können wir das quadratische Optimierungsproblem mit Hilfe des linearen Gleichungssystems (2.9) lösen.
Hinter dem Aktive-Mengen-Verfahren steht die Idee, die Menge der aktiven Ungleichungen $\mathcal{A}(\hat{x})$ durch eine Menge $\tilde{\mathcal{A}} \subseteq \mathcal{J} := \{p+1, \ldots, p+q\}$ zu schätzen. Mit dieser Menge $\tilde{\mathcal{A}}$ formulieren wir ein quadratisches Hilfsproblem.

**Problem 2.5 (Quadratisches Hilfsproblem)**

$$\min_{d \in \mathbb{R}^{n_x}} \frac{1}{2} (x_k + d)^\top W (x_k + d) + c^\top (x_k + d)$$

unter den Nebenbedingungen

$$a_i^\top d = 0 \qquad \forall i \in \mathcal{I} \cup \tilde{\mathcal{A}}$$

mit der symmetrischen und positiv definiten Matrix $W \in \mathbb{R}^{n_x \times n_x}$ und den Vektoren $c \in \mathbb{R}^{n_x}$ und $a_i \in \mathbb{R}^{n_x}$ für $i = 1, \ldots, p + q$.

In Abhängigkeit von der Lösung $d$ des Problems 2.5 und den zugehörigen Lagrangemultiplikatoren $\lambda_i \left( i \in \mathcal{I} \cup \tilde{\mathcal{A}} \right)$ treten verschiedene Fallunterscheidungen auf, die nach [43] folgenden Algorithmus ergeben.

**Aktive-Mengen-Verfahren**

i Bestimme eine für Problem 2.3 zulässige Startschätzung $x_0$, setze $k = 0$ und

$$\tilde{A}_0 := \left\{ i \in \mathcal{J} | a_i^\top x_0 = b_i \right\}.$$

ii Bestimme eine Lösung des Hilfsproblems aus Problem 2.5 durch das Lösen des linearen Gleichungssystems

$$\begin{bmatrix} W & A_{\mathcal{I} \cup \tilde{\mathcal{A}}_k}^\top \\ A_{\mathcal{I} \cup \tilde{\mathcal{A}}_k} & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda_{\mathcal{I} \cup \tilde{\mathcal{A}}_k} \end{bmatrix} = \begin{bmatrix} -(W x_k + c) \\ 0 \end{bmatrix}.$$

iii Falls $d = 0$ und $\lambda_{\tilde{\mathcal{A}}_k} \geq 0$, so gilt für alle übrigen Lagrangemultiplikatoren $\lambda_i = 0$ für $i \in \mathcal{J} \backslash \tilde{\mathcal{A}}_k$. Beende den Algorithmus mit der Lösung $(x_k, \lambda)$.

iv Falls $d = 0$ und $\lambda_u := \min \left\{ \lambda_i | i \in \tilde{\mathcal{A}}_k \right\} < 0$, so erfolgt ein Deaktivierungsschritt

$$\tilde{\mathcal{A}}_{k+1} := \tilde{\mathcal{A}}_k \backslash \{u\}.$$

Setzte $k := k + 1$ und gehe zu Schritt ii.

v Falls $a_i^\top (x_k + d) \leq b_i, i \in \mathcal{J}\backslash\tilde{\mathcal{A}}_k$, so setze

$$x_{k+1} := x_k + d \qquad \tilde{\mathcal{A}}_{k+1} := \tilde{\mathcal{A}}_k \qquad k := k + 1$$

und gehe zu Schritt ii.

vi Bestimme einen Index $r \in \mathcal{J}\backslash\tilde{\mathcal{A}}_k$ mit $a_r^\top d > 0$ und

$$\alpha_k := \frac{b_r - a_r^\top x_k}{a_r^\top d} = \min\left\{\frac{b_i - a_i^\top x_k}{a_i^\top d} : i \in \mathcal{J}\backslash\tilde{\mathcal{A}}_k, a_i^\top d > 0\right\}.$$

Setze

$$x_{k+1} := x_k + \alpha_k d \qquad \tilde{\mathcal{A}}_{k+1} := \tilde{\mathcal{A}}_k \cup \{r\} \qquad k := k + 1$$

und gehe zu Schritt ii.

**Bemerkung 2.6.** *Weitere Details zur numerischen Lösung von quadratischen Optimierungsproblemen, unter anderem eine Methode zur Berechnung einer zulässigen Startlösung, können in [43] gefunden werden.*

**Bemerkung 2.7.** *Eine Alternative zum Aktive-Mengen-Verfahren bietet zum Beispiel die Verwendung eines Innere-Punkte-Verfahrens zur Lösung von quadratischen Optimierungsproblemen.*

### 2.2.3 Globalisierung des SQP-Verfahrens

Für das lokale SQP-Verfahren aus Abschnitt 2.2.1 lassen sich nach [43] lokale Konvergenzeigenschaften zeigen. Diese gelten jedoch nur in einer unbekannten Umgebung um das gesuchte Minimum. Für den praktischen Einsatz des SQP-Verfahren ist es somit notwendig, den Algorithmus zu globalisieren. Der Unterschied besteht dabei bei der Berechnung der neuen Iterierten

$$x_{k+1} := x_k + \alpha d_k,$$

die um eine Schrittweite $\alpha > 0$ ergänzt wird. Das Ziel der Globalisierung ist es die Schrittweite $\alpha$ so zu wählen, dass sich bezüglich einer Bewertungsfunktion für jede fortlaufende Iterierte $x_{k+1}$ ein Abstieg ergibt. Die Globalisierung unterteilen wir dabei in zwei Schritte:

1. Wahl einer geeigneten Bewertungsfunktion.

2. Wahl einer geeigneten Schrittweite $\alpha$.

### Bewertungsfunktion für das SQP-Verfahren

Bei der Wahl einer geeigneten Bewertungsfunktion haben wir mehrere Möglichkeiten. Im unrestringierten Fall können wir einfach die Zielfunktion als Bewertungsfunktion verwenden. Bei restringierten Optimierungsproblemen wie Problem 2.1 muss eine geeignete Bewertungsfunktion definiert werden, die eine Abstiegseigenschaft bezüglich aller vom SQP-Verfahren berechneten Suchrichtungen hat. In [43] wird dazu beispielsweise die $\ell_1$-Penaltyfunktion

$$\ell_1(x;\eta) := f(x) + \eta\left(\sum_{i=1}^{q}\max(0, g_i(x)) + \sum_{j=1}^{p}|h_j(x)|\right)$$

vorgeschlagen.

**Bemerkung 2.8.** *Die $\ell_1$-Penaltyfunktion ist aufgrund der Betragsfunktion nicht differenzierbar, aber dennoch richtungsdifferenzierbar.*

Die Richtungsableitung bezüglich einer Richtung $d \in \mathbb{R}^{n_x}$ der $\ell_1$-Penaltyfunktion ist nach [43] durch

$$\begin{aligned}\ell_1'(x;d;\eta) = \nabla f(x)^\top d + \eta \sum_{i:g_i(x)>0} \nabla g_i(x)^\top d + \eta \sum_{i:g_i(x)=0} \max\left(0, \nabla g_i(x)^\top d\right) + \\ \eta \sum_{j:h_j(x)>0} \nabla h_j(x)^\top d - \eta \sum_{j:h_j(x)<0} \nabla h_j(x)^\top d + \eta \sum_{j:h_j(x)=0} \left|\nabla h_j(x)^\top d\right|\end{aligned}$$

definiert. Bezüglich der Abstiegseigenschaft ergibt sich folgender Satz.

**Satz 2.5 (Abstiegseigenschaft der $\ell_1$-Penaltyfunktion für das SQP-Verfahren)**

Sei $d := [d^\top, \lambda_h^\top, \lambda_g^\top]^\top \neq 0$ ein KKT-Punkt von Problem 2.2. Es gilt unter der Bedingung

$$\eta \geq \max\left(\lambda_{h_1}, \ldots, \lambda_{h_p}, |\lambda_{g_1}|, \ldots, \left|\lambda_{g_q}\right|\right),$$

dass sich die Richtungsableitung der $\ell_1$-Penaltyfunktion für eine positiv definite Hessematrix der Lagrangefunktion $\mathcal{L}_{xx}(x_k, \lambda_{h,k}, \lambda_{g,k})$ (2.6) durch

$$\ell_1'(x;d;\eta) < 0$$

abschätzen lässt.

Der Beweis von Satz 2.5 kann unter anderem in [43] gefunden werden.

### Armijo Liniensuche

Zur Wahl einer geeigneten Schrittweite verwenden wir die Armijo Liniensuche. Mit Hilfe der gewählten Bewertungsfunktion

$$\ell_1(x;\eta)$$

können sich auch die Nebenbedingungen des Optimierungsproblems in den Abstiegseigenschaften wiederfinden. Wir nehmen an, dass wir für einen Zustand $x_k$ mit dem SQP-Verfahren eine Abstiegsrichtung $d_k$ ermittelt haben. Die nächste Iterierte des Zustands

$$x_{k+1} = x_k + \alpha d_k$$

ist insbesondere von der Schrittweite $\alpha$ abhängig. Zur Bestimmung von einer geeigneten Schrittweite betrachten wir einen Schnitt in der Bewertungsfunktion

$$\varphi(\alpha) = \ell_1(x_k + \alpha d_k;\eta)$$

entlang der Abstiegsrichtung $d_k$. Die Funktion $\varphi(\alpha)$ ist in Abbildung 2.2 exemplarisch

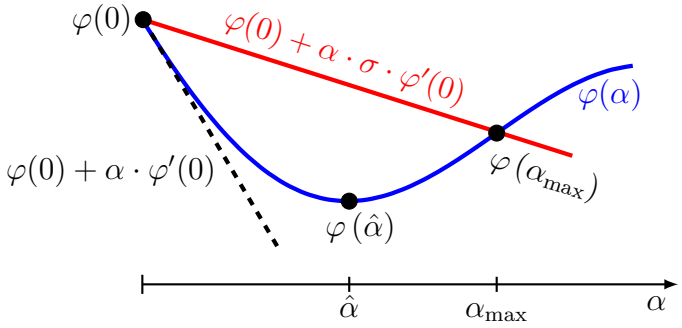


Abbildung 2.2: Skizze zur Armijo Liniensuche

dargestellt. Mit $d_k$ als Suchrichtung des SQP-Verfahrens gilt für die Richtungsableitung der Bewertungsfunktion nach Satz 2.5

$$\varphi'(0) = \ell_1'(x_k; d_k; \eta) < 0.$$

Die Wahl der Schrittweite kann durch eine Liniensuche charakterisiert werden. Diese ermittelt, wie die Schrittweite $\alpha$ gewählt werden muss, um einen möglichst großen Abstieg

in der Bewertungsfunktion zu bewirken. Bestenfalls kann man an diesem Punkt eine exakte Liniensuche durchführen, mit der sich die bestmögliche Schrittweite $\hat{\alpha}$ ermitteln ließe. Jedoch gilt eine exakte Liniensuche in der Regel als zu aufwändig für diesen Schritt. Behelfen können wir uns mit sogenannten Schrittweitenstrategien, wie beispielsweise der Armijoregel bzw. Armijo Liniensuche.

**Armijoregel:**

Sei $d_k \in \mathbb{R}^{n_x}$ eine Abstiegsrichtung einer Bewertungsfunktion $\ell_1(x_k;\eta)$ mit $\ell_1'(x_k;d_k;\eta) < 0$ bezüglich des Vektors $x_k \in \mathbb{R}^{n_x}$. Sei außerdem

$$\varphi(\alpha) = \ell_1(x_k + \alpha d_k;\eta)$$

die Schnittfunktion der Bewertungsfunktion entlang der Suchrichtung $d_k$, so ist die Armijobedingung für eine Schrittweite $\alpha$ durch

$$\varphi(\alpha) \leq \varphi(0) + \alpha \cdot \sigma \cdot \varphi'(0), \tag{2.10}$$

mit $\sigma \in (0,1)$ definiert.

**Algorithmus Armijo Liniensuche:**

Sei $d_k \in \mathbb{R}^{n_x}$ eine Abstiegsrichtung einer Bewertungsfunktion $\ell_1(x_k;\eta)$ bezüglich des Zustands $x_k \in \mathbb{R}^{n_x}$ und die beiden Skalare $\beta \in (0,1)$ und $\sigma \in (0,1)$ fest definiert. Wir setzen zu Beginn $\alpha = \alpha_{\max}$.

i Falls die Ungleichung (2.10) für die aktuelle Schrittweite $\alpha$ erfüllt ist, gebe die Schrittweite $\alpha$ zurück. Sonst gehe weiter zu ii.

ii Reduziere die Schrittweite durch

$$\alpha = \beta \cdot \alpha$$

und gehe zurück zu i.

**Bemerkung 2.9.** *Große und leistungsfähige Softwarepakete, welche das SQP-Verfahren implementieren, sind zum Beispiel SNOPT aus [44] und WORHP aus [21]. Speziell zur Lösung von quadratischen Optimierungsproblemen ist auch die Softwarebibliothek qpOA-*

*SES aus [34] sehr gut geeignet. Im Rahmen dieser Arbeit verwenden wir für das SQP-Verfahren das Softwarepaket sqpfiltertoolbox, welches von der Optimalsteuerungssoftware* OCPID-DAE1 *aus [36] verwendet wird.*

## 2.3 Optimalsteuerungsprobleme

### 2.3.1 Problemstellung

In Problem 2.1 ist ein restringiertes nichtlineares diskretes Optimierungsproblem definiert. Für einen Bahnplanungsalgorithmus zur Berechnung von Rennlinien müssen wir kontinuierliche Optimalsteuerungsprobleme betrachten. Eine Vielzahl von Methoden und Algorithmen für diese Art von Optimierungsproblem sind in [13] und [40] beschrieben. Wir definieren das stetige Optimalsteuerungsproblem.

**Problem 2.6 (Allgemeines Optimalsteuerungsproblem (OSP))**

Bestimme die Zustandsvektorfunktion $y : [t_0, t_f] \to \mathbb{R}^{n_y}$, den Parametervektor $p \in \mathbb{R}^{n_p}$ und die zugehörige Steuerungsvektorfunktion $u : [t_0, t_f] \to \mathbb{R}^{n_u}$, so dass das Zielfunktional

$$\varphi(y(t_0), y(t_f), p) \tag{2.11}$$

minimal wird, unter den nichtlinearen Nebenbedingungen

$$y'(t) = F(t, y(t), u(t), p) \tag{2.12}$$

$$\psi_{\min} \leq \psi(y(t_0), y(t_f), p) \leq \psi_{\max} \tag{2.13}$$

$$v_{\min} \leq v(t, y(t), u(t), p) \leq v_{\max} \qquad t \in [t_0, t_f] \tag{2.14}$$

und den Boxbeschränkungen

$$y_{\min} \leq y(t) \leq y_{\max} \qquad t \in [t_0, t_f] \tag{2.15}$$

$$u_{\min} \leq u(t) \leq u_{\max} \qquad t \in [t_0, t_f] \tag{2.16}$$

$$p_{\min} \leq p \leq p_{\max}. \tag{2.17}$$

Für die Nebenbedingungen aus Problem 2.6 gilt

$$\begin{aligned} F &: [t_0, t_f] \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_y} \\ \psi &: \mathbb{R}^{n_y} \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_\psi} \\ v &: [t_0, t_f] \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_v}. \end{aligned}$$

Die Gleichung (2.12) entspricht einem Differentialgleichungssystem erster Ordnung mit der Dimension des Zustandsvektors $n_y$. Nichtlineare Randbedingungen können wir in (2.13) und nichtlineare Steuer-, Parameter- und Zustandsbeschränkungen in (2.14) formulieren. Konstante Zustands- und Steuerungsbeschränkungen sind als Boxschranken in den Ungleichungen (2.15) und (2.16) für das Zeitintervall $[t_0, t_f]$ gegeben. Die Boxschranken für den Parametervektor $p \in \mathbb{R}^{n_p}$ sind in (2.17) angegeben.
Problem 2.6 beschreibt eine allgemeine Form der Art von Optimalsteuerungsproblemen, welche im Rahmen dieser Arbeit gelöst werden sollen. Zur Lösung von Optimalsteuerungsproblemen, wie Problem 2.6, können unterschiedliche Herangehensweisen verwendet werden. Zunächst wird in der Optimalsteuerung zwischen direkten und indirekten Verfahren unterschieden. Bei indirekten Verfahren werden die notwendigen Bedingungen bereits für das Optimalsteuerungsproblem oder ein Variationsproblem formuliert und anschießend diskretisiert. Wohingegen direkte Verfahren das Optimalsteuerungsproblem zunächst auf ein restringiertes nichtlineares diskretes Optimierungsproblem, wie Problem 2.1, diskretisieren und dieses anschließend mit einem Optimierungsalgorithmus lösen. Im Rahmen dieser Arbeit werden wir uns auf direkte Verfahren zur Lösung von Optimalsteuerungsproblemen beschränken.

### 2.3.2 Direkte Schießverfahren

Die meisten Nebenbedingungen in Problem 2.6 können wir punktweise auf einem Gitter

$$\mathbb{G} = \{t_k\}_{k=0,\dots,N} \qquad \text{mit } t_0 < t_1 < \cdots < t_N, \tag{2.18}$$

und $t_N = t_f$ diskretisieren. Die Steuerungen $u(t)$ lassen sich beispielsweise stückweise konstant oder stückweise linear modellieren. Anders verhält es sich bei der Nebenbedingung der Zustände (2.12)

$$y'(t) = F(t, y(t), u(t), p),$$

die sich aus einem System gewöhnlicher Differentialgleichungen ergeben. Zur Diskretisierung des Differentialgleichungssystems der Zustände kommen in der Regel zwei Klassen

von Diskretisierungsverfahren zum Einsatz. Einerseits gibt es die Schießverfahren und andererseits die volle Diskretisierung oder auch Kollokation des Optimalsteuerungsproblems. Bei einer Diskretisierung durch Kollokation wird das Differentialgleichungssystem punktweise auf dem Gitter, durch ein Integrationsverfahren für Differentialgleichungen diskretisiert. Eine Durchführung der Kollokation mit Trapezregel findet sich in Kapitel 4. Im ersten Teil dieser Arbeit befassen wir uns zunächst mit den Schießverfahren.
Zuerst definieren wir für die Schießverfahren Zustände und Steuerungen, hier stückweise linear, auf dem punktweisen Gitter (2.18)

$$\left[y_0^\top, \ldots, y_N^\top\right]^\top \qquad y_k = y(t_k) \in \mathbb{R}^{n_y},\ \forall k = 0, \ldots, N \tag{2.19}$$

$$\left[u_0^\top, \ldots, u_N^\top\right]^\top \qquad u_k = u(t_k) \in \mathbb{R}^{n_u},\ \forall k = 0, \ldots, N. \tag{2.20}$$

Für die Parameter $p \in \mathbb{R}^{n_p}$ ändert sich in diskreter Form nichts, da sie keine zeitliche Abhängigkeit besitzen.

**Bemerkung 2.10.** *In [40] ist das Schießverfahren auch für eine Diskretisierung von Zuständen und Steuerungen auf zwei verschiedenen Gittern beschrieben. Für diese Arbeit genügt es jedoch, ein gemeinsames Gitter zu verwenden.*

Das Ziel bei den Schießverfahren ist es, die diskreten Zustände mit der durch das Differentialgleichungssystem gegebenen Dynamik zu koppeln. Bei den Schießverfahren wird zwischen Einfach- und Mehrfachschießverfahren unterschieden. Die Grundidee liegt darin, bei jeder Auswertung der Nebenbedingung (2.12), ein Integrationsverfahren für Differentialgleichungen wie beispielsweise ein Runge-Kutta-Verfahren durchzuführen. Der Unter-

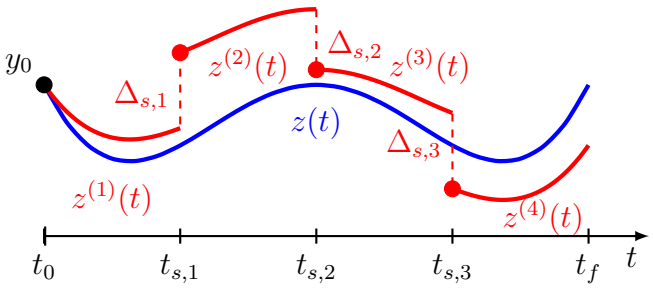


Abbildung 2.3: Skizze Einfach- und Mehrfachschießverfahren

schied von Einfach- zu Mehrfachschießverfahren lässt sich anhand von Abbildung 2.3 aufzeigen. Beim Einfachschießverfahren wird für das ganze Integrationsintervall ein Integrationsverfahren durchgeführt. Diese Integration ist in Abbildung 2.3 durch die Funktion

$z : [t_0, t_f] \to \mathbb{R}^{n_y}$ dargestellt. Durch die Integration auf dem Gitter (2.18) ergeben sich für die Funktion $z$ die diskreten Auswertepunkte

$$\left[z_0^\top, \ldots, z_N^\top\right]^\top \qquad z_k = z(t_k; u_0, \ldots, u_N, y_0, p) \in \mathbb{R}^{n_y},\ \forall k = 0, \ldots, N. \tag{2.21}$$

Das diskretisierte Optimierungsproblem sieht bei Verwendung des Einfachschießverfahrens wie folgt aus.

**Problem 2.7 (Optimierungsproblem mit Einfachschießverfahren)**

Minimiere

$$\varphi(y_0, y_N, p)$$

bezüglich der Zustände $y_k \in \mathbb{R}^{n_y}$ aus (2.19), der Steuerungen $u_k \in \mathbb{R}^{n_u}$ aus (2.20) und der Parameter $p \in \mathbb{R}^{n_p}$ für $k = 0, \ldots, N$, unter den Nebenbedingungen

$$\begin{aligned} y_k &= z_k && \forall k = 0, \ldots, N \\ \psi_{\min} \le \psi(y_0, y_N, p) &\le \psi_{\max} \\ v_{\min} \le v(t_k, y_k, u_k, p) &\le v_{\max} && \forall k = 0, \ldots, N \end{aligned}$$

und den Boxbeschränkungen

$$\begin{aligned} y_{\min} \le y_k &\le y_{\max} && \forall k = 0, \ldots, N \\ u_{\min} \le u_k &\le u_{\max} && \forall k = 0, \ldots, N \\ p_{\min} \le p &\le p_{\max}. \end{aligned}$$

Die Vektoren $z_k$ für $k = 0, \ldots, N$ seien durch ein Integrationsverfahren und Formel (2.21) bestimmt.

Für das Mehrfachschießverfahren wird das Zeitintervall $[t_0, t_f]$ in mehrere Integrationsintervalle unterteilt. Die Anzahl der Schießintervalle sei durch $N_s < N$ gegeben. Die Schießzeitpunkte $t_{s,l}$ mit $l = 0, \ldots, N_s$ müssen dazu auch im Gitter (2.18) vorhanden sein, allerdings muss nicht zwangsläufig jeder Gitterpunkt ein Schießzeitpunkt sein. Die Schießzeitpunkte bezüglich des Gitters notieren wir durch den Zusammenhang $t_{k(s,l)} = t_{s,l}$. Der Index $k(s, l)$ entspricht dabei dem Index des Zeitpunktes $t_{s,l}$ auf dem Gitter (2.18). Für den Anfangs- und Endzeitpunkt gilt $t_{s,0} = t_0$ und $t_{s,N_s} = t_N = t_f$. In Abbildung 2.3 sind

verschiedene Integrationsintervalle durch die Funktionen $z^{(l)} : [t_{s,l-1}, t_{s,l}) \to \mathbb{R}^{n_y}$ dargestellt. Um die Notation bezüglich des Gitters (2.18) zu vereinfachen, definieren wir die Integrationsintervalle in diskreter Form auf dem Gesamtintervall $[t_0, t_N]$

$$\begin{aligned} &\left[z_0^{(l)\top}, \ldots, z_N^{(l)\top}\right]^\top \\ &z_k^{(l)} = z^{(l)}(t_k; u_{k(s,l-1)}, \ldots, u_{k(s,l)}, y_{k(s,l-1)}, p) \in \mathbb{R}^{n_y}, \quad \text{mit } k(s,l-1) \le k \le k(s,l) \end{aligned} \tag{2.22}$$

für alle Schießintervalle $l = 1, \ldots, N_s$. Für den Algorithmus und die spätere Implementierung benötigen wir die diskreten Schießintervalle aus (2.22) nur auf dem Gitterabschnitt $\left[t_{k(s,l-1)}, t_{k(s,l)}\right)$. Beim Mehrfachschießverfahren entstehen, im Vergleich zum Einfachschießverfahren, an den Übergangspunkten Lücken $\Delta_{s,l}$ im Verlauf der Zustandsfunktion. Diese müssen durch zusätzliche diskrete Nebenbedingungen

$$z_{k(s,l)}^{(l+1)} - y_{k(s,l)} = \Delta_{s,l} = 0 \qquad \forall l = 1, \ldots, N_s - 1$$

geschlossen werden. Für das Mehrfachschießverfahren ergibt sich folgendes diskretisierte Optimierungsproblem.

**Problem 2.8 (Optimierungsproblem mit Mehrfachschießverfahren)**

Minimiere

$$\varphi(y_0, y_N, p)$$

bezüglich der Zustände $y_k \in \mathbb{R}^{n_y}$ aus (2.19), der Steuerungen $u_k \in \mathbb{R}^{n_u}$ aus (2.20) und der Parameter $p \in \mathbb{R}^{n_p}$ für $k = 0, \ldots, N$, unter den Nebenbedingungen

$$\begin{aligned} y_k &= z_k^{(l)} && \forall l = 1, \ldots, N_s \\ & && \forall k \text{ mit } t_k \in \left[t_{k(s,l-1)}, t_{k(s,l)}\right) \\ y_{k(s,l)} &= z_{k(s,l)}^{(l+1)} && \forall l = 1, \ldots, N_s - 1 \\ \psi_{\min} &\le \psi(y_0, y_N, p) \le \psi_{\max} && \\ v_{\min} &\le v(t_k, y_k, u_k, p) \le v_{\max} && \forall k = 0, \ldots, N \end{aligned}$$

und den Boxbeschränkungen

$$\begin{aligned} y_{\min} &\le y_k \le y_{\max} && \forall k = 0, \ldots, N \\ u_{\min} &\le u_k \le u_{\max} && \forall k = 0, \ldots, N \\ p_{\min} &\le p \le p_{\max}. && \end{aligned}$$

Die Vektoren $z_k^{(l)}$ seien durch ein Integrationsverfahren und Formel (2.22) bestimmt.

**Bemerkung 2.11.** *Weitere Details zur Diskretisierung von Optimalsteuerungsproblemen können in [13] und [40] nachgelesen werden, speziell zu Mehrfachschießverfahren auch in [19].*

## 2.4 Dynamische Programmierung

In diesem Abschnitt definieren wir den Lösungsansatz der dynamischen Programmierung für ein diskretes Optimalsteuerungsproblem, vgl. [47]. Eine detaillierte Darstellung der Algorithmen zur dynamischen Programmierung finden sich in [11], [12] und [8]. Die Diskretisierung von Optimalsteuerungsproblemen für die Anwendung der dynamischen Programmierung erfolgt im Rahmen dieser Arbeit problemspezifisch. Wir definieren auf dem Intervall $[t_0, t_f]$, mit $t_f = t_{n_t}$, das auf einem Gitter diskretisierte Optimalsteuerungsproblem:

**Problem 2.9 (Gitter Diskretisiertes Optimalsteuerungsproblem (DOSP))**

$$\min F_0(x,u) := \varphi(x(t_{n_t})) + \sum_{j=0}^{n_t-1} f_0(t_j, x(t_j), u(t_j))$$

mit den Nebenbedingungen

$$\left.\begin{aligned} x(t_{j+1}) &= f(t_j, x(t_j), u(t_j)) && j = 0, \dots, n_t - 1 \\ x(t_0) &= x_0 \\ x(t_j) &\in X(t_j) && j = 0, \dots, n_t \\ u(t_j) &\in U(t_j, x(t_j)) && j = 0, \dots, n_t. \end{aligned}\right\} \tag{2.23}$$

Dieses diskrete Optimalsteuerungsproblem ist auf dem Gitter

$$\mathbb{G}_t := \{t_j \mid j = 0, \dots, n_t\}$$

definiert. Die Steuerungen und Zustände entsprechen den Gitterfunktionen

$$u : \mathbb{G}_t \longrightarrow \mathbb{R}^{n_u} \qquad t_j \longmapsto u(t_j)$$

und

$$x : \mathbb{G}_t \longrightarrow \mathbb{R}^{n_x} \qquad t_j \longmapsto x(t_j)$$

mit $j = 0, \dots, n_t$.

Für das diskretisierte Optimalsteuerungsproblem aus Problem 2.9 soll durch eine Rückwärtsrechnung, ausgehend vom letzten Zeitpunkt $t_n$, eine Wertefunktion berechnet werden. Diese Rückwärtsrechnung verwendet im Berechnungsverlauf nur entsprechende Teilprobleme von Problem 2.9. Diese Teilprobleme definieren wir für den Zeitpunkt $t_k$, $k < n_t$ wie folgt:

**Problem 2.10 (DOSP im Zeitpunkt $t_k$)**

$$\min F_k(x,u) := \varphi(x(t_{n_t})) + \sum_{j=k}^{n_t-1} f_0(t_j, x(t_j), u(t_j))$$

mit den Nebenbedingungen

$$\left.\begin{array}{rll} x(t_{j+1}) &= f(t_j, x(t_j), u(t_j)) & j = k, \ldots, n_t - 1 \\ x(t_k) &= x_k & \\ x(t_j) &\in X(t_j) & j = k, \ldots, n_t \\ u(t_j) &\in U(t_j, x(t_j)) & j = k, \ldots, n_t. \end{array}\right\} \tag{2.24}$$

Dieses diskrete Optimalsteuerungsproblem ist auf dem Gitter

$$\mathbb{G}_t := \{t_j \mid j = k, \ldots, n_t\}$$

definiert. Die Steuerungen und Zustände entsprechen den Gitterfunktionen

$$u : \mathbb{G}_t \longrightarrow \mathbb{R}^{n_u} \qquad t_j \longmapsto u(t_j)$$

und

$$x : \mathbb{G}_t \longrightarrow \mathbb{R}^{n_x} \qquad t_j \longmapsto x(t_j)$$

mit $j = k, \ldots, n_t$.

Für $t_k = t_0$ entspricht Problem 2.10 dem Ausgangsproblem auf dem Intervall $[t_0, t_f]$, mit $t_f = t_{n_t}$ aus Problem 2.9.

Das Optimalitätsprinzip von Bellman definiert eine Rekursionsformel für eine optimale

Wertefunktion

$$W(t_k, x_k) := \begin{cases} \inf\limits_{x,u \text{ mit } (2.24)} F_k(x,u) & \text{falls Problem 2.9 lösbar ist} \\ \infty & \text{sonst} \end{cases},$$

durch die das diskrete Optimalsteuerungsproblem aus Problem 2.9 gelöst werden kann. Für unzulässige Zustände $x(t_j) \notin X(t_j)$ gilt $f_0(t_j, x(t_j), u(t_j)) = \infty$. Angenommen die optimale Wertefunktion $W(t_{j+1}, x)$ ist für den Zeitpunkt $t_{j+1}$ bekannt, dann können wir die optimale Wertefunktion im Zeitpunkt $t_j$ durch

$$W(t_j, x) = \min_{u \in U(t_j, x)} \{f_0(t_j, x, u) + W(t_{j+1}, f(t_j, x, u))\} \tag{2.25}$$

bestimmen. Für den letzten Zeitpunkt $t_{n_t}$ erfolgt eine Initialisierung der Wertefunktion durch

$$W(t_{n_t}, x_{n_t}) := \begin{cases} \varphi(x_{n_t}) & \text{für } x_{n_t} \in X(t_{n_t}) \\ \infty & \text{sonst} \end{cases}. \tag{2.26}$$

**Methode der dynamischen Programmierung nach Bellman:**

i Initialisierung der Wertefunktion im Zeitpunkt $t_{n_t}$ durch (2.26):

$$W(t_{n_t}, x_{n_t}) := \begin{cases} \varphi(x_{n_t}) & \text{für } x_{n_t} \in X(t_{n_t}) \\ \infty & \text{sonst} \end{cases}.$$

ii Rückwärtsrechnung:

(a) Berechne für $j = n_t - 1, \ldots, 0$, die Lösung $W(t_j, x)$ von (2.25).

(b) Speichere für jeden diskreten Zustand $x$ zu jedem Zeitpunkt $t_j$ die berechnete diskrete optimale Steuerung $u^*(t_j, x) \in U(t_j, x)$.

iii Vorwärtsrechnung:

(a) Wähle einen Startzustand $x(t_0) \in X(t_0)$ zum Zeitpunkt $t_0$.

(b) Berechne für $j = 0, \ldots, n_t - 1$, den Zustand $x(t_{j+1}) = f(t_j, x(t_j), u^*(t_j, x(t_j)))$ mit der optimalen diskreten Steuerung $u^*(t_j, x(t_j))$ aus der Rückwärtsrechnung.

**Bemerkung 2.12.** *Im Zusammenhang mit der dynamischen Programmierung als Lösungsstrategie für ein Optimalsteuerungsproblem fällt oft auch der Ausdruck „Fluch der Dimensionen“. Dieser Begriff resultiert daraus, dass der Speicher- und Rechenaufwand dieses Optimalsteuerungsverfahrens mit zunehmender Dimension des Optimalsteuerungsproblems exponentiell anwächst.*

## 2.5 Nichtlineare modellprädiktive Regelung

### 2.5.1 Grundlagen nichtlineare modellprädiktive Regelung

Im Vergleich zur klassischen Regelung fließt bei einer modellprädiktiven Regelung auch ein Modell der Systemdynamik in den Regelungsalgorithmus ein. Ein ausführlicher Überblick zur modellprädiktiven Regelung mit verschiedensten Varianten von Algorithmen kann in [45] gefunden werden. In dieser Arbeit wollen wir zunächst nur die grundlegenden Varianten der modellprädiktiven Regelung einführen.
Für dynamische Systeme, deren Dynamik durch gewöhnliche Differentialgleichungen beschrieben werden kann, formulieren wir die Problemstellung der modellprädiktiven Regelung als Optimalsteuerungsproblem.

**OSP 2.1 (Standard Optimalsteuerungsproblem für nichtlineare modellprädiktive Regelung)**

Bestimme eine Zustandsfunktion $y : [t_k, t_k + T] \to \mathbb{R}^{n_y}$, die zugehörige Steuerung $u : [t_k, t_k + T] \to \mathbb{R}^{n_u}$ und den zugehörigen Parametervektor $p \in \mathbb{R}^{n_p}$, so dass

$$\alpha_0 \underbrace{\varphi(y(t_k + T), p)}_{\substack{\text{Ökonomischer} \\ \text{Zielfunktionsanteil}}} + \alpha_1 \underbrace{\int_{t_k}^{t_k+T} \|y_{\text{ref}}(t) - y(t)\|_2 \, dt}_{\text{Trackinganteil}}$$

minimal wird, unter den Nebenbedingungen

$$\begin{aligned} y'(t) &= F(t, y(t), u(t), p) \\ v_{\min} \le v(t, y(t), u(t), p) &\le v_{\max} \qquad t \in [t_k, t_k + T] \end{aligned}$$

und den Boxbeschränkungen

$$\begin{aligned} y_{\min} \le y(t) &\le y_{\max} \qquad t \in [t_k, t_k + T] \\ u_{\min} \le u(t) &\le u_{\max} \qquad t \in [t_k, t_k + T] \\ p_{\min} \le p &\le p_{\max} \end{aligned}$$

und den Moving Horizon Anfangsbedingungen

$$y(t_k) = y_k.$$

Für die Nebenbedingungen aus OSP 2.1 gilt:

$$F : [t_0, t_f] \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_y}$$
$$v : [t_0, t_f] \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_v}.$$

In OSP 2.1 ist ein Optimalsteuerungsproblem definiert, das die nichtlineare modellprädiktive Regelung in einen Zusammenhang mit Optimalsteuerung bringt. Charakteristisch ist zum Einen die Zielfunktion und zum Anderen die Moving Horizon Anfangsbedingungen, vgl. [45, 23, 39, 42]. Der Begriff Moving Horizon ist darin als mitbewegter Vorausschau-Horizont der Länge $T$ zu verstehen.
In der Zielfunktion unterscheiden wir in OSP 2.1 in die beiden Anteile:

- **Ökonomischer Zielfunktionsanteil:** In diesem Term kann eine Kostenfunktion angegeben werden. Ziel der Regelung ist es somit, die ökonomischen Kosten möglichst gering zu halten.
- **Trackinganteil:** Dieser Term minimiert den Abstand zu einer Referenztrajektorie, welche durch $y_{\text{ref}} : [t_k, t_k + T] \to \mathbb{R}^{n_y}$ definiert ist.

Für $\alpha_0 = 0$ und $\alpha_1 > 0$ erhalten wir einen reinen Trackingregler, der versucht eine vorgegebene Bahn möglichst genau nachzuregeln.
Für $\alpha_0 > 0$ und $\alpha_1 = 0$ in der Zielfunktion von OSP 2.1 erhalten wir eine ökonomische modellprädiktive Regelung, welche auch unter dem Namen „Economic Model Predictive Control“ (EMPC) bekannt ist. Ein aktueller Überblick für die ökonomische modellprädiktive Regelung kann in [31] gefunden werden. Im Vergleich zur klassischen Regelung zum Verfolgen einer Referenztrajektorie wird hier auch ein ökonomisches Kostenfunktional betrachtet. Dies ist zum Beispiel in der Produktionstechnik interessant, wo die Produktionsgeschwindigkeit oder der Verbrauch von Energie optimiert werden kann. Ein anderes Beispiel wäre der Betrieb eines Satelliten, der mit möglichst wenig Treibstoffverbrauch Kollisionen mit anderen kosmischen Objekten vermeiden soll. Des Weiteren ist ökonomische modellprädiktive Regelung auch für die Bahnplanung autonomer Fahrzeuge relevant. Dazu kann beispielsweise nach dem Kraftstoffverbrauch, dem Komfort oder der schnellsten Trajektorie optimiert werden.

Je nach der Gewichtung der Faktoren $\alpha_0$ und $\alpha_1$ ist auch eine gemischte Variante von tracking und ökonomischer modellprädiktiver Regelung möglich.
Durch die Moving Horizon Anfangsbedingungen

$$y(t_k) = y_k \tag{2.27}$$

kann der Startzustand des Optimalsteuerungsproblems jeweils durch den aktuell gemessenen Systemzustand definiert werden. Damit lässt sich für die nichtlineare modellprädiktive Regelung folgender Algorithmus definieren.

**Standard nichtlineare modellprädiktive Regelung:**

i Messe den aktuellen Zustand $y_k \in \mathbb{R}^{n_y}$ und den aktuellen Zeitpunkt $t_k$.

ii Löse für $y_k \in \mathbb{R}^{n_y}$ das Optimalsteuerungsproblem aus OSP 2.1 auf dem Vorausschau-Horizont $T$.

iii Setze die neue Steuerung des Systems $u : [t_k, t_k + \Delta t] \to \mathbb{R}^{n_u}$ und wende diese auf das System an. Wobei $\Delta t$ für die Abtastrate der Regelung steht.

iv Gehe zurück zu i.

Eines der größten Probleme des Standard Algorithmus zur nichtlinearen modellprädiktiven Regelung ist, dass das Lösen von Optimalsteuerungsproblemen einen komplexen iterativen Prozess darstellt, der oft eine nicht vernachlässigbare Rechenzeit benötigt. Dementsprechend können während des Optimierens Zeitverzögerungen entstehen. Abhilfe kann hierbei eine mehrschritt nichtlineare modellprädiktive Regelung, oder auch Multistep NMPC, sein bei welcher der aktuelle Zustand $y_k \in \mathbb{R}^{n_y}$ mit einem Zeitversatz $M\Delta t$ in die Zukunft prognostiziert wird. Der Faktor $M > 0 \in \mathbb{N}$ steht dabei für die Anzahl der übersprungenen Schritte. Während des Zeitversatzes $M\Delta t$ wird weiterhin die vorherige MPC Lösung verwendet. Kann der Optimierer eine neue Lösung während des Zeitintervalls $M\Delta t$ finden, so wird diese bei erreichen des Zeitversatzes übernommen. Andernfalls muss ein weiteres Mal projiziert werden und der Optimierer versucht für den nächsten Zeitschritt eine Lösung zu finden.

**Multistep nichtlineare modellprädiktive Regelung:**

i **Initialisierung:** Messe den aktuellen Zustand $y_k \in \mathbb{R}^{n_y}$ und den aktuellen Zeitpunkt $t_k$ und löse für $y_k \in \mathbb{R}^{n_y}$ das Optimalsteuerungsproblem aus OSP 2.1 auf dem Vorausschau-Horizont $T$.

ii Setze die neue Steuerung des Systems $u : [t_k, t_k + M\Delta t] \to \mathbb{R}^{n_u}$ und wende diese auf das System an.

iii Messe den aktuellen Zustand $y_k \in \mathbb{R}^{n_y}$ und den aktuellen Zeitpunkt $t_k$ und projiziere den Zustand, auf der berechneten Lösung des Optimalsteuerungsproblems, für den Zeitpunkt $\tilde{t}_k = t_k + M\Delta t$. Daraus folgt der Zustand $\tilde{y}_k \in \mathbb{R}^{n_y}$.

iv Löse für $\tilde{y}_k \in \mathbb{R}^{n_y}$ und $\tilde{t}_k$ das Optimalsteuerungsproblem aus OSP 2.1 auf dem Vorausschau-Horizont $T$. Falls der Zeitpunkt $\tilde{t}_k$ erreicht wird, bevor das Optimalsteuerungsproblem gelöst werden konnte, wiederhole Schritt iii mit $\tilde{t}_k = \tilde{t}_k + M\Delta t$, solange $\tilde{t}_k < t_k + T$.

v Warte bis der Zeitpunkt $\tilde{t}_k$ erreicht wird und gehe zurück zu ii.

Die Vorteile dieses Regelkonzepts sind:

- Die Referenztrajektorie muss die Systemdynamik nicht strikt erfüllen.
- Die Rückführung auf die Referenztrajektorie erfolgt durch ein Modell des Systems und hält auch Beschränkungen für Zustände, Steuerungen und andere Nebenbedingungen ein.
- Die Kosten- bzw. Zielfunktion legt den Fokus auf eine optimale Prozessdurchführung. Die Gewichtung der Faktoren $\alpha_0$ und $\alpha_1$ legt dabei fest, wie die Kostenfunktion zum Trackingterm gewichtet wird.

Die Nachteile dieses Regelkonzepts sind:

- Es muss ein Modell des Systems in Form eines Differentialgleichungssystems bekannt sein.
- Dieser Ansatz steht und fällt mit der Robustheit und Berechnungsgeschwindigkeit des Optimierers. Sind die Rechenzeiten zu lang oder kann der Optimierer das Opti-

malsteuerungsproblem nicht lösen, so scheitert das Verfahren, wenn das System den Vorausschau-Horizont der letzten MPC Lösung erreicht hat.

- Die Referenztrajektorie muss vorher festgelegt werden. Es wird nicht überprüft ob diese den Prozess des Systems, bezüglich der Kosten- oder Zielfunktion, optimal steuert. Demnach ist die Lösung stark von der Wahl der Faktoren $\alpha_0$ und $\alpha_1$ abhängig.

**Bemerkung 2.13.** *Zahlreiche weitere Varianten von modellprädiktiven Regelungskonzepten können in [45] gefunden werden. Im Rahmen dieser Arbeit fokussieren wir uns auf eine ökonomische modellprädiktive Regelung mit hierarchischem Regelkonzept.*

### 2.5.2 Ökonomische modellprädiktive Regelung mit hierarchischem Regelkonzept

Die ökonomische modellprädiktive Regelung, welche auch unter dem Namen „Economic Model Predictive Control“ (EMPC) bekannt ist, definiert einen optimal gesteuerten Regelungsprozess. Die ökonomische modellprädiktive Regelung ergibt sich für das Optimalsteuerungsproblem aus OSP 2.1 im Fall $\alpha_0 > 0$ und $\alpha_1 = 0$. Somit betrachten wir ausschließlich ein ökonomisches Kostenfunktional.

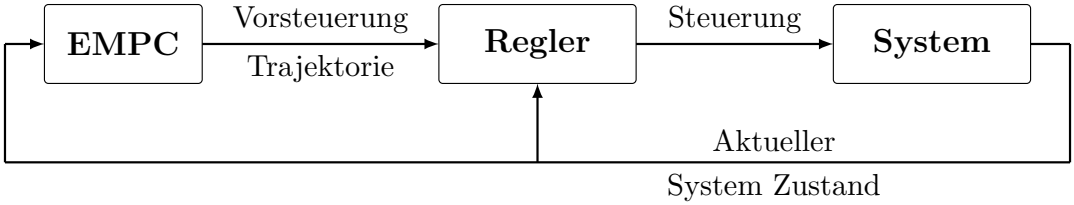


Abbildung 2.4: Schematischer Aufbau eines hierarchischen Regelkonzepts mit ökonomischer modellprädiktiver Regelung.

Im Rahmen dieser Arbeit verwenden wir zusammen mit der ökonomischen modellprädiktiven Regelung ein zweistufiges hierarchisches Regelkonzept. Dieses Regelkonzept ist in Abbildung 2.4 schematisch dargestellt. Die Steuerung der ökonomischen modellprädiktiven Regelung wird in einem Regler als Vorsteuerung verwendet. Der Regler soll dabei die optimierte Trajektorie nachregeln. Wichtig ist, dass der Regler und die ökonomische modellprädiktive Regelung in unterschiedlichen Frequenzen laufen können. Dabei ist die Frequenz des Reglers höher zu wählen, als die Frequenz der ökonomischen modellprädiktiven Regelung.

**Bemerkung 2.14.** *Der detaillierte Aufbau des hierarchischen Regelkonzepts wird im Verlauf dieser Arbeit jeweils für den konkreten Versuchsaufbau beschrieben.*

Die Vorteile dieses Regelkonzepts sind:

- Es ist keine vordefinierte Referenztrajektorie notwendig.
- Die Kosten- bzw. Zielfunktion legt den Fokus auf eine optimale Prozessdurchführung.
- Im Regelkonzept aus Abbildung 2.4 können Regler und Optimalsteuerung auch mit unterschiedlichen Frequenzen arbeiten.

Die Nachteile dieses Regelkonzepts sind:

- Es muss ein Modell des Systems in Form eines Differentialgleichungssystems bekannt sein.
- Dieser Ansatz steht und fällt mit der Robustheit und Berechnungsgeschwindigkeit des Optimierers. Sind die Rechenzeiten zu lang oder kann der Optimierer das Optimalsteuerungsproblem nicht lösen, so scheitert das Verfahren, wenn das System den Vorausschau-Horizont der letzten MPC Lösung erreicht hat.

Prinzipiell ist es auch möglich ohne die zusätzliche Reglerkomponente in Abbildung 2.4 auszukommen. Dabei müssten die Steuergrößen der ökonomischen modellprädiktiven Regelung, wie in Abschnitt 2.5.1, direkt verwendet werden. Bei einer solchen Regelung würde allerdings die Positionsablage zur berechneten optimalen Trajektorie nicht in die Regelung eingehen. Somit würden Modellierungsfehler in der Fahrzeugdynamik nicht ausgeregelt. Um eine robuste online-Steuerung für autonome Fahrzeuge zu erhalten, werden wir im Rahmen dieser Arbeit eine ökonomische modellprädiktive Regelung für autonome Onlinebahnplanung mit untergeordnetem Bahnfolgeregler entwickeln. Der Bahnfolgeregler wird dabei von der Forschungs- und Entwicklungsabteilung der Volkswagen AG zur Verfügung gestellt.

**Bemerkung 2.15.** *Zur Vereinfachung verwenden wir im Folgenden für die ökonomische modellprädiktive Regelung die Abkürzung MPC.*

# Kapitel 3

# Bahnplanung für autonome Fahrzeuge

## 3.1 Fahrzeugmodellierung

Eine der zentralsten Rollen in der optimalen Steuerung von Fahrzeugen spielt das Fahrzeugmodell. Durch eine geschickte Modellierung versucht man ein komplexes System zu beschreiben, welches bei einem Fahrzeug aus vielen verschiedenen mechanischen, elektronischen oder auch werkstoffspezifischen Komponenten besteht. Da Fahrzeugmodelle mit zunehmender Komplexität immer rechenaufwändiger werden, muss eine Abwägung zwischen Detailtreue und Rechenaufwand erfolgen. Ein umfassender Überblick zu verschiedenen Fahrzeugmodellen von Kraftfahrzeugen kann in [58] und [62] gefunden werden.

### 3.1.1 Kinematisches Fahrzeugmodell

Eines der einfachsten Fahrzeugmodelle kann durch reine Kinematik beschrieben werden. In [49] ist der Begriff Kinematik definiert. Laut Definition betrachtet die kinematische Modellierung nur die Geometrie der Bewegung. Die verwendeten Größen bestehen dabei aus Zeit, Ort, Geschwindigkeit, Richtung und geometrischen Beziehungen. Die Kinematik betrachtet keine Kräfte, welche die Ursache der Bewegungen sind.
Die geometrischen Beziehungen für das kinematische Fahrzeugmodell sind in Abbildung 3.1 dargestellt. Für die einzelnen Größen aus Abbildung 3.1 gilt:

- $x$: Kartesische $x$-Koordinate des Mittelpunkts der Hinterachse des Fahrzeugs.
- $y$: Kartesische $y$-Koordinate des Mittelpunkts der Hinterachse des Fahrzeugs.

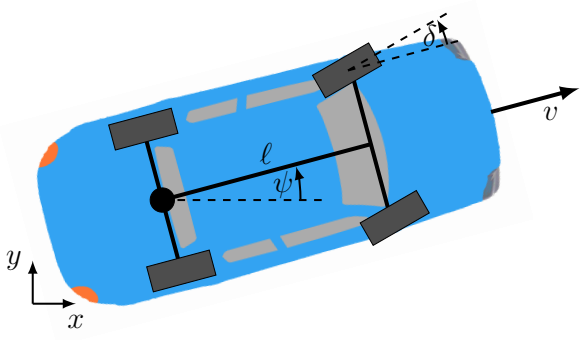


Abbildung 3.1: Schematische Darstellung des kinematischen Fahrzeugmodells

- $\psi$: Gierwinkel $\psi$ im Bezug auf die $x$-Achse des kartesischen Koordinatensystems.
- $v$: Fahrzeuggeschwindigkeit $v$ entlang der Fahrzeugausrichtung, welche durch den Gierwinkel $\psi$ gegeben ist.
- $\delta$: Lenkwinkel $\delta$, welcher an der Vorderachse des Fahrzeugs anliegt.
- $\ell$: Geometrischer Abstand zwischen Vorder- und Hinterachse.

**DGL-System 3.1 (Kinematisches Fahrzeugmodell)**

$$
\begin{aligned}
x'(t) &= v(t)\cos(\psi(t)) \\
y'(t) &= v(t)\sin(\psi(t)) \\
\psi'(t) &= \frac{v(t)}{\ell}\tan(\delta(t))
\end{aligned}
$$

Das Differentialgleichungssystem 3.1 beschreibt die kinematischen Bewegungsgleichungen der Hinterachse eines Fahrzeugs in Abhängigkeit von den Steuergrößen Geschwindigkeit $v$ und Lenkwinkel $\delta$. Verschiedene Modellierungen für die Kinematik von Kraftfahrzeugen werden in [62] vorgestellt.

### 3.1.2 Einspurmodell

Das Einspurmodell kann in unterschiedlichen Detailstufen formuliert werden. Im Rahmen dieser Arbeit wollen wir ein verhältnismäßig komplexes Einspurmodell nach dem

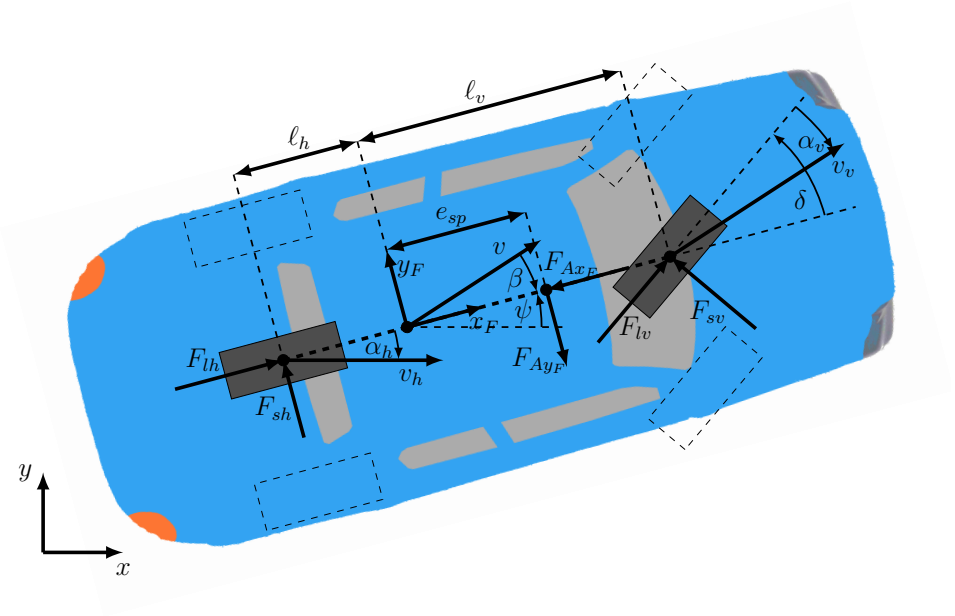


Abbildung 3.2: Schematische Darstellung des Einspurmodells

Vorbild von [42] verwenden. In Abbildung 3.2 sind die dynamischen Großen des Einspurmodells skizziert. Im Unterschied zum kinematischen Fahrzeugmodell, modelliert das Einspurmodell auch angreifende Kräfte am Fahrzeug. In Abbildung 3.2 sind zwei Koordinatensysteme dargestellt. Zum Einen das ortsfeste kartesische $xy$-Koordinatensystem und zum Anderen das körperfeste $x_F y_F$-Koordinatensystem, welches im Schwerpunkt des Fahrzeugs definiert ist. Für die einzelnen Größen aus Abbildung 3.2 gilt:

- $x$: Kartesische $x$-Koordinate des Schwerpunkts des Fahrzeugs.
- $y$: Kartesische $y$-Koordinate des Schwerpunkts des Fahrzeugs.
- $\psi$: Gierwinkel $\psi$ im Bezug auf die $x$-Achse des kartesischen Koordinatensystems.
- $v$: Fahrzeuggeschwindigkeit $v$ am Schwerpunkt.
- $v_v$: Fahrzeuggeschwindigkeit $v_v$ an der Vorderachse.
- $v_h$: Fahrzeuggeschwindigkeit $v_h$ an der Hinterachse.
- $\alpha_v$: Schräglaufwinkel $\alpha_v$ an der Vorderachse.
- $\alpha_h$: Schräglaufwinkel $\alpha_h$ an der Hinterachse.
- $\beta$: Schwimmwinkel $\beta$ am Schwerpunkt.

- $\delta$: Lenkwinkel $\delta$, welcher an der Vorderachse des Fahrzeugs anliegt.
- $F_{lv}$: Längskraft $F_{lv}$ an der Vorderachse.
- $F_{lh}$: Längskraft $F_{lh}$ an der Hinterachse.
- $F_{sv}$: Querkraft $F_{sv}$ an der Vorderachse.
- $F_{sh}$: Querkraft $F_{sh}$ an der Hinterachse.
- $F_{Ax_F}$: Aerodynamische Kraft $F_{Ax_F}$ entlang der körperfesten $x_F$-Achse.
- $F_{Ay_F}$: Aerodynamische Kraft $F_{Ay_F}$ entlang der körperfesten $y_F$-Achse.
- $\ell_v$: Geometrischer Abstand zwischen Schwerpunkt und Vorderachse.
- $\ell_h$: Geometrischer Abstand zwischen Schwerpunkt und Hinterachse.
- $e_{sp}$: Geometrischer Abstand zwischen Schwerpunkt und aerodynamischem Kraftangriffspunkt.

Weitere Größen, die für die Definition des Einspurmodells notwendig sind:

- $v_x$: Geschwindigkeit des Fahrzeugschwerpunkts entlang der $x$-Achse des ortsfesten $xy$-Koordinatensystems.
- $v_y$: Geschwindigkeit des Fahrzeugschwerpunkts entlang der $y$-Achse des ortsfesten $xy$-Koordinatensystems.
- $w_\psi$: Winkelgeschwindigkeit des Gierwinkels $\psi$.
- $w_\delta$: Winkelgeschwindigkeit des Lenkwinkels $\delta$.
- $F_{x_F}$: Kombinierte Gesamtkraft $F_{x_F}$ entlang der körperfesten $x_F$-Achse.
- $F_{y_F}$: Kombinierte Gesamtkraft $F_{y_F}$ entlang der körperfesten $y_F$-Achse.

Das Einspurmodell ist mit diesen Größen nach [42] wie folgt definiert.

**DGL-System 3.2 (Einspurmodell)**

$$
\begin{aligned}
x'(t) &= v_x(t) \\
y'(t) &= v_y(t) \\
\psi'(t) &= w_\psi(t) \\
v_x'(t) &= \frac{1}{m}\left[F_{x_F}(t)\cdot\cos\psi(t) - F_{y_F}(t)\cdot\sin\psi(t)\right] \\
v_y'(t) &= \frac{1}{m}\left[F_{x_F}(t)\cdot\sin\psi(t) + F_{y_F}(t)\cdot\cos\psi(t)\right] \\
w_\psi'(t) &= \frac{1}{I_{zz}}\left[F_{sv}(t)\cdot l_v\cdot\cos\delta(t) - F_{sh}(t)\cdot l_h - F_{Ay_F}(t)\cdot e_{sp} + F_{lv}(t)\cdot l_v\cdot\sin\delta(t)\right] \\
\delta'(t) &= w_\delta(t).
\end{aligned}
$$

Aus dem Schwimmwinkel $\beta$ können die Schräglaufwinkel

$$\alpha_v = \delta - \arctan\left(\frac{l_v\psi' - v\sin\beta}{v\cos\beta}\right), \tag{3.1}$$

$$\alpha_h = \arctan\left(\frac{l_h\psi' + v\sin\beta}{v\cos\beta}\right) \tag{3.2}$$

berechnet werden. Die Kräfte in $x_F$- bzw. $y_F$-Richtung ergeben sich aus

$$
\begin{aligned}
F_{x_F} &= F_{lh} - F_{Ax_F} + F_{lv}\cdot\cos(\delta) - F_{sv}\cdot\sin(\delta) \\
F_{y_F} &= F_{sh} - F_{Ay_F} + F_{lv}\cdot\sin(\delta) + F_{sv}\cdot\cos(\delta).
\end{aligned}
$$

Für den aus der Widerstandsfläche $A$ resultierenden Luftwiderstand gilt

$$
\begin{aligned}
F_{Ax_F} &= \frac{1}{2}\cdot c_w\cdot\rho\cdot A\cdot v^2 \\
F_{Ay_F} &\approx 0.
\end{aligned}
$$

Zur Berechnung der Querkräfte an Vorder- und Hinterachse wird die „Magic Formula von Pacejka“ nach [42]

$$
\begin{aligned}
F_{sv}(\alpha_v) &= \mu D_v\cdot\sin(C_v\cdot\arctan(B_v\alpha_v - E_v(B_v\alpha_v - \arctan(B_v\alpha_v)))) \\
F_{sh}(\alpha_h) &= \mu D_h\cdot\sin(C_h\cdot\arctan(B_h\alpha_h - E_h(B_h\alpha_h - \arctan(B_h\alpha_h))))
\end{aligned}
$$

angewandt. Der Parameter $\mu \in (0, 1]$ gibt dabei den Reibungsbeiwert der Haftreibung an. Bei einer trockenen Fahrbahn kann ein Reibungsbeiwert von $\mu = 1$ verwendet werden, während bei nasser oder schneebedeckter Fahrbahn ein kleinerer Wert auftritt. Die Querkräfte hängen damit direkt von den Schräglaufwinkeln $\alpha_v$ und $\alpha_h$, der Vorder- bzw. Hinterachse ab. Die Längskräfte setzen sich nach [42] aus der Radreibung und der Bremskraft

$$F_{lv} = -F_{bv} - F_{rv}$$
$$F_{lh} = -F_{bh} - F_{rh}$$

an Vorder- und Hinterachse zusammen.
Brems- und Antriebskraft werden zu $F_B$ zusammengefasst und mit dem Vorzeichen einer Bremskraft modelliert. Mit der Formel

$$F_{bv} = \begin{cases} s_v F_B & \text{falls } F_B \geq \Delta \\ a_0 + a_1 F_B + a_2 F_B^2 + a_3 F_B^3 & \text{falls } 0 \leq F_B < \Delta \\ b_0 + b_1 F_B + b_2 F_B^2 + b_3 F_B^3 & \text{falls } -\Delta < F_B < 0 \\ \sigma_v F_B & \text{falls } F_B \leq -\Delta \end{cases}$$

soll ein stetig differenzierbarer Übergang an der Stelle $F_B = 0$ gewährleistet werden. Aus den Bedingungen

$$\begin{aligned} F_{bv}(0) &= 0 = a_0 = b_0 \\ F'_{bv}(0) &= a_1 = b_1 \\ \frac{1}{2} F''_{bv}(0) &= a_2 = b_2 \\ F_{bv}(\Delta) &= s_v \Delta \\ F'_{bv}(\Delta) &= s_v \\ F_{bv}(-\Delta) &= -\sigma_v \Delta \\ F'_{bv}(-\Delta) &= \sigma \end{aligned}$$

folgt das lineare Gleichungssystem

$$\begin{bmatrix} 1 & \Delta & \Delta^2 & 0 \\ 1 & 2\Delta & 3\Delta^2 & 0 \\ 1 & -\Delta & 0 & \Delta^2 \\ 1 & -2\Delta & 0 & 3\Delta^2 \end{bmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ b_3 \end{pmatrix} = \begin{pmatrix} s_v \\ s_v \\ \sigma_v \\ \sigma_v \end{pmatrix}.$$

Zusammengefasst ergibt sich nach dem Lösen des linearen Gleichungssystems für die Koeffizienten

$$\begin{aligned} a_0 &= b_0 = 0 \\ a_1 &= b_1 = \frac{s_v + \sigma_v}{2} \\ a_2 &= b_2 = \frac{s_v - \sigma_v}{\Delta} \\ a_3 &= -\frac{s_v - \sigma_v}{2\Delta^2} \\ b_3 &= \frac{s_v - \sigma_v}{2\Delta^2}. \end{aligned}$$

Die Vorzeichen behaftete Bremskraft der Vorderachse

$$F_{bv} = \begin{cases} s_v F_B & \text{falls } F_B \geq \Delta \\ \frac{s_v+\sigma_v}{2} F_B + \frac{s_v-\sigma_v}{\Delta} F_B^2 - \frac{s_v-\sigma_v}{2\Delta^2} F_B^3 & \text{falls } 0 \leq F_B < \Delta \\ \frac{s_v+\sigma_v}{2} F_B + \frac{s_v-\sigma_v}{\Delta} F_B^2 + \frac{s_v-\sigma_v}{2\Delta^2} F_B^3 & \text{falls } -\Delta < F_B < 0 \\ \sigma_v F_B & \text{falls } F_B \leq -\Delta \end{cases} \tag{3.3}$$

ist durch einen Parameter $\sigma_v$ charakterisiert. In diesen geht die Antriebstopologie in der Form

$$\sigma_v : \begin{cases} \sigma_v = 1 & \text{für Frontradantrieb} \\ 0 < \sigma_v < 1 & \text{für Allradantrieb} \\ \sigma_v = 0 & \text{für Heckradantrieb} \end{cases}$$

ein. Der Faktor $s_v = l_h/(l_v + l_h)$ gibt den Anteil der Bremskraft an der Vorderachse an. Die Formel (3.3) gilt analog auch für die Hinterachse mit $\sigma_h = 1 - \sigma_v$ und $s_h = 1 - s_v$. Für den Übergangsbereich aus Formel (3.3) verwenden wir $\Delta = 0{,}01$.
Die Rollreibung

$$F_{rv} = f_r(v) \cdot F_{zv} \qquad\qquad F_{rh} = f_r(v) \cdot F_{zh}$$

setzt sich nach [42] aus den statischen Radlasten

$$\begin{aligned} F_{zv} &= m \cdot g \cdot s_v \\ F_{zh} &= m \cdot g \cdot s_h \end{aligned} \tag{3.4}$$

und der empirischen Formel

$$f_r(v) = f_{r0} + f_{r1} \frac{v}{100} + f_{r4} \left(\frac{v}{100}\right)^4 \tag{3.5}$$

zusammen. Wobei die Geschwindigkeit $v$ aus (3.5) nach [42] in der Einheit $\frac{\text{km}}{\text{h}}$ eingeht. Mit Hilfe der Radkräfte kann auch der zur Beschränkung der Haftreibung verwendete Kammsche Kreis

$$\frac{\sqrt{F_{lv}^2 + F_{sv}^2}}{F_{zv}} - \mu \leq 0 \tag{3.6}$$

$$\frac{\sqrt{F_{lh}^2 + F_{sh}^2}}{F_{zh}} - \mu \leq 0 \tag{3.7}$$

definiert werden. Hierbei sind die maximalen Reifenkräfte für Vorder- (3.6) bzw. Hinterachse (3.7) auf einem Kreis, entsprechend der Längs- und Querkräfte beschränkt. Der Parameter $\mu \in (0, 1]$ gibt wie auch in der „Magic Formula von Pacejka“ den Reibungsbeiwert der Haftreibung an.

### 3.1.3 Krummlinige Koordinaten

Krummlinige Koordinaten bieten eine Möglichkeit, die Fahrzeugdynamik in einem geeigneteren Koordinatensystem zu betrachten. Eine Optimierung von Rundenzeiten unter der Verwendung von krummlinigen Koordinaten findet sich bereits in [55]. In diesem Abschnitt wollen wir die kartesisch modellierten Fahrzeugmodelle in ein nach der Bogenlänge parametrisiertes Koordinatensystem überführen. Das bedeutet im zweidimensionalen Fall, die beiden Differentialgleichungen

$$x'(t) = v_x(t) \tag{3.8}$$

$$y'(t) = v_y(t) \tag{3.9}$$

durch Differentialgleichungen der Bogenlänge $s(t)$ und des Abstands $r(t)$ von einer Kurve $\gamma : [0, L] \longrightarrow \mathbb{R}^2$, in unserem Fall die Mittellinie, mit

$$\gamma(s) := \begin{pmatrix} x_m(s) \\ y_m(s) \end{pmatrix} \tag{3.10}$$

zu ersetzen. Zwei bedeutende Größen der Kurve $\gamma$, welche von der Bogenlängen abhängen sind durch den Winkel $\psi(s)$ und die Krümmung $\kappa(s)$ gegeben. Der Winkel berechnet sich durch die ersten Ableitungen

$$\psi_m(s) = \arctan\left(\frac{y_m'(s)}{x_m'(s)}\right) \tag{3.11}$$

nach der Bogenlänge. Vorausgesetzt die Kurve $\gamma$ ist bezüglich der Bogenlänge $s$ mit $\|\gamma'(s)\| = 1$ parametrisiert.

**Bemerkung 3.1.** *Die rechten Seiten der Gleichungen (3.8) und (3.9) sind durch Differentialgleichungen beschrieben. Wichtig für eine allgemeine Herleitung ist lediglich, dass die Differentialgleichungen von $v_x(t)$ und $v_y(t)$ nicht von $x(t)$ oder $y(t)$ abhängen dürfen, was bei Fahrzeugmodellen in der Regel nicht der Fall ist.*

Mit der Kurve aus (3.10) als Mittellinie der Fahrbahn, können wir jeden Punkt $(x, y)$ der Fahrbahn durch die Größen $s \in [0, L]$ und $r \in \left[-\frac{b(s)}{2}, \frac{b(s)}{2}\right]$

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} x_m(s(t)) \\ y_m(s(t)) \end{pmatrix} + n_m(s(t))r(t) \tag{3.12}$$

beschreiben. Die Ausdehnung der Bahn ist durch eine Funktion der Bahnbreite $b(s) > 0$, abhängig von der Bogenlänge $s \in [0, L]$ charakterisiert. Als Normalenvektor der Kurve (3.10) können wir bei einer, nach der Bogenlänge parametrisieren Kurve, mit $\|\gamma'(s)\| = 1$,

$$n_m(s) = \begin{pmatrix} y'_m(s) \\ -x'_m(s) \end{pmatrix} \tag{3.13}$$

verwenden. Unsere Wahl des Normalenvektors (3.13) nach Abbildung 3.3 führt dazu, dass

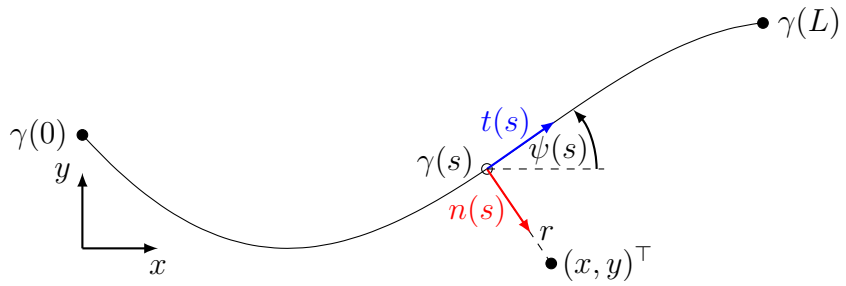


Abbildung 3.3: Bogenparametrisierung der Koordinaten

Punkte mit positiven $r$-Werten rechts und Punkte mit negativen $r$-Werten links der Kurve $\gamma(s)$ liegen. Mit (3.13) ergibt sich aus Gleichung (3.12)

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} x_m(s(t)) + r(t)y'_m(s(t)) \\ y_m(s(t)) - r(t)x'_m(s(t)) \end{pmatrix}. \tag{3.14}$$

Im nächsten Schritt werden die beiden Gleichungen aus (3.14) bezüglich der Zeit $t$ abgeleitet und mit (3.8) und (3.9) gleichgesetzt. Daraus ergibt sich

$$v_x(t) = x'(t) = \left[x'_m(s(t)) + r(t)y''_m(s(t))\right] s'(t) + r'(t)y'_m(s(t)) \tag{3.15}$$

$$v_y(t) = y'(t) = \left[y'_m(s(t)) - r(t)x''_m(s(t))\right] s'(t) - r'(t)x'_m(s(t)). \tag{3.16}$$

Um eine Gleichung für $s'(t)$ zu erhalten, multiplizieren wir (3.15) mit $x'_m(s(t))$ und die Gleichung (3.16) mit $y'_m(s(t))$. Anschließend addieren wir die beiden entstandenen Gleichungen. Unter der Berücksichtigung, dass $\|\gamma(s)\|^2 = 1$ ergibt sich

$$x'_m(s(t))v_x(t) + y'_m(s(t))v_y(t) = [1 + r(t)(x'_m(s(t))y''_m(s(t)) - x''_m(s(t))y'_m(s(t)))]\, s'(t).$$

Wir erhalten die Differentialgleichung für $s(t)$ gegeben durch

$$s'(t) = \frac{x'_m(s(t))v_x(t) + y'_m(s(t))v_y(t)}{1 + r(t)[x'_m(s(t))y''_m(s(t)) - x''_m(s(t))y'_m(s(t))]}.$$

Analog können wir die Gleichung (3.15) mit $y'_m(s(t))$ und (3.16) mit $-x'_m(s(t))$ multiplizieren und die resultierenden Gleichungen addieren. Unter Verwendung der Eigenschaften, $\|\gamma(s)\|^2 = 1$ und $\langle \gamma'(s), \gamma''(s) \rangle = 0$, einer nach der Bogenlänge parametrisierten Kurve ergibt sich für $r(t)$ die Differentialgleichung

$$r'(t) = v_x(t)y'_m(s(t)) - v_y(t)x'_m(s(t)).$$

Zusammengefasst können wir somit das Differentialgleichungssystem, bestehend aus den Gleichungen (3.8) und (3.9) durch das Differentialgleichungssystem

$$s'(t) = \frac{x'_m(s(t))v_x(t) + y'_m(s(t))v_y(t)}{1 + r(t)\kappa_m(s(t))} \tag{3.17}$$

$$r'(t) = v_x(t)y'_m(s(t)) - v_y(t)x'_m(s(t)) \tag{3.18}$$

ersetzen. Hierbei verwenden wir die Krümmung der Kurve $\gamma$

$$\kappa_m(s) = x'_m(s(t))y''_m(s(t)) - x''_m(s(t))y'_m(s(t)). \tag{3.19}$$

Kurven $\gamma$, wie die Mittellinie oder die Fahrbahnränder des Testkurses, modellieren wir im Rahmen dieser Arbeit durch kubische Splines vgl. [32]. Für geschlossene Kurven verwenden wir dabei periodische, für offene Kurven natürliche Randbedingungen bei der Berechnung des kubischen Splines.

**Bemerkung 3.2.** *Ein Algorithmus zur Projektion von kartesischen Koordinaten auf kubische Splines wird in [47], einer im Rahmen dieser Arbeit entstandenen Publikation vorgestellt.*

## 3.2 Hindernismodellierung

### 3.2.1 Hindernisse am Fahrbahnrand

Hindernisse an den Fahrbahnrändern können durch Hindernisfunktionen modelliert werden. Um den zweimal stetig differenzierbaren Übergang der Hindernisfunktion zu gewährleisten wird eine stückweise Übergangsfunktion definiert. Die Funktion der aufsteigenden

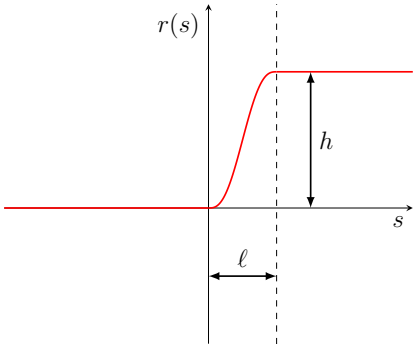


Abbildung 3.4: Aufsteigender Hindernisübergangsbereich

Hinderniskurve ergibt sich aus

$$r(s) = \begin{cases} 0 & \text{falls } s \leq 0 \\ a_0 + a_1 s + a_2 s^2 + a_3 s^3 + a_4 s^4 + a_5 s^5 & \text{falls } 0 < s < \ell \\ h & \text{falls } s \geq \ell \end{cases} .$$

Wir forden an den Übergängen einen zweimal stetig differenzierbaren Übergang mit den Bedingungen

$$\begin{aligned} r(0) &= 0 & r(\ell) &= h \\ r'(0) &= 0 & r'(\ell) &= 0 \\ r''(0) &= 0 & r''(\ell) &= 0. \end{aligned}$$

Hieraus folgt das lineare Gleichungssystem

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & \ell & \ell^2 & \ell^3 & \ell^4 & \ell^5 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2\ell & 3\ell^2 & 4\ell^3 & 5\ell^4 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 6\ell & 12\ell^2 & 20\ell^3 \end{bmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{pmatrix} = \begin{pmatrix} 0 \\ h \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} .$$

Zusammengefasst ergibt sich nach dem Lösen des linearen Gleichungssystems für die Koeffizienten

$$\begin{aligned} a_0 &= 0, & a_1 &= 0, \\ a_2 &= 0, & a_3 &= \frac{10h}{\ell^3}, \\ a_4 &= -\frac{15h}{\ell^4}, & a_5 &= \frac{6h}{\ell^5}. \end{aligned}$$

Die aufsteigende Hindernisfunktion ergibt sich damit zu

$$r(s) = \begin{cases} 0 & \text{falls } s \leq 0 \\ \frac{10h}{\ell^3}s^3 - \frac{15h}{\ell^4}s^4 + \frac{6h}{\ell^5}s^5 & \text{falls } 0 < s < \ell \\ h & \text{falls } s \geq \ell \end{cases},$$

$$r'(s) = \begin{cases} 0 & \text{falls } s \leq 0 \\ \frac{30h}{\ell^3}s^2 - \frac{60h}{\ell^4}s^3 + \frac{30h}{\ell^5}s^4 & \text{falls } 0 < s < \ell \\ 0 & \text{falls } s \geq \ell \end{cases}.$$

Um die Fahrbahn nach dem Hindernis wieder anzupassen wird eine Rückführungsfunktion benötigt. Für die Rückführungskurve kann der gleiche Ansatz

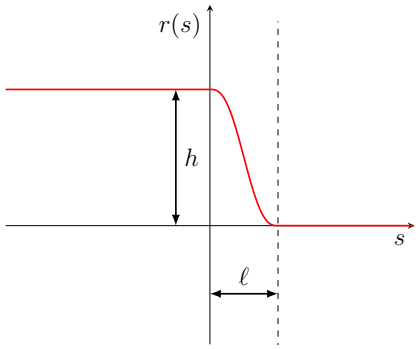


Abbildung 3.5: Absteigender Hindernisübergangsbereich

$$r(s) = \begin{cases} 0 & \text{falls } s \leq 0 \\ b_0 + b_1 s + b_2 s^2 + b_3 s^3 + b_4 s^4 + b_5 s^5 & \text{falls } 0 < s < \ell \\ h & \text{falls } s \geq \ell \end{cases}$$

verwendet werden. Die angepassten Übergangsbedingungen ergeben sich zu

$$\begin{aligned} r(0) &= h & r(\ell) &= 0 \\ r'(0) &= 0 & r'(\ell) &= 0 \\ r''(0) &= 0 & r''(\ell) &= 0. \end{aligned}$$

Im linearen Gleichungssystem

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & \ell & \ell^2 & \ell^3 & \ell^4 & \ell^5 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2\ell & 3\ell^2 & 4\ell^3 & 5\ell^4 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 6\ell & 12\ell^2 & 20\ell^3 \end{bmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix} = \begin{pmatrix} h \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

verändert sich lediglich die rechte Seite. Zusammengefasst ergibt sich nach dem Lösen des linearen Gleichungssystems für die Koeffizienten

$$b_0 = h, \qquad b_1 = 0,$$
$$b_2 = 0, \qquad b_3 = -\frac{10h}{\ell^3},$$
$$b_4 = \frac{15h}{\ell^4}, \qquad b_5 = -\frac{6h}{\ell^5}.$$

Die Rückführungsfunktion gilt damit

$$r(s) = \begin{cases} h & \text{falls } s \leq 0 \\ h - \frac{10h}{\ell^3}s^3 + \frac{15h}{\ell^4}s^4 - \frac{6h}{\ell^5}s^5 & \text{falls } 0 < s < \ell \\ 0 & \text{falls } s \geq \ell \end{cases},$$
$$r'(s) = \begin{cases} 0 & \text{falls } s \leq 0 \\ -\frac{30h}{\ell^3}s^2 + \frac{60h}{\ell^4}s^3 - \frac{30h}{\ell^5}s^4 & \text{falls } 0 < s < \ell \\ 0 & \text{falls } s \geq \ell \end{cases}.$$

## 3.3 Versuchsaufbau für offline-Fahrversuche

Alle Fahrversuche, welche im Rahmen dieser Arbeit durchgeführt wurden, waren Teil eines Projekts mit der Forschungs- und Entwicklungsabteilung der Volkswagen AG. Als Testfahrzeug kam ein VW Golf GTI aus Abbildung 3.6 zum Einsatz. Die Fahrversuche fanden auf abgesperrten und abgesicherten Rennstrecken statt. Zum Einen wurden diese auf der Rennstrecke Autodrom Most, aus Abbildung 3.7, in Most (Tschechien) und zum Anderen auf der Rennstrecke Autodromo do Algarve, aus Abbildung 3.8 in Portimao (Portugal) durchgeführt.

Abbildung 3.6: VW Golf GTI.


Abbildung 3.7: Satellitenbild der Rennstrecke in Most aus Google Maps.


Abbildung 3.8: Satellitenbild der Rennstrecke in Portimao aus Google Maps.

Ziel der offline-Fahrversuche ist es, zuvor berechnete Bahnen mit einem Bahnfolgeregler nachzufahren und die Trajektorien zusammen mit Testfahrern hinsichtlich ihrer Optimalität zu bewerten. In Abbildung 3.9 ist der Aufbau zur Durchführung der offline-Fahrversuche skizziert.

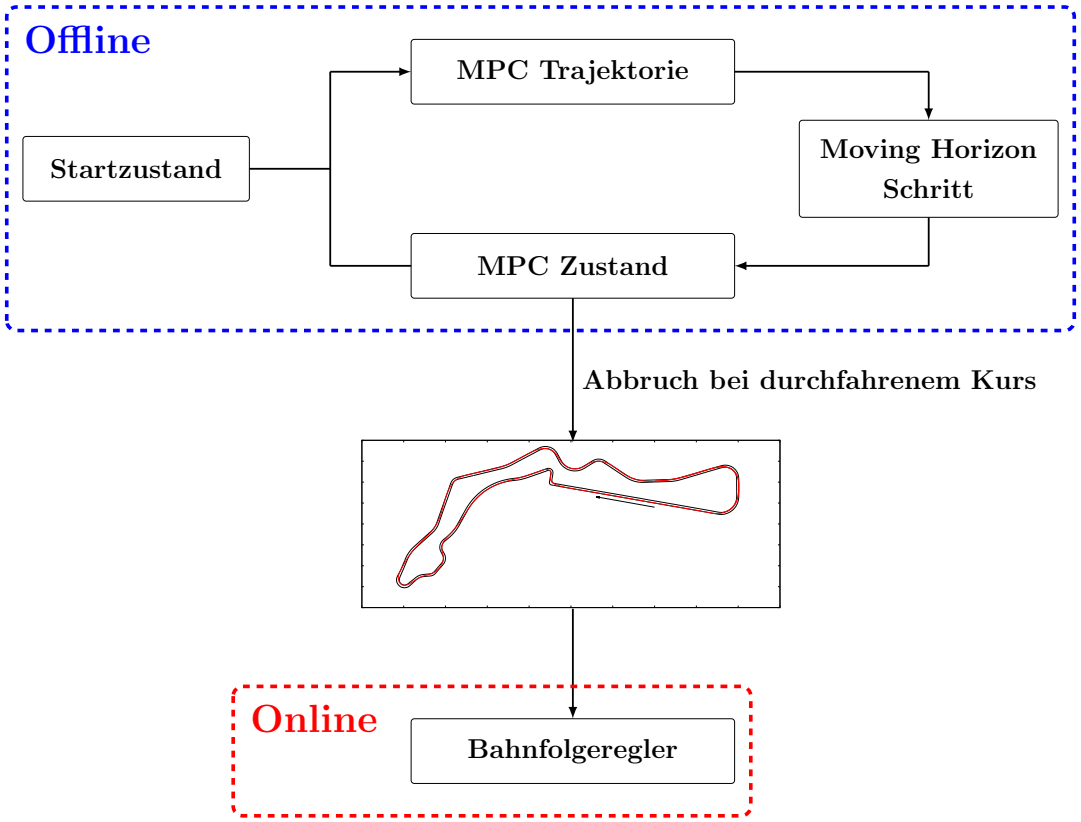


Abbildung 3.9: Schematischer Aufbau der offline-Fahrversuche

**Bemerkung 3.3.** *Der in verwendete Bahnfolgeregler wurde von der Forschungs- und Entwicklungsabteilung der Volkswagen AG bereitgestellt. Die Entwicklung, Anpassung und Inbetriebnahme des Reglers wurde vom Projektpartner übernommen. Im Rahmen der Fahrversuche dieser Arbeit verwenden wir diesen Bahnfolgeregler dementsprechend als Blackbox. Der verwendete Regler benötigt zum Abfahren einer berechneten Rennlinie zusätzlich zur Bahn auch ein Krümmungs-, Geschwindigkeits- und Beschleunigungsprofil der Trajektorie.*

Die Optimierung der Bahn führen wir hierbei komplett offline, das heißt vor der Testfahrt durch. Die MPC-Idee passen wir so an, dass nach jeder berechneten MPC Trajektorie ein Moving Horizon Schritt mit konstanter Schrittweite angewandt wird. Durch diese Methode

setzt sich die resultierende Bahn aus kleinen MPC Segmenten zusammen, die nach einem kompletten Durchlaufen des Testkurses eine optimierte Trajektorie ergeben. Diese entstandene Rennlinie kann anschließend im Testfahrzeug an den Bahnfolgeregler übergeben werden, der diese daraufhin online nachfährt. So kann ein kontrollierter Vergleich zwischen manuell gefahrenen und autonomen Trajektorien erfolgen, der anschließend zusammen mit erfahrenen Testfahrern bewertet werden kann.

## 3.4 Bahnplanung mit dynamischer Programmierung

### 3.4.1 Motivation

Das Verfahren der dynamischen Programmierung, zur Lösung von Optimalsteuerungsproblemen, ist bereits in Abschnitt 2.4 definiert. Im wesentlichen unterteilen wir das Verfahren der dynamischen Programmierung dort in zwei Lösungsphasen. Zunächst muss für das Optimalsteuerungsproblem eine Rückwärtsrechnung erfolgen, bei der eine Wertefunktion auf einem diskreten Zustandsgitter berechnet wird. Diese Rückwärtsrechnung ist in der Regel sehr rechenaufwändig, da sowohl Speicher- als auch der Rechenaufwand exponentiell mit der Dimension der Zustände steigen. Interessant im Bezug auf online-Optimierungen ist hingegen die zweite Phase der Vorwärtsrechnung. Sind zusätzlich zur Wertefunktion auch die optimalen Steuerungen auf dem Zustandsgitter abgelegt worden, so kann die Vorwärtsrechnung durch eine einfache Interpolation auf das Zustandsgitter erfolgen. Dies ist in der Regel auch online lösbar.
Ein weiterer Vorteil der dynamischen Programmierung im Vergleich zu anderen Algorithmen ist, das es sich um ein gradientenfreies Verfahren handelt. Somit werden weder die ersten noch zweiten Ableitungen von Zielfunktion, Nebenbedingungen und des zugrunde liegenden Differentialgleichungssystems benötigt.

### 3.4.2 Diskretisierung des Optimalsteuerungsproblems

Für die dynamische Programmierung müssen Gitterfunktionen über alle Zustände abgespeichert und berechnet werden. Dies verlangt nach einer effizienten Implementierung, so dass möglichst wenige unzulässige Zustände im Gitter enthalten sind. Kartesische Koordinaten erweisen sich bei unseren Problemen als ineffizient, da bei einem Rundkurs, wie in Abbildung 3.10, sehr viele Zustände abseits des Kurses auftreten.
Da der Speicher- und Rechenaufwand exponentiell zu der Anzahl der verwendeten Zustände wächst, ist es ratsam, ein verhältnismäßig einfaches Fahrzeugmodell, wie das ki-

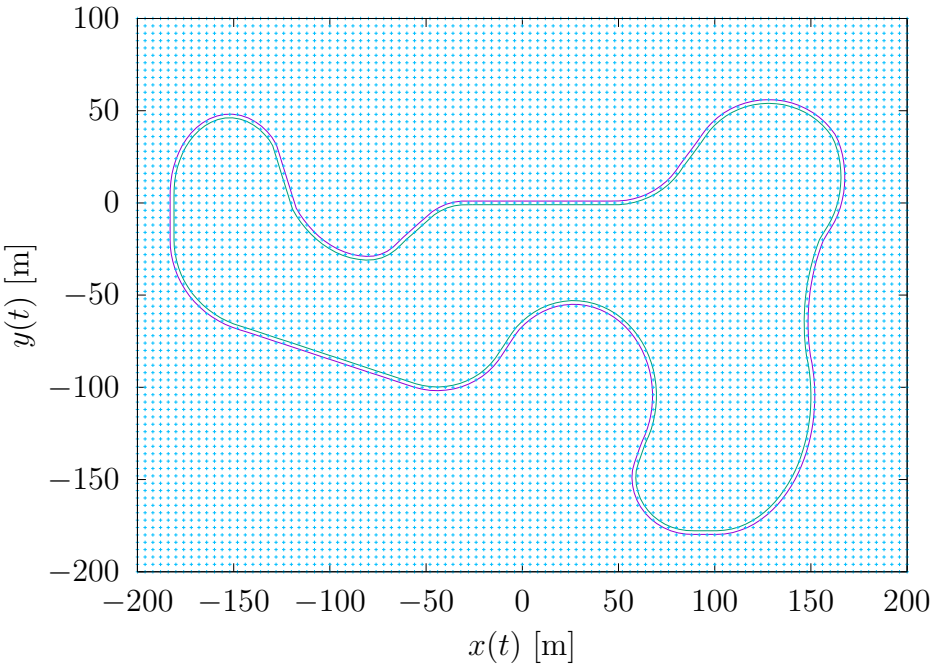


Abbildung 3.10: Kartesisches Gitter über einem Testkurs

nematische Fahrzeugmodell zu verwenden.

**DGL-System 3.3 (Kurven-parametrisiertes kinematisches Fahrzeugmodell)**

$$
\begin{aligned}
s'(t) &= \frac{v(t)\left(x'_m(s(t))\cos(\psi(t)) + y'_m(s(t))\sin(\psi(t))\right)}{1 + r(t)\kappa_m(s(t))} \\
r'(t) &= v(t)\left(y'_m(s(t))\cos(\psi(t)) - x'_m(s(t))\sin(\psi(t))\right) \\
\psi'(t) &= \frac{v(t)}{\ell}\tan(\delta(t))
\end{aligned}
$$

Die Äquivalenz des Differentialgleichungssystems 3.3 zu Formel (3.17) und (3.18) folgt aus $v_x = v(t)\cos(\psi(t))$ und $v_y = v(t)\sin(\psi(t))$, passend zum kinematischen Fahrzeugmodell. Da das kinematische Fahrzeugmodell keine Beschränkung der Geschwindigkeit beinhaltet, muss eine Beschränkung der Zentripetalbeschleunigung durch eine Nebenbedingung erfolgen. Im Vergleich zu zusätzlichen Differentialgleichungen sind zusätzliche Nebenbedingungen mit dem Ansatz der dynamischen Programmierung günstig, im Bezug auf die

Rechenzeit. In [58] ist für die maximale Zentripetalbeschleunigung bei einer konstanten Kreisfahrt die vereinfachte Formel

$$\left(\frac{v^2}{\rho}\right)_{\max} = \mu \cdot g$$

angegeben. Diese Abschätzung entspricht gerade der oberen Schranke des Kammschen Kreis, im Fall einer konstanten Längsbeschleunigung. Auf die Längsbeschleunigung müssen wir verzichten, da das Hinzufügen eine zusätzliche Differentialgleichung und dem entsprechend auch einen weit höheren Rechen- und Speicheraufwand bedeuten würde. Die Variable $\mu$ entspricht dem Reibungsbeiwert der Straße, $g$ der Erdbeschleunigung und $\rho$ dem Kurvenradius. Setzen wir statt einem konstanten Kurvenradius die Krümmung $\kappa = \frac{1}{\rho}$ ein, ergibt sich für die maximale Geschwindigkeit die Formel

$$v_{\max} = \sqrt{\frac{\mu \cdot g}{|\kappa|}}. \tag{3.20}$$

Den Reibungsbeiwert nehmen wir mit dem Wert $\mu = 1$ an, was einer trockenen Fahrbahn entspricht. Die Variable $g$ entspricht der Gravitationskonstante $g = 9{,}81\frac{\mathrm{m}}{\mathrm{s}^2}$ und $\kappa$ der Krümmung der Bahn. Die Krümmung des kinematischen Fahrzeugmodells ist durch

$$\kappa = \frac{\partial \psi}{\partial \hat{s}} = \frac{\frac{v}{\ell}\tan(\delta)}{v} = \frac{\tan(\delta)}{\ell} \tag{3.21}$$

gegeben, wobei $\partial \hat{s}$ der partiellen Ableitung nach der tatsächlich gefahrenen Bahnlänge $\hat{s}$ entspricht.
Wir definieren das Optimalsteuerungsproblem:

**Problem 3.1 (Moving Horizon Optimalsteuerungsproblem nach [47])**

Minimiere bezüglich der Steuergrößen $\delta(t)$ und $v(t)$ die Zielfunktion

$$-s(T)$$

auf dem Zeitintervall $t \in [0, T]$, unter Erfüllung des Differentialgleichungssystems

$$\begin{aligned}
s'(t) &= \frac{v(t)\left(x'_m(s(t))\cos(\psi(t)) + y'_m(s(t))\sin(\psi(t))\right)}{1 + r(t)\kappa_m(s(t))} \\
r'(t) &= v(t)\left(y'_m(s(t))\cos(\psi(t)) - x'_m(s(t))\sin(\psi(t))\right) \\
\psi'(t) &= \frac{v(t)}{\ell}\tan(\delta(t))
\end{aligned}$$

mit $s(t) \in [0, L]$, $r(t) \in [-B(s(t))/2, B(s(t))/2]$ und den Nebenbedingungen

$$v(t) \leq \sqrt{\frac{\mu \cdot g \cdot \ell}{|\tan(\delta(t))|}}$$
$$\delta_{\min} \leq \delta(t) \leq \delta_{\max}$$
$$v_{\min} \leq v(t) \leq v_{\max}$$

mit einem festen Anfangspunkt $(s(0), r(0), \psi(0)) = (s_0, r_0, \psi_0)$ und $t \in [0, T]$.

Problem 3.1 entspricht einem Optimalsteuerungsproblem aus [47], einer im Rahmen dieser Dissertation veröffentlichen Publikation, welches durch dynamische Programmierung auf einem Zustandsgitter für einen Moving Horizon gelöst werden kann. Um das Problem 3.1 zu diskretisieren, müssen wir Zustände, Zeitpunkte und Steuerungen diskretisieren. Die Steuerungen $u_1(t_k) = \delta(t_k)$ und $u_2(t_k) = v(t_k)$ betrachten wir in der diskreten Menge

$$U(t_k) := \left\{ u(t_k) = \begin{pmatrix} \delta(t_k) \\ v(t_k) \end{pmatrix} \middle| \begin{array}{l} \delta(t_k) \in \{\delta_0, \delta_1, \ldots, \delta_{n_d}\} \\ v(t_k) \in \{v_0, v_1, \ldots, v_{n_v}\} \end{array} \right\}. \tag{3.22}$$

In der dynamischen Programmierung wollen wir die Ziel- bzw. Wertefunktion in jedem diskreten Zustand und zu jedem diskreten Zeitpunkt auswerten. Wir führen somit ein Zeit- und Zustandsgitter

$$\mathbb{G} = \left\{ g = \begin{pmatrix} t_k \\ s_i \\ r_j \\ \psi_l \end{pmatrix} \middle| k = 0, 1, \ldots, n_t; i = 0, 1, \ldots, n_s; j = 0, 1, \ldots, n_r; l = 0, 1, \ldots, n_\psi \right\} \tag{3.23}$$

ein.
Wir definieren das Optimalsteuerungsproblem:

**Problem 3.2 (Dynamische Programmierung mit Moving Horizon Optimalsteuerungsproblem nach [47])**

Minimiere die Zielfunktion

$$F(x, u) := -s(t_{n_t})$$

auf dem Gitter $\mathbb{G}$, mit einem festen Anfangspunkt $(s(0), r(0), \psi(0)) = (s_0, r_0, \psi_0)$. Die Zustände $x(t_k) = (s(t_k), r(t_k), \psi(t_k))^\top$ sind durch die Nebenbedingungen

$$\begin{aligned}
x(t_{k+1}) &= x(t_k) + h\Phi(t_k, x(t_k), u(t_k)) && k = 0, \ldots, n_t - 1 \\
s(t_k) &\in [0, L] && k = 0, \ldots, n_t \\
r(t_k) &\in [-B(s(t_k))/2, B(s(t_k))/2] && k = 0, \ldots, n_t \\
u(t_k) &\in U(t_k) && k = 0, \ldots, n_t \\
v(t_k) &\leq \sqrt{\frac{\mu \cdot g \cdot \ell}{|\tan(\delta(t_k))|}} && k = 0, \ldots, n_t
\end{aligned}$$

beschränkt.

Problem 3.2 entspricht einer diskretisierten Form von Problem 3.1. Die diskrete Menge $U(t_k)$ der möglichen Steuerungen ist durch (3.22) definiert. Das Differentialgleichungssystem 3.3 wird im Einzelnen durch ein Runge-Kutta-Verfahren mit der zugehörigen Inkrementfunktion $\Phi(t_k, x(t_k), u(t_k))$ gelöst, dessen Lösung bei der Auswertung der Wertefunktion auf das Gitter interpoliert wird. Dieses Modell, das bereits in [47] gelöst wurde, wollen wir in dieser Arbeit im Bezug auf den Speicheraufwand optimieren.
Um das weitere Vorgehen zu motivieren, folgt eine Beispielrechnung für den Speicher- und Rechenaufwand. Nehmen wir an, wir wollen mit $n_t = 400$ Zeitpunkten, $n_s = 400$ Bogenlängenwerten, $n_r = 51$ Bahnabständen und $n_\psi = 300$ Winkelzuständen eine durch dynamische Programmierung berechnete Wertefunktion $W(x(t), t) : \mathbb{G} \longmapsto \mathbb{R}$ mit doppelter Genauigkeit (8 Byte) abspeichern. Es ergäbe sich ein Speicherbedarf von

$$\frac{(n_t n_s n_r n_\psi)8\mathrm{B}}{1024^3 \frac{\mathrm{B}}{\mathrm{GB}}} \approx 18{,}24\mathrm{GB}.$$

Beim Rechenaufwand müssen auch die Dimensionen der diskreten Steuerungsmengen (3.22) berücksichtigt werden. Das bedeutet für $n_d = 51$ Lenkwinkel und $n_v = 50$ Geschwindigkeiten, dass das Differentialgleichungssystem 3.3

$$n_t n_s n_r n_\psi n_d n_v \approx 6{,}24 \cdot 10^{12}$$

mal gelöst werden muss. Der enorme Speicher- und Rechenaufwand wird die Frage auf, ob das Problem nicht um eine Dimension reduziert werden kann.
Wir sollten uns zunächst fragen, welche Art von Lösung wir suchen und welche Eigenschaften diese Lösung besitzen soll. Wir gehen davon aus, dass die zurückgelegte Bogenlänge

$s(t)$, bei einer optimalen Lösung, streng monoton mit der Zeit $t$ anwächst, und entfernen deshalb die Zeit aus dem Gitter

$$\mathbb{G} = \left\{ x = \begin{pmatrix} s_i \\ r_j \\ \psi_l \end{pmatrix} \middle| i = 0, 1, \ldots, n_s; j = 0, 1, \ldots, n_r; l = 0, 1, \ldots, n_\psi \right\}. \tag{3.24}$$

Da wir für die Integration des Differentialgleichungssystems 3.3 diskrete Zeitinkremente benötigen führen wir diese als diskrete Steuerung ein und erhalten die neue Menge von Steuerungen

$$U(s_i) := \left\{ u(s_i) = \begin{pmatrix} \delta(s_i) \\ v(s_i) \\ \Delta t(s_i) \end{pmatrix} \middle| \begin{matrix} \delta(s_i) \in \{\delta_0, \delta_1, \ldots, \delta_{n_d}\} \\ v(s_i) \in \{v_0, v_1, \ldots, v_{n_v}\} \\ \Delta t(s_i) \in \{\Delta t_0, \ldots, \Delta t_{n_v}\} \end{matrix} \right\}. \tag{3.25}$$

Wesentlich, bei der Optimalsteuerung ist die Wahl der Ziel- bzw. Wertefunktion. Im Gegensatz zum Moving Horizon Ansatz aus [47], [39] und [42] haben wir, durch die eingeführten Zeitinkremente, die Möglichkeit, die benötigte Zeit

$$T \approx \sum_{i=0}^{n_s} \Delta t(s_i) \tag{3.26}$$

zum Erreichen eines Endpunkts zu minimieren. Um zu verhindern, dass das Zeitinkrement $\Delta t \to 0$ geht, führen wir eine zusätzliche Nebenbedingung

$$s_{i+1} - s_i \leq \Delta t(s_i) \cdot s_i'(x(s_i), u(s_i))$$

ein. Diese soll sicherstellen, dass die nächste Bogenlängenebene erreicht wird. In der letzten Bogenlängenebene des Gitters können wir die Wertefunktion auf $W(s_{n_s}, x_{r,\psi}) = 0$ setzen.

**Problem 3.3 (Dynamische Programmierung mit Moving Horizon Optimalsteuerungsproblem mit reduzierter Gitterdimension)**

Minimiere die Zielfunktion

$$F(x, u) := \sum_{i=0}^{n_s} \Delta t(s_i)$$

auf dem Gitter $\mathbb{G}$, mit einem festen Anfangspunkt $(s(0), r(0), \psi(0)) = (s_0, r_0, \psi_0)$. Die

Zustände $x(s_i) = (s_i, r(s_i), \psi(s_i))^\top$ sind durch die Nebenbedingungen

$$
\begin{aligned}
x_{r,\psi}(s_{i+1}) &= x_{r,\psi}(s_i) + h\Phi_{r,\psi}(x(s_i), u(s_i)) && i = 0, \ldots, n_s - 1 \\
r(s_i) &\in [-B(s_i)/2, B(s_i)/2] && i = 0, \ldots, n_s \\
u(s_i) &\in U(s_i) && i = 0, \ldots, n_s \\
v(s_i) &\leq \sqrt{\frac{\mu \cdot g \cdot \ell}{|\tan(\delta(t_j))|}} && i = 0, \ldots, n_s \\
s_i + h\Phi_s(x(s_i), u(s_i)) - s_i &\leq \Delta t(s_i) \cdot s_i'(x(s_i), u(s_i)) && i = 0, \ldots, n_s
\end{aligned}
$$

beschränkt.

Die diskrete Menge $U$ der möglichen Steuerungen ist durch (3.25) definiert. Die Auswertung der Differentialgleichung beruht darauf, dass zur Integration des Differentialgleichungssystems 3.3 keine expliziten Zeiten verwendet werden, sondern lediglich der Integrationszeitraum benötigt wird, welcher in Form von $\Delta t$ ein Teil der Steuerungen $u(s)$ ist. Um die Wertefunktion der letzten Bogenlängenebene auswerten zu können, müssen wir wieder zwischen den Gitterpunkten interpolieren.

**Bemerkung 3.4.** *Im Vergleich zu Problem 3.2, reduziert Problem 3.3 den Speicheraufwand um eine Dimension. Durch die zusätzliche Steuerung, bleibt die Dimension des Rechenaufwands gleich.*

### 3.4.3 Ergebnisse

Für das Problem 3.3 wird die Methode der dynamischen Programmierung nach Bellman aus Abschnitt 2.4 angewandt. Dabei benötigt die Rückwärtsrechnung trotz einer Parallelisierung mit 40 Threads für den gesamten Kurs in Most immer noch mehrere Stunden. Ohne Parallelisierung können Rechenzeiten von über einer Woche auftreten. Die Vorwärtsrechnung kann fast ohne Rechenaufwand durchgeführt werden. In Abbildung 3.11 ist die berechnete Trajektorie für etwas mehr als einer Runde, anhand des Testkurses in Most dargestellt. Beim Betrachten der Trajektorie stellen wir ein schlechtes Krümmungsverhalten der Rennlinie fest. Im Vergleich zu von Rennfahrern gefahrenen Trajektorien werden hier Kurven zu eng eingelenkt, was zu hohen Maximalkrümmungen führt. Der Grund für das schlechte Krümmungsverhalten lässt sich anhand des Geschwindigkeitsprofils in Abbildung 3.12 erkennen. Hier sind die Geschwindigkeit und der Lenkwinkel über die Bogenlänge dargestellt. In wesentlichen erkennen wir, dass die Geschwindigkeit

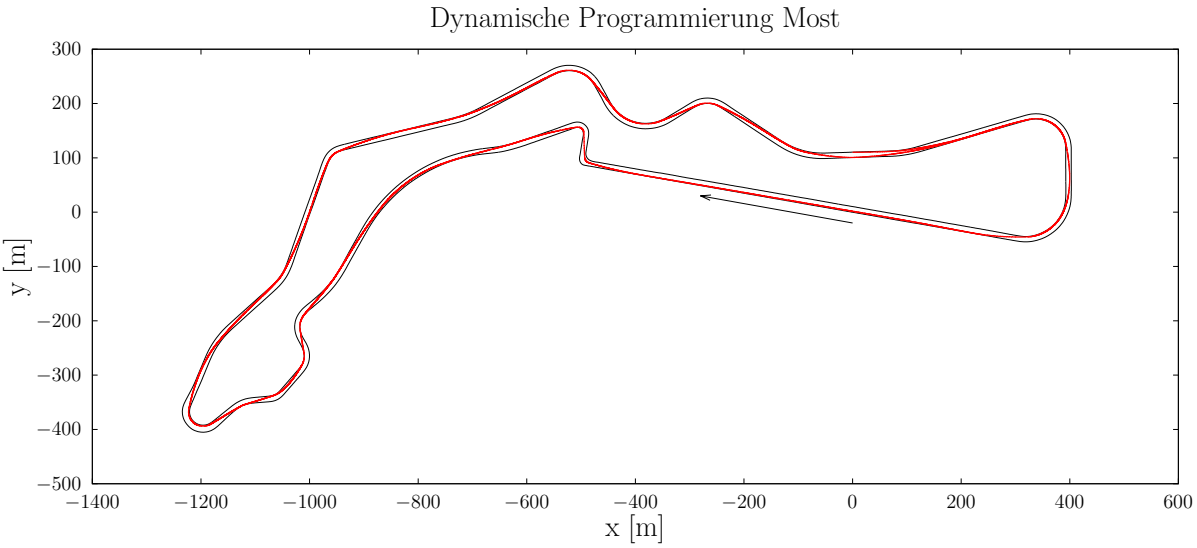


Abbildung 3.11: Mit dynamischer Programmierung berechnete Trajektorie für Most, mit Begrenzung der Kurvengeschwindigkeit

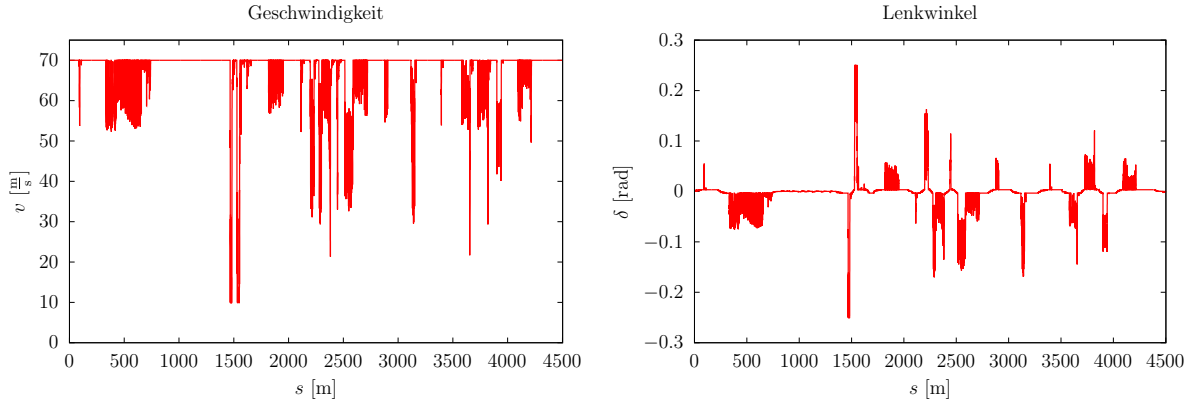


Abbildung 3.12: Geschwindigkeitsprofil und Lenkwinkel für Most mit dynamischer Programmierung

bevorzugt der Maximalgeschwindigkeit folgt und nur an einigen Stellen aufgrund der Beschränkung (3.20) einbricht. Auch der Lenkwinkel aus Abbildung 3.12 wird sprunghaft gesteuert. Diese Verläufe sind der Tatsache geschuldet, dass im Differentialgleichungssystem 3.3 der Lenkwinkel und die Geschwindigkeit direkt gesteuert werden können, weshalb die Lenkwinkeländerung, Brems- und Antriebsbeschleunigung nicht beschränkt sind. So-

mit stellt die in Abbildung 3.11 dargestellte Trajektorie eine Rennlinie für ein Fahrzeug mit unendlichem Beschleunigungspotential dar. Hinzu kommt, dass Formel (3.20) die Längsbeschleunigungen vernachlässigt, was zu einer Verletzung des Kammschen Kreises führen kann.

Neben einer schlechten Qualität der berechneten Trajektorie lässt sich als weiterer Nachteil anführen, dass die langen Rechenzeiten der Rückwärtsrechnung ungeeignet für die Praxistauglichkeit des Ansatzes sind. Die Rückwärtsrechnung muss nicht nur für jeden Testkurs durchgeführt werden, sondern auch wenn beispielsweise Hindernisse auftreten, die noch nicht berücksichtigt sind. Dementsprechend ist dieser Ansatz für die Onlineberechnung von Rennlinien, welche mit der Vorwärtsrechnung einer bereits berechneten Wertefunktion möglich wäre, in der Praxis untauglich.

**Bemerkung 3.5.** *Aufgrund einer zu schlechten Bewertung der Trajektorie im Bezug auf Geschwindigkeit und Krümmungsverhalten, wurden für die dynamische Programmierung keine offline- oder online-Tests am realen Fahrzeug durchgeführt. In [47] fand lediglich eine offline-Nachregelung mit einem Modellfahrzeug im Maßstab 1:8 statt. Diese Tests erfolgten allerdings mit einer notwendigen Skalierung der berechneten Trajektorie mit dem Faktor 1:70, um diese auf eine Laborfläche anzupassen.*

## 3.5 Optimalsteuerung mit dem Einspurmodell

### 3.5.1 Grundlage der offline-Optimierung mit dem kartesischen Einspurmodell

Als Grundlage für die Berechnung einer optimalen Rennlinie verwenden wir im Rahmen dieser Arbeit ein Vorläuferprojekt aus [42]. Das in [42] definierte Fahrzeugmodell entspricht, in weiten Teilen, dem in Abschnitt 3.1.2 eingeführten Modell. Erweitert wurde dieses Modell aus Abschnitt 3.1.2 um eine Konfigurationsmöglichkeit für Front- und Heckantrieb. Das zugehörige Optimalsteuerungsproblem der einzelnen MPC Schritte ist wie folgt definiert.

**OSP 3.1 (Einspurmodell Moving Horizon Optimalsteuerungsproblem nach [42])**

Minimiere die Zielfunktion

$$-c_2\left(s_r(T+\Delta T)-s_r(T)+s_l(T+\Delta T)-s_l(T)\right)+c_1\int_T^{T+\Delta T} w_\delta(t)^2 dt$$

unter Erfüllung des Differentialgleichungssystems

$$\begin{aligned}
x'(t) &= v_x(t)\\
y'(t) &= v_y(t)\\
\psi'(t) &= w_\psi(t)\\
v_x'(t) &= \frac{1}{m}\left[F_x(t)\cdot\cos\psi(t)-F_y(t)\cdot\sin\psi(t)\right]\\
v_y'(t) &= \frac{1}{m}\left[F_x(t)\cdot\sin\psi(t)+F_y(t)\cdot\cos\psi(t)\right]\\
w_\psi'(t) &= \frac{1}{I_{zz}}\left[F_{sv}(t)\cdot l_v\cdot\cos\delta(t)-F_{sh}(t)\cdot l_h-F_{Ay}(t)\cdot e_{sp}+F_{lv}(t)\cdot l_v\cdot\sin\delta(t)\right]\\
\delta'(t) &= w_\delta(t)\\
s_l'(t) &= \frac{x'(t)x_l'+y'(t)y_l'}{(x_l')^2+(y_l')^2-(x(t)-x_l)x_l''-(y(t)-y_l)y_l''}\\
s_r'(t) &= \frac{x'(t)x_r'+y'(t)y_r'}{(x_r')^2+(y_r')^2-(x(t)-x_r)x_r''-(y(t)-y_r)y_r''},
\end{aligned}$$

mit den Nebenbedingungen

$$\begin{aligned}
\left[\begin{pmatrix}x(t)-x_l\\ y(t)-y_l\end{pmatrix}+\frac{b}{2}\frac{n_l}{\|n_l\|}\right]\frac{n_l}{\|n_l\|} &\le 0\\
\left[\begin{pmatrix}x(t)-x_r\\ y(t)-y_r\end{pmatrix}+\frac{b}{2}\frac{n_r}{\|n_r\|}\right]\frac{n_r}{\|n_r\|} &\le 0\\
\frac{\sqrt{F_{lv}^2(t)+F_{sv}^2(t)}}{F_{zv}(t)}-\mu &\le 0\\
\frac{\sqrt{F_{lh}^2(t)+F_{sh}^2(t)}}{F_{zh(t)}}-\mu &\le 0\\
\sqrt{v_x^2(t)+v_y^2(t)} &\ge v_{\min}
\end{aligned}$$

und entsprechenden Moving Horizon Anfangsbedingungen.

Die beiden zusätzlichen Differentialgleichungen für $s_l'(t)$ und $s_r'(t)$ entsprechen je dem Fortschritt im Bezug zur linken beziehungsweise rechten Fahrbahngrenze. Die Fahrbahn-

grenzen sind als Kurven bzw. kubische Splines hinterlegt, deren Normalenvektoren $n_l = (-y_l', x_l')^\top$ und $n_r = (y_l', -x_l')^\top$ von der Fahrbahn weg zeigen. Die beiden ersten Nebenbedingungen bilden Projektionen auf die Fahrbahngrenzen ab, die unsere zulässigen Positionen auf die Fahrbahn beschränken. Die dritte und vierte Nebenbedingung entsprechen der in (3.6) und (3.7) definierten Einhaltung des Kammschen Kreises für die dynamischen Radlasten. Außerdem wird noch eine zulässige Mindestgeschwindigkeit $v_{\min} > 0$ vorgegeben, welche für das Einspurmodell notwendig ist. Die Referenzpositionen auf den Randkurven $x_l$, $x_r$, $y_l$ und $y_r$ ergeben sich aus einer Auswertung der kubischen Splines der Randkurven bezüglich $s_l(t)$ und $s_r(t)$. Selbes gilt für die Ableitungen $x_l'$, $x_r'$, $y_l'$, $y_r'$, $x_l''$, $x_r''$, $y_l''$ und $y_r''$, welche ebenfalls durch eine Auswertung der kubischen Splines berechnet werden können. Als Zielfunktion wird eine Kombination aus Streckenfortschritt

$$-c_2 \left(s_r(T + \Delta T) - s_r(T) + s_l(T + \Delta T) - s_l(T)\right)$$

und Steuerungsaufwand

$$c_1 \int_T^{T+\Delta T} w_\delta(t)^2 dt$$

verwendet.

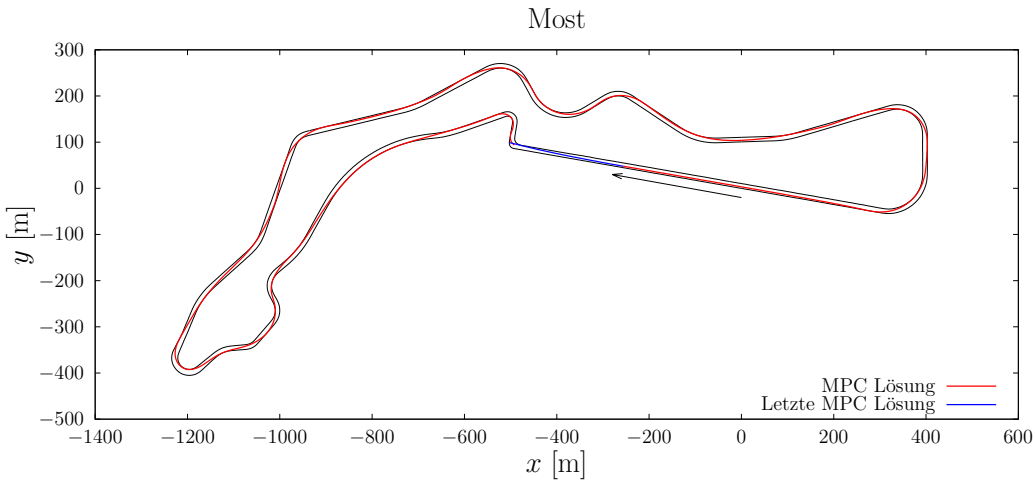


Abbildung 3.13: Lösung für den Testkurs Most bei einem Vorausschau-Horizont von 4 s

Zunächst betrachten wir das in OSP 3.1 bzw. [42] definierte Verfahren für den Testkurs in Most. In Abbildung 3.13 ist eine optimierte Rennlinie dargestellt, welche einer aus Segmenten zusammengesetzten Moving Horizon Trajektorie entspricht. Die tatsächliche

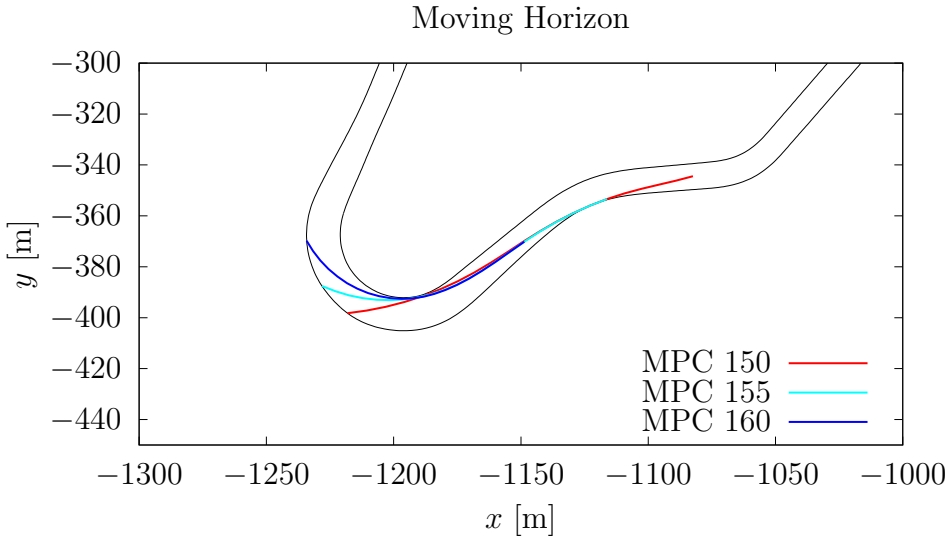


Abbildung 3.14: Moving Horizon für drei MPC Lösungen mit einem Abstand von fünf Zwischenlösungen

Länge der Vorausschau hängt in den einzelnen MPC Lösungen insbesondere von der Geschwindigkeit ab. So zeigt Abbildung 3.14 eine reduzierte MPC Trajektorienlänge beim Durchfahren einer stark gekrümmten Kurvenpassage in Most. Ein weiterer Effekt ist die zu beobachtende Fächerung der MPC Trajektorien. Diese wird maßgeblich durch den Anteil des Fortschritts, im Bezug auf die Randkurven, in der Zielfunktion erzeugt.

### 3.5.2 Bewertung des kartesischen Einspurmodells

Zur Bewertung der Rennlinie wurden Testfahrten mit einer offline berechneten Trajektorie auf dem Testkurs in Most durchgeführt. Eine Schwierigkeit bereitet hierbei vor allem, dass sich der Vorausschau-Horizont nicht beliebig vergrößern lässt, ohne die Robustheit des Verfahrens zu verringern. Die in Abbildung 3.13 verwendeten 4 Sekunden reichen nicht aus, um bei hoher Geschwindigkeit rechtzeitig für die Kurve nach der Start-Ziel Geraden bremsen zu können. In der vergrößerten Abbildung 3.15 ist die letzte berechnete MPC Trajektorie blau eingezeichnet. Da der Optimierer im nächsten Schritt versuchen müsste, in die Kurve einzufahren und dementsprechend das Fahrzeug hinreichend stark abbremsen muss, schlägt die Optimierung fehl. Mit der verfügbaren Vorausschau von 4 Sekunden bleibt nicht genügend Zeit, um das Fahrzeug stark genug zu verzögern. Eine weitere

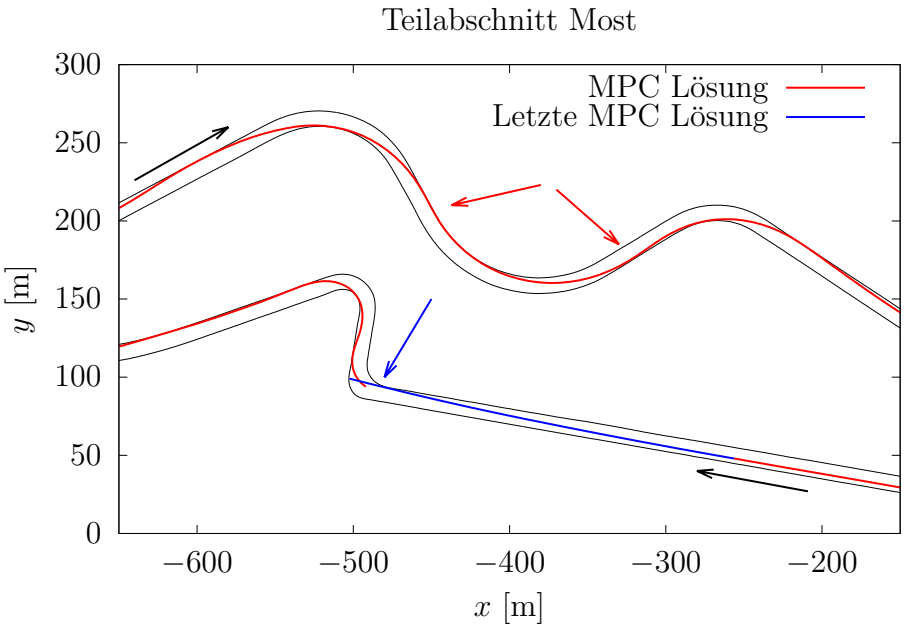


Abbildung 3.15: Vergrößerter Kursausschnitt von Abbildung 3.13.

Erhöhung der Vorausschau, bei Verwendung des kartesischen Einspurmodells, senkt die Robustheit des Optimierers.

Weitere Nachteile der kurzen Vorausschau zeigen sich bei Testfahrten auf der, mit dem kartesischen Einspurmodell, offline berechneten Rennlinie. Mit einem Blick auf die MPC Trajektorien aus Abbildung 3.14 wird deutlich, dass die Vorausschau nur kurze aufeinanderfolgende Kurvenabschnitte berücksichtigt. Ein guter Rennfahrer passt seine Trajektorie hingegen meist auf ganze Kurvenpassagen an. Ein Beispiel hierfür ist in Abbildung 3.15 mit roten Pfeilen markiert. Beim Durchfahren der S-förmigen Kurvenpassage wird die erste Kurve von der Außenbahn angefahren und auch auf der Außenbahn ausgefahren, was bei einer einzelnen Kurve korrekt wäre. Allerdings hat dies zur Folge, dass die darauffolgenden Kurven, die in Abbildung 3.15 markiert sind, je von der Innenseite der Kurve angefahren werden. Daraus resultiert, dass das Fahrzeug die letzte Kurve mit einer deutlich geringeren Geschwindigkeit verlässt, wie bei einer vergleichbaren Rennlinie eines Testfahrers. Besser wäre es zu versuchen, die beiden Kurven von der Außenseite anzufahren, um eine höhere Kurvenausgangsgeschwindigkeit zu erzielen. Auch diese Eigenschaften der berechneten Rennlinie sind auf die kurze Vorausschau zurückzuführen.

### 3.5.3 Offline-Bahnplanung mit einem kurven-parametrisiertem Einspurmodell

#### Anpassungen des Optimalsteuerungsproblems

Das Ziel der Verwendung eines kurven-parametrisierten Einspurmodells ist es, die Robustheit des Optimierers im Hinblick auf höhere Vorausschau-Horizonte zu erhöhen. Dementsprechend definieren wir ein Moving Horizon Optimalsteuerungsproblem mit einem kurven-parametrisierten Einspurmodell.

**OSP 3.2 (Kurven-parametrisiertes Einspurmodell Moving Horizon Optimalsteuerungsproblem)**

Minimiere die Zielfunktion

$$\int_{T}^{T+\Delta T} \left(c_1 \cdot w_\delta^2(t) + c_2 \cdot \alpha_v^2(t)\right) \mathrm{d}t - c_3 \left(s\left(T+\Delta T\right) - s\left(T\right)\right)$$

unter Erfüllung des Differentialgleichungssystems

$$\begin{aligned}
s'(t) &= \frac{x'_m(s(t))v_x(t) + y'_m(s(t))v_y(t)}{1 + r(t)\kappa_m(s(t))} \\
r'(t) &= v_x(t)y'_m(s(t)) - v_y(t)x'_m(s(t)) \\
\psi'(t) &= w_\psi(t) \\
v'_x(t) &= \frac{1}{m}\left[F_x(t) \cdot \cos\psi(t) - F_y(t) \cdot \sin\psi(t)\right] \\
v'_y(t) &= \frac{1}{m}\left[F_x(t) \cdot \sin\psi(t) + F_y(t) \cdot \cos\psi(t)\right] \\
w'_\psi(t) &= \frac{1}{I_{zz}}\left[F_{sv}(t) \cdot l_v \cdot \cos\delta(t) - F_{sh}(t) \cdot l_h - F_{Ay}(t) \cdot e_{sp} + F_{lv}(t) \cdot l_v \cdot \sin\delta(t)\right] \\
\delta'(t) &= w_\delta(t),
\end{aligned}$$

mit den Nebenbedingungen

$$\begin{aligned}
r^2(t) - \frac{b^2(s(t))}{4} &\leq 0 \\
\frac{\sqrt{F_{lv}^2(t) + F_{sv}^2(t)}}{F_{zv}(t)} - \mu &\leq 0 \\
\frac{\sqrt{F_{lh}^2(t) + F_{sh}^2(t)}}{F_{zh}(t)} - \mu &\leq 0 \\
\sqrt{v_x^2(t) + v_y^2(t)} &\geq v_{\min}
\end{aligned}$$

und entsprechenden Moving Horizon Anfangsbedingungen.

Im Vergleich zum Optimalsteuerungsproblem aus OSP 3.1 verändern sich sowohl die Zielfunktion, das Differentialgleichungssystem, als auch die Nebenbedingungen.
Zunächst fällt auf, dass die beiden zusätzlichen Differentialgleichungen

$$s_l'(t) = \frac{x'(t)x_l' + y'(t)y_l'}{(x_l')^2 + (y_l')^2 - (x(t) - x_l)x_l'' - (y(t) - y_l)y_l''}$$
$$s_r'(t) = \frac{x'(t)x_r' + y'(t)y_r'}{(x_r')^2 + (y_r')^2 - (x(t) - x_r)x_r'' - (y(t) - y_r)y_r''}$$

aus OSP 3.1 nicht mehr benötigt werden. Da im kurven-parametrisierten Einspurmodell bereits eine Differentialgleichung für den Streckenfortschritt $s(t)$ im Bezug auf die Mittellinie existiert, benötigen wir nur noch einen kubischen Spline für die Streckenbreite, um die Straßenränder zu definieren. Der kubische Spline für die Streckenbreite besitzt eine Auswertung für $b(s(t))$, die in der ersten Nebenbedingung

$$r^2(t) - \frac{b^2(s(t))}{4} \leq 0$$

verwendet wird. Diese Nebenbedingung ersetzt die beiden komplexeren Nebenbedingungen

$$\left[\begin{pmatrix} x(t) - x_l \\ y(t) - y_l \end{pmatrix} + \frac{b}{2}\frac{n_l}{\|n_l\|}\right]\frac{n_l}{\|n_l\|} \leq 0,$$
$$\left[\begin{pmatrix} x(t) - x_r \\ y(t) - y_r \end{pmatrix} + \frac{b}{2}\frac{n_r}{\|n_r\|}\right]\frac{n_r}{\|n_r\|} \leq 0.$$

Diese Änderungen verringern die Komplexität des Optimalsteuerungsproblems und sorgen dementsprechend für mehr Robustheit, beim Lösen des Problems.
Weitere Änderungen zeigen sich in der Zielfunktion

$$\int_T^{T+\Delta T} \left(c_1 \cdot w_\delta^2(t) + c_2 \cdot \alpha_v^2(t)\right) \mathrm{d}t - c_3 \left(s\left(T + \Delta T\right) - s\left(T\right)\right).$$

Für den Anteil des Streckenfortschritts verwenden wir hier lediglich die Projektion auf die Mittellinie $s(t)$. Zusätzlich zu dem Anteil des Lenkaufwands fügen wir auch einen Anteil zum Schräglaufwinkel der Vorderachse hinzu. Dies hat den Zweck, ein Untersteuern des Fahrzeugs zu vermeiden.

### Auswertungen der Testfahrten

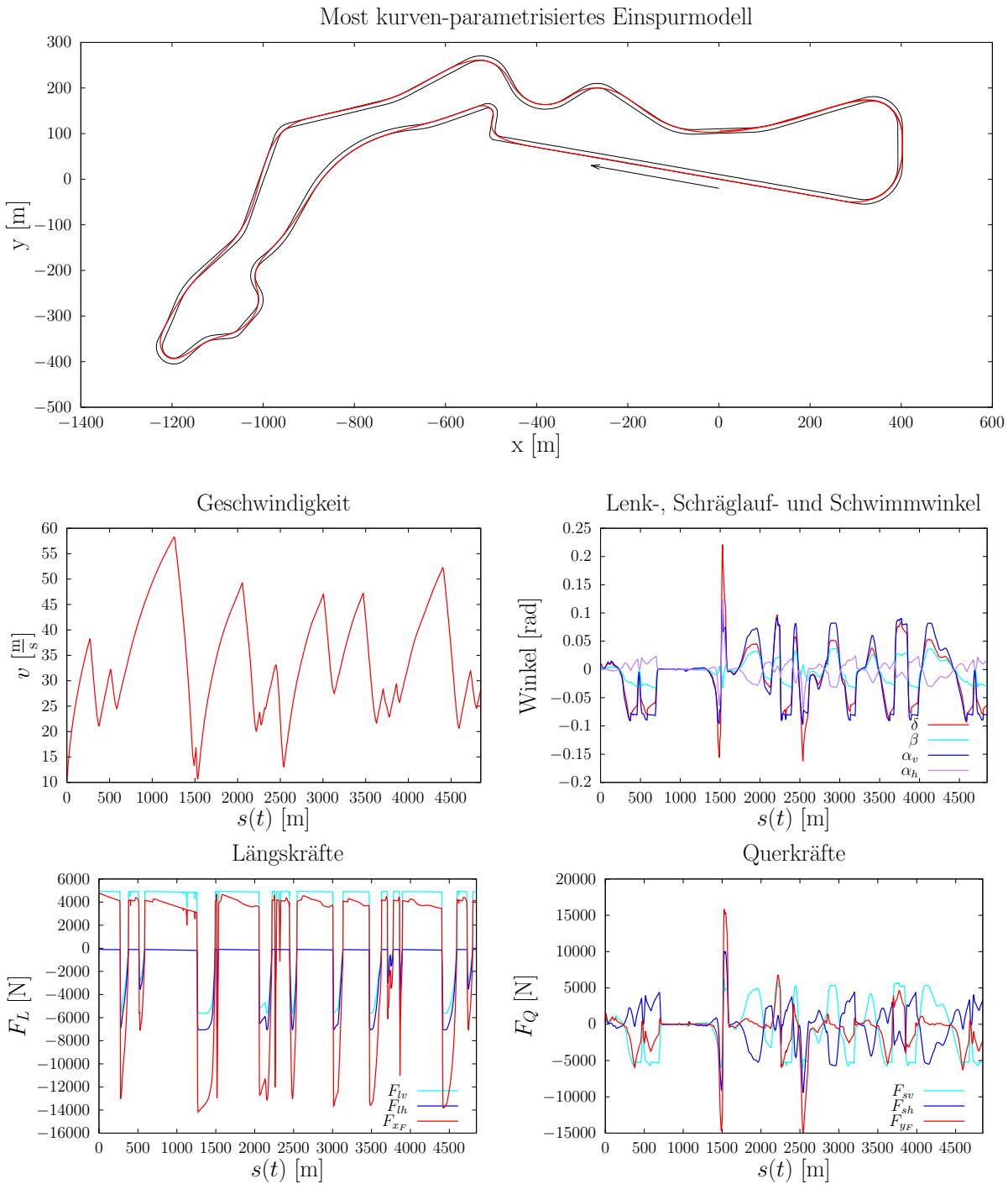


Abbildung 3.16: Kurven-parametrisiertes Einspurmodell für den Testkurs in Most mit einer Vorausschau von 7 s.

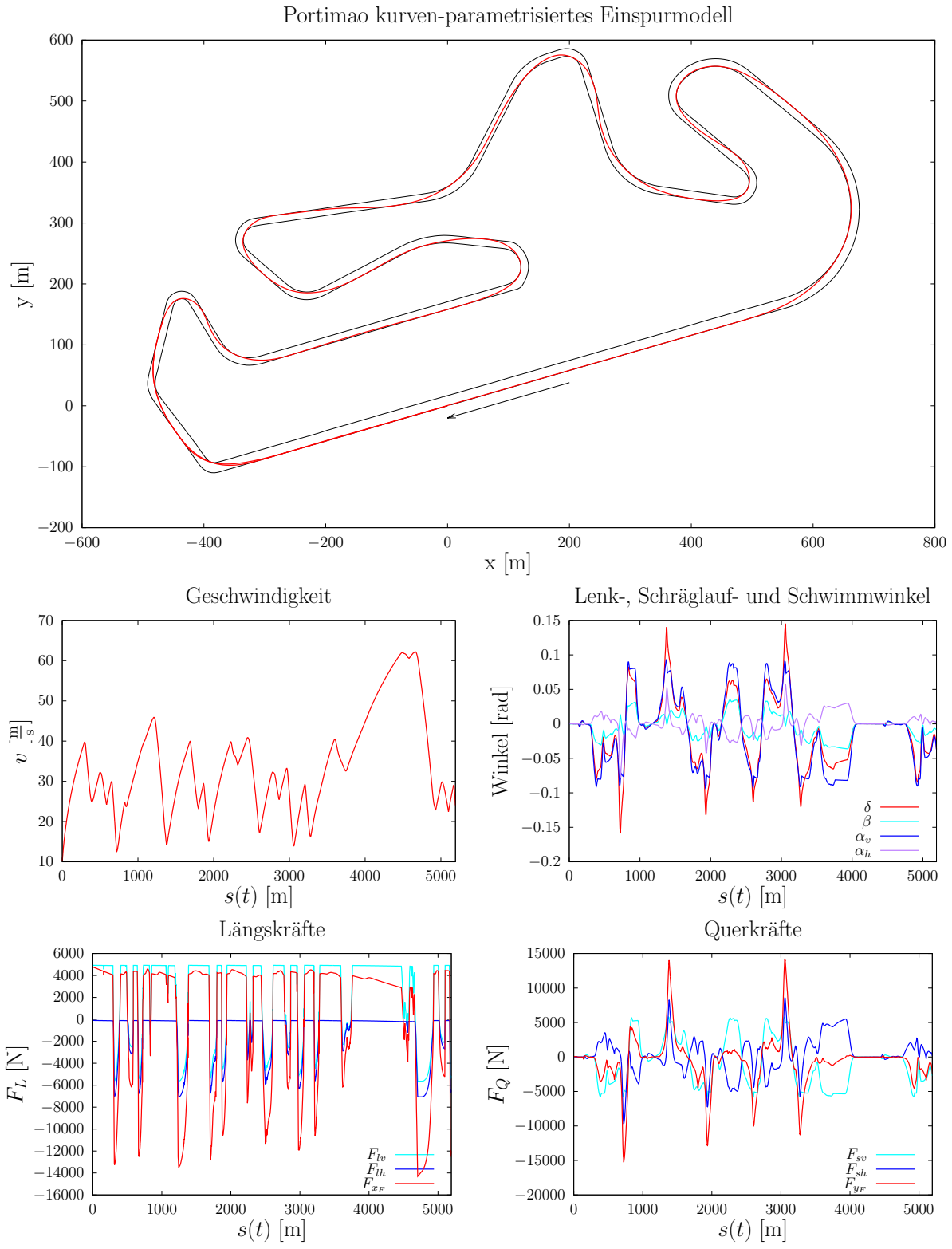


Abbildung 3.17: Kurven-parametrisiertes Einspurmodell für den Testkurs in Portimao mit einer Vorausschau von 7 s.

Für das kurven-parametrisierte Einspurmodell wurden, wie in Abbildung 3.9 schematisch dargestellt, offline-Fahrversuche durchgeführt. Bei den Fahrversuchen wurden Trajektorien mit verschiedenen Gewichten in der Zielfunktion von OSP 3.2 getestet. In den Abbildungen 3.16 und 3.17 sind jeweils die durch Testfahrer am besten bewerteten Ergebnisse, für die Testkurse in Most und Portimao dargestellt. Im Vergleich zu den Ergebnissen mit dem kartesischen Einspurmodell mit nur 4 Sekunden Vorausschau aus Abschnitt 3.5.2, konnten wir mit dem kurven-parametrisierten Einspurmodel die Vorausschau auf 7 Sekunden erhöhen. Diese der Erhöhung der Vorausschau führt, bei einem direkten Vergleich anhand des Testkurses in Most, zu einem deutlich verbesserten Verhalten der Trajektorie. Analog zu Abbildung 3.15 zeigt Abbildung 3.18 die kritischen Stellen der MPC Lösung

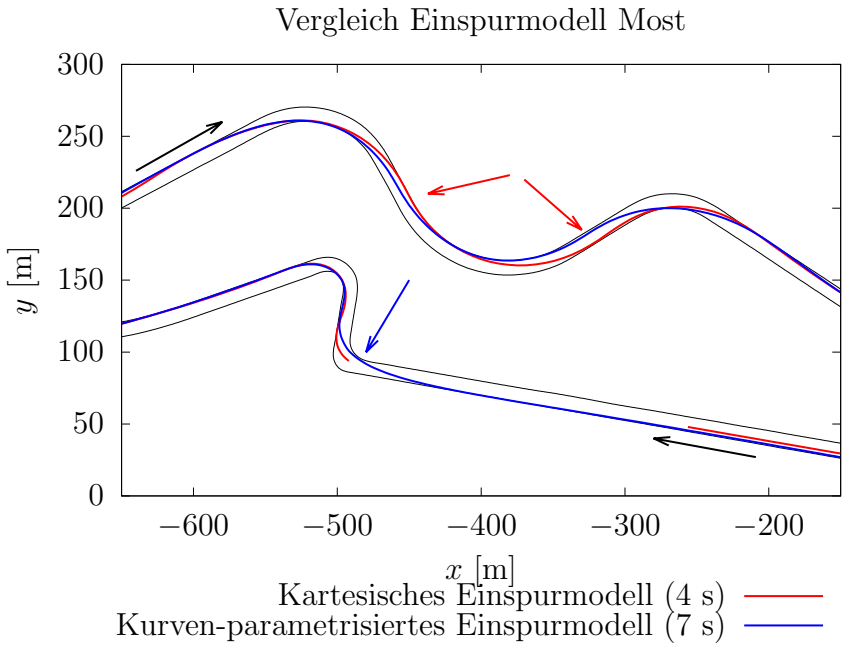


Abbildung 3.18: Vergleich von kartesischem und kurven-parametrisiertem Einspurmodell, bei einer Vorausschau von 4 bzw. 7 s.

des kartesischen Einspurmodells, bei einer Vorausschau von 4 Sekunden. Einer der Kritikpunkte aus Abschnitt 3.5.2 beschreibt das Verhalten der kartesischen Einspurmodell-Trajektorie an den mit roten Pfeilen markierten Stellen des Testkurses. An beiden Stellen fährt die Trajektorie die Kurven von der Innenseite aus an. Wie in Abschnitt 3.5.2 erläutert, liegt dies an der zu kleinen Vorausschau. Im Vergleich kann hier das kurven-parametrisierte Einspurmodell mit der höheren Vorausschau, die sich bei diesem Modell nun auch robust lösen lässt punkten. Der Kurvenabschnitt wird, wie von Testfahrern erwartet, so durchfahren, dass man die letzte Kurve von der Außenseite anfährt. Dies

zahlt sich in Form einer höheren Geschwindigkeit beim Verlassen des Kurvenabschnitts aus. Ein weiterer Unterschied, der sich aus der erhöhten Vorausschau ergibt ist, dass auch Kurve nach der Zielgeraden durchfahren werden kann. Die 7 Sekunden reichen hierbei aus, um eine Lösung zu finden, bei der das Fahrzeug noch gebremst werden kann. Allerdings fährt das Fahrzeug die darauffolgende Kurve von der Innenseite aus an. Die Vorausschau scheint dementsprechend noch nicht für eine optimales Durchfahren der Kurve auszureichen. Eine weitere Erhöhung der Vorausschau zeigte sich allerdings auch beim kurven-parametrisiertem Einspurmodell als weniger robust.
Für den Testkurs in Portimao wurde ebenfalls eine Lösung mit dem kurven-parametrisierten Einspurmodell berechnet. Die in Abbildung 3.17 dargestellte Trajektorie zeigt, wie auch die Trajektorie von Most ein weitgehend gutes Verhalten bei den offline-Testfahrten. Dennoch ist auch hier eine Stelle dabei, die von Testfahrern anders bewertet wurde. Ei-

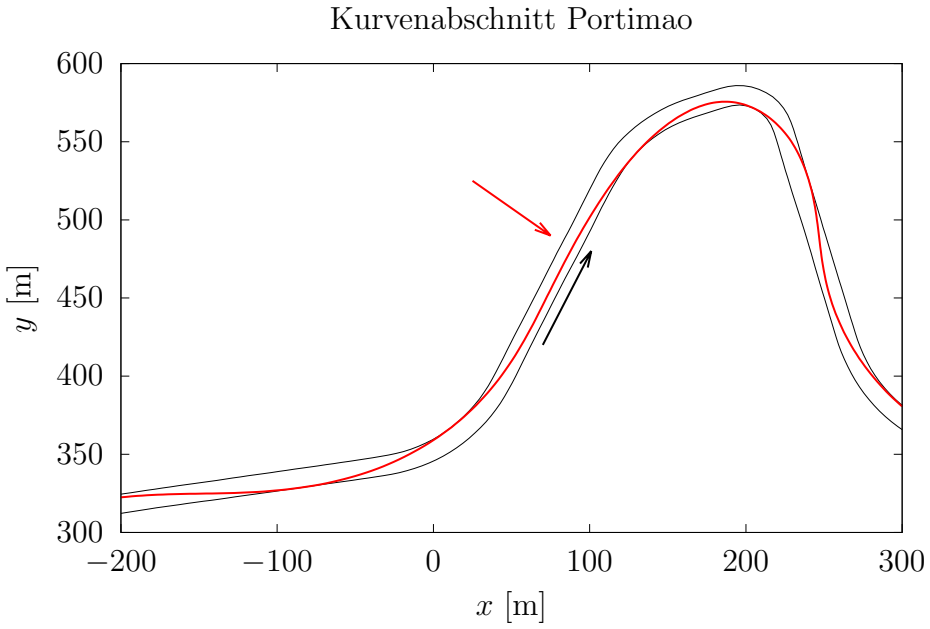


Abbildung 3.19: Vergrößerung eines Kurvenabschnitts in Portimao

ne Vergrößerung der entsprechenden Kurvenpassage ist in Abbildung 3.19 dargestellt. Hier ist wieder eine Stelle rot markiert, an der Testfahrer versuchen, die darauf folgende Kurvenpassage, von außen anzufahren.
Zusammenfassend zeigt das kurven-parametrisierte Einspurmodell ein sehr robustes Verhalten, das verglichen mir dem kartesischen Modell höhere Vorausschau-Horizonte zulässt. Allerdings müssen für höhere Vorausschau-Horizonte hohe Rechenzeiten in Kauf genom-

men werden. Deshalb eignet sich dieses Modell im Zusammenspiel mit dem verwendeten Optimierer nicht für eine optimale Onlineberechnung von Rennlinien. Im folgenden Abschnitt führen wir ein Optimalsteuerungsproblem ein, das die Lösungsqualität des kurvenparametrisierten Einspurmodells mit einem vereinfachten Fahrzeugmodell nachstellt. Dieses Fahrzeugmodell soll anschließend mit online fähigen Rechenzeiten verwendet werden können.

## 3.6 Vereinfachtes kurven-parametrisiertes Fahrzeugmodell

### 3.6.1 Motivation

Komplexe Fahrzeugmodelle wie das Einspurmodell aus dem Differentialgleichungssystem 3.2 sind in der online-Optimalsteuerung eine besondere Herausforderung. Dies liegt unter anderem am hohen Grad der Nichtlinearitäten, der sowohl in den Differentialgleichungen als auch in den Beschränkungen auftritt. Dies wirkt sich insbesondere auf die Rechenzeiten beim Lösen der Optimalsteuerungsprobleme aus. Unter Umständen können auch keine Lösungen gefunden werden, falls das numerische Verfahren scheitert. Zusätzlich sind für komplexe Fahrzeugmodelle auch viele Fahrzeugparameter zu bestimmen. Dies kann zum Einen zu Messfehlern führen und zum Anderen hängen die Fahrzeugparameter auch von variablen Faktoren, wie Wetter oder Reifenalter ab. Ein weiterer Punkt ist, dass in dieser Arbeit im Fahrzeug ein Bahnfolgeregler verwendet wird, der zur Regelung nur die Trajektoriendaten der Position, des Gierwinkels, der Krümmung, der Geschwindigkeit und der Längsbeschleunigung verwendet. Interne Fahrzeuggrößen wie Reifenkräfte, Schwimmwinkel, etc. werden von der Bahnplanung nicht benötigt. Deshalb führen wir in diesem Abschnitt ein vereinfachtes Fahrzeugmodell ein, welches sich für den online fähigen Einsatz eignet.

### 3.6.2 Modell

Als Grundlage für unser vereinfachtes kurven-parametrisiertes Fahrzeugmodell verwenden wir das kinematisches Fahrzeugmodell aus dem Differentialgleichungssystem 3.1, welches wir mit der Kurvenparametrisierung aus (3.17) und (3.18) umformen.

**DGL-System 3.4 (Kurven-parametrisiertes kinematisches Fahrzeugmodell)**

$$
\begin{aligned}
s'(t) &= \frac{v(t)\left(x'_m(s(t))\cos(\psi(t)) + y'_m(s(t))\sin(\psi(t))\right)}{1 + r(t)\kappa_m(s(t))},\\
r'(t) &= v(t)\left(y'_m(s(t))\cos(\psi(t)) - x'_m(s(t))\sin(\psi(t))\right),\\
\psi'(t) &= \frac{v(t)}{\ell}\tan(\delta(t))
\end{aligned}
$$

Wir reduzieren dieses Fahrzeugmodell in dem wir annehmen, dass die Geschwindigkeit $v(t) = 1$ ist, für alle $t \in [t_0, t_f]$.

**Bemerkung 3.6.** *Die Beschränkung $v(t) = 1$ kann dadurch gerechtfertigt werden, dass sich bei Zeit optimalen Bahnen, in welchen die Geschwindigkeit nicht durch die Form der Bahn beschränkt wird, beim kinematischen Fahrzeugmodell die maximal mögliche Geschwindigkeit einstellt. Durch diese Beschränkung entsteht einer Parametrisierung nach der Bogenlänge, das heißt der Zeitpunkt $t$ entspricht gerade der gefahrenen Strecke $\zeta$. Unser Ziel bei der Parametrisierung nach der Bogenlänge ist es, die Bahnplanung von der Berechnung des Geschwindigkeitsprofils zu entkoppeln.*

Dadurch kann mit dem Intervall $[t_0, t_f]$ die gefahrene Strecke $[\zeta_0, \zeta_f]$ angegeben werden, wodurch nur noch die Form der Bahn betrachtet wird. Zusätzlich vereinfachen wir die dritte Gleichung, indem wir die neue Steuerung

$$u(\zeta) = \frac{\tan(\delta(\zeta))}{\ell}$$

einführen. Damit ergibt sich das neue Differentialgleichungssystem

$$
\begin{aligned}
s'(\zeta) &= \frac{x'_m(s(\zeta))\cos(\psi(\zeta)) + y'_m(s(\zeta))\sin(\psi(\zeta))}{1 + r(\zeta)\kappa_m(s(\zeta))},\\
r'(\zeta) &= y'_m(s(\zeta))\cos(\psi(\zeta)) - x'_m(s(\zeta))\sin(\psi(\zeta)),\\
\psi'(\zeta) &= u(\zeta).
\end{aligned}
$$

Zusätzlich möchten wir noch eine weitere Umformung der vornehmen, indem wir den Gierwinkel $\psi(\zeta)$ durch

$$\chi(\zeta) = \psi_m(s(\zeta)) - \psi(\zeta) \qquad \text{und} \qquad \chi'(\zeta) = \kappa_m(s(\zeta))s'(\zeta) - u(\zeta)$$

ersetzen. Unter Berücksichtigung der Zusammenhänge von $x'_m(s(\zeta)) = \cos(\psi_m(s(\zeta)))$, $y'_m(s(\zeta)) = \sin(\psi_m(s(\zeta)))$ und der Additionstheoreme von Sinus und Kosinus

$$\sin(\alpha \pm \beta) = \sin(\alpha)\cos(\beta) \pm \cos(\alpha)\sin(\beta)$$
$$\cos(\alpha \pm \beta) = \cos(\alpha)\cos(\beta) \mp \sin(\alpha)\sin(\beta)$$

erhalten wir das vereinfachte kurven-parametrisierte Fahrzeugmodell.

**DGL-System 3.5 (Vereinfachtes kurven-parametrisiertes Fahrzeugmodell)**

$$\begin{aligned} s'(\zeta) &= \frac{\cos(\chi(\zeta))}{1 + r(t)\kappa_m(s(\zeta))}, \\ r'(\zeta) &= \sin(\chi(\zeta)), \\ \chi'(\zeta) &= \kappa_m(s(\zeta))s'(\zeta) - u(\zeta). \end{aligned}$$

**Bemerkung 3.7.** *Die Steuerung $u(\zeta)$ beschreibt in diesem Fahrzeugmodell nicht mehr den Lenkwinkel, sondern die Krümmung der optimierten Bahn. Diese kann vom Bahnfolgeregler verwendet werden. Die für den Regler benötigte Geschwindigkeit und Längsbeschleunigung sind über ein separates Optimierungsproblem zu berechnen.*

Für das Differentialgleichungssystem 3.5 muss noch eine passende Zielfunktion gefunden werden, um es in ein Optimalsteuerungsproblem zu überführen.

**OSP 3.3 (Vereinfachtes kurven-parametrisiertes Optimalsteuerungsproblem)**

Minimiere die Zielfunktion

$$-\alpha_0 s(\zeta_f) + \alpha_1 \int_0^{\zeta_f} u(\zeta)^2 d\zeta$$

unter den Nebenbedingungen

$$\begin{aligned} s'(\zeta) &= \frac{\cos(\chi(\zeta))}{1 + r(\zeta)\kappa_m(s(\zeta))}, \\ r'(\zeta) &= \sin(\chi(\zeta)), \\ \chi'(\zeta) &= \kappa_m(s(\zeta))s'(\zeta) - u(\zeta), \\ u(\zeta) &\in [u_{\min}, u_{\max}], \\ r(\zeta) &\in [r_{\min}(s(\zeta)), r_{\max}(s(\zeta))], \end{aligned}$$

und den Anfangsbedingungen

$$s(0) = s_0, \qquad r(0) = r_0, \qquad \chi(0) = \chi_0.$$

Die Zielfunktion kann in die beiden Terme

$$-\alpha_0 s(\zeta_f) \qquad \text{und} \qquad \alpha_1 \int_0^{\zeta_f} u(\zeta)^2 d\zeta$$

unterteilt werden. Der erste Term entspricht wie schon beim Einspurmodell dem Fortschritt auf der Bahn im Bezug auf die Mittellinie. Dies entspricht dem kürzesten gefahrenen Weg, um eine möglichst große Endbogenlänge im Bezug auf die Mittellinie zu erreichen. Der Wert der gefahrenen Trajektorie ist die Länge $L = \zeta_f - \zeta_0$ festgelegt und die Endbogenlänge im Bezug auf die Mittellinie frei. Der zweite Term beschreibt eine Minimierung des Krümmungsaufwands.

Mit einem Blick auf das Optimalsteuerungsproblem aus OSP 3.3 erkennen wir, dass dort nur zwei skalare Faktoren $\alpha_0$ und $\alpha_1$ auftreten. In das Differentialgleichungssystem 3.5 gehen keine Faktoren, oder andere Fahrzeugparameter ein. Das heißt, dass wir unser spezifisches Fahrzeug ausschließlich durch die beiden Faktoren in der Zielfunktion charakterisieren müssen. Dafür veranschaulichen wir uns zunächst die Bedeutung der beiden Faktoren im Bezug auf die berechnete Trajektorie.

Für besonders beschleunigungsstarke Fahrzeuge, beispielsweise ein sportlicher Rennwagen, überwiegt der erste Term. Dies liegt darin begründet, dass ein beschleunigungsstarkes Fahrzeuge für Kurven stärker Abbremsen kann, da es die Geschwindigkeit schnell wieder aufbauen kann. Dementsprechend ist es für ein sehr sportliches Fahrzeug sinnvoll eng und auf dem möglichst kürzestem Weg durch Kurven zu fahren. Anders verhält es sich bei beschleunigungsschwachen Fahrzeugen. Hier dauert das Aufbauen von Geschwindigkeit

länger. Dementsprechend wird versucht für Kurven möglichst wenig abzubremsen, was durch die Minimierung der gefahrenen Kurvenkrümmung erreicht werden kann.
Die Faktoren $\alpha_0$ und $\alpha_1$ müssen zueinander gewichtet werden, so dass sie das Verhalten der verwendeten Fahrzeugs möglichst gut nachbilden.

### 3.6.3 Vergleich mit dem kurven-parametrisierten Einspurmodell

Um die optimierten Trajektorien des vereinfachten kurven-parametrisierten Fahrzeugmodells bewerten zu können, vergleichen wir das Modell mit den berechneten Rennlinien des kurven-parametrisierten Einspurmodells aus Abschnitt 3.5.3. Dafür wählen wir die skalaren Faktoren $\alpha_0$ und $\alpha_1$ aus der Zielfunktion in OSP 3.3 so, dass wir möglichst gut an die Lösung des kurven-parametrisierten Fahrzeugmodells herankommen.

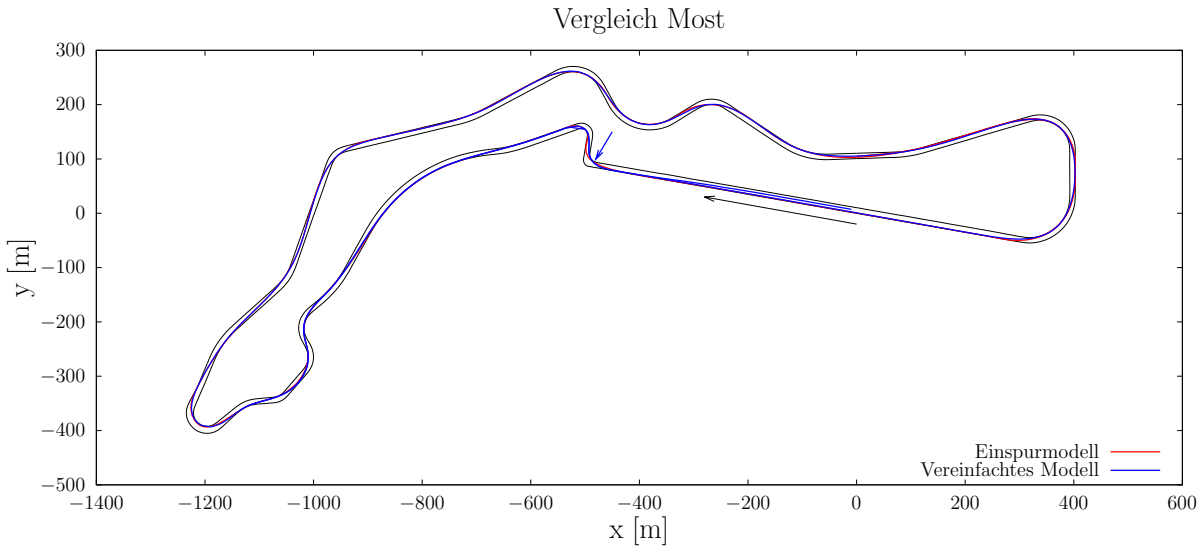


Abbildung 3.20: Vergleich anhand des Testkurses in Most, mit dem kurven-parametrisierten Einspurmodell aus Abbildung 3.16. Die verwendete Vorausschau des vereinfachten kurven-parametrisierten Fahrzeugmodells beträgt 250 m.

Abbildung 3.20 stellt beide Lösungen, der durch Moving Horizon fortgesetzten MPC Trajektorien, über dem Testkurs in Most dar. Da die Rennlinie des Einspurmodells unter der Trajektorie des vereinfachten kurven-parametrisierten Fahrzeugmodells gezeichnet wurde, lassen sich die Unterschiede zum Einspurmodell anhand der teils roten Hinterlegung der

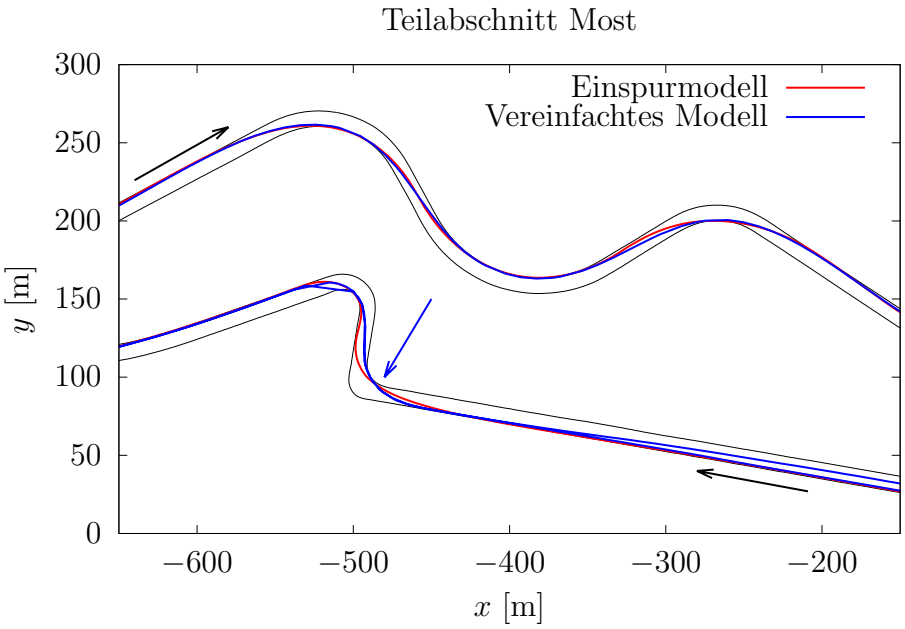


Abbildung 3.21: Vergrößerter Ausschnitt von Abbildung 3.20.

blau eingezeichneten Trajektorie erkennen. Wir sehen daran deutlich, dass das vereinfachte kurven-parametrisierte Fahrzeugmodell die Lösung des Einspurmodells sehr gut wieder spiegelt. Einen größeren Unterschied gibt es lediglich an der mit einem blauen Pfeil gekennzeichneten Stelle. In Abbildung 3.21 ist dieser Kursabschnitt noch einmal vergrößert dargestellt. Hier lässt die Vorausschau des Einspurmodells von 7 Sekunden gerade noch zu, dass die Kurve durchfahren werden kann, während der feste Vorausschau-Horizont von 250 m die Kurve optimaler durchfahren kann. Durch die geringere Komplexität von OSP 3.3 im Vergleich zu OSP 3.2 kann die optimale Trajektorie deutlich robuster und schneller berechnet werden. Dementsprechend scheint mit diesem Modell auch an eine Onlineberechnung der Rennlinie möglich zu sein.

### 3.6.4 Berechnung eines Geschwindigkeitsprofils

Zur Berechnung des Geschwindigkeitsprofils betrachten wir die longitudinale Bewegung des Fahrzeugs entlang einer berechneten Trajektorie.

**DGL-System 3.6 (Longitudinale Bewegungsgleichungen nach [14] und [16])**

$$
\begin{aligned}
s'(t) &= v(t) \\
v'(t) &= u(t) - c_F \cdot v(t) - c_D \cdot v(t)^2
\end{aligned}
$$

Für das Differentialgleichungssystem 3.6 wurde bereits in [16] ein Optimalsteuerungsproblem formuliert und durch dynamische Programmierung gelöst. Hierbei wird das Differentialgleichungssystem 3.6 mit Hilfe des Satzes für implizite Funktionen auf den Bogenlängenbereich transformiert und wir erhalten nach [16]

$$
\begin{aligned}
t'(s) &= \frac{1}{v(s)} \\
v'(s) &= \frac{u(s)}{v(s)} - c_F - c_D \cdot v(s).
\end{aligned}
$$

Das Optimalsteuerungsproblem aus Problem 3.4 zeigt im Vergleich zu [16] eine, für unseren Anwendungsfall, angepasste Formulierung.

**Problem 3.4 (Optimalsteuerungsproblem zur Berechnung des Geschwindigkeitsprofils)**

Minimiere die Zielfunktion

$$
\int_0^L \frac{1}{v(s)} ds + c_{v_f} \left(v(L) - v_f\right)^2
$$

unter den Nebenbedingungen

$$
\begin{aligned}
t'(s) &= \frac{1}{v(s)} \\
v'(s) &= \frac{u(s)}{v(s)} - c_F - c_D \cdot v(s) \\
\sqrt{\kappa(s)^2 v(s)^4 + u(s)^2} &\leq \mu \cdot g \\
v(s) &\in [v_{\min}, v_{\max}] \\
u(s) &\in [u_{\min}, u_{\max}] \\
t(0) &= t_0 \\
v(0) &= v_0.
\end{aligned}
$$

Die Nebenbedingung $\sqrt{\kappa(s)^2 v(s)^4 + u(s)^2} \leq \mu \cdot g$ kann aus dem Kammschen Kreis, vgl. [9]

$$\begin{aligned} \frac{\sqrt{F_l^2 + F_s^2}}{F_z} - \mu &\leq 0 \\ \frac{\sqrt{(m \cdot a_l)^2 + (m \cdot a_s)^2}}{m \cdot g} &\leq \mu \\ \sqrt{a_l^2 + a_s^2} &\leq \mu \cdot g \\ \sqrt{\kappa^2 v(s)^4 + a_s^2} &\leq \mu \cdot g, \end{aligned}$$

mit $a_l(s) = |\kappa(s)| \cdot v(s)^2$, ermittelt werden. Die Zielfunktion von Problem 3.4 besitzt zwei Anteile. Der Anteil

$$\int_0^L \frac{1}{v(s)} ds$$

entspricht einer Minimierung der Endzeit und

$$c_{v_f} \left( v(L) - v_f \right)^2$$

einem Penalty Term für die Endgeschwindigkeit. Dieser ist aus Sicherheitsgründen für die online-Tests eingeführt worden, um die Endgeschwindigkeit an den MPC Trajektorien zu reduzieren. Dies ist vorteilhaft, falls keine optimalen Trajektorien mehr gefunden werden, da der Testfahrer im Ernstfall bei einer weniger hohen Geschwindigkeit eingreifen muss.

Wie auch in [16] überführen wir das Optimalsteuerungsproblem 3.4 für die dynamische Programmierung in eine diskrete Form.

**Problem 3.5 (Diskretes Optimalsteuerungsproblem zur Berechnung des Geschwindigkeitsprofils)**

Minimiere die Zielfunktion

$$h \cdot \sum_{i=0}^{n_s - 1} \frac{1}{v(s_i)} + c_{v_f} \left( v(s_{n_s}) - v_f \right)^2$$

unter den Nebenbedingungen

$$
\begin{aligned}
t(s_{i+1}) &= t(s_i) + h \cdot \frac{1}{v(s_i)} && i = 1, \dots, n_s - 1 \\
v(s_{i+1}) &= v(s_i) + h \cdot \left( \frac{u(s_i)}{v(s_i)} - c_F - c_D \cdot v(s_i) \right) && i = 1, \dots, n_s - 1 \\
\sqrt{\kappa(s_i)^2 v(s_i)^4 + u(s_i)^2} &\leq \mu \cdot g && i = 1, \dots, n_s \\
v(s_i) &\in [v_{\min}, v_{\max}] && i = 1, \dots, n_s \\
u(s_i) &\in [u_{\min}, u_{\max}] && i = 1, \dots, n_s - 1 \\
t(s_0) &= t_0 \\
v(s_0) &= v_0.
\end{aligned}
$$

In Optimalsteuerungsproblem 3.5 wird ein expliziter Euler Schritt zur Integration des Differentialgleichungssystems und der Zielfunktion verwendet. Dieses diskrete Optimalsteuerungsproblem lässt sich mit der dynamischen Programmierung nach Abschnitt 2.4 lösen. Aufgrund der geringen Zustands- und Steuerungsdimension sind dabei sowohl die Rückwärts- als auch die Vorwärtsrechnung der dynamischen Programmierung online fähig. Als Krümmung $\kappa(s)$ kann zur Berechnung die Steuerung der optimalen MPC Trajektorie verwendet werden.
Wir testen dieses Verfahren für eine Gerade mit 50 Gitterpunkten und 200 Metern Länge. In Abbildung 3.22 fällt auf, dass ab etwa 150 Metern die Penalty-Beschränkung aus der Zielfunktion aktiv wird. Hier zeigt das optimierte Beschleunigungsprofils jedoch einen Nachteil. Beim moderaten Bremsen springt die Beschleunigung zwischen dem Maximalwert und keiner bis kaum Bremsbeschleunigung hin und her. Um dieses Verhalten in den Online-Fahrversuchen zu verhindern verwenden wir einen gleitenden Zwischenwertfilter, um das Beschleunigungsprofil zu glätten. Um steile Beschleunigungsflanken nicht herauszufiltern, welche die Bremspunkte des Fahrzeugs angeben, verwenden wir den Filter nur in Punkten, an denen ein Alternieren in der Beschleunigung vorliegt. Der Zwischenwert bildet sich jeweils über drei Punkte

$$a_i = a_i + \frac{\min(a_{i-1} - a_i, a_{i+1} - a_i)}{2}.$$

Nach mehrfacher Anwendung, in unserem Fall dreimalig, können die Sprünge reduziert werden.

**Bemerkung 3.8.** *In [9] und [10] sind noch weitere Maßnahmen zur Glättung der Be-*

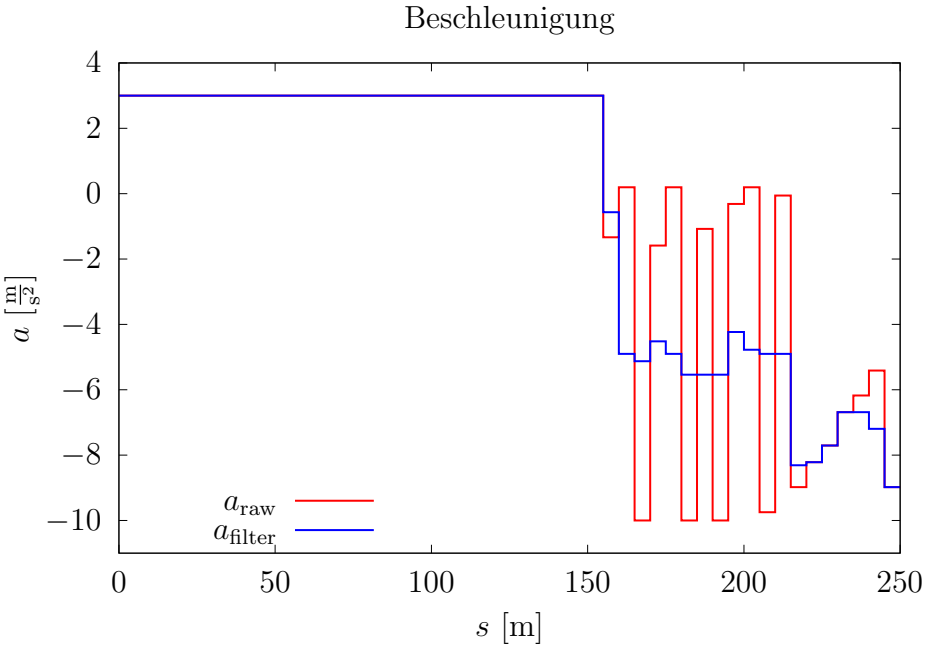


Abbildung 3.22: Filterung des Beschleunigungsprofils

*schleunigung beschrieben, wie das Hinzufügen des Beschleunigungsaufwands zur Zielfunktion. Für die Online-Fahrversuche im Rahmen dieser Arbeit wurde der gleitende Zwischenwertfilter verwendet.*

**Bemerkung 3.9.** *Die Anpassung und Implementierung der Berechnung des Geschwindigkeits- und Beschleunigungsprofils wurde in Zusammenarbeit mit Andreas Britzelmeier durchgeführt. Als Grundlage für den Code diente die Implementierung aus [16].*

## 3.7 Software zur Offlinebahnplanung

Zur Berechnung von optimierten offline-Trajektorien wurden grafische Benutzeroberflächen erstellt, die zur Überprüfung und Darstellung der berechneten Lösung während der Laufzeit dienen. Dies ist insbesondere für das vereinfachte kurven-parametrisierte Fahrzeugmodell wichtig, da mit Hilfe der grafischen Benutzeroberfläche das Laufzeitverhalten des Optimierers außerhalb des Fahrzeugs getestet werden kann. Etwaige Fehlerquellen können wir dadurch bereits vor den Online-Fahrversuchen eliminieren. Im Folgenden stellen wir die Benutzeroberflächen für das kurven-parametrisierte Einspurmodell aus Abschnitt 3.5.3 und das vereinfachte kurven-parametrisierte Fahrzeugmodell aus Abschnitt 3.6 vor.

### 3.7.1 Benutzeroberfläche für das kurven-parametrisierte Einspurmodell

Für das in Abschnitt 3.5.3 beschriebene kurven-parametrisierte Einspurmodell wurde eine grafische Benutzeroberfläche erstellt. Diese Oberfläche dient zur Ermittlung geeigneter Optimierungsparameter, um die Qualität der Rennlinie des Einspurmodells zu verbessern. Die in Abschnitt 3.5.3 dargestellten Ergebnisse sind unter Verwendung dieser Benutzeroberfläche entstanden und anschließend auch durch offline-Fahrversuche validiert worden.


Abbildung 3.23: Darstellung des Kurses in der grafischen Benutzeroberfläche des kurven-parametrisierten Einspurmodells

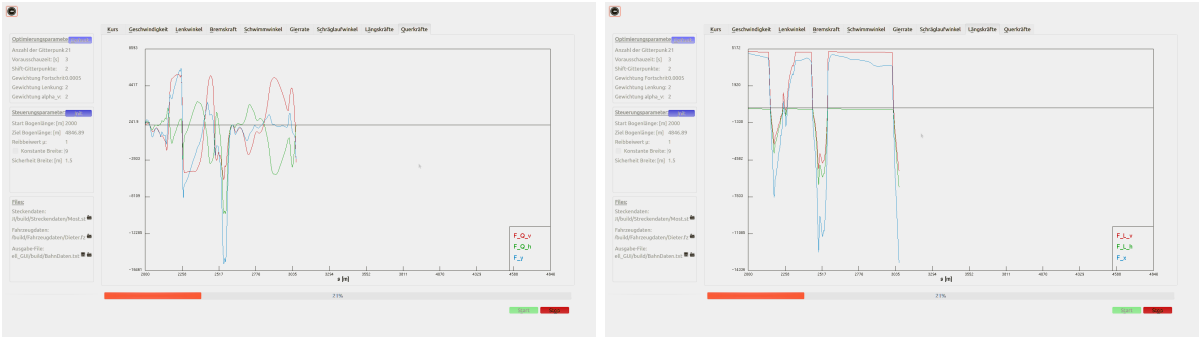

Abbildung 3.24: Darstellung der Quer- und Längskräfte in der grafischen Benutzeroberfläche des kurven-parametrisierten Einspurmodells

Die grafische Benutzeroberfläche des kurven-parametrisierten Einspurmodells ist in den Abbildungen 3.23 und 3.24 dargestellt. Links in der Oberfläche befindet sich ein Konfigurationsmenü für die Optimierungsparameter. Im zentralen Bereich der Benutzeroberfläche sind zum Einen der Kurs und zum Anderen die internen Fahrzeuggrößen dargestellt. Die Abbildungen 3.23 und 3.24 zeigen im Vergleich zu Abschnitt 3.5.3 eine Trajektorie für eine kurze Vorausschau von nur 3 Sekunden. Die Berechnungen mit einer höheren Vorausschau, wie in Abschnitt 3.5.3 dargestellt, benötigen deutlich längere Rechenzeiten.
Unter Verwendung dieser grafischen Benutzeroberfläche konnten wir passende Optimierungsparameter finden, die auf gut bewertete Rennlinien führen. Die Bewertung der Rennlinien fand in Zusammenarbeit mit Testfahrern statt. Die Rechenzeiten und die Stabilität der optimalen Steuerung reichen dieser Softwarezusammensetzung jedoch nicht aus, um plausible Online-Fahrversuche mit dem kurven-parametrisierte Einspurmodell zu realisieren.

### 3.7.2 Benutzeroberfläche für das vereinfachte kurven-parametrisierte Fahrzeugmodell

Die Benutzeroberfläche für das vereinfachte kurven-parametrisierte Fahrzeugmodell stellt sich einfacher dar als die des Einspurmodells. Dies liegt daran, dass weniger interne Fahrzeuggrößen berechnet werden.
In Abbildung 3.25 ist die Übersichtsdarstellung der grafischen Benutzeroberfläche während der Laufzeit für den Testkurs Most dargestellt. Die Benutzeroberfläche unterteilt sich in das Steuermenü auf der linken Seite und die grafische Darstellung der Trajektorie und internen Fahrzeuggrößen im zentralen Bereich der Oberfläche. Im Vergleich zur Benutzeroberfläche des Einspurmodells sind keine Optimierungsparameter mehr einstellbar. In den Analysen von Abschnitt 3.6.3 haben wir bereits Parameter ermittelt, welche die optimale Trajektorie des Einspurmodells gut widerspiegeln. Diese Benutzeroberfläche soll uns vor allem im Bezug auf die Laufzeiteigenschaften des Optimierers Aufschluss geben.
In Abbildung 3.25 sehen wir vier Diagramme, welche die berechneten Daten einer MPC-Trajektorie darstellen. Die oberen beiden Diagramme stellen die Lösung der letzten optimierten Trajektorie dar. In der Kursdarstellung links oben wird die aktuelle Trajektorie grün dargestellt. Die Abbildung des Kurses lässt sich in der Benutzeroberfläche auch isolieren und vergrößern, wie Abbildung 3.26 zeigt. Die bereits gefahrene Bahn ist rot eingezeichnet. Rechts vom Kurs wird die berechnete Steuerung der Optimierung, welche der Krümmung entspricht, für die aktuelle MPC-Trajektorie abgebildet. Bei einer Beobachtung während der Laufzeit lässt sich erkennen, dass die Krümmung der einzelnen

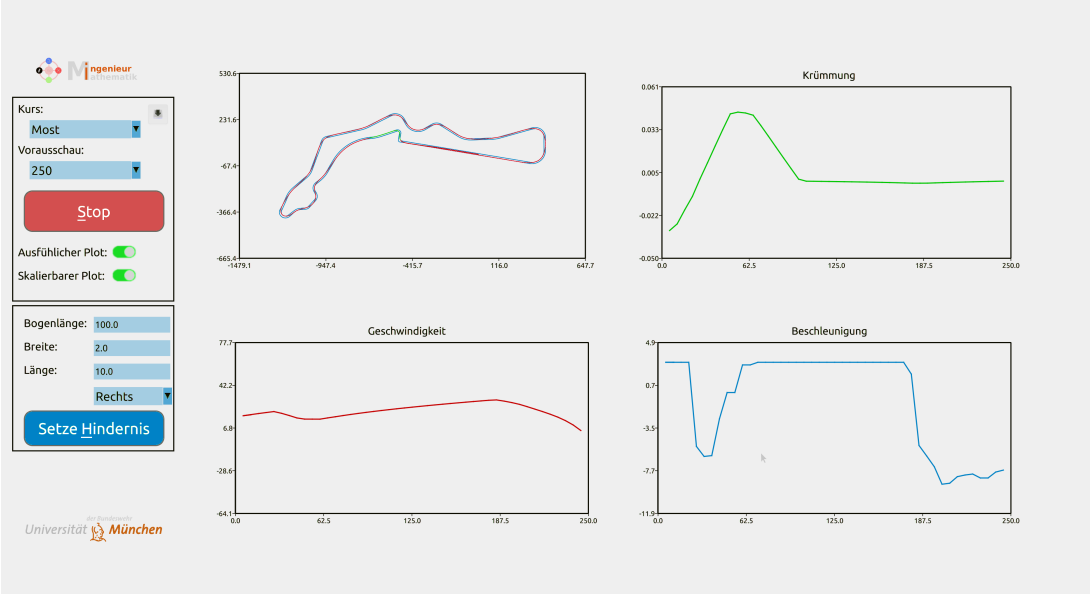


Abbildung 3.25: Gesamtdarstellung einer Berechnung für den Testkurs Most in der grafischen Benutzeroberfläche für das vereinfachte kurven-parametrisierte Fahrzeugmodell

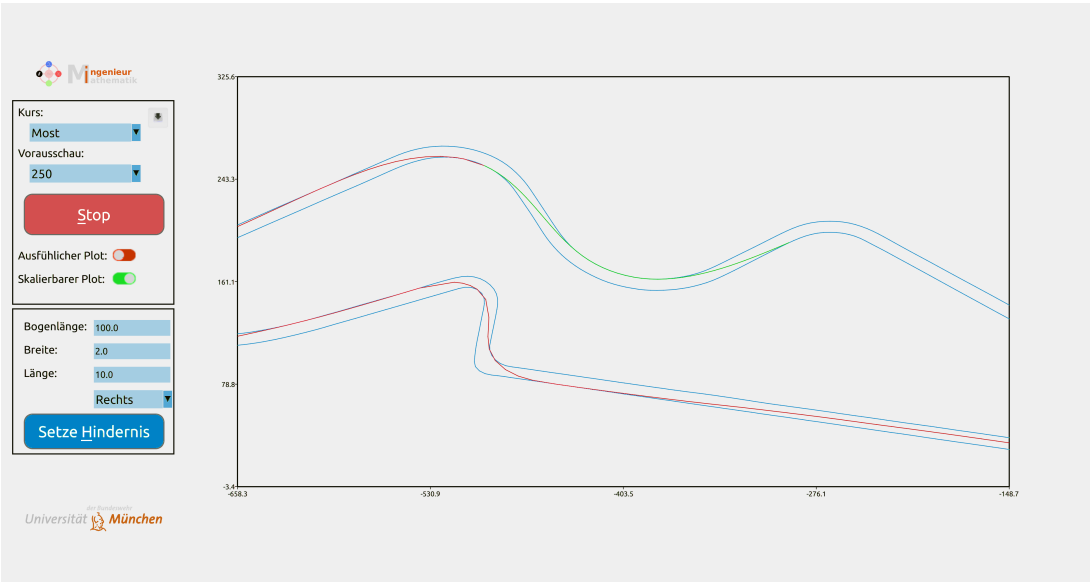


Abbildung 3.26: Detaildarstellung für den Kurs Most in der grafischen Benutzeroberfläche für das vereinfachte kurven-parametrisierte Fahrzeugmodell

MPC-Trajektorien stetig verläuft und keine großen Sprünge aufweist. Anhand des Diagramms sehen wir, dass der Krümmungsverlauf in Abbildung 3.25 nicht bei 0 beginnt. Dahinter steckt die Absicht auch fehlgeschlagene Optimierungen zu visualisieren. Schlägt

eine Optimierung fehl, wird die bisherige MPC-Trajektorie weiter abgefahren. Dies macht sich beim Verlauf der Krümmung dadurch bemerkbar, dass der Verlauf vom Anfang weg verschwindet. Durch diese Art der Darstellung können wir feststellen, ob die Optimierung ins Stocken gerät. Für den in Abbildung 3.25 dargestellten Testkurs Most geschieht dies jedoch bei einer Vorausschau von 250 Metern kaum.
Im unteren Bereich von Abbildung 3.25 sind noch zwei weitere Diagramme dargestellt, welche die Geschwindigkeit und die Beschleunigung über der aktuellen MPC-Trajektorie abbilden. Berechnet werden diese beiden Verläufe wie in Abschnitt 3.6.4 beschrieben. Diese Verläufe dienen vor allem dazu, die Qualität von Geschwindigkeits- und Beschleunigungsprofil zu überprüfen und zu verbessern. Zwei Verbesserungen, welche bereits in Abschnitt 3.6.4 beschrieben wurden, sind die Filterung des Beschleunigungsprofils und Beschränkung des Geschwindigkeitsprofils am Endpunkt. Durch die Filterung glätten wir das Beschleunigungsprofil, so dass der Verlauf für Online-Fahrversuche praktikabel wird. Die Beschränkung des Endpunkt des Geschwindigkeitsprofils erfolgt ebenfalls aus praktikablen Gründen. Die Geschwindigkeit am Ende der Trajektorie wird stark verzögert, um in Online-Fahrversuchen das Risiko im Testfahrzeug beim Fehlschlagen der Berechnung von neuen MPC-Trajektorien zu minimieren.
Die grafische Benutzeroberfläche zeigt, dass das Zusammenspiel der Berechnungen von MPC-Trajektorie und Geschwindigkeitsprofil funktioniert. Somit können auch Online-Fahrversuche durchgeführt werden.

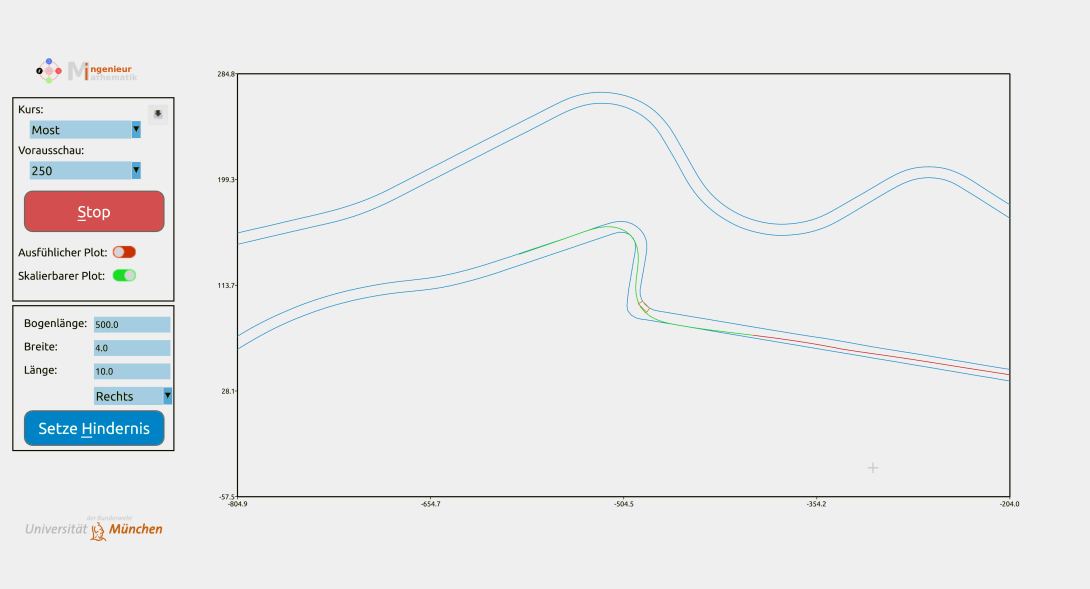


Abbildung 3.27: Detaildarstellung mit Hindernis für den Kurs Most in der grafischen Benutzeroberfläche für das vereinfachte kurven-parametrisierte Fahrzeugmodell

Des Weiteren können in der grafischen Benutzeroberfläche während der Laufzeit auch Randhindernisse, wie sie in Abschnitt 3.2.1 beschrieben sind, eingebracht werden. Die Trajektorie muss sich daraufhin auf die Hindernisse anpassen. Ein Beispiel für MPC-Trajektorie mit Randhindernis für den Testkurs Most ist in Abbildung 3.27 dargestellt. Nach den ausgiebigen Tests der Algorithmen hinsichtlich ihrer Laufzeiteigenschaften können wir damit beginnen Online-Fahrversuche mit einem Versuchsträger durchzuführen.

## 3.8 Onlinebahnplanung mit vereinfachtem Fahrzeugmodell

### 3.8.1 Aufbau des Fahrversuchs

Im Vergleich zu den offline-Versuchsreihen aus Abschnitt 3.3, findet bei online-Versuchen auch die Berechnung der MPC Trajektorien während der Fahrt, auf einem Computer im Fahrzeug statt.

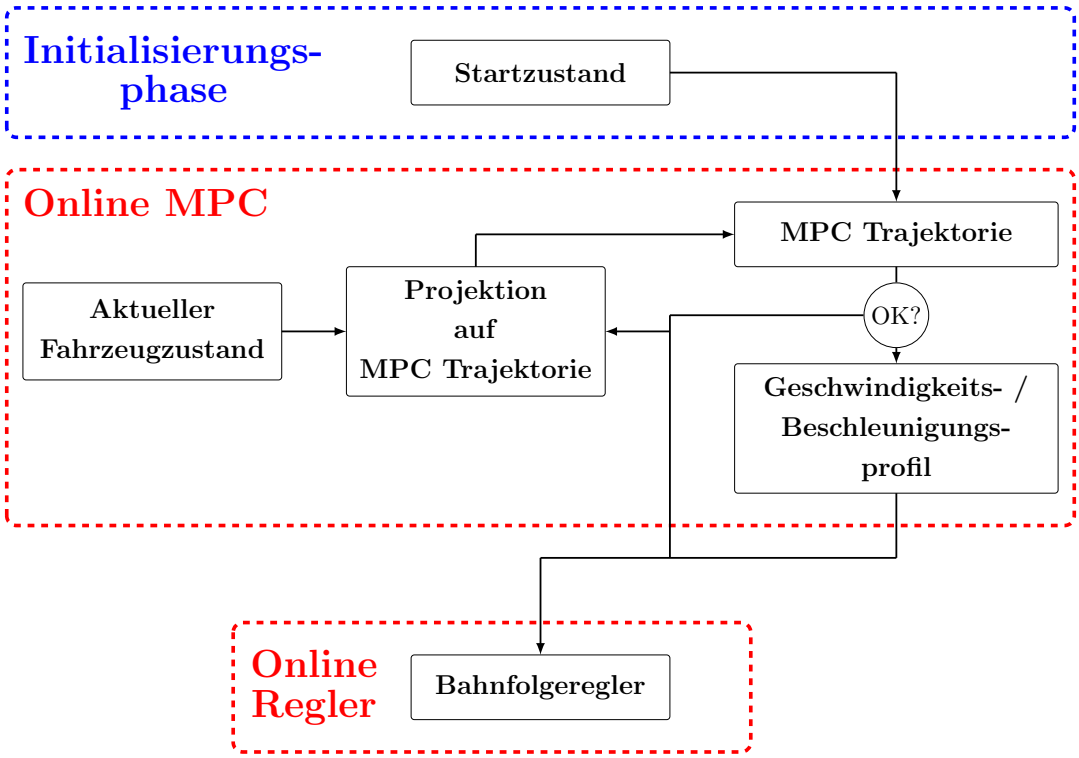


Abbildung 3.28: Schematischer Aufbau der Online-Fahrversuche

In Abbildung 3.28 ist der Aufbau der Online-Fahrversuche schematisch skizziert. Das au-

tonome Fahrzeug beginnt zunächst mit einer Initialisierungsphase, in der für eine aktuelle Position auf dem Testkurs eine MPC Trajektorie mit Geschwindigkeits- und Beschleunigungsprofil berechnet wird. Erst ab dem Zeitpunkt zu dem eine zulässige MPC Trajektorie aus der Initialisierungsphase vorliegt, kann mit der online-Optimierung und der autonomen Testfahrt begonnen werden.
Nach der Initialisierungsphase unterteilen sich die autonomen Fahrversuche in zwei übergeordnete Blöcke. Zum Einen in die Onlinebahnplanung und zum Anderen in den Bahnfolgeregler. Diese beiden Einheiten laufen mit unterschiedlichen Frequenzen, was es ermöglicht die Bahnplanung auf einer eigenen Hardware zu berechnen. Während der Bahnfolgeregler auf einem Echtzeit-System läuft, kann die Bahnplanung auch auf einem normalen Betriebssystem auf einem im Fahrzeug verbautem Versuchsrechner ausgeführt werden. Neu berechnete MPC Trajektorien werden an den Regler gesendet und Fahrzeugdaten, wie die Fahrzeugposition, an beide Subsysteme verteilt.
Im Vergleich zu den offline-Fahrversuchen liegt der größte Unterschied bei dem in Abbildung 3.28 skizzierten Aufbau der Online-Fahrversuche im MPC Block. Während der Laufzeit wird der initialisierte Optimierer in einer regelmäßigen Frequenz mit dem aktuellen Fahrzeugzustand aufgerufen. Die Fahrzeugposition wird anschließend auf die letzte MPC Trajektorie projiziert. Dies ist notwendig, um sicherzustellen, dass der Regler keine Sprünge in der Trajektorie erhält. Würde mit jedem MPC Schritt die aktuelle Position verwendet, so setzt diese mit jeder Aktualisierung die Ablage oder auch den Fehler des Reglers zurück. Dies führt zu einer schwierigen Kopplung zwischen Regler und Optimierer. Um diese Kopplung zu umgehen projizieren wir den aktuellen Fahrzeugzustand auf die aktuelle Trajektorie. Die Projektion ist in Abbildung 3.29 skizziert. Für die Projek-

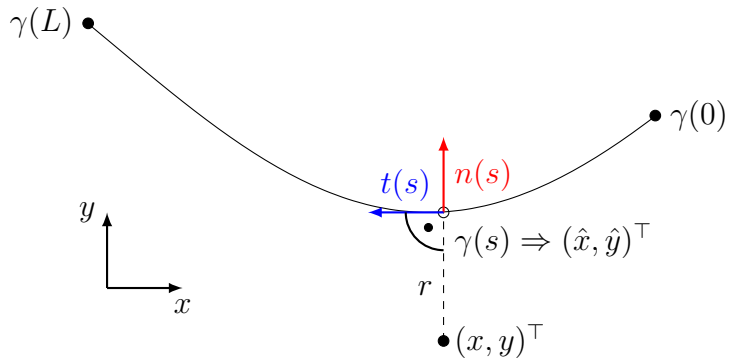


Abbildung 3.29: Projektion auf Trajektorie für online MPC

tion berechnen wir einen kubischen Spline über die Gitterpunkte der MPC Trajektorie. Diesen Spline betrachten wir analog zu in Abschnitt 3.1.3 als Kurve $\gamma : [0, L] \longrightarrow \mathbb{R}^2$.

Mit Hilfe der Kurve können wir durch die Verwendung eines lokalen Newton-Verfahrens eine Position auf die Trajektorie projizieren. Der Projektionsalgorithmus wird in einer im Rahmen dieser Arbeit entstandenen Publikation [47] erläutert.
In unserem Versuchsaufbau aus Abbildung 3.28 wird die projizierte Position an den Optimierer übergeben, der für diese Startposition eine neue MPC Trajektorie optimiert. Nachdem die Optimierung abgeschlossen ist überprüfen wir, ob die Lösung verwendet werden kann. Schlägt die Optimierung fehl oder benötigt diese zu viel Rechenzeit, so wird die Lösung verworfen und mit einem aktuelleren Fahrzeugzustand neu gestartet. Das Verwerfen von Trajektorien, die eine zu lange Rechenzeit benötigt haben, liegt darin begründet, dass sich das Fahrzeug während der Optimierung weiter bewegt. Für den Fall, dass die Trajektorie zulässig ist, berechnen wir für diese, wie in Abschnitt 3.6.4 beschrieben, ein Geschwindigkeits- und Beschleunigungsprofil. Das für den Regler benötigte Krümmungsprofil, kann direkt aus der Steuerung des vereinfachten kurven-parametrisierten Fahrzeugmodells erstellt werden. Zuletzt erfolgt eine Übertragung der Daten an den Bahnfolgeregler, welcher diese als neue Trajektorie übernimmt.

**Bemerkung 3.10.** *Wie bereits in Abschnitt 3.3 wird der Bahnfolgeregler, welcher vom Projektpartner der Volkswagen AG bereitgestellt wurde, als Blackbox behandelt und nicht im Detail beschrieben.*

### 3.8.2 Ergebnisse

Die online-Testfahrten auf Rennstrecken, konnten ausschließlich auf dem Testkurs in Most durchgeführt werden. Hierbei kam ein Audi RS-7 aus Abbildung 3.30 zum Einsatz. Im Vorhinein wurden lediglich einzelne Tests, auf einer freien Dynamikfläche durchgeführt, um die Grundfunktionalität des Optimierers zu überprüfen. Die Inbetriebnahme, sowie die ersten Testfahrten auf der Rennstrecke in Most, fanden mit manueller Längsführung statt. Der in diesem Abschnitt präsentierte Fahrversuch verwendet eine autonome Längsführung auf Basis der in Abschnitt 3.6.4 berechneten Geschwindigkeits- und Beschleunigungsprofile. Um dem Testfahrer genügend Spielraum für Eingriffe in Notsituationen zu geben, werden vom Regler nur 40% der berechneten Längsdynamik verwendet. Die online auf dem Versuchs-

Abbildung 3.30: Online-Testfahrt Audi RS-7

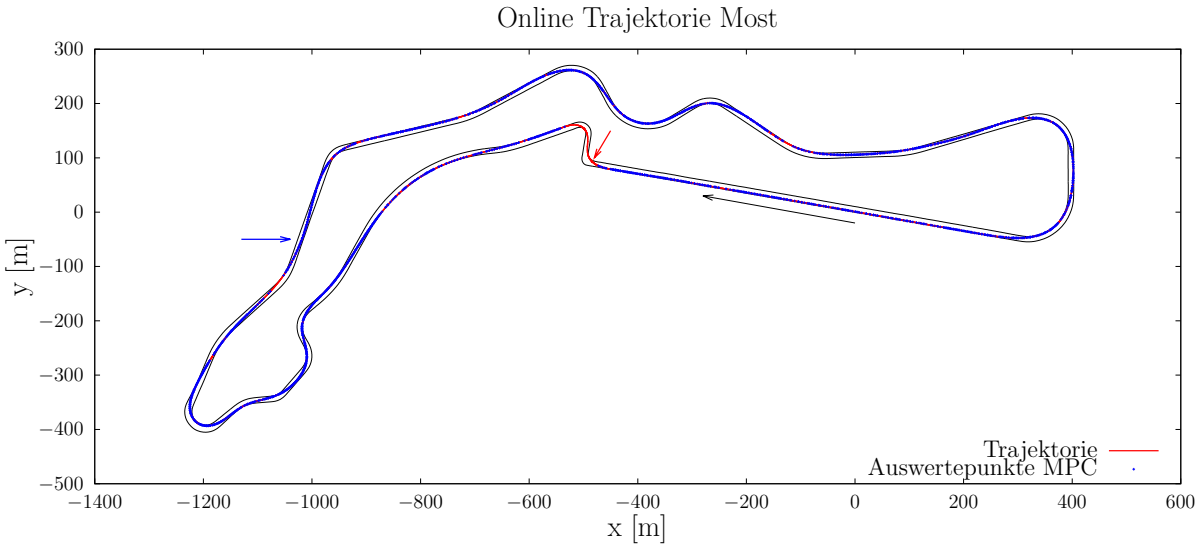


Abbildung 3.31: Online-Trajektorie für den Testkurs Most mit einem Vorausschau-Horizont von 200 m.

fahrzeug optimierte Rennlinie eines Tests mit 200 Metern Vorausschau-Horizont, bei dem der gesamte Kurs fast dreimal abgefahren wurde, ist in Abbildung 3.31 dargestellt. Die Rennlinien der drei aufeinanderfolgenden Runden fallen dabei so gut zusammen, dass in Abbildung 3.31 keine Unterschiede zu erkennen sind. Die Vorausschau von 200 Metern ist, der begrenzten CPU Taktfrequenz geschuldet, welche auf dem Testfahrzeug zur Verfügung steht. Bei Tests mit 250 Metern Vorausschau-Horizont waren die Rechenzeiten zu groß, um reproduzierbare und robuste online-Ergebnisse zu produzieren. Somit gibt es auch leichte Unterschiede im Vergleich zur Trajektorie aus Abbildung 3.20. Der größte Unterschied ist das in Abbildung 3.31 mit einen blauen Pfeil markierte Pendeln der Rennlinie, welches mit einem höheren Vorausschau-Horizont geringer ausfällt. Im Hinblick auf die Testfahrten ist dieses Pendeln als unkritisch zu bewerten.

**Bemerkung 3.11.** *Die Begrenzung der Rechenzeit wurde bei den online-Versuchen auf* 0,1 s *gelegt, um den Optimierer mit* 10 Hz *aufrufen zu können. Bei einem Vorausschau-Horizont von* 200 m *konnte ein überschreiten der maximalen Rechenzeit für die meisten MPC-Trajektorien vermieden werden.*

Zusätzlich sind die Startpunkte aller erfolgreich berechneten und verwendeten MPC Trajektorien in Abbildung 3.31 mit blauen Markern gekennzeichnet. Auf diese Art lässt sich kennzeichnen, in welchen Bereichen, des Testkurses in Most, der Optimierer ins Stocken

gerät. Deutlich zu erkennen ist hierbei wieder eine bereits bekannte Stellte des Kurses nach der Zielgeraden. Die Trajektorie zum Durchfahren der Kurvenpassage wird unmittelbar vor der Kurve berechnet, kann aber während der Kurvenpassage nicht aktualisiert werden. Erst nach dem Beenden der zweiten Kurve wird wieder eine neue MPC Trajektorie erfolgreich berechnet. Dies liegt an der Prüfung der Zulässigkeit der Trajektorie. Zum Einen schlägt hier der Optimierer mehrfach fehl und zum Anderen kann für berechnete MPC Trajektorien die Zeitbeschränkung, der Rechenzeit nicht eingehalten werden. Wünschenswert wäre an dieser Stelle eine höhere Vorausschau als 200 Meter, da in der Kurve ein Großteil des verfügbaren Vorausschau-Horizonts bereits abgefahren ist.

**Bemerkung 3.12.** *Später in Kapitel 4 wird ein neuer Optimierer mit einem anderen Diskretisierungs- und Optimierungsverfahren vorgestellt, welcher sowohl die Rechenzeiten senken, als auch höhere Vorausschau-Horizonte zulassen soll.*

Da wir die Geschwindigkeits- und Beschleunigungsprofile wie in Abschnitt 3.6.4 beschrieben separat nach der Bestimmung der MPC Trajektorie berechnen, betrachten wir auch diese in der Auswertung der Fahrversuche. Das Geschwindigkeits- und Beschleunigungs-

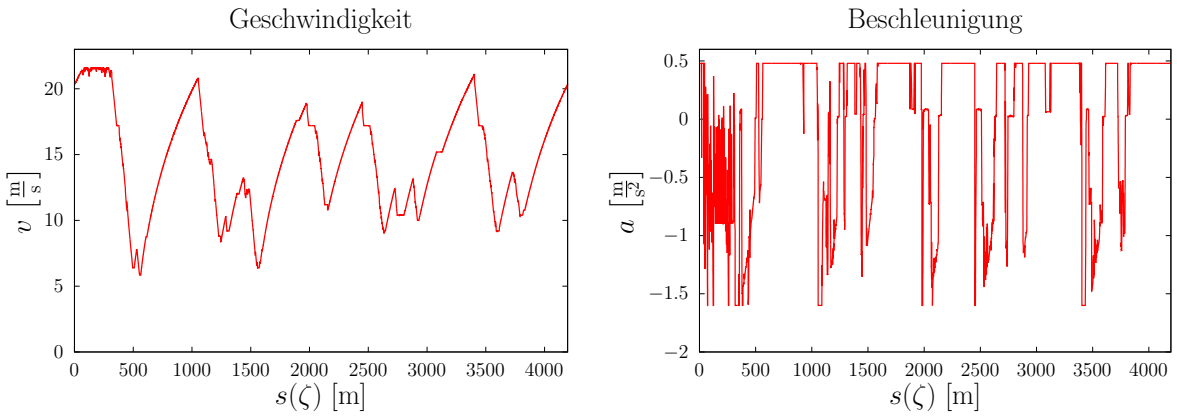


Abbildung 3.32: Berechnetes Geschwindigkeits- und Beschleunigungsprofil einer online gefahrenen Runde auf dem Testkurs Most

profil für eine komplette Runde ist in Abbildung 3.32 dargestellt. Es handelt sich hierbei um die zweite Runde der in Abbildung 3.31 dargestellten Trajektorie. Wie bereits angegeben verwendet der Regler nur 40% der berechneten Längsdynamik was beim Geschwindigkeitsprofil deutlich wird. Der grundlegende Verlauf des Geschwindigkeitsprofils spiegelt ein weitgehend ähnliches Verhalten, wie das Geschwindigkeitsprofil des Einspurmodells aus Abbildung 3.16 wieder. Ein schlechteres Ergebnis zeichnet sich beim Beschleunigungsprofil ab. Das Beschleunigungsprofil des Einspurmodells aus Abbildung 3.16 zeigt zwar auch

Sprünge in der Steuerung, diese spiegeln allerdings weitgehend auch das Verhalten eines Rennfahrers wieder, der vor Kurven stark abbremst und nach Kurven stark beschleunigt. Das in 3.32 dargestellte Beschleunigungsprofil hat deutlich mehr Rauschen in Vergleich zum Einspurmodell. Besonders drastisch ist dieses Rauschen auf der Start- bzw. Zielgeraden, wo die maximal zulässige Geschwindigkeit erreicht wird. Hier versucht das berechnete Beschleunigungsprofil die Geschwindigkeit konstant zu halten, ein leichtes Rauschen lässt sich hier allerdings auch in der Geschwindigkeit feststellen.
Letztlich sind sowohl das Geschwindigkeits- als auch das Beschleunigungsprofil fahrbar. Allerdings sollte für das Fahren im dynamischen Grenzbereich, mit 100% der berechneten Längsdynamik, das Beschleunigungsprofil noch verbessert werden.

**Bemerkung 3.13.** *Bis zu* 80% *der berechneten Längsdynamik, konnten auf einer freien Dynamikfläche mit der in Abschnitt 3.6.4 beschriebenen Berechnung von Geschwindigkeits- und Beschleunigungsprofil getestet werden.*

### 3.8.3 Ergebnisse mit Hindernissen

Nachdem der Optimierer bei Online-Fahrversuchen robuste MPC Trajektorien berechnet und daraus gute Rennlinien resultieren, können wir uns mit dem Thema von Hindernissen auf der Fahrbahn befassen. Dazu verwenden wir die in Abschnitt 3.2.1 definierten Randhindernisse. Da Einzelhindernisse im wesentlichen nur zu einer Querablage von der optimalen Rennlinie führen, ist es für Fahrversuche interessanter Doppelhindernisse zu betrachten. Die Abbildung 3.33 zeigt die Position des Doppelhindernisses für den Testkurs in Most. Der Abstand zwischen den beiden Hindernissen beträgt 50 Meter. Das erste Hindernis ist am linken Fahrbahnrand und ragt auf einer Länge von 10 Metern, 5 Meter in die Fahrbahn hinein. Das zweite Hindernis folgt auf dem rechten Fahrbahnrand und besitzt eine Breite von 7 Metern. Die Länge des zweiten Hindernisses entspricht ebenfalls 10 Meter. Bei diesem Hindernis bleibt dem Fahrzeug nur etwas mehr als eine Fahrzeugbreite Platz, um das Hindernis zu passieren. Dementsprechend ist dieses Szenario eher eine extreme Hindernissituation. Die Hindernisse werden bei den Online-Fahrversuchen rein softwareseitig eingespielt, was bedeutet, dass keine realen Hindernisse an den entsprechenden Positionen vorliegen.
Die beschriebene Hindernissituation wurde online, durch mehrere Fahrversuche getestet. In Abbildung 3.34 ist die online optimierte Bahn dargestellt, welche in Most gefahren wurde. Anhand der blauen Referenzlinie, welche der online MPC Trajektorie ohne Hindernis entspricht sehen wir, dass das erste Hindernis gerade die Innenseite der Kurve blockiert an der die optimale Bahn verläuft. Dementsprechend weicht das Fahrzeug in etwa zur

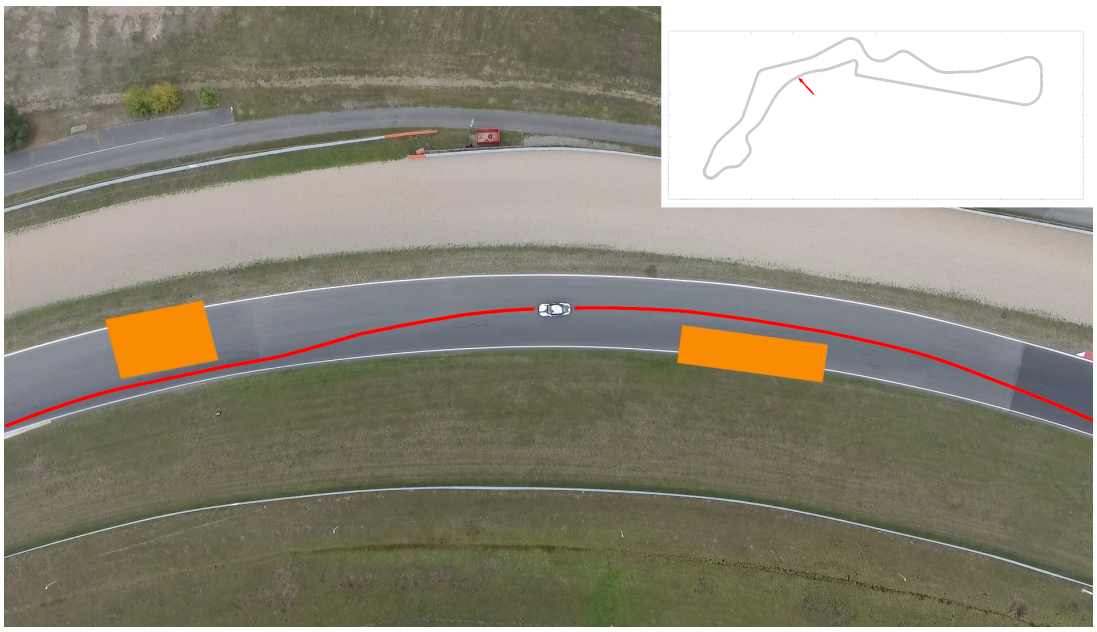

Abbildung 3.33: Qualitative Darstellung der Position des Doppelhindernisses auf dem Testkurs in Most

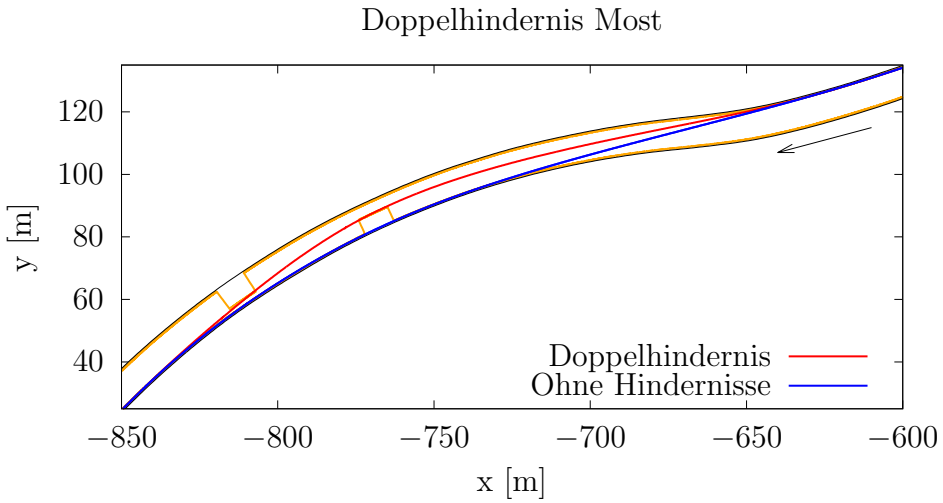


Abbildung 3.34: MPC Trajektorie für ein Doppelhindernis

Mitte der Bahn aus. Durch das zweite Hindernis wird das Fahrzeug wieder zurück zum linken Fahrbahnrand gezwungen, was aufgrund des geringen Abstands der Hindernisse

eine Herausforderung darstellt. Trotz der extremen Ausdehnung des zweiten Hindernisses lassen sich immer noch robust MPC Trajektorien finden, welche dem Doppelhindernis ausweichen können.

**Bemerkung 3.14.** *Der in Abschnitt 3.2.1 definierte Übergangsbereich, welcher einen stetig differenzierbaren Übergang vom Straßenrand zur Hindernisaußenkante ermöglicht, ist für diese Versuche mit* 0,2 *m angegeben. Der Übergangsbereich ist, aufgrund der kurzen Länge, in Abbildung 3.34 nicht zu erkennen.*

Insgesamt wurde das Doppelhindernis dreimal durchfahren. Zwei Durchfahrten verliefen dabei tadellos, wie in Abbildung 3.34 gezeigt. Bei einer der drei Durchfahrten trat während der Umfahrung des ersten Hindernisses eine Umplanung auf. Der Begriff Umplanung beschreibt den Verlust der Reproduzierbarkeit der Ergebnisse durch eine springende Trajektorie. Die zugehörigen, online berechneten MPC Trajektorien sind in Abbildung

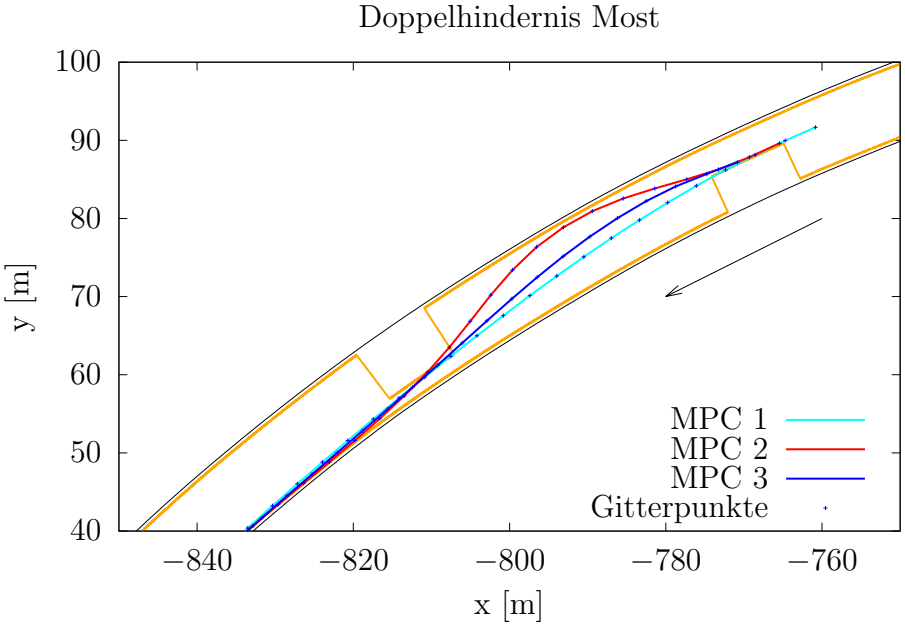


Abbildung 3.35: Umplanung der MPC Trajektorie bei Doppelhindernis

3.35 dargestellt. Während die erste Trajektorie das Doppelhindernis gut passiert, findet bei der zweiten in rot eingezeichneten MPC Trajektorie eine Umplanung statt. Die dritte MPC Trajektorie korrigiert anschließend diese Umplanung wieder zurück. Etwa zum Zeitpunkt der dritten MPC Planung kam es zu einem Eingriff des Testfahrers. Das Fahrzeug führte beim Übergang zur roten Trajektorie nämlich einen starken Ruck in der Lenkung aus, der in den vorherigen Runden nicht vorhanden war.

Um den genauen Grund für die Umplanung herauszufinden wurde zunächst überprüft, ob einer der Startpunkte der MPC Trajektorien gerade in den Übergangsbereich der Hindernisse fällt. Dabei fällt auf, dass der Startpunkt der roten Trajektorie zwar nahe an dem Übergangsbereich liegt, jedoch dennoch knapp außerhalb. Da der Optimierer die Nebenbedingungen punktweise auswertet, kann ein Einfluss des Übergangsbereichs am Startpunkt ausgeschlossen werden. Dem tatsächlichen Grund sind wir damit allerdings bereits auf der Spur. Dass die Nebenbedingungen nur punktweise erfüllt sind, wird nicht zuletzt dadurch deutlich, dass die rote Trajektorie das zweite Hindernis zwischen zwei ihrer Gitterpunkte deutlich sichtbar schneidet. Um diesen Effekt noch genauer zu visua-

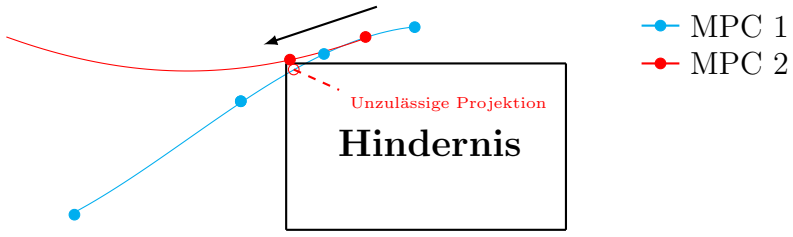


Abbildung 3.36: Schematische Darstellung der Umplanung

lisieren können wir Abbildung 3.36 betrachten. Die erste Trajektorie erfüllt in Abbildung 3.36 punktweise die Nebenbedingung. Der Startpunkt der neuen Trajektorie ist auf der letzten MPC Trajektorie interpoliert und erfüllt ebenfalls die Nebenbedingung, jedoch wäre der zweite Gitterpunkt, wenn er auf der ersten Trajektorie liegen würde unzulässig. Dementsprechend muss der Optimierer diesen Gitterpunkt so lange korrigieren, bis die Trajektorie punktweise zulässig ist. Dadurch entsteht eine nicht reproduzierbare Umplanung.

**Bemerkung 3.15.** *Ist bereits der Startpunkt einer MPC Trajektorie unzulässig kann grundsätzlich keine Lösung gefunden werden.*

Da immerhin noch zulässige Trajektorien entstehen und die Umplanung bereits in der nächsten MPC Optimierung korrigiert wird, ist dieses Verhalten zumindest nicht Sicherheitskritisch. Dennoch ist es ein größer Eingriff in den Fahrkomfort und die Optimalität der Lösung. Hinzu kommt, dass man im Bezug auf feste Hindernisse eine reproduzierbare Lösung erwarten würde. Als alternativen Ansatz könnte man die Hindernisse auch in Form von Penalty Termen in die Zielfunktion aufnehmen, um so unzulässige Zustände zu vermeiden. Je nach Gewichtung in der Zielfunktion, beeinflusst dies jedoch auch die Optimalität der MPC Trajektorien. Da die Fahrversuche nach diesen Tests beendet waren, wird dies im Rahmen dieser Arbeit nicht weiter untersucht.

**Bemerkung 3.16.** *Die Onlinebahnplanung wurde mit dem Optimierer* OCPIDlight*, einer im Umfang reduzierten Variante von* OCPID-DAE1 *durchgeführt.* OCPIDlight *verwendet ein direktes Einfachschießverfahren. Dieser Optimierer limitiert die verwendbare Vorausschaulänge und die Auflösung der Diskretisierung.*

# Kapitel 4

# Strukturausnutzung eines Optimalsteuerungsproblems in einem Innere-Punkte-Verfahren

Aufgrund der Limitierungen bezüglich der verwendbaren Vorausschaulänge und der Auflösung der Diskretisierung entwickeln wir im Folgenden einen alternativen Optimierer zu OCPID-DAE1. Hierbei verwenden wir eine direkte Diskretisierung durch Kollokation zusammen mit einem Innere-Punkte-Verfahren für große und dünn besetzte Optimierungsprobleme. Um die Leistungsfähigkeit des Optimierers noch weiter zu erhöhen, nutzen wir zur Lösung des linearen Gleichungssystems im Innere-Punkte-Verfahren gezielt die Struktur des diskretisieren Optimalsteuerungsproblems aus.

## 4.1 Direkte Diskretisierung eines Optimalsteuerungsproblems

### 4.1.1 Problemstellung

Analog zu [13] betrachten wir das bereits in Problem 2.6 definierte allgemeine Optimalsteuerungsproblem.

**Problem 4.1 (Allgemeines Optimalsteuerungsproblem (OSP))**

Bestimme die Zustandsvektorfunktion $y : [t_0, t_f] \to \mathbb{R}^{n_y}$, den Parametervektor $p \in \mathbb{R}^{n_p}$ und die zugehörige Steuerungsvektorfunktion $u : [t_0, t_f] \to \mathbb{R}^{n_u}$, so dass das

Zielfunktional

$$\varphi(y(t_0), y(t_f), p)$$

minimal wird, unter den nichtlinearen Nebenbedingungen

$$\begin{aligned} y'(t) &= F(t, y(t), u(t), p) \\ \psi_{\min} &\leq \psi(y(t_0), y(t_f), p) \leq \psi_{\max} \\ v_{\min} &\leq v(t, y(t), u(t), p) \leq v_{\max} \qquad t \in [t_0, t_f] \end{aligned}$$

und den Boxbeschränkungen

$$\begin{aligned} y_{\min} &\leq y(t) \leq y_{\max} \qquad t \in [t_0, t_f] \\ u_{\min} &\leq u(t) \leq u_{\max} \qquad t \in [t_0, t_f] \\ p_{\min} &\leq p \leq p_{\max}. \end{aligned}$$

Für die Nebenbedingungen aus Problem 4.1 gilt

$$\begin{aligned} F &: [t_0, t_f] \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_y} \\ \psi &: \mathbb{R}^{n_y} \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_\psi} \\ v &: [t_0, t_f] \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_v}. \end{aligned}$$

Dieses Optimalsteuerungsproblem wollen wir auf einem äquidistanten Zeitgitter

$$\mathbb{G} = \{t_k\}_{k=0,\ldots,N} \qquad \text{mit } t_k = t_0 + h \cdot k, \tag{4.1}$$

mit der Schrittweite $h = \frac{t_f - t_0}{N}$ diskretisieren, um es in ein restringiertes nichtlineares Optimierungsproblem zu überführen. Wir definieren das nichtlineare restringierte Optimierungsproblem:

**Problem 4.2 (Restringiertes nichtlineares Optimierungsproblem (NLP))**

Bestimme den Vektor der Optimierungsvariablen $x \in \mathbb{R}^n$, so dass die Zielfunktion

$$f(x)$$

minimal wird, unter den Nebenbedingungen

$$h(x) = 0$$
$$g(x) \leq 0$$

mit $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^p$ und $g : \mathbb{R}^n \to \mathbb{R}^q$.

Bei der direkten Diskretisierung oder auch Kollokation wird hierbei das verwendete Integrationsverfahren für die gewöhnlichen Differentialgleichungen mit in die Diskretisierung eingearbeitet.

### 4.1.2 Kollokation mit Trapezregel-Diskretisierung

Mit Hilfe des äquidistanten Zeitgitters aus Formel (4.1) mit der Schrittweite $h$, erzeugen wir einen zusammengesetzten diskreten Vektor der Optimierungsvariablen

$$x^\top = (y_0, u_0, y_1, \ldots, y_N, u_N, p) \qquad \text{mit } n_x = (N+1) \cdot (n_y + n_u) + n_p, \tag{4.2}$$

der sich aus den Zuständen $y_k = y(t_k)$, den Steuerungen $u_k = u(t_k)$ für $k = 0, \ldots, N$ und den Parametern $p$ zusammensetzt. Es ist dabei zu beachten, dass die Steuerungen bei dieser Diskretisierung durch die Trapezregel nicht durch stückweise konstante Funktionen zwischen den Gitterpunkten modelliert werden. Stattdessen verwenden wir stückweise lineare Funktionen mit den Stützstellen $u_k = u(t_k)$ für $k = 0, \ldots, N$. Wir definieren die Defekte durch die Trapezregel als

$$\zeta_k = y_{k+1} - y_k - \frac{h}{2}\left(F(t_k, y_k, u_k, p) + F(t_{k+1}, y_{k+1}, u_{k+1}, p)\right) \quad k = 0, \ldots, N-1. \tag{4.3}$$

Die Defekte können wir auch in vektorieller Form

$$\zeta(x) = \begin{bmatrix} \zeta_0 \\ \vdots \\ \zeta_{N-1} \end{bmatrix} \qquad \text{mit } \zeta : \mathbb{R}^{n_x} \to \mathbb{R}^{n_\zeta}$$

notieren. Für die diskretisierten Ungleichungsnebenbedingungen erhalten wir

$$c(x) = \begin{bmatrix} \psi(y(t_0), y(t_f), p) \\ -\psi(y(t_0), y(t_f), p) \\ v(t_0, y(t_0), u(t_0), p) \\ -v(t_0, y(t_0), u(t_0), p) \\ y(t_0) \\ -y(t_0) \\ u(t_0) \\ -u(t_0) \\ \vdots \\ v(t_N, y(t_N), u(t_N), p) \\ -v(t_N, y(t_N), u(t_N), p) \\ y(t_N) \\ -y(t_N) \\ u(t_N) \\ -u(t_N) \\ p \\ -p \end{bmatrix} \qquad c_{\max} = \begin{bmatrix} \psi_{\max} \\ -\psi_{\min} \\ v_{\max} \\ -v_{\min} \\ y_{\max} \\ -y_{\min} \\ u_{\max} \\ -u_{\min} \\ \vdots \\ v_{\max} \\ -v_{\min} \\ y_{\max} \\ -y_{\min} \\ u_{\max} \\ -u_{\min} \\ p_{\max} \\ -p_{\min} \end{bmatrix},$$

mit $c : \mathbb{R}^{n_x} \to \mathbb{R}^{n_c}$ und $c_{\max} \in \mathbb{R}^{n_c}$. Zu den Zeitpunkten $t_0$ und $t_N = t_f$ gelten die Randbedingungen aus Problem 2.6.
Mit dem diskreten Vektor der Optimierungsvariablen $x$ aus (4.2) definieren wir ein diskretes restringiertes nichtlineares Optimierungsproblem.

**Problem 4.3 (Diskretes restringiertes nichtlineares Optimierungsproblem (Trapez))**

Bestimme den Vektor der Optimierungsvariablen $x \in \mathbb{R}^{n_x}$, so dass

$$f(x) = \varphi(y_0, y_N, p) \tag{4.4}$$

minimal wird, unter den Nebenbedingungen

$$h(x) = \zeta(x) = 0 \qquad \text{mit } \zeta : \mathbb{R}^{n_x} \to \mathbb{R}^{n_\zeta} \tag{4.5}$$

$$g(x) = c(x) - c_{\max} \leq 0 \qquad \text{mit } c : \mathbb{R}^{n_x} \to \mathbb{R}^{n_c} \text{ und } c_{\max} \in \mathbb{R}^{n_c} \tag{4.6}$$

mit $f : \mathbb{R}^{n_x} \to \mathbb{R}$, $h : \mathbb{R}^{n_x} \to \mathbb{R}^{n_\zeta}$ und $g : \mathbb{R}^{n_x} \to \mathbb{R}^{n_c}$.

Dieses Optimalsteuerungsproblem ist äquivalent zu Problem 2.1. Die Dimensionen der diskretisierten Nebenbedingungen entsprechen

$$\begin{aligned} n_\zeta &= N \cdot n_y \\ n_c &= 2\left(n_\psi + (N+1) \cdot n_v + n_x\right). \end{aligned}$$

Für dieses Optimierungsproblem können wir die Lagrangefunktion

$$\mathcal{L}(x,\lambda) = \varphi(y(t_0), y(t_f), p) + \lambda_\zeta^\top \zeta(x) + \lambda_c^\top \left(c(x) - c_{\max}\right) \tag{4.7}$$

und auch deren Hessematrix

$$\nabla_{xx}^2 \mathcal{L} = \begin{bmatrix} W_{00} & & & & \Phi^\top & P_0^\top \\ & W_{11} & & & & P_1^\top \\ & & \ddots & & & \vdots \\ & & & W_{N-1N-1} & & P_{N-1}^\top \\ \Phi & & & & W_{NN} & P_N^\top \\ P_0 & P_1 & \cdots & P_{N-1} & P_N & P \end{bmatrix} \tag{4.8}$$

aufstellen. Diese Matrixstruktur ergibt sich explizit aus der Trapezregel-Diskretisierung. Aus der Lagrangefunktion (4.7) resultieren die Teilmatrizen

$$W_{ii} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial y_i^2} & \frac{\partial \mathcal{L}}{\partial y_i \partial u_i} \\ \frac{\partial \mathcal{L}}{\partial u_i \partial y_i} & \frac{\partial \mathcal{L}}{\partial u_i^2} \end{bmatrix} \qquad \Phi = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial y_0 \partial y_f} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$P_i = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial p \partial y_i} & \frac{\partial \mathcal{L}}{\partial p \partial u_i} \end{bmatrix} \qquad P = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial p^2} \end{bmatrix}.$$

Die Jacobimatrix der nichtlinearen Nebenbedingungen ist durch

$$G = \begin{bmatrix} \nabla \zeta(x)^\top \\ \nabla c(x)^\top \end{bmatrix} \in \mathbb{R}^{(n_\zeta + n_c) \times n_x} \tag{4.9}$$

definiert. Da die Jacobimatrix $G$ sehr dünn besetzt ist, sollte die Struktur der Matrix bei deren Berechnung verwendet werden. Die Jacobimatrix ergibt sich zu

$$\begin{bmatrix} \nabla \zeta(x)^\top \\ \nabla c(x)^\top \end{bmatrix} = \left[\begin{array}{cccc:c} M_0 & S_0 & & & R_0 \\ & \ddots & \ddots & & \vdots \\ & & M_{N-1} & S_{N-1} & R_{N-1} \\ \hdashline \Psi_0 & & & \Psi_N & R \\ C_0 & & & & Q_0 \\ & C_1 & & & Q_1 \\ & & \ddots & & \vdots \\ & & & C_N & Q_N \\ & & & & Q \end{array}\right]. \tag{4.10}$$

Die Teilmatrizen der Jacobimatrix ergeben sich aus

$$M_i = \begin{bmatrix} -\mathbf{I} - \frac{h}{2}\frac{\partial F_i}{\partial y_i} & -\frac{h}{2}\frac{\partial F_i}{\partial u_i} \end{bmatrix} \quad S_i = \begin{bmatrix} \mathbf{I} - \frac{h}{2}\frac{\partial F_{i+1}}{\partial y_{i+1}} & -\frac{h}{2}\frac{\partial F_{i+1}}{\partial u_{i+1}} \end{bmatrix} \quad R_i = \begin{bmatrix} -\frac{h}{2}\left(\frac{\partial F_i}{\partial p} + \frac{\partial F_{i+1}}{\partial p}\right) \end{bmatrix}$$

$$\Psi_0 = \begin{bmatrix} \frac{\partial \psi}{\partial y_0} & \mathbf{0} \\ -\frac{\partial \psi}{\partial y_0} & \mathbf{0} \end{bmatrix} \quad \Psi_N = \begin{bmatrix} \frac{\partial \psi}{\partial y_N} & \mathbf{0} \\ -\frac{\partial \psi}{\partial y_N} & \mathbf{0} \end{bmatrix} \quad R = \begin{bmatrix} \frac{\partial \psi}{\partial p} \\ -\frac{\partial \psi}{\partial p} \end{bmatrix}$$

$$C_i = \begin{bmatrix} \frac{\partial F_i}{\partial y_i} & \frac{\partial F_i}{\partial u_i} \\ -\frac{\partial F_i}{\partial y_i} & -\frac{\partial F_i}{\partial u_i} \\ \mathbf{I} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \\ \mathbf{0} & -\mathbf{I} \end{bmatrix} \quad Q_i = \begin{bmatrix} \frac{\partial F_i}{\partial p} \\ -\frac{\partial F_i}{\partial p} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \quad Q = \begin{bmatrix} \mathbf{I} \\ -\mathbf{I} \end{bmatrix}.$$

## 4.2 Innere-Punkte-Verfahren

Innere-Punkte-Verfahren sind einer der am weit verbreitetsten Algorithmen in der restringierten Optimierung. Das Innere-Punkte-Verfahren ist neben dem SQP-Verfahren aus Abschnitt 2.2.1 ein weiterer Algorithmus um Ungleichungsnebenbedingungen in restringierten Optimierungsproblemen, wie Problem 2.1 zu behandeln. Die Verbreitung des Innere-Punkte-Algorithmus liegt überwiegend an der Vielseitigkeit des Verfahrens, das neben linearen und quadratischen auch nichtlineare Optimierungsprobleme lösen kann. Das heißt die numerische Berechnung eines Minimums eines nichtlinearen restringierten Optimierungsproblems muss mit dem Innere-Punkte-Verfahren nicht zwangsläufig, wie beim SQP-Verfahren über die Berechnung quadratischer Teilprobleme erfolgen. Eine der am meisten verwendeten Software-Bibliotheken für Innere-Punkte-Verfahren ist Ipopt aus [22], [70] und [71]. In diesem Kapitel wird in eine Strukturausnutzung unter Verwendung eines Innere-Punkte-Verfahrens durchgeführt. Dazu musste eine eigene Implementierung eines Innere-Punkte-Verfahrens erfolgen. Diese wurde in Form einer Software-Bibliothek mit Namen IPbasic implementiert. Zur besseren Vergleichbarkeit orientiert sich der im Folgenden beschriebene Algorithmus von IPbasic an der Implementierung von Ipopt.

### 4.2.1 Problemstellung

Das Optimierungsproblem aus Problem 2.1, welches äquivalent zu Problem 4.3 ist, wird bei der Verwendung eines Innere-Punkte-Verfahrens in eine Folge von Barriereproblemen zerlegt. Wir definieren das Barriereproblem:

**Problem 4.4 (Barriereproblem des Innere-Punkte-Verfahrens, vgl. [22])**

Bestimme den Vektor der Optimierungsvariablen $x \in \mathbb{R}^n$ und den Vektor der Schlupfvariablen $s = (s^{(1)}, \ldots, s^{(q)})^\top \in \mathbb{R}^q_+$, so dass die Zielfunktion

$$f(x) - \mu \sum_{i=1}^{q} \ln\left(s^{(i)}\right) \tag{4.11}$$

minimal wird, unter den Nebenbedingungen

$$h(x) = 0 \tag{4.12}$$

$$g(x) + s = 0, \tag{4.13}$$

mit $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^p$ und $g : \mathbb{R}^n \to \mathbb{R}^q$. Der skalare Faktor $\mu > 0$ sei dabei ein Element einer Nullfolge $\{\mu_j\}$ mit $\mu_j \in \mathbb{R}$.

Das Barriereproblem eliminiert die Ungleichungsnebenbedingungen aus Problem 2.1 durch die Einführung von Schlupfvariablen. Durch die Nullfolge $\{\mu_j\}$ wird mit dem Innere-Punkte-Verfahren eine Folge von Barriereproblem gelöst.

### 4.2.2 Bestimmung der Suchrichtung

Das Barriereproblem aus Problem 4.11 führt unter der Voraussetzung der Erfüllung der LICQ aus Definition 2.3 auf eine Lagrangefunktion der Art

$$\mathcal{L}\left(x, s, \lambda_h, \lambda_g; \mu\right) = f(x) - \mu \sum_{i=1}^{q} \left[\ln(s^{(i)})\right] + \lambda_h^\top h(x) + \lambda_g^\top \left(g(x) + s\right). \tag{4.14}$$

Ausgehend von der Lagrangefunktion (4.14) und einer Erfüllung der LICQ können die KKT-Bedingungen des Barriereproblems

$$\nabla f(x) + \nabla h(x)\lambda_h + \nabla g(x)\lambda_g = 0 \tag{4.15}$$

$$-\mu e + S\lambda_g = 0 \tag{4.16}$$

$$h(x) = 0 \tag{4.17}$$

$$g(x) + s = 0 \tag{4.18}$$

bestimmt werden.

**Bemerkung 4.1.** *Für $\mu \Rightarrow 0$ erhalten wir die ursprünglichen KKT-Bedingungen aus Satz 2.2.*

Die Gleichungen (4.15) und (4.16) ergeben sich durch das Differenzieren der Lagrangefunktion (4.14) nach $x$ und $s$, wobei Gleichung (4.16) mit $S$ multipliziert wurde. Dabei entspricht $S$ einer Diagonalmatrix mit dem Vektor $s$ auf der Hauptdiagonalen und der Vektor $e = [1, \ldots, 1]^\top \in \mathbb{R}^q$. Die Gleichungen (4.17) und (4.18) entsprechen den Nebenbedingungen (4.12) und (4.13). Zur Bestimmung der Nullstellen der KKT-Bedingungen definieren wir ein Newton-Verfahren

$$\begin{bmatrix} H_k & \widetilde{A}_k^\top \\ A_k & 0 \end{bmatrix} \begin{bmatrix} d_x \\ d_s \\ d_\lambda \end{bmatrix} = - \begin{bmatrix} \gamma_k + B_k \lambda_k \\ C_k \end{bmatrix} \tag{4.19}$$

für den Iterationsschritt $k+1$. Die linke Seite von Gleichung (4.19) entsteht durch eine weitere Differenzierung der KKT-Bedingungen des Barriereproblems nach dem Vektor $z = [x, s, \lambda_h, \lambda_g]^\top$. Die Teilmatrix

$$H_k = \begin{bmatrix} \nabla^2_{xx} \mathcal{L} & 0 \\ 0 & \Lambda_g \end{bmatrix}_k \tag{4.20}$$

entspricht der Hessematix der Lagrangefunktion (4.14) im Iterationsschritt $k$ mit $\nabla^2_{xx}\mathcal{L} = \nabla^2_{xx}\mathcal{L}\,(x, s, \lambda_h, \lambda_g; \mu)$. Die Teilmatrix $\Lambda_g$ entspricht einer Diagonalmatrix mit dem Vektor $\lambda_g$ auf den Hauptdiagonalen. Die Teilmatrix $A_k$ der linken Seite von Gleichung (4.19) ergibt sich durch die Differenzierung der Nebenbedingungen (4.17) und (4.18) und die Teilmatrix $\widetilde{A}_k^\top$ aus der Differenzierung der Gleichungen (4.15) und (4.16) nach $\lambda_g$. Das heißt:

$$A_k = \begin{bmatrix} \nabla h(x)^\top & 0 \\ \nabla g(x)^\top & I \end{bmatrix}_k \qquad \text{und} \qquad \widetilde{A}_k = \begin{bmatrix} \nabla h(x)^\top & 0 \\ \nabla g(x)^\top & S \end{bmatrix}_k . \tag{4.21}$$

**Bemerkung 4.2.** *Die Matrix des linearen Gleichungssystems (4.19) ist aufgrund von Gleichung (4.21) nicht symmetrisch.*

Zur Bestimmung der rechten Seite des Newton-Verfahrens (4.19) müssen wir lediglich die KKT-Bedingungen des Barriereproblems im Iterationsschritt $k$ auswerten. Das heißt $\gamma_k = \left[\nabla f(x)^\top, -\mu e^\top\right]_k^\top$, $\lambda_k = \left[\lambda_h^\top, \lambda_g^\top\right]_k^\top$ und

$$B_k = \begin{bmatrix} \nabla h(x) & \nabla g(x) \\ 0 & S \end{bmatrix}_k , \qquad C_k = \begin{bmatrix} h(x) \\ g(x) + s \end{bmatrix}_k .$$

### 4.2.3 Reduktion der Dimensionen des linearen Gleichungssystems

Durch die Gleichungen (4.19), (4.20) und (4.21) erhalten wir ein lineares Gleichungssystem in der Form

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L} & 0 & \nabla h(x) & \nabla g(x) \\ 0 & \Lambda_g & 0 & S \\ \nabla h(x)^\top & 0 & 0 & 0 \\ \nabla g(x)^\top & I & 0 & 0 \end{bmatrix}_k \begin{bmatrix} d_x \\ d_s \\ d_{\lambda_h} \\ d_{\lambda_g} \end{bmatrix} = - \begin{bmatrix} \nabla_x \mathcal{L}\left(x, s, \lambda_h, \lambda_g; \mu\right) \\ -\mu e + S\lambda_g \\ h(x) \\ g(x) + s \end{bmatrix}_k . \tag{4.22}$$

Der vierte Zeilenblock von (4.22) ergibt zusammengefasst

$$\nabla g(x_k)^\top d_x + d_s = - \left(g(x_k) + s_k\right).$$

Diese Gleichung lässt sich nach

$$d_s = -\nabla g(x_k)^\top d_x - \left(g(x_k) + s_k\right) \tag{4.23}$$

auflösen. In den übrigen Zeilenblöcken der Matrix des linearen Gleichungssystems (4.22) wird der Suchrichtungsanteil $d_s$ lediglich mit einer Diagonalmatrix der Lagrangemultiplikatoren multipliziert. Da die Gleichung (4.23) sowohl konstante als auch von $d_x$ abhängige Anteile besitzt verändern sich sowohl die Matrix, als auch die rechte Seite des linearen Gleichungssystems (4.22), wenn wir $d_s$ durch Gleichung (4.23) ersetzen. Es entsteht das lineare Gleichungssystem

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L} & \nabla h(x) & \nabla g(x) \\ \nabla h(x)^T & 0 & 0 \\ -\Lambda_g \nabla g(x)^T & 0 & S \end{bmatrix}_k \begin{bmatrix} d_x \\ d_{\lambda_h} \\ d_{\lambda_g} \end{bmatrix} = - \begin{bmatrix} \nabla_x \mathcal{L}\left(x, s, \lambda_h, \lambda_g; \mu\right) \\ h(x) \\ -\mu - \Lambda_g g(x) \end{bmatrix}_k .$$

Nach dem wir den dritten Zeilenblock mit $-\Lambda_g^{-1}$ multiplizieren, ergibt sich das reduzierte lineare Gleichungssystem

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L} & \nabla h(x) & \nabla g(x) \\ \nabla h(x)^T & 0 & 0 \\ \nabla g(x)^T & 0 & -\Lambda_g^{-1} S \end{bmatrix}_k \begin{bmatrix} d_x \\ d_{\lambda_h} \\ d_{\lambda_g} \end{bmatrix} = - \begin{bmatrix} \nabla_x \mathcal{L}\left(x, s, \lambda_h, \lambda_g; \mu\right) \\ h(x) \\ \Lambda_g^{-1}\mu e + g(x) \end{bmatrix}_k . \tag{4.24}$$

Die Bestimmung der inversen Matrix $\Lambda_g^{-1}$ ist aufgrund der diagonalen Struktur dieser Matrix trivial.

**Bemerkung 4.3.** *Ein weiterer Vorteil des reduzierten linearen Gleichungssystems (4.24) besteht darin, dass die resultierende Matrix im Gegensatz zur Matrix aus Gleichung (4.22) im Allgemeinen symmetrisch ist.*

### 4.2.4 Update Formeln

Ist die Suchrichtung eines Iterationsschritts bestimmt, so können wir die neuen Itereierten analog zu [22] durch

$$x_{k+1} = x_k + \alpha_k d_{x_k} \tag{4.25}$$

$$s_{k+1} = s_k + \alpha_k d_{s_k} \tag{4.26}$$

$$\lambda_{h,k+1} = \lambda_{h,k} + \alpha_k d_{\lambda_h,k} \tag{4.27}$$

$$\lambda_{g,k+1} = \lambda_{g,k} + \alpha_{g,k} d_{\lambda_g,k} \tag{4.28}$$

bestimmen. Im Anschuss ermitteln wir einen für große Lagrangemultipikatoren skalierten Fehler in den KKT-Bedingungen des Barriereproblems nach [22]

$$\Delta_\mu(x,s,\lambda) := \max\left\{\frac{\|\nabla f(x) + \nabla c(x)\lambda\|_\infty}{\delta_d}, \|c(x)\|_\infty, \frac{-\mu e + S\lambda_g}{\delta_c}\right\}, \tag{4.29}$$

mit

$$\delta_d = \max\left\{\delta_{\max}, \frac{\|\lambda\|}{N_c}\right\}/\delta_{\max} \qquad \delta_c = \max\left\{\delta_{\max}, \frac{\|\lambda_g\|}{N_g}\right\}/\delta_{\max}$$

und

$$c(x) = \begin{bmatrix} h(x) \\ g(x) + s \end{bmatrix}.$$

Als Abbildungsvorschrift der Barriereparameter $\mu$ verwenden wir nach dem Vorbild von IPOPT

$$\mu_{k+1} = \max\left(\mu_{\text{tol}}, \min\left(\kappa_\mu \mu_k, \mu_k^{\theta_\mu}\right)\right),$$

mit $\theta_\mu = \frac{3}{2}$ und $\kappa_\mu = 0{,}9$. Die Toleranz $\mu_{\text{tol}}$ beschreibt den minimalen Barriereparameter und kann frei gewählt werden.

### 4.2.5 Schrittweitensteuerung

#### Maximalbereich der Schrittweiten

Eine zentrale Rolle für die Konvergenzeigenschaften des Innere-Punkte-Verfahrens spielt die Schrittweitensteuerung der Schrittweiten $\alpha_k$ und $\alpha_{g,k}$. Zulässige Maximalwerte für die Schrittweiten können vergleichsweise einfach analog zu [22]

$$\alpha_k^{\max} = \max\{\alpha \in (0,1] : s_k + \alpha d_{s_k} \geq (1-\tau_k)s_k\} \tag{4.30}$$

$$\alpha_{g,k}^{\max} = \max\{\alpha \in (0,1] : \lambda_{g,k} + \alpha d_{\lambda_{g,k}} \geq (1-\tau_k)\lambda_{g,k}\} \tag{4.31}$$

gefunden werden. Um sicherzustellen, dass weder die Schlupfvariablen noch die Lagrangemultiplikatoren der Ungleichungen zu schnell an die Ränder des zulässigen Gebiets konvergieren wird ein Abstandsparameter

$$\tau_k = \max\{\tau_{\min}, 1 - \mu_k\}$$

festgelegt. Nach [22] kann für die Schrittweite der Lagrangemultiplikatoren der Ungleichungsnebenbedingungen bereits die maximale Schrittweite $\alpha_{g,k} = \alpha_{g,k}^{\max}$ verwendet werden.

### Armijoregel mit $\ell_1$ Penaltyfunktion

Die Schrittweitensteuerung des Innere-Punkte-Verfahrens erfolgt analog zur Globalisierung des SQP-Verfahrens aus Abschnitt 2.2.3 durch eine Armijo Liniensuche. Um die Schrittweite $\alpha_k$ durch die Armijoregel

$$\ell(x_k + \alpha_k d_{x_k}, s_k + \alpha_k d_{s_k}) \leq \ell(x_k, s_k) + \alpha_k \cdot \sigma \cdot \ell'(x_k, s_k; d_{x_k}, d_{s_k}) \tag{4.32}$$

und den Algorithmus der Armijo Liniensuche aus Abschnitt 2.2.3 bestimmen zu können, müssen wir eine passende Bewertungsfunktion $\ell(x_k, s_k)$ und deren Richtungsableitung $\ell'(x_k, s_k; d_{x_k}, d_{s_k})$ in die Richtung $[d_{x_k}, d_{s_k}]^\top$ wählen. Eine mögliche Bewertungsfunktion für das Innere-Punkte-Verfahren beschreibt, analog zum SQP-Verfahren, die $\ell_1$-Penaltyfunktion

$$\ell_1(x, s; \eta, \xi) := f(x) - \mu \sum_{i=1}^{q} \ln\left(s^{(i)}\right) + \sum_{i=1}^{q} \eta_i \left|g_i(x) + s^{(i)}\right| + \sum_{j=1}^{p} \xi_j \left|h_j(x)\right|. \tag{4.33}$$

**Bemerkung 4.4.** *Die $\ell_1$-Penaltyfunktion des Innere-Punkte-Verfahrens aus Gleichung (4.33) ist aufgrund der Betragsfunktion nicht differenzierbar, aber dennoch richtungsdifferenzierbar.*

Die Richtungsableitung der $\ell_1$-Penaltyfunktion lässt sich mit Hilfe einer Fallunterschei-

dung des Betrags als

$$\begin{aligned}\ell_1'\left(x,s;d_x,d_s;\eta,\xi\right) =& \nabla_x f(x)^\top d_x - \mu\sum_{i=1}^{q}\frac{d_{s^{(i)}}}{s^{(i)}} + \sum_{i:\ g_i(x)+s^{(i)}>0}\eta_i\left(\nabla g_i\left(x\right)^\top d_x + d_{s^{(i)}}\right) \ + \\ & \sum_{i:\ g_i(x)+s^{(i)}=0}\eta_i\left|\nabla g_i\left(x\right)^\top d_x + d_{s^{(i)}}\right| \ - \\ & \sum_{i:\ g_i(x)+s^{(i)}<0}\eta_i\left(\nabla g_i\left(x\right)^\top d_x + d_{s^{(i)}}\right) \ + \sum_{j:\ h_j(x)>0}\xi_j\nabla h_j(x)^\top d_x \ + \\ & \sum_{j:\ h_j(x)=0}\xi_j\left|\nabla h_j(x)^\top d_x\right| \ - \sum_{j:\ h_j(x)<0}\xi_j\nabla h_j(x)^\top d_x\end{aligned} \tag{4.34}$$

für $i = 1, \ldots, q$ und $j = 1, \ldots, p$ darstellen.
Wie bereits beim SQP-Verfahren benötigen wir eine Abstiegseigenschaft der $\ell_1$-Penaltyfunktion entlang der Suchrichtung des Innere-Punkte-Verfahrens.

**Satz 4.1 (Abstiegseigenschaft der $\ell_1$-Penaltyfunktion für das Innere-Punkte-Verfahren)**

Sei $d := [d_x^\top, d_s^\top, d_{\lambda_h}^\top, d_{\lambda_g}^\top]^\top \neq 0$ eine durch (4.22) berechnete Suchrichtung des Innere-Punkte-Verfahrens. Es gilt unter den Bedingungen

$$\begin{aligned}\eta_i &> \left|\lambda_g^{(i)} + d_{\lambda_g^{(i)}}\right| && \forall\ i = 1, \ldots, q\\ \xi_j &> \left|\lambda_h^{(j)} + d_{\lambda_h^{(j)}}\right| && \forall\ j = 1, \ldots, p\\ s_k^{(i)} &\geq 0 && \forall\ i = 1, \ldots, q\\ \lambda_{g,k}^{(i)} &\geq 0 && \forall\ i = 1, \ldots, q,\end{aligned}$$

dass sich die Richtungsableitung der $\ell_1$-Penaltyfunktion für eine positiv definite Hessematrix der Lagrangefunktion $\mathcal{L}_{xx}(x_k, s_k, \lambda_{h,k}, \lambda_{g,k}; \mu)$ durch

$$\ell_1'\left(x,s;d_x,d_s;\eta,\xi\right) < 0$$

abschätzen lässt.

Der Beweis von Satz 4.1 erfolgt analog zu [43]. Der wesentliche Unterschied ergibt sich dadurch, dass die Grundvoraussetzungen für den Beweis statt durch die erfüllten KKT-Bedingungen des quadratischen Teilproblems über das lineare Gleichungssystem (4.22) bestimmt werden.

***Beweis:*** Zur vereinfachten Schreibweise verwenden wir in diesem Beweis $f := f(x_k)$, $g := g(x_k)$, $h := h(x_k)$, $\mathcal{L}_{xx} := \mathcal{L}_{xx}(x_k, s_k, \lambda_{h,k}, \lambda_{g,k}; \mu)$ und $\ell_1' := \ell_1'(x, s; d_x, d_s; \eta, \xi)$. Sei $d := [d_x^\top, d_s^\top, d_{\lambda_h}^\top, d_{\lambda_g}^\top]^\top \neq 0$ eine Lösung von (4.22) für $x := x_k$, $s := s_k$, $\lambda_h := \lambda_{h,k}$ und $\lambda_g := \lambda_{g,k}$, dann gilt:

$$\begin{aligned}
\mathcal{L}_{xx} d_x + \nabla h d_{\lambda_h} + \nabla g d_{\lambda_g} &= -\nabla f - \nabla h \lambda_h - \nabla g \lambda_g \\
\Lambda_g d_s + S d_{\lambda_g} &= \mu e - S \lambda_g \\
\nabla h^\top d_x &= -h \\
\nabla g^\top d_x + d_s &= -g - s.
\end{aligned}$$

Umgeformt ergeben sich die Gleichungen

$$\nabla f^\top d_x = d_x^\top \nabla f = -d_x^\top \mathcal{L}_{xx} d_x - d_x^\top \nabla h \left(\lambda_h + d_{\lambda_h}\right) - d_x^\top \nabla g \left(\lambda_g + d_{\lambda_g}\right) \tag{4.35}$$

$$S^{-1} \mu e = S^{-1} \Lambda_g d_s + \lambda_g + d_{\lambda_g} \tag{4.36}$$

$$\nabla h^\top d_x = -h \tag{4.37}$$

$$d_x^\top \nabla h = -h^\top \tag{4.38}$$

$$\nabla g^\top d_x = -g - s - d_s \tag{4.39}$$

$$d_x^\top \nabla g = -g^\top - s^\top - d_s^\top. \tag{4.40}$$

Wir setzten Gleichung (4.35), unter der Verwendung von (4.36), (4.37), (4.38), (4.39) und (4.40), in die Richtungsableitung der $\ell_1$-Penaltyfunktion (4.34) ein und erhalten

$$\begin{aligned}
\ell_1' = &- d_x^\top \mathcal{L}_{xx} d_x \; + \sum_{j=1}^{p} h_j \left(\lambda_h^{(j)} + d_{\lambda_h^{(j)}}\right) \; + \\
&\sum_{i=1}^{q} \left(g_i + s^{(i)} + d_{s^{(i)}}\right) \left(\lambda_g^{(i)} + d_{\lambda_g^{(i)}}\right) \; - \\
&\sum_{i=1}^{q} \left[\frac{d_{s^{(i)}}^2 \lambda_g^{(i)}}{s^{(i)}} + \left(\lambda_g^{(i)} + d_{\lambda_g^{(i)}}\right) d_{s^{(i)}}\right] \; - \sum_{i:\; g_i + s^{(i)} > 0} \eta_i \left(g_i + s^{(i)}\right) \; + \\
&\sum_{i:\; g_i + s^{(i)} < 0} \eta_i \left(g_i + s^{(i)}\right) \; - \sum_{j:\; h_j > 0} \xi_j h_j \; + \sum_{j:\; h_j < 0} \xi_j h_j.
\end{aligned}$$

Dabei ist zu beachten, dass die beiden Summen über die Beträge mit $j : h_j = 0$ bzw. $i : g_i + s^{(i)} = 0$ aus (4.34) unter der Verwendung von (4.37) und (4.39) eliminiert werden können. Bei den Summen über $i = 1, \ldots, q$ hebt sich der Term $\sum_{i=1}^{q} \left(\lambda_g^{(i)} + d_{\lambda_g^{(i)}}\right) d_{s^{(i)}}$ auf. Unter der Berücksichtigung, dass $s^{(i)} \geq 0$ und $\lambda_g^{(i)} \geq 0$ für alle $i = 1, \ldots, q$, gilt

$$\sum_{i=1}^{q} \frac{d_{s^{(i)}}^2 \lambda_g^{(i)}}{s^{(i)}} \geq 0.$$

Damit lässt sich die Richtungsableitung der $\ell_1$-Penaltyfunktion wie folgt abschätzen

$$\begin{aligned}\ell_1' \leq & - d_x^\top \mathcal{L}_{xx} d_x \; + \sum_{i:\; g_i+s^{(i)}>0} \left(\lambda_g^{(i)} + d_{\lambda_g^{(i)}} - \eta_i\right)\left(g_i + s^{(i)}\right) \; + \\ & \sum_{i:\; g_i+s^{(i)}<0} \left(\lambda_g^{(i)} + d_{\lambda_g^{(i)}} + \eta_i\right)\left(g_i + s^{(i)}\right) \; + \\ & \sum_{j:\; h_j>0} \left(\lambda_h^{(j)} + d_{\lambda_h^{(j)}} - \xi_j\right) h_j \; + \sum_{j:\; h_j<0} \left(\lambda_h^{(j)} + d_{\lambda_h^{(j)}} + \xi_j\right) h_j.\end{aligned}$$

Unter den Bedingungen

$$\begin{aligned}\eta_i &> \left|\lambda_g^{(i)} + d_{\lambda_g^{(i)}}\right| && \forall\; i = 1, \ldots, q \\ \xi_j &> \left|\lambda_h^{(j)} + d_{\lambda_h^{(j)}}\right| && \forall\; j = 1, \ldots, p,\end{aligned}$$

aus Satz 4.1, gilt für eine positiv definite Hessematrix der Lagrangefunktion

$$\ell_1' < -d_x^\top \mathcal{L}_{xx} d_x < 0.$$

□

**Kopplungsstrategien der Schrittweitensteuerung**

Die im Rahmen dieser Arbeit entwickelte Softwarebibliothek IPBASIC besitzt die Möglichkeit, verschiedene Kopplungsstrategien für die Schrittweite zu verwenden. In Abschnitt 4.2.4 sind die Updateformeln

$$\begin{aligned}x_{k+1} &= x_k + \alpha_k d_{x_k} \\ s_{k+1} &= s_k + \alpha_k d_{s_k} \\ \lambda_{h,k+1} &= \lambda_{h,k} + \alpha_k d_{\lambda_h,k} \\ \lambda_{g,k+1} &= \lambda_{g,k} + \alpha_{g,k} d_{\lambda_g,k}\end{aligned}$$

gegeben, welche in IPOPT und im Rahmen der Beispiele dieser Arbeit verwendet werden. Hierbei sind die Schrittweiten der Zustände und Schlupfvariablen gekoppelt. Die Kopplung entsteht dadurch, dass die maximale Schrittweite vor Beginn der Armijo Liniensuche durch Formel (4.30) beschränkt wird. Anschließend führen wir mit Hilfe der $\ell_1$-Penaltyfunktion die Armijo Liniensuche durch und erhalten die gekoppelte Schrittweite $\alpha_k$. Die Schrittweite der Lagrangemultiplikatoren $\alpha_{g,k}$ ergibt sich ohne Kopplung durch Formel (4.31).
IPBASIC besitzt die Kopplungsstrategien:

- Kopplung von Zuständen und Schlupfvariablen: Wahl der Schrittweite der Lagrangemultiplikatoren durch Formel (4.31).
- Kopplung von Zuständen, Schlupfvariablen und Lagrangemultiplikatoren: Wahl der maximale Schrittweite vor Beginn der Armijo Liniensuche durch das Minimum von $\alpha_{s,k}^{\max}$ und $\alpha_{g,k}^{\max}$ aus (4.30) und (4.31).
- Keine Kopplung der Schrittweiten: Wahl der Schrittweite der Zustände durch Armijo Liniensuche mit maximaler Schrittweite $\alpha^{\max} = 1$. Wahl der Schrittweiten der Schlupfvariablen und Lagrangemultiplikatoren durch die Formeln (4.30) und (4.31).

In der Praxis erweist sich die Kopplung von Zustände und Schlupfvariablen als besonders robust. Die Konvergenz des Innere-Punkte-Verfahrens ist sehr stark von der Startschätzung für die Zustände, Schlupfvariablen und Lagrangemultiplikatoren abhängig. Für schlechte Startschätzungen kann es schnell passieren, dass die Armijo Liniensuche gegen die minimale Schrittweite läuft. Da insbesondere die Schätzung der Lagrangemultiplikatoren eine Herausforderung darstellt, ist es praktikabel diese bei der Schrittweitensteuerung zu entkoppeln. So ist es möglich in der Armijo Liniensuche die minimale Schrittweite zu akzeptieren und dennoch Fortschritt in den Lagrangemultiplikatoren zu erhalten. Ist die Schrittweite der Lagrangemultiplikatoren in einem solchen Fall gekoppelt, führt dies in der Regel zu einer Stagnation des Innere-Punkte-Verfahrens.

### 4.2.6 Ablauf des Innere-Punkte-Verfahrens

Nachdem wir in den Abschnitten 4.2.1, 4.2.2, 4.2.3, 4.2.4 und 4.2.5 die einzelnen Lösungsschritte des Innere-Punkte-Verfahrens beschrieben haben, ergibt sich zusammengefasst für das Innere-Punkte-Verfahren folgender Algorithmus.

**Algorithmus Innere-Punkte-Verfahren**

i Bestimme Startschätzung $(x_0, s_0, \lambda_{h,0}, \lambda_{g,0})^\top$ und setze $k = 0$ und $\mu = \mu_0 > 0$.

ii Falls $\Delta_\mu(x_k, s_k, \lambda_{h,k}, \lambda_{g,k}) < \varepsilon_{\text{tol}}$ aus (4.29) und $\mu \leq \mu_{\text{tol}}$: Beende das Verfahren.

iii Berechne die Lösung $\left(d_x, d_{\lambda_h}, d_{\lambda_g}\right)^\top$ des linearen Gleichungssystems aus (4.24) und bestimme durch Formel (4.23) die Suchrichtung der Schlupfvariablen $d_s$.

iv Bestimme mit der Armijoregel als Schrittweitensteuerung und den Formeln (4.30) und (4.31) die Schrittweiten $\alpha_k$ und $\alpha_{g,k}$.

v Berechne die neuen Iterierten $(x_{k+1}, s_{k+1}, \lambda_{h,k}, \lambda_{g,k})^\top$ durch die Formeln (4.25), (4.26), (4.27) und (4.28). Dann setze $k := k + 1$ und gehe zurück zu ii.

### 4.2.7 Beispiel für das Innere-Punkte-Verfahren

Die Funktionsweise des Innere-Punkte-Verfahrens wollen wir zunächst anhand eines einfachen Beispiels zeigen. Dazu verwenden wir ein restringiertes nichtlineares Optimierungsproblem, das in der zweidimensionalen xy-Ebene dargestellt werden kann.

**Beispiel 4.1.**

$$\min f(x, y) = x^2 + y^2$$

*unter den Nebenbedingungen*

$$\begin{aligned} g_1(x, y) &= 1{,}5 - y \leq 0 \\ g_2(x, y) &= e^x - y \leq 0. \end{aligned}$$

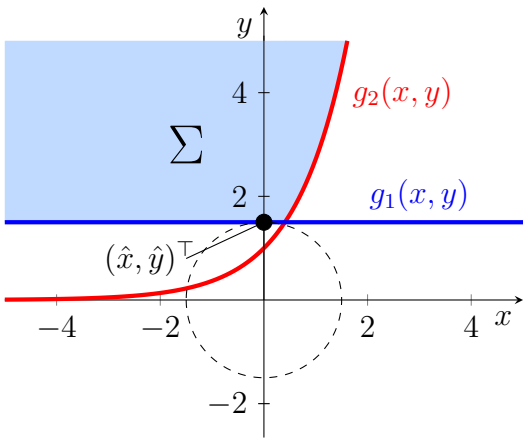


Abbildung 4.1: Skizze zu Beispiel 4.1

In Beispiel 4.1 ist ein einfaches nichtlineares restringiertes Optimierungsproblem definiert. In Worten formuliert wird in Beispiel 4.1 ein Punkt im zulässigen Bereich $\Sigma$ gesucht, der den Abstand zum Ursprung $(0, 0)^\top$ minimiert. In Abbildung 4.1 wird dieses Optimum veranschaulicht. Durch den kleinsten Kreis um den Ursprung, der den zulässigen Bereich $\Sigma$ tangiert, können wir das Minimum im Punkt $(\hat{x}, \hat{y})^\top = (0, 1{,}5)^\top$ ermitteln.

Für die numerische Lösung von Beispiel 4.1 können wir die für das Innere-Punkte-Verfahren benötigten Ableitungen

$$\nabla f(x,y) = \begin{bmatrix} 2x \\ 2y \end{bmatrix}, \qquad \nabla g(x,y) = \begin{bmatrix} 0 & e^x \\ -1 & -1 \end{bmatrix},$$
$$\nabla^2 f(x,y) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \qquad \lambda^\top \nabla^2 g(x,y) = \begin{bmatrix} \lambda_2 e^x & 0 \\ 0 & 0 \end{bmatrix}$$

bestimmen. Damit ergibt sich das Gleichungssystem zur Bestimmung der Suchrichtung aus Formel (4.22) zu

$$\begin{bmatrix} 2+\lambda_2 e^x & 0 & 0 & 0 & 0 & e^x \\ 0 & 2 & 0 & 0 & -1 & -1 \\ 0 & 0 & \lambda_1 & 0 & s_1 & 0 \\ 0 & 0 & 0 & \lambda_2 & 0 & s_2 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ e^x & -1 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_x \\ d_y \\ d_{s_1} \\ d_{s_2} \\ d_{\lambda_1} \\ d_{\lambda_2} \end{bmatrix} = - \begin{bmatrix} 2x + \lambda_2 e^x \\ 2y - \lambda_1 - \lambda_2 \\ s_1\lambda_1 - \mu \\ s_2\lambda_2 - \mu \\ 1{,}5 - y + s_1 \\ e^x - y + s_2 \end{bmatrix}.$$

**Bemerkung 4.5.** *Für dieses kleine Testbeispiel kann das lineare Gleichungssystem der Suchrichtungsbestimmung auch in nicht reduzierter Form verwendet werden.*

| | $\mu$ | x | y | KKT-Error | $\mu$ | x | y | KKT-Error |
|---|---|---|---|---|---|---|---|---|
| 0 | – | 0,6 | 2 | 3,6383 | – | 0,6 | 2 | 3,6383 |
| 1 | 0,5 | 0,38179 | 1,7601 | 1,7404 | 1e-08 | 0,34953 | 1,5124 | 1,4516 |
| 2 | 0,05 | 0,16896 | 1,5334 | 0,41614 | 1e-08 | 0,27833 | 1,5008 | 0,76856 |
| 3 | 0,005 | 0,00103 | 1,5002 | 0,01753 | 1e-08 | 0,08477 | 1,5 | 0,35159 |
| 4 | 1e-08 | 6,17e-05 | 1,5 | 0,00160 | 1e-08 | 0,01113 | 1,5 | 0,17447 |
| 5 | 1e-08 | -1,54e-07 | 1,5 | 1,47e-07 | 1e-08 | 0,00123 | 1,5 | 0,15266 |
| 6 | 1e-08 | -1e-08 | 1,5 | 6,11e-14 | 1e-08 | 0,00015 | 1,5 | 0,15035 |
| 7 | – | – | – | – | 1e-08 | 2,04e-05 | 1,5 | 0,15005 |
| 8 | – | – | – | – | 1e-08 | 2,66e-06 | 1,5 | 0,15001 |
| 9 | – | – | – | – | 1e-08 | 3,38e-07 | 1,5 | 0,15 |
| 10 | – | – | – | – | 1e-08 | 3,53e-08 | 1,5 | 0,15 |
| 11 | – | – | – | – | 1e-08 | -4,09e-09 | 1,5 | 5,58e-09 |

Tabelle 4.1: Ergebnisse zu Beispiel 4.1: Links absteigendem $\mu$. Rechts mit $\mu = 10^{-8}$.

Beide Ergebnisreihen aus Tabelle 4.1 wurden mit der gleichen Armijo-Schrittweitensuche unter Verwendung der $\ell_1$ Penaltyfunktion aus Abschnitt 4.2.5 berechnet. Als Startschätzung für die Schlupfvariablen wurden die Nebenbedingungen für den Startzustand ausgewertet. Für die Lagrange-Multiplikatoren verwenden wir die Startwerte $\lambda_1 = \lambda_2 = 1$. Die

Wahl des Barriereparameters $\mu$ erfolgte für dieses Beispiel frei ohne Updateformel. Wir erkennen anhand von Tabelle 4.1, den Einfluss der Wahl der Nullfolge des Barriereparameters $\mu$. Durch die schrittweise Verkleinerung des Barriereparameters in der linken Tabellenhälfte wird vermieden, dass die Iterationspunkte zu stark gegen die Beschränkungen des Problems laufen.

## 4.3 Strukturausnutzung im Innere-Punkte-Verfahren

Nachdem wir mit Problem 4.3 ein Optimalsteuerungsproblem in ein allgemeines restringiertes nichtlineares Optimierungsproblem in der Form von Problem 2.1 überführt haben, können wir darauf das Innere-Punkte-Verfahren anwenden. Im Innere-Punkte-Verfahren betrachten wir im speziellen die Struktur des linearen Gleichungssystems aus der Bestimmung der Suchrichtung.

### 4.3.1 Umsortierung des linearen Gleichungssystems im Innere-Punkte-Verfahren

Unser Ziel in diesem Abschnitt ist es, das im Innere-Punkte-Verfahren zu lösende lineare Gleichungssystem (4.24) in eine möglichst einer Bandmatrix ähnliche Struktur zu bringen. Im Allgemeinen lässt sich die Matrix nicht in eine Bandmatrix umformen, was an den Parameter- und Randbedingungsanteilen liegt. Mit der durch die Trapezregel-Diskretisierung definierten Hessematrix der Lagrangefunktion (4.8) und der Jacobimatrix (4.10) können wir das lineare Gleichungssystem (4.24), welches im Innere-Punkte-Verfah-

ren zur Bestimmung der Suchrichtung verwendet wird aufstellen:

$$\left[\begin{array}{ccccc:ccc:ccccc}
W_{00} & & & \Phi^\top & P_0^\top & M_0^\top & & & \Psi_0^\top & C_0^\top & & & \\
& W_{11} & & & P_1^\top & S_0^\top & \ddots & & & & C_1^\top & & \\
& & \ddots & & \vdots & & \ddots & M_{N-1}^\top & & & & \ddots & \\
\Phi & & & W_{NN} & P_N^\top & & & S_{N-1}^\top & \Psi_N^\top & & & & C_N^\top \\
P_0 & P_1 & \cdots & P_N & P & R_0^\top & \cdots & R_{N-1}^\top & R^\top & Q_0^\top & Q_1^\top & \cdots & Q_N^\top \; Q^\top \\
\hdashline
M_0 & S_0 & & & R_0 & & & & & & & & \\
& \ddots & \ddots & & \vdots & & & & & & & & \\
& & M_{N-1} & S_{N-1} & R_{N-1} & & & & & & & & \\
\hdashline
\Psi_0 & & & \Psi_N & R & & & & D & & & & \\
C_0 & & & & Q_0 & & & & & E_0 & & & \\
& C_1 & & & Q_1 & & & & & & E_1 & & \\
& & \ddots & & \vdots & & & & & & & \ddots & \\
& & & C_N & Q_N & & & & & & & & E_N \\
& & & & Q & & & & & & & & & E
\end{array}\right]. \tag{4.41}$$

Durch symmetrische Zeilen- und Spaltenblockumformungen können wir diese Matrix (4.41) in eine Matrix der Form

$$\left[\begin{array}{ccc:ccc:ccc:cc}
W_{00} & C_0^\top & M_0^\top & & & & & & \Phi^\top & & & \Psi_0^\top & P_0^\top \\
C_0 & E_0 & & & & & & & & & & & Q_0 \\
M_0 & & & S_0 & & & & & & & & & R_0 \\
\hdashline
& & S_0^\top & W_{11} & C_1^\top & M_1^\top & & & & & & & P_1^\top \\
& & & C_1 & E_1 & & & & & & & & Q_1 \\
& & & M_1 & & & \ddots & & & & & & R_1 \\
\hdashline
& & & & & \ddots & \ddots & \ddots & \ddots & & & & \vdots \\
& & & & & & \ddots & \ddots & & & & & \vdots \\
& & & & & & \ddots & & & S_{N-1} & & & R_{N-1} \\
\hline
\Phi & & & & & & & S_{N-1}^\top & W_{NN} & C_N^\top & & \Psi_N^\top & P_N^\top \\
& & & & & & & & C_N & E_N & & & Q_N \\
& & & & & & & & & & E & & Q \\
\hdashline
\Psi_0 & & & & & & & & \Psi_N & & & D & R \\
P_0 & Q_0^\top & R_0\top & P_1 & Q_1^\top & R_1^\top & \cdots & R_{N-1}^\top & P_N & Q_N^\top & Q^\top & R^\top & P
\end{array}\right] \tag{4.42}$$

umformen.

**Bemerkung 4.6.** *In [48], einer im Rahmen dieser Dissertation entstandenen Publikation, wurde diese Umformung auch für eine volle Diskretisierung mit Runge-Kutta-*

*Verfahren durchgeführt. Diese Diskretisierung erzeugt im Vergleich zur Trapezregel jedoch ein breiteres Band in der Matrix.*

### 4.3.2 Lösung des linearen Gleichungssystems mit Strukturausnutzung

Verglichen mit der ursprünglichen Matrix (4.41) hat die umsortierte Matrix (4.42) einen großen Teilblock, welcher eine Bandmatrix enthält. Wir können somit das lineare Gleichungssystem in die Blockstruktur

$$\begin{bmatrix} W & A^\top \\ A & B \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{4.43}$$

unterteilen. Die Matrix $W \in \mathbb{R}^{n_W \times n_W}$ entspricht hierbei dem Teilblock mit der Bandmatrix, während die Blöcke $A \in \mathbb{R}^{n_B \times n_W}$ und $B \in \mathbb{R}^{n_B \times n_B}$ den nicht Band diagonalen Anteilen der Blockstruktur entsprechen. Besonders interessant wird diese Blockstruktur für den Fall, dass die Dimension der Bandmatrix sehr viel größer ist als die Dimension der Restblöcke $n_W \gg n_B$. Dieser Zusammenhang ist insbesondere dann erfüllt, wenn die Dimension des Gitters der Diskretisierung $N$ groß ist. Das liegt darin begründet, dass nur die Dimension $n_W$ von $N$ abhängt, während die Dimension $n_B$ für das jeweilige Optimalsteuerungsproblem konstant bleibt. Aufgrund der kleinen Dimension $n_B$ können die Matrizen $A$ und $B$ in der Implementierung dicht besetzt behandelt werden.
Nehmen wir an uns steht eine schnelle Berechnungsmethode für die LR-Zerlegung der Bandmatrix $W$ zur Verfügung. Für den Teilvektor $x$ erhalten wir dann

$$x = W^{-1}\left(\alpha - A^\top y\right). \tag{4.44}$$

Diesen können wir in den zweiten Block des linearen Gleichungssystems 4.43 einsetzen und erhalten ein lineares Gleichungssystem für den Teilvektor $y$

$$\left(B - AW^{-1}A^\top\right) y = \beta - AW^{-1}\alpha. \tag{4.45}$$

Da die Dimension der Matrix $\left(B - AW^{-1}A^\top\right) \in \mathbb{R}^{n_B \times n_B}$ klein ist, können wir (4.45) durch eine dicht besetzte LR-Zerlegung lösen. Setzen wir im Anschluss die Lösung des Teilvektors $y$ in (4.44) ein, erhalten wir auch den Teilvektor $x$.
Für Bandmatrizen wie der Matrix $W$ gibt es eigene Algorithmen zur LR-Zerlegung. Die Software-Bibliothek LAPACK stellt für Bandmatrizen zur Faktorisierung die Routine `(d/s)gbtrf` und zur Rückwärtssubstitution die Routine `(d/s)gbtrs` bereit. Zur Lösung des linearen Gleichungssystems (4.43) genügt es die Faktorisierung der Bandmatrix $W$ nur

einmal zu berechnen. Die Rückwärtssubstitution muss hingegen $n_B + 2$ mal durchgeführt werden um die Gleichungen (4.44) und (4.45) zu lösen.

**Bemerkung 4.7.** *In [48] haben wir zur Lösung dieser Art von linearen Gleichungssystem die Software-Bibliothek* ALGLIN[1] *von Enrico Bertolazzi verwendet. Für diese Arbeit wurde eine eigenständige Programmierung mit* LAPACK *vorgenommen und um eine Parallelisierung erweitert. Bei dieser wird die Rückwärtssubstitution des Ausdrucks $W^{-1}A^\top$ aus (4.45), für alle $n_B$ rechten Seiten in $A^\top$, parallel berechnet. Da die* LAPACK *Routinen* `(d/s)gbtrf` *und* `(d/s)gbtrs` *sequenziell arbeiten, macht eine Parallelisierung an dieser Stelle Sinn.*

**Algorithmus zur Lösung des linearen Gleichungssystems:**

i Berechne die LR-Faktorisierung der Bandmatrix $W$.

ii Berechne durch die Rückwärtssubstitution der LR-Faktorisierung der Bandmatrix $W$ den Ausdruck $W^{-1}\alpha$. Bestimme damit die rechte Seite $\beta - AW^{-1}\alpha$ von Formel (4.45).

iii Berechne die Rückwärtssubstitutionen der LR-Faktorisierung der Bandmatrix $W$ parallel für alle Spalten der Matrix $A^\top$ und erhalte die Lösung von $W^{-1}A^\top$. Berechne damit die dicht besetzte Matrix $\left(B - AW^{-1}A^\top\right)$ aus Formel (4.45).

iv Berechne die LR-Faktorisierung der dicht besetzten Matrix $\left(B - AW^{-1}A^\top\right)$.

v Erhalte durch die Rückwärtssubstitution der LR-Faktorisierung der Matrix $\left(B - AW^{-1}A^\top\right)$ mit der rechten Seite $\beta - AW^{-1}\alpha$ den Lösungsvektor $y$ aus Formel (4.45).

vi Berechne den Vektor $\left(\alpha - A^\top y\right)$. Erhalte durch die Rückwärtssubstitution der LR-Faktorisierung Bandmatrix $W$ mit der rechten Seite $\left(\alpha - A^\top y\right)$ den Lösungsvektor $x$ aus Formel (4.44).

Der beschriebene Algorithmus stellt eine alternative Lösungsmethode für lineare Gleichungssysteme in der Form von (4.43) dar. Die meisten Optimierungssoftwarepakete für große dünn besetzte Probleme, wie beispielsweise IPOPT, arbeiten mit Algorithmen zur

[1] www: https://bitbucket.org/ebertolazzi/alglin

LR-Faktorisierung für dünn besetzte Matrizen. Einige bekannte Softwarepakete hierfür sind MUMPS, SuperLU, HSL und PARDISO. Im Vergleich zu den Algorithmen zur LR-Faktorisierung von allgemeinen dünn besetzten Matrizen, besitzt der hier eingeführte Algorithmus zwei wesentliche Vorteile.
Die LR-Faktorisierung für dünn besetzte Matrizen hat in der Regel mit einem Auffüllen einiger Nullelement in der Matrix zu kämpfen. Zum Einen erhöht dies den Speicheraufwand und zum Anderen müssen die Algorithmen zahlreiche Permutationen durchführen, um das Auffüllen von Nullelementen so weit wie möglich zu vermeiden.
Der zweite Vorteil besteht in der effizienteren Speicherstruktur. Der Speicheraufwand für Bandmatrizen und dicht besetzte Matrizen bleibt bei gleich bleibenden Dimensionen konstant. Effiziente lineare Algebra Algorithmen versuchen auch gezielt diese fest definierte Speicherstruktur auszunutzen, um die Häufigkeit des Ladens von Speicherblöcken zu reduzieren. Dadurch haben Algorithmen mit einer fest definierten Speicherstruktur eine Vorteil gegenüber verteilter Speicherstrukturen. Dünn besetzte Matrizen sind in der Regel verteilte und dynamische Speicherstrukturen. Dementsprechend ergibt sich für die Algorithmen zur LR-Faktorisierung von dünn besetzten Matrizen ein entscheidender Nachteil im Bezug auf die Speicherstruktur.

## 4.4 Berechnung von Ableitungen

Solange wir keine approximierten Hessematrizen verwenden, müssen wir für Optimalsteuerungsprobleme Ableitungen bis zur zweiten Ordnung berechnen. Da die Berechnung der Ableitungen durch finite Differenzen für zweite Ableitungen numerisch instabil sein kann, sind für uns insbesondere symbolische Ableitungen interessant. Die Berechnung von symbolischen Ableitungen per Hand kann sehr aufwändig und fehleranfällig sein. Deshalb verwenden wir für die Beispiele von OCPbasic eine Software für symbolische Differentialrechnung mit dem Namen GiNaC aus [15]. Mit Hilfe von GiNaC können C++ Klassen generiert werden, die wir als Optimalsteuerungsproblem in OCPbasic verwenden. Zu beachten ist, dass wir stückweise Funktionen, wie kubische Splines, in GiNaC als Platzhalter definieren und die entsprechenden Funktionen und Ableitungen eigenständig in die C++ Klassen einarbeiten müssen.

**Bemerkung 4.8.** *Der Vorteil von GiNaC gegenüber anderen Softwarepaketen symbolischen Differentialrechnung ist, dass es sich um freie Software handelt. Natürlich könnten wir auch kommerzielle Software für symbolische Algebra, wie die Matlab Symbolic Math Toolbox oder Maple verwenden.*

**Bemerkung 4.9.** *Beim Lösen von komplexeren Optimalsteuerungsproblemen kann auch algorithmisches Differenzieren verwendet werden. Dabei steigt die Komplexität der Ableitungen im Vergleich zur symbolischen Differentialrechnung weniger stark an. Ein freies Softwarepaket für algorithmisches Differenzieren ist ADOL-C aus [69].*

## 4.5 Regularisierung des Innere-Punkte-Verfahrens

Um im Innere-Punkte-Verfahren eine Abstiegsrichtung zu erhalten, benötigen wir nach Satz 4.1 die positive Definitheit der Hessematrix. Auch wenn die notwendige Optimalitätsbedingung zweiter Ordnung aus Abschnitt 2.1.3 bereits die positive Semidefinitheit und die hinreichende Optimalitätsbedingung aus Abschnitt 2.1.4 sogar positive Definitheit in einem Minimum zumindest auf einem kritischen Kegel fordert, muss dies nicht für jeden Iterationspunkt gelten. Eine symmetrische Hessematrix ist genau dann positiv definit, falls alle Eigenwerte der Hessematrix $\lambda > 0$ sind.
Eine der einfachsten und verbreitetsten Regularisierungen entsteht aus Gerschgorin Kreisen. Nach dem Kreissatz von Gerschgorin [37] ergibt sich eine Abschätzung der Eigenwerte unserer symmetrischen Hessematrix $H \in \mathbb{R}^{n\times n}$ zu

$$|\lambda - h_{ii}| \leq \sum_{j=1,\dots,n \;\; j\neq i} |h_{ij}| \qquad\qquad i = 1, \dots, n.$$

In der komplexen Zahlenebene verhalten sich die Gerschgorin Kreise, wie in Abbildung

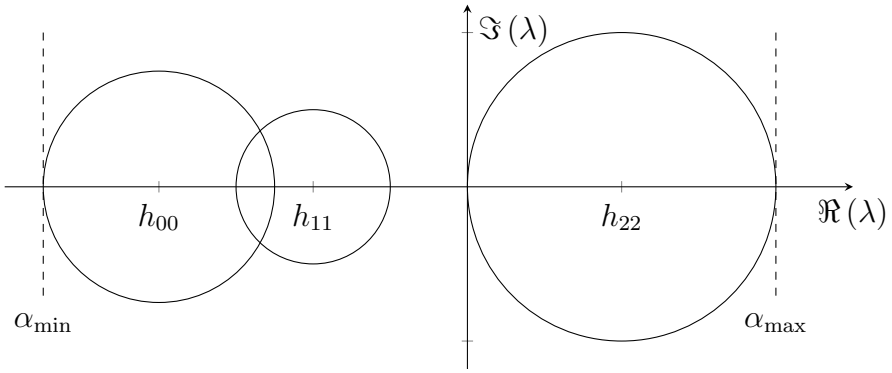


Abbildung 4.2: Skizze zu Gerschgorin Kreisen

4.2 dargestellt. Da die verwendete Hessematrix symmetrisch ist, treten ausschließlich reelle Eigenwerte auf. Somit kann mit Hilfe der Gerschgorin Kreise eine Abschätzung für den

kleinsten Eigenwert

$$\lambda_{\min} \geq \alpha_{\min} = \min_{i=1,\dots,n} \left( h_{ii} - \sum_{j=1,\dots,n \; j \neq i} |h_{ij}| \right)$$

ermittelt werden. Mit dieser Abschätzung für den kleinsten Eigenwert können wir eine regularisierte Hessematrix

$$H - (\alpha_{\min} - \Delta_{\text{tol}})I \qquad \text{mit } \Delta_{\text{tol}} > 0 \text{ und } \Delta_{\text{tol}} \lll 1 \tag{4.46}$$

definieren. Die Regularisierung erfolgt nur für den Fall $\alpha_{\min} \leq 0$, da andernfalls bereits eine positiv definite Hessematrix vorliegt.
Eine weitere in OCPbasic implementierte Regularisierung befasst sich mit der speziellen Struktur der Hessematrix. Die Hessematrix besteht, wie in (4.8) angeben aus

$$H = \nabla^2_{xx}\mathcal{L} = \begin{bmatrix} W_{00} & & & & \Phi^\top & P_0^\top \\ & W_{11} & & & & P_1^\top \\ & & \ddots & & & \vdots \\ & & & W_{N-1N-1} & & P_{N-1}^\top \\ \Phi & & & & W_{NN} & P_N^\top \\ P_0 & P_1 & \cdots & P_{N-1} & P_N & P \end{bmatrix}.$$

Wir gehen für die Regularisierung der Matrix davon aus, dass wir die Anteile der Parameter und Randbedingungen vernachlässigen können und betrachten stattdessen die Matrix

$$\tilde{H} = \begin{bmatrix} W_{00} & & & & \\ & W_{11} & & & \\ & & \ddots & & \\ & & & W_{N-1N-1} & \\ & & & & W_{NN} \end{bmatrix}.$$

Mit dem minimalen Eigenwert

$$\lambda_{\min,\tilde{H}} = \min_{i=0,\dots,N} \lambda_{\min,W_{ii}},$$

der Matrix $\tilde{H}$, kann analog zu (4.46) mit

$$H - (\lambda_{\min,\tilde{H}} - \Delta_{\text{tol}})I \qquad \text{mit } \Delta_{\text{tol}} > 0 \text{ und } \Delta_{\text{tol}} \lll 1 \tag{4.47}$$

eine Regularisierung der Hessematrix erfolgen. Die Regularisierung erfolgt ebenfalls nur für den Fall $\lambda_{\min,\tilde{H}} \leq 0$. Der Eigenwert $\lambda_{\min,W_{ii}}$ entspricht jeweils dem kleinsten Eigenwert der Teilmatrizen $W_{ii}$. Dieser kann für $i = 0, \ldots, N$ parallel, durch eine numerische Eigenwertberechnung ermittelt werden.

**Bemerkung 4.10.** *In den nachfolgenden Beispielen wird, sofern nichts anderes angegeben ist, zunächst keine Regularisierung der Hessematrix angewandt.*

## 4.6 Beispiele

### 4.6.1 Minimale Energie

Um die Unterschiede des Optimierers OCPBASIC im Vergleich zur Version der Veröffentlichung [48] OCPCOLL zu veranschaulichen führen wir ein Beispiel zur minimalen Energie aus [48] ein.

**OSP 4.1 (Minimale Energie)**

Minimiere die Zielfunktion

$$x_3(1)$$

unter den Nebenbedingungen

$$\begin{aligned} x_1'(t) &= x_2(t) \\ x_2'(t) &= u(t) \\ x_3'(t) &= \frac{1}{2}u(t)^2 \\ x_1(t) &\leq \frac{1}{9} \end{aligned}$$

und unter den Randbedingungen

$$\begin{aligned} x_1(0) &= 0 & \qquad x_1(1) &= 0 \\ x_2(0) &= 1 & \qquad x_2(1) &= -1 \\ x_3(0) &= 0. \end{aligned}$$

Um vergleichbare Aussagen über die Rechenzeiten treffen zu können, wurden die Berechnungen auf der gleichen Hardware, wie in der Veröffentlichung [48] durchgeführt. Dabei

handelt es sich um einen Intel Core i7-6700K, $4 \times 4.0$GHz Prozessor.

**Bemerkung 4.11.** *Die Namensänderung des Optimierers zu* OCPBASIC *dient zur besseren Unterscheidbarkeit von der Vorgängerversion* OCPCOLL. *Der Optimierer* OCPBASIC *wurde zusammen mit seinem verwendeten Innere-Punkte-Verfahren* IPBASIC *gänzlich neu implementiert, um die Leistungsfähigkeit des Optimierers im Hinblick auf die Strukturausnutzung zu verbessern.*

Die numerische und analytische Lösung von OSP 4.1 wurde bereits in [48] angegeben. Zunächst einmal vergleichen wir die Rechenzeiten der beiden Optimierer Versionen mit IPOPT für unterschiedliche Anzahl von Gitterpunkten.

| $N+1$ | OCPBASIC | | | OCPCOLL | | | IPOPT | |
|---|---|---|---|---|---|---|---|---|
| | $T_{\text{ges}}$ | $T_{\text{lin}}$ | Iter | $T_{\text{ges}}$ | $T_{\text{lin}}$ | Iter | $T_{\text{ges}}$ | Iter |
| 20 | 0,004 s | 0,003 s | 21 | 0,005 s | 0,002 s | 21 | 0,008 s | 9 |
| 200 | 0,055 s | 0,035 s | 30 | 0,055 s | 0,023 s | 30 | 0,065 s | 19 |
| 2000 | 0,67 s | 0,46 s | 38 | 0,67 s | 0,30 s | 57 | 0,66 s | 46 |
| 20000 | 3,3 s | 2,0 s | 28 | 6,4 s | 3,1 s | 51 | 10,2 s | 76 |

Tabelle 4.2: Vergleich der Rechenzeiten von OCPBASIC, OCPCOLL und IPOPT anhand von OSP 4.1 (Zeiten von OCPCOLL und IPOPT aus [48])

Die Anzahl der Gitterpunkte ist in Tabelle 4.2 mit $N+1$ gerade so angepasst, dass die Anzahl in beiden Versionen äquivalent sind. Aufgrund der Verwendung des selben Verfahrens schneiden OCPBASIC und OCPCOLL in Tabelle 4.2 ähnlich ab, jedoch lässt sich eine leichte Verbesserung in OCPBASIC feststellen. Im Vergleich zu IPOPT zeigt OCPBASIC deutlich schnellere Rechenzeiten. Die Bedeutung der Strukturausnutzung lässt sich allerdings anhand von einem Vergleich mit IPOPT nur bedingt messen. Das Innere-Punkte-Verfahren von IPBASIC ist zwar an das Verfahren in IPOPT angelehnt, jedoch nicht äquivalent. Um die Bedeutung der Strukturausnutzung besser untersuchen zu können testen wir deshalb, wie auch bereits in [48], den Unterschied zu einem anderen linearen Gleichungssystem Löser MA57 für dünn besetzte Matrizen. Hierfür wird in OCPBASIC der lineare Gleichungssystem Löser durch MA57 ausgetauscht.

**Bemerkung 4.12.** *Bei 2000 Gitterpunkten beträgt die Iterationszahl von* MA57 *39, im Vergleich zu 38 Iterationen bei* OCPBASIC *mit* LAPACK. *Dies liegt an den unterschiedlichen Residuen, welche die beiden lineare Gleichungssystem Löser liefern. Ein Unterschied in der Anzahl der Iterationen kann auch größer sein, falls durch die unterschiedlichen Residuen beispielsweise ein anderer Armijoschritt ausgelöst wird. Für die restlichen Rechenzeiten in Tabelle 4.3 ist die Iterationszahl gleich.*

| $N+1$ | OCPbasic LAPACK | | OCPbasic LAPACK par. | | OCPbasic MA57 | | Iter |
|---|---|---|---|---|---|---|---|
| | $T_{\text{ges}}$ | $T_{\text{lin}}$ | $T_{\text{ges}}$ | $T_{\text{lin}}$ | $T_{\text{ges}}$ | $T_{\text{lin}}$ | |
| 20 | 0,004 s | 0,003 s | 0,003 s | 0,002 s | 0,003 | 0,002 s | 21 |
| 200 | 0,055 s | 0,035 s | 0,055 s | 0,046 s | 0,038 | 0,028 s | 30 |
| 2000 | 0,67 s | 0,46 s | 0,28 s | 0,19 s | 0,49 | 0,37 s | 38 - 39 |
| 20000 | 3,3 s | 2,0 s | 2,0 s | 1,4 s | 3,6 | 2,9 s | 28 |
| 50000 | 6,9 s | 5,0 s | 5,0 s | 3,7 s | 9,5 | 7,4 s | 29 |
| 100000 | 14,1 s | 10,9 s | 11,2 s | 8,1 s | 21,6 | 17,5 s | 32 |

Tabelle 4.3: Vergleich OCPbasic unter Verwendung von LAPACK und MA57 anhand von OSP 4.1

Die in Tabelle 4.3 dargestellten Rechenzeiten umfassen auch die parallelisierte Variante des linearen Gleichungssystem Lösers aus Abschnitt 4.3.2. Bereits die sequenzielle Variante dominiert MA57 bei hohen Anzahlen von Gitterpunkten. Der parallelisierten Variante gelingt es die Rechenzeit noch weiter zu senken.

**Bemerkung 4.13.** *Für die parallelisierte Variante des linearen Gleichungssystem Lösers aus Abschnitt 4.3.2 wurden mit folgenden OpenMP Umgebungsvariablen durchgeführt:*
`export OMP_PROC_BIND=TRUE`
`export OMP_WAIT_POLICY=PASSIVE`

### 4.6.2 Kinematisches Fahrzeugmodell

Wir definieren unter Verwendung des kinematischen Fahrzeugmodells aus dem Differentialgleichungssystem 3.1 das Ausweichproblem.

**OSP 4.2 (Ausweichproblem mit kinematischem Fahrzeugmodell)**

Minimiere die Zielfunktion

$$t_f + c_s \int_0^{t_f} \delta(t)^2 dt$$

unter den Nebenbedingungen

$$\begin{aligned} x'(t) &= v(t)\cos(\psi(t)) \\ y'(t) &= v(t)\sin(\psi(t)) \\ \psi'(t) &= \frac{v(t)}{\ell}\tan(\delta(t)) \\ r^2 \leq x(t)^2 + y(t)^2 &\leq \infty \\ y(t) &\geq -0{,}01 \end{aligned}$$

und unter den Randbedingungen

$$\begin{aligned} x(0) &= -1 & x(t_f) &= 1 \\ y(0) &= 0 & y(t_f) &= 0 \\ \psi(0) &= 0 & \psi(t_f) &= 0. \end{aligned}$$

In OSP 4.2 ist ein Optimalsteuerungsproblem zur Berechnung einer zeitoptimierten Ausweichtrajektorie um einen Kreis mit Radius $0 \leq r \leq 0{,}5$ definiert. Die Steuerungen sind dabei durch

$$-0{,}5 \leq \delta(t) \leq 0{,}5 \qquad \text{und} \qquad 0 \leq v(t) \leq 3$$

beschränkt. Um auch die Richtung des Ausweichens vorzugeben wird zudem der Zustand $y(t) \geq -0{,}01$ beschränkt. In Abbildung 4.3 ist eine Lösung dieses Optimalsteuerungs-

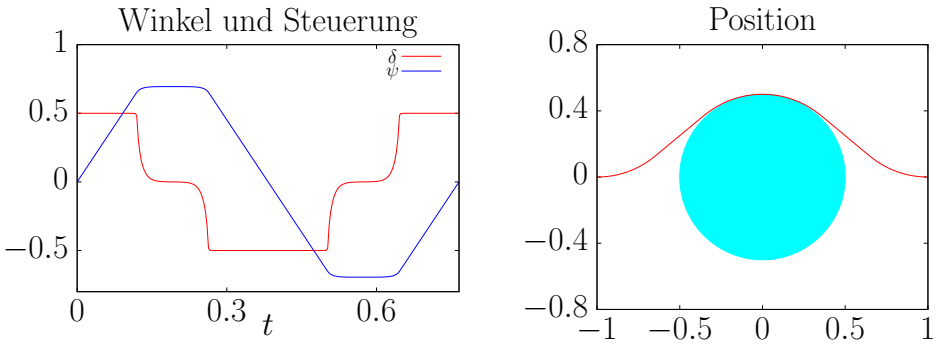


Abbildung 4.3: Lösung von OSP 4.2 für $\ell = 0{,}3$, $r = 0{,}5$ und $c_s = 0{,}005$

problems für 250 Gitterpunkte dargestellt. Die Steuerung $v(t)$ entspricht aufgrund der Zeitoptimalität und der Tatsache, dass sie lediglich durch $v_{\max} = 3$ beschränkt ist, gerade

$v(t) = 3\ \forall t \in [0, t_f]$ und ist deshalb in Abbildung 4.3 nicht dargestellt. Die optimale Endzeit beträgt $t_f \approx 0{,}77$s. Der Verlauf des Lenkwinkels $\delta(t)$ in Abbildung 4.3 verdeutlicht die Bedeutung des Zielfunktionsterms $c_s \int_0^{t_f} \delta(t)^2 dt$. Durch diesen Term werden Sprungstellen in der Steuerung vermieden, welche sich für $c_s = 0$ ausprägen. Dies ist im Hinblick auf den allgemeinen Fahrkomfort natürlich wünschenswert.

### 4.6.3 Vereinfachtes kurven-parametrisiertes Fahrzeugmodell

Da in diesem Abschnitt nicht mit den Rechenzeiten aus [48] verglichen werden muss, konnten die Berechnungen dieses Abschnitts auf einem neueren Prozessor berechnet werden. Dabei handelt es sich um einen Intel Core i7-7700K, $4 \times 4.2$GHz Prozessor. Für alle Beispiele mit dem vereinfachten kurven-parametrisierten Fahrzeugmodell wird in OCPbasic eine Regularisierung der Hessematrix mit dem minimalen Eigenwert der Teilmatrizen $W_{ii}$ aus Abschnitt 4.5 angewandt.

#### Vereinfachtes kurven-parametrisiertes Fahrzeugmodell mit Vorausschau-Horizont

Zunächst können wir uns bei dem vereinfachten kurven-parametrisierten Fahrzeugmodell wie in Abschnitt 3.6 auf einen Vorausschau-Horizont beschränken. Wir definieren unter Verwendung des vereinfachten kurven-parametrisierten Fahrzeugmodells aus dem Differentialgleichungssystem 3.5 das Optimalsteuerungsproblem.

**OSP 4.3 (Vereinfachtes kurven-parametrisiertes Fahrzeugmodell mit Vorausschau-Horizont)**

Minimiere die Zielfunktion

$$-\alpha_0 s(\zeta_f) + \alpha_1 \int_0^{\zeta_f} u(\zeta)^2 d\zeta$$

unter den Nebenbedingungen

$$\begin{aligned} s'(\zeta) &= \frac{\cos(\chi(\zeta))}{1 + r(\zeta)\kappa_m(s(\zeta))} \\ r'(\zeta) &= \sin(\chi(\zeta)) \\ \chi'(\zeta) &= \kappa_m(s(\zeta))\frac{\cos(\chi(\zeta))}{1 + r(\zeta)\kappa_m(s(\zeta))} - u(\zeta) \\ r_{\min}(\zeta) &\le r(\zeta) \le r_{\max}(\zeta) \end{aligned}$$

und unter den Anfangsbedingungen

$$s(0) = s_0 \qquad r(0) = r_0 \qquad \chi(0) = \chi_0,$$

mit $u(\zeta) \in [u_{\min}, u_{\max}]$.

Das Optimalsteuerungsproblem aus OSP 4.3 lässt sich vereinfacht für den Testkurs in Most mit $r_{\min}(\zeta) = -5{,}0$ m und $r_{\max}(\zeta) = 5{,}0$ m lösen. Die Lösung von OSP 4.3 ist für

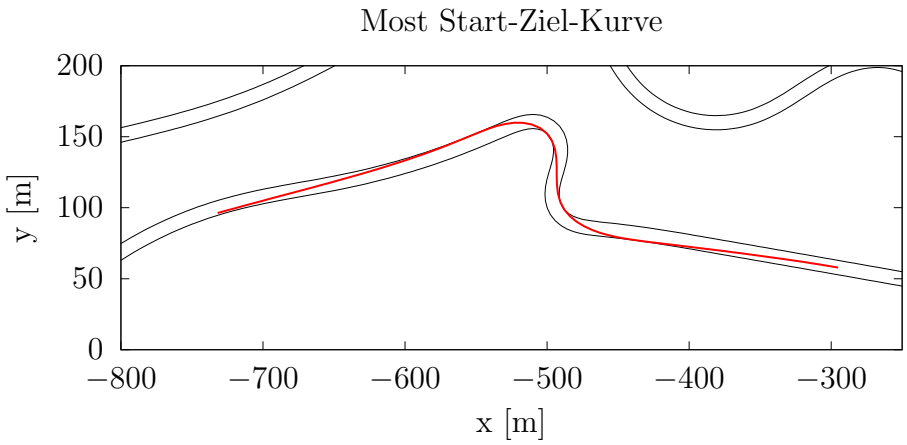


Abbildung 4.4: Optimale Lösung für Kurvenabschnitt 1 und 2 des Testkurses Most

die Start-Ziel-Kurve von Most in Abbildung 4.4 dargestellt. Der Vorausschau-Horizont liegt bei dieser Berechnung bereits bei 500 m und die Rechenzeit mit der parallelisierten Version von OCPbasic beträgt 0,022 s.

Um genauer analysieren zu können, wie sich die Rechenzeiten bei zunehmender Anzahl von Gitterpunkten entwickelt, führen wir ein Benchmark Problem ein. Dabei wollen wir einen möglichst einfachen Kurs definieren, welcher durch einen periodischen kubischen Spline der Mittellinie repräsentiert wird. Als Stützpunkte des periodischen kubischen Splines verwenden wir:

| x: | y: |
|---|---|
| 0 m | 0 m |
| 200 m | 200 m |
| 0 m | 400 m |
| -200 m | 200 m |
| 0 m | 0 m |

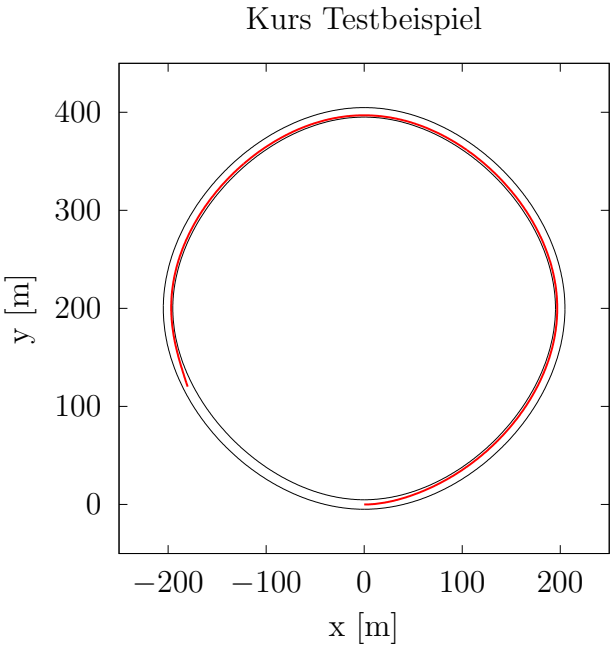


Abbildung 4.5: Optimierte Trajektorie für das Testbeispiel des Benchmarks mit 1000 Gitterpunkten.

Zusätzlich definieren wir, wie bereits bei unserem vorherigen Beispiel, $r_{\min}(\zeta) = -5{,}0$ m und $r_{\max}(\zeta) = 5{,}0$ m. Wir verwenden einen konstanten Vorausschau-Horizont von 1000 m und lösen das Optimalsteuerungsproblem mit einer verschiedenen Anzahl von Gitterpunkten. In Abbildung 4.5 ist eine Lösung für dieses Testbeispiel für 1000 Gitterpunkte dargestellt.
Tabelle 4.4 zeigt den Vergleich der Rechenzeiten für die verschiedenen lineare Gleichungssystem Löser. Es zeichnet sich zunächst ein ähnliches Verhalten, wie in OSP 4.1 ab. Besonders auffällig ist allerdings, dass die Rechenzeiten von MA57 bei vielen Gitterpunkten bedeutend größer sind als die Vergleichszeiten bei OCPbasic mit Verwendung von LAPACK. Bei einer Anzahl von Gitterpunkten mit $N = 150000$ schlägt der lineare

| Gitterpunkte | OCPbasic LAPACK | | OCPbasic LAPACK par. | | OCPbasic MA57 | | Iter |
|---|---|---|---|---|---|---|---|
| | $T_{\text{ges}}$ | $T_{\text{lin}}$ | $T_{\text{ges}}$ | $T_{\text{lin}}$ | $T_{\text{ges}}$ | $T_{\text{lin}}$ | |
| 10 | 0,004 s | 0,002 s | 0,003 s | 0,001 s | 0,003 s | 0,002 s | 14 |
| 100 | 0,029 s | 0,018 s | 0,022 s | 0,009 s | 0,026 s | 0,020 s | 13 |
| 1000 | 0,33 s | 0,18 s | 0,15 s | 0,08 s | 0,56 s | 0,49 s | 14 |
| 10000 | 2,1 s | 0,89 s | 1,6 s | 0,75 s | 10,1 s | 9,4 s | 11 |
| 100000 | 12,0 s | 8,1 s | 10,9 s | 7,0 s | 427,9 s | 423,4 s | 10 |
| 150000 | 18,1 s | 12,1 s | 17,2 s | 10,8 s | 933,3 s | 926,5 s | 10 |

Tabelle 4.4: Benchmark für vereinfachtes kurven-parametrisiertes Fahrzeugmodell

Gleichungssystem Löser mit LAPACK, MA57 ca. um den Faktor 76 bis 77.

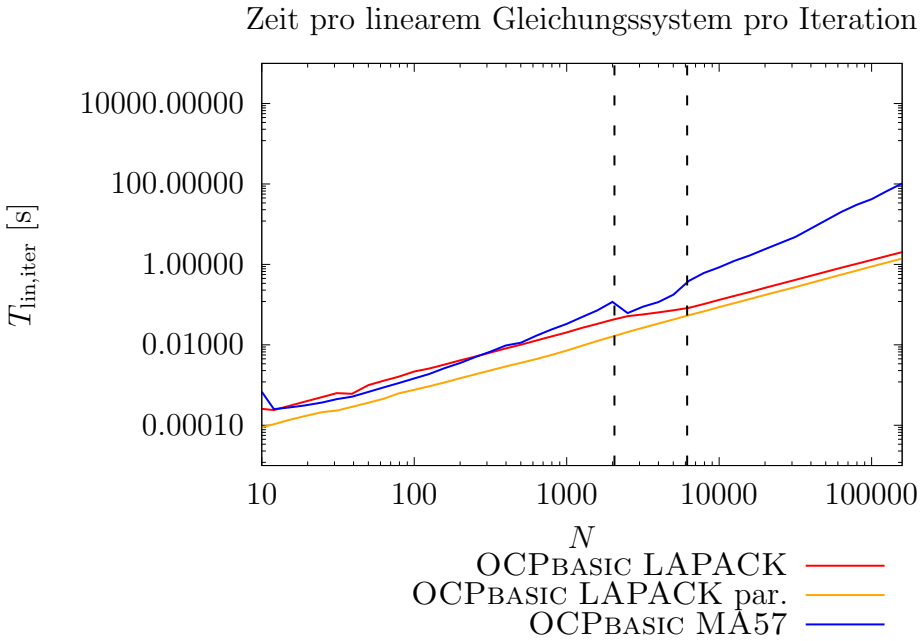


Abbildung 4.6: Rechenzeiten $T_{\text{lin,iter}}$ zur Lösung des linearen Gleichungssystems pro Iteration, in doppelt logarithmischer Darstellung

Um den Verlauf der Rechenzeiten besser darstellen zu können sind in Abbildung 4.6 die Rechenzeiten zur Lösung eines linearen Gleichungssystems für über die Anzahl der Gitterpunkte, in doppelt logarithmischer Darstellung abgebildet. In der Abbildung lässt sich gut erkennen, dass die Rechenzeiten von MA57 ab einer gewissen Anzahl von Gitterpunkten einen Knick aufweisen und deutlich steiler steigen. In doppelt logarithmischer

Darstellung entspricht eine lineare Funktion gerade einer polynomialen Funktion $\mathcal{O}(mx^p)$. Wobei der Faktor $m$ die lineare Funktion in doppelt logarithmischer Darstellung parallel verschiebt und der Exponent $p$ die Steigung der Geraden angibt. In Abbildung 4.6 liegen die Rechenzeiten der parallelen Variante von OCPbasic mit Verwendung von LAPACK näherungsweise auf einer Geraden. Vergleichen wir den Faktor, der sich zwischen $N = 10$ und $N = 1000$ Gitterpunkten einstellt, kommen wir etwa auf den Faktor 100, was einem Exponent von $p = 1$ entspricht. Der gemessene Rechenaufwand beträgt somit ca. $\mathcal{O}(mN)$. In erster Näherung ist der Rechenaufwand der sequenziellen Version von OCPbasic mit Verwendung von LAPACK in Abbildung 4.6 dazu parallel verschoben, was die in Abschnitt 4.3.2 eingeführte parallele Variante der Rückwärtssubstitution auch erwarten lässt.

**Bemerkung 4.14.** *Für die parallelisierte Variante des linearen Gleichungssystem Lösers aus Abschnitt 4.3.2 wurden, wie auch bei OSP 4.1, mit OpenMP Umgebungsvariablen durchgeführt:*
`export OMP_PROC_BIND=TRUE`
`export OMP_WAIT_POLICY=PASSIVE`

Anmerken lässt sich zu Abbildung 4.6, dass mit den Rechenzeiten pro Iteration nicht nur der Rechenaufwand gemessen wird, sondern auch der Speicheraufwand. Gerade bei großen Speichermengen sind Effekte, wie das Laden von Arbeitsspeicher in den Prozessor Cache nicht trivial. Um den Anstieg der Rechenzeiten für MA57 besser interpretieren zu können, vergleichen wir die verwendete Speichermenge im Zusammenhang mit der Anzahl von Gitterpunkten mit dem Cache Speicher des Prozessors (8 MByte). Da sich nur der Anteil der Bandmatrix aus (4.42) mit der Anzahl von Gitterpunkten $N$ in Zeilen- und Spaltendimension vergrößert, vernachlässigen wir in der Abschätzung den konstanten Speicheranteil der Restmatrix. Die Bandmatrix besteht aus $N$ symmetrischen Blöcken, mit den für OSP 4.3 dimensionierten Matrizen

$$
\begin{aligned}
W_{ii} \in \mathbb{R}^{5\times 5} &\underset{\text{sym}}{\longrightarrow} 15 \text{ Werte} & C_i \in \mathbb{R}^{5\times 5} &\longrightarrow 25 \text{ Werte} \\
M_i \in \mathbb{R}^{4\times 5} &\longrightarrow 20 \text{ Werte} & S_i \in \mathbb{R}^{4\times 5} &\longrightarrow 20 \text{ Werte} \\
E_i \in \mathbb{R}^{5\times 5} &\underset{\text{diag}}{\longrightarrow} 5 \text{ Werte,}
\end{aligned}
$$

was in der Summe 85 Werte pro Block bedeutet. Für MA57 werden diese Werte in einer symmetrischen Sparsematrix abgespeichert, die pro Eintrag zwei 4 Byte Integer und einen 8 Byte Double an Speicher benötigt. Das heißt für unsere kritische Anzahl von

Gitterpunkten

$$N_{\text{krit}} = \frac{8 \cdot 1024 \cdot 1024 \text{ Byte}}{85 \cdot 16 \text{ Byte}} \approx 6168.$$

Intern verwendet MA57 auch einen Verarbeitungsspeicherbereich, der in unserem Fall die dreifache Größe der verwendeten Sparsematrix umfasst. Dieser besitzt die kritische Anzahl von Gitterpunkten

$$N_{\text{krit,work}} = \frac{8 \cdot 1024 \cdot 1024 \text{ Byte}}{85 \cdot 16 \text{ Byte} \cdot 3} \approx 2056.$$

In Abbildung 4.6 sind zwei schwarze Linien eingezeichnet, welche gerade der kritischen Anzahl von Gitterpunkten entsprechen. Da sich die Rechenzeit von MA57 ab diesem Bereich drastisch verschlechtert liegt die Vermutung nahe, dass es sich hierbei um einen Effekt der Speicherverarbeitung handelt.
Für den Gleichungssystem Löser von OCPbasic mit Verwendung von LAPACK haben die beiden Linien kaum Aussagekraft, da eine vollkommen andere Speicherstruktur vorliegt. Die Speicherstruktur der verwendeten Bandmatrix des Softwarepakets LAPACK, ist auf die Struktur einer Bandmatrix angepasst. Der Gleichungssystem Löser MA57 ist für allgemeine symmetrische Sparsematrizen implementiert und kann deshalb die Speicherstruktur nicht derart effizient nutzen.

### Vereinfachtes kurven-parametrisiertes Fahrzeugmodell als Rundenoptimierung

Motiviert durch die schnellen Rechenzeiten für hohe Vorausschau-Horizonte mit vielen Gitterpunkten wollen wir im weiteren versuchen das Optimalsteuerungsproblem von einem Vorausschau-Horizont auf eine ganze Runde zu erweitern.
Wir definieren unter Verwendung des vereinfachten kurven-parametrisierten Fahrzeugmodells aus dem Differentialgleichungssystem 3.5 das Optimalsteuerungsproblem.

**OSP 4.4 (Vereinfachtes kurven-parametrisiertes Fahrzeugmodell für Gesamtkurs)**

Minimiere die Zielfunktion

$$\alpha_0 \zeta_f + \alpha_1 \int_0^{\zeta_f} u(\zeta)^2 d\zeta$$

unter den Nebenbedingungen

$$
\begin{aligned}
s'(\zeta) &= \frac{\cos(\chi(\zeta))}{1 + r(\zeta)\kappa_m(s(\zeta))} \\
r'(\zeta) &= \sin(\chi(\zeta)) \\
\chi'(\zeta) &= \kappa_m(s(\zeta))\frac{\cos(\chi(\zeta))}{1 + r(\zeta)\kappa_m(s(\zeta))} - u(\zeta) \\
r_{\min}(\zeta) &\leq r(\zeta) \leq r_{\max}(\zeta)
\end{aligned}
$$

und unter den Randbedingungen

$$
\begin{aligned}
s(0) &= 0 & s(t_f) &= s_{\max} \\
r(0) &= r(\zeta_f) & \chi(0) &= \chi(\zeta_f),
\end{aligned}
$$

mit $u(\zeta) \in [u_{\min}, u_{\max}]$.

Durch die Randbedingungen wird ein stetiger Übergang am periodischen Anfangs- bzw. Endpunkt der Trajektorie gefordert.

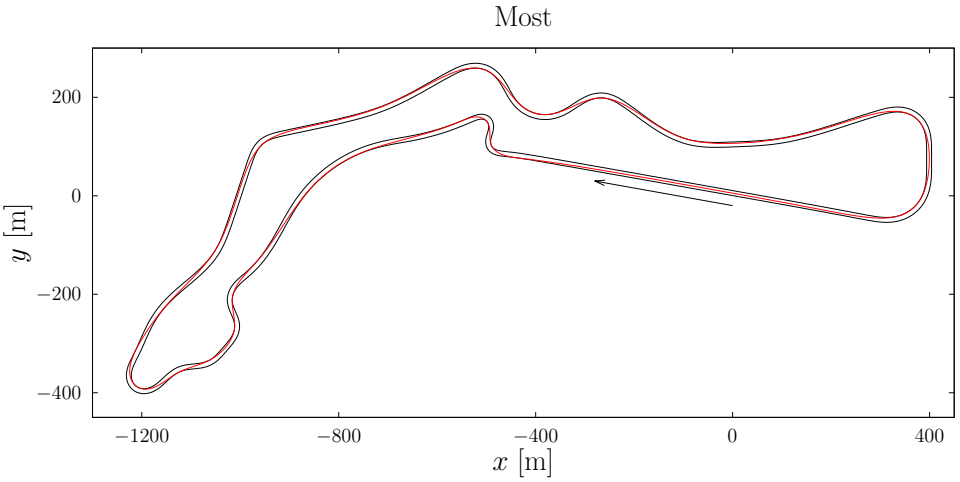


Abbildung 4.7: Optimale Lösung für der Testkurs Most

In Abbildung 4.7 ist die Lösung des Optimalsteuerungsproblems aus OSP 4.4 für einen Abschnitt des Testkurses in Most dargestellt. Das Lösen des Optimalsteuerungsproblems benötigt mit 600 Gitterpunkten gerade einmal 0,26 s.

**Bemerkung 4.15.** *Es sei angemerkt, dass sich die Optimierung eines gesamten Testkurses deutlich weniger robust verhält, als das Optimalsteuerungsproblem auf einem Vorausschau-Horizont. Die Robustheit hängt dabei insbesondere von den Faktoren $\alpha_0$ und $\alpha_1$ der Zielfunktion, der Anzahl der Gitterpunkte $N$ und der Komplexität des Testkurses ab.*

### Vereinfachtes kurven-parametrisiertes Fahrzeugmodell als MPC

Abschließend testen wir den Optimierer OCPbasic auch für ein MPC Beispiel, analog zu Abbildung 3.20 aus Abschnitt 3.6.3, durch einen Vergleich mit dem kurven-parametrisierten Einspurmodell. In Abbildung 4.8 ist die mit OCPbasic berechnete Lösung des

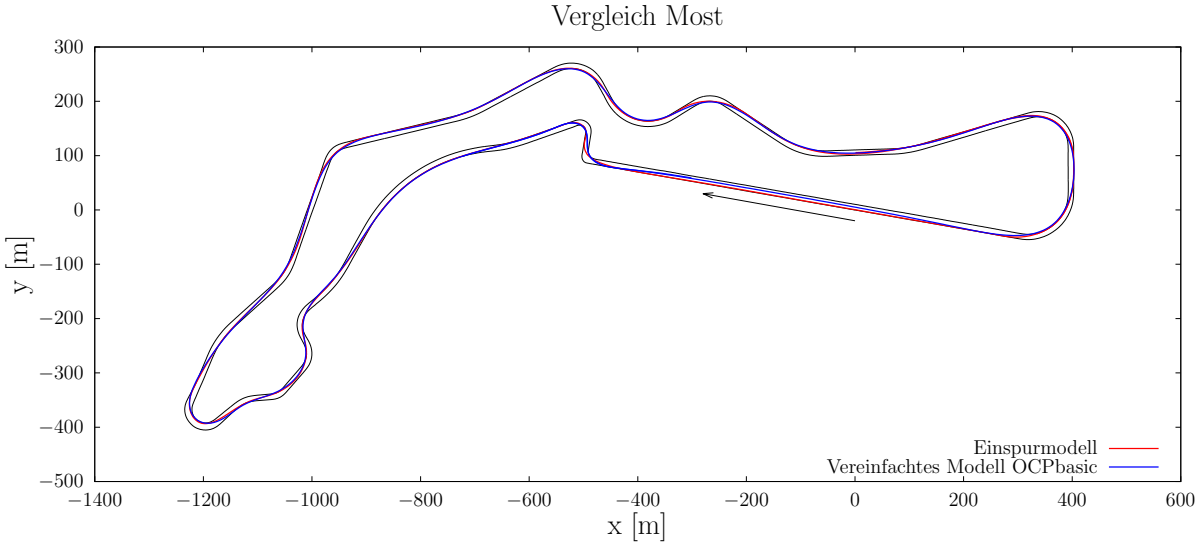


Abbildung 4.8: Vergleich anhand des Testkurses Most, mit dem kurven-parametrisierten Einspurmodell aus Abbildung 3.16. Die verwendete Vorausschau des vereinfachten kurven-parametrisierten Fahrzeugmodells in OCPbasic beträgt 500 m.

vereinfachten kurven-parametrisierten Fahrzeugmodells, zusammen mit der in Abbildung 3.16 dargestellten Einspurmodell-Trajektorie, für den Testkurs in Most abgebildet. Wie in Abschnitt 3.6.3 können wir den Unterschied zum Einspurmodell anhand der roten Hinterlegung der blauen Trajektorie erkennen. Aufgrund der erhöhten Vorausschau von 500 Metern, müssen auch die Faktoren der Zielfunktion $\alpha_0$ und $\alpha_1$, leicht modifiziert werden, um die Lösung des vereinfachten kurven-parametrisierten Fahrzeugmodells an das Einspurmodell anzugleichen. Wie bereits in Abbildung 3.6.3 gleicht sich das vereinfach-

te kurven-parametrisierte Fahrzeugmodell über fast den gesamten Testkurs gut an die Lösung des Einspurmodells an. Unterschiede zeigen sich zum einen am Ende der Zielgeraden, wo die Trajektorie des vereinfachten kurven-parametrisierten Fahrzeugmodells besser verläuft, da die Vorausschau des Einspurmodells, wie in Abschnitt 3.6.3 beschrieben, nur ausreicht um die Kurve gerade noch durchfahren zu können. Ein neuer Unterschied entsteht beim Verlauf der Trajektorie auf der Zielgeraden. Während die Lösung des Einspurmodells hier am linken Straßenrand verläuft, wechselt die mit OCPbasic berechnete Rennlinie zur Mitte der Fahrbahn. Dies liegt am verwendeten Optimierungsverfahren. Da statt dem SQP-Verfahren in OCPbasic ein Innere-Punkte-Verfahren zum Einsatz kommt, unterliegt die blaue Trajektorie minimal veränderten Optimalitätskriterien. Hinsichtlich der Optimalität ist es auf einer Geraden nahezu egal, in welchem Abstand das Fahrzeug zur Mittellinie fährt. Deshalb wirken sich hier die schwachen Barriereterme, welche die Lösung von den Beschränkungen fernhalten, sichtbar aus.

**Bemerkung 4.16.** *Hinsichtlich der Praxistauglichkeit der Trajektorie, ist ein autonomes Durchfahren der Zielgeraden in der Mitte der Straße sogar wünschenswert. Da Zielgeraden typischerweise an den Fahrbahnrändern durch Mauern begrenzt sind, geben Trajektorien dieser Art den Sicherheitsfahrern des autonomen Fahrzeugs, deutlich verbesserte Eingriffsmöglichkeiten im Falle technischer Ausfälle des Versuchsträgers, die zum Abweichen von der Bahn führen.*

Zusammenfassend ist festzustellen, dass sich mit OCPbasic auch höhere Vorausschau-Horizonte immer noch in einer online fähigen Zeit berechnen lassen. Dementsprechend müssten für den Einsatz im Versuchsfahrzeug keine reduzierten Vorausschau-Horizonte verwendet werden. Die durchschnittliche Zeit zur Berechnung einer optimalen Lösung liegt mit OCPbasic bei der oben genannten Hardware bei 0,03 s, unter Verwendung der parallelisierten Variante mit OpenMP Umgebungsvariablen. Ein weiterer Vorteil von OCPbasic besteht darin, dass der Lösungsprozess abhängig von der bereits benötigten Zeit abgebrochen werden kann. In diesem Beispiel lag diese maximale Optimierungszeit bei 0,1 s. Falls der Optimierer in dieser Zeit keine Lösung findet, wird die MPC Trajektorie nicht akzeptiert und das Optimierungsverfahren für einen neuen MPC Schritt angewandt. Solange noch genügend Vorausschau von der letzten berechneten Trajektorie übrig ist, kann der Optimierer immer weiter versuchen eine neue MPC Trajektorie zu liefern. Dieses Verhalten entspricht gerade dem, in Abschnitt 3.8.1 beschriebenen, Verfahren zur online-Optimierung. Somit eignet sich der Optimierer OCPbasic im Hinblick auf die Konfigurationsmöglichkeiten für die Implementierung von online-Versuchen.

# Kapitel 5

# Fazit

## 5.1 Zusammenfassung

Das Ziel dieser Arbeit war es, verschiedene Methoden und Algorithmen aus der Optimalsteuerung in dem Anwendungsbereich der Onlinebahnplanung für autonome Fahrzeuge im dynamischen Grenzbereich zu untersuchen.
Zunächst haben wir einen Algorithmus mit dynamischer Programmierung zur Lösung eines Optimalsteuerungsproblems, unter Verwendung eines kurven-parametrisierten kinematischen Fahrzeugmodells vorgestellt. Motiviert war dieser Ansatz durch die Vorwärtsrechnung der dynamischen Programmierung, welche als Feedback Gesetz auch online anwendbar wäre. Durch die schlechte Qualität der berechneten Trajektorien hat sich dieser Ansatz als unzureichend herausgestellt. Der Grund hierfür liegt an der Minimierung der Anzahl von Differentialgleichungen, welche der Ansatz der dynamischen Programmierung erfordert und eine dementsprechend einfache Wahl des Fahrzeugmodells notwendig macht. Des Weiteren ist die Rückwärtsrechnung der dynamischen Programmierung zu rechenaufwändig. Dies liegt vor allem daran, dass die Rückwärtsrechnung für alle noch nicht berücksichtigten Anpassungen, wie zum Beispiel Hindernisse und andere Testkurse, erneut durchgeführt werden muss.
Weiterführend an ein Vorläuferprojekt aus [42] wurde im nächsten Schritt ein Einspurmodell zur Optimalsteuerung verwendet. Hierbei kam eine direkte Diskretisierung mit Schießverfahren und verwendetem SQP-Verfahren des Optimierers OCPID-DAE1 zum Einsatz. Es erfolgte eine Anpassung des kartesischen Fahrzeugmodells aus [42] hin zu einem kurven-parametrisierten Einspurmodell. Damit ließen sich die Vorausschau-Horizonte so weit robust steigern, dass eine zufriedenstellende optimale Rennlinie auf verschiedenen Testkursen berechnet werden konnte. Dabei fand auch eine Anpassung der Zielfunktion des

Optimalsteuerungsproblems statt. Zur Bewertung der Trajektorien konnten offline berechnete Rennlinien mit einem Versuchsfahrzeug und unter Verwendung eines Bahnfolgereglers nachgefahren werden. Zusammen mit Testfahrern fand anschließend eine Auswertung der Qualität der Rennlinien statt. Die Zielfunktion wurde so angepasst, dass die resultierenden Trajektorien zufriedenstellend waren. Aufgrund der Rechenzeiten eignet sich dieser Ansatz im Zusammenspiel mit dem verwendeten Optimierer nicht für Onlineberechnungen. Die berechneten Lösungen mit dem kurven-parametrisierten Einspurmodell konnten allerdings in späteren Teilen der Arbeit als Vergleichslösung verwendet werden.

Um auch eine online lauffähige Bahnplanung mit dem Optimierer OCPID-DAE1 zu realisieren, wurde in Abschnitt 3.6 ein vereinfachtes kurven-parametrisiertes Fahrzeugmodell eingeführt, das auch in einer Onlinebahnplanung Verwendung findet. Durch eine geschickte Wahl der Zielfunktion ist es uns gelungen das vereinfachte kurven-parametrisierte Fahrzeugmodell an die Lösungen des Einspurmodells anzunähern. Unter Verwendung des Optimierers OCPID-DAE1 war es möglich das Optimalsteuerungsproblem mit dem vereinfachten kurven-parametrisierten Fahrzeugmodell auf einem Versuchsfahrzeug der Volkswagen AG zu implementieren. Im Anschluss fanden Online-Fahrversuche mit diesem Bahnplaner und einem Vorausschau-Horizont von 200 Metern statt. Die Versuche führten zu einer reproduzierbaren Rennlinie, welche qualitativ mit offline berechneten Trajektorien mit einem Vorausschau-Horizont von 200 Metern übereinstimmt. Die guten Ergebnisse im Bezug auf die Qualität der Rennlinie motivierten weitere Fahrversuche mit Hindernissen, welche in die Fahrbahnbeschränkungen integriert wurden. Den Einfluss von Hindernissen auf die Trajektorie und die Robustheit der Optimierung testeten wir mit Hilfe eines Doppelhindernisses, bei dem die beiden Hindernisse einen Abstand von 50 Metern hatten. Bei den online-Versuchsfahrten konnte der Optimierer eine Ausweichtrajektorie für das Doppelhindernis ohne Einbußen bezüglich der online-Fähigkeit der Rechenzeiten berechnen. Es sei angemerkt, dass bei einer von drei Durchfahrten des Doppelhindernisses eine Umplanung der Trajektorie ausgelöst wurde, welche durch die punktweise Auswertung der Nebenbedingungen erklärt werden kann. Dementsprechend ist die Reproduzierbarkeit der Trajektorien stark von der Art der Beschränkungen, in diesem Fall der Form der Hindernisse abhängig. Das dynamische Limit wurde in den Online-Fahrversuchen, aus Sicherheitsgründen, auf 40% der verfügbaren Längsdynamik beschränkt. Auf freier Fläche konnte der Bahnplaner bei bis zu 80% der verfügbaren Längsdynamik getestet werden. Eine weitere Erhöhung der verwendeten Längsdynamik erfordert eine Anpassung und Verbesserung der Berechnung des Geschwindigkeitsprofils. Damit ist das Ziel dieser Arbeit bezüglich der Onlinebahnplanung für autonome Fahrzeuge im dynamischen Grenzbereich weitgehend erreicht.

Weiterführend wurde in Kapitel 4 eine Strukturausnutzung eines Innere-Punkte-Verfahrens im Zusammenspiel mit einer direkten Diskretisierung durch die Trapezregel untersucht. Aus diesen Untersuchungen ging ein an IPOPT orientiertes, aber komplett eigenständiges Innere-Punkte-Verfahren namens IPBASIC hervor, das sich durch seine Implementierung speziell zur Strukturausnutzung eignet. Mit IPBASIC konnte ein komplett neuer Optimierer OCPBASIC, zur Lösung von Optimalsteuerungsproblemen, mit Strukturausnutzung entwickelt werden. Der Optimierer OCPBASIC schafft es zeitgemäße lineare Gleichungssystem Löser für dünn besetzte Matrizen, wie MA57, bezüglich der Rechenzeiten zu schlagen. Gerade bei großen Dimensionen beziehungsweise einer großen Anzahl von Gitterpunkten wird dieser Unterschied deutlich. Zusammen mit dem vereinfachten kurven-parametrisierten Fahrzeugmodell aus Abschnitt 3.6 lassen sich durch OCPBASIC auch längere Vorausschau-Horizonte, wie in Abbildung 4.4 mit 500 Metern, mit einer online fähigen Rechenzeit realisieren. Fairerweise muss man hierbei anmerken, dass die für die Onlineberechnung interessanten Fälle mit verhältnismäßig wenig Gitterpunkten $N < 200$ auskommen. Dementsprechend wäre auch die Verwendung eines Standardlösers für lineare Gleichungssysteme mit dünn besetzten Matrizen, im Bezug auf die Rechenzeiten denkbar. Folglich könnten auch Softwarepakete wie FALCON.m aus [61], FORCES NLP und FORCES PRO aus [74] und [27] oder CasADi aus [3] zur Lösung der Optimalsteuerungsprobleme verwendet werden.

## 5.2 Ausblick

Vor allem die Verwendung des vereinfachten kurven-parametrisierten Fahrzeugmodells aus Abschnitt 3.6 ist für den weiteren Einsatz von Optimalsteuerung im Bereich des autonomen Fahrens interessant. Neben den aktuellen Fahrzeugdaten wird hierfür lediglich die Beschränkung der Bahn durch den rechten und linken Fahrbahnrand benötigt. Die Offlineberechnung einer Startlösung ist nicht notwendig. Somit könnte dieses Verfahren im Zusammenspiel mit einer robusten Umwelterkennung für Hindernisse und Fahrbahnränder eingesetzt werden. Hierbei würde die Umwelterkennung für einen Vorausschau-Horizont die Hindernisse und Fahrbahnränder zur Verfügung stellen, welche der Optimierer verwenden kann, um eine optimale Trajektorie zu berechnen. Dabei könnte auch zunehmend der Fahrkomfort optimiert werden, so dass die Insassen eines autonomen Fahrzeugs weniger Fahrbelastungen ausgesetzt sind.
Die Auslegung des vereinfachten kurven-parametrisierten Fahrzeugmodells für den dynamischen Grenzbereich hängt überwiegend von der Qualität der berechneten Geschwindigkeits- und Beschleunigungsprofile ab. Eine Verbesserung der Berechnungsmethoden

ist bereits Bestandteil aktueller Forschungen und kann in [16], [10] und [9] verfolgt werden. Darin wird die Berechnung von Geschwindigkeits- und Beschleunigungsprofilen im Hinblick auf Kollisionsvermeidung mit mehreren Fahrzeugen untersucht.
Die Verwendung des Optimierers OCPbasic für Online-Fahrversuche zusammen mit dem vereinfachten kurven-parametrisierten Fahrzeugmodell ist für weiterführende Forschungen bezüglich der Trajektorienoptimierung auf Rennstrecken besonders interessant. Dazu wäre auch eine Weiterentwicklung von OCPbasic denkbar. Im Bezug auf die Strukturausnutzung wäre beispielsweise eine cyclic reduction für tridiagonale Blockmatrizen, aus [1] und [59], statt des LAPACK Bandmatrix Lösers anwendbar. Die cyclic reduction hat den Vorteil, dass sich die Faktorisierung parallelisieren lässt. Da die verwendete tridiagonale Blockmatrizen aus 4.42, für konstante Blockdimensionen, besonders viele Nulleinträge in den nebendiagonalen Blocks besitzt, könnte auch eine cyclic reduction für nicht homogene Blockdimensionen entwickelt werden.
Weiterführend an diese Arbeit kann eine erneute Optimierung mit dem kurven-parametrisierten Einspurmodell unter Verwendung von OCPbasic entwickelt werden. Hierbei stößt jedoch die direkte symbolische Differentialrechnung an ihre Grenzen. Durch die Verwendung algorithmischer Differenzierungstechniken, wie ADOL-C aus [69], sollten sich die Probleme der direkten symbolischen Differentialrechnung lösen lassen.

# Literaturverzeichnis


[1] Akimova, E. N. und Belousov, D. V. *Parallel algorithms for solving linear systems with block-tridiagonal matrices on multi-core CPU with GPU*. Journal of Computational Science, 3 (6); 445 – 449, 2012. ISSN 1877-7503. Next Generation Computational Scientists: Russian Federation.

[2] Athans, M. und Falb, P. *Optimal Control: An Introduction to the Theory and Its Applications*. Lincoln Laboratory publications. McGraw-Hill, 1966.

[3] Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B. und Diehl, M. *CasADi – A software framework for nonlinear optimization and optimal control*. Mathematical Programming Computation, In Press, 2018.

[4] Biral, F., Bertolazzi, E. und Bosetti, P. *Notes on Numerical Methods for Solving Optimal Control Problems*. IEEJ Journal of Industry Applications, 5; 154–166, 2015.

[5] Biral, F., Bertolazzi, E. und Bosetti, P. *Notes on Numerical Methods for Solving Optimal Control Problems*. IEEJ Journal of Industry Applications, 5 (2); 154–166, 2016.

[6] Bianco, N. D., Bertolazzi, E., Biral, F. und Massaro, M. *Comparison of direct and indirect methods for minimum lap time optimal control problems*. Vehicle System Dynamics, 57 (5); 665–696, 2019.

[7] Bertolazzi, E., Biral, F. und Lio, M. D. *Symbolic-numeric efficient solution of optimal control problems for multibody systems*. Journal of Computational and Applied Mathematics, 185 (2); 404 – 421, 2006. ISSN 0377-0427. Special Issue: International Workshop on the Technological Aspects of Mathematics.

[8] Bellman, R. E. und Dreyfus, S. E. *Applied Dynamic Programming*. Princetown University Press, 1962.

[9] Britzelmeier, A. und Dreves, A. *A Decomposition Algorithm for Nash Equilibria in Intersection Management*. 2019. Accepted for publication.

[10] Britzelmeier, A., Dreves, A. und Gerdts, M. *Numerical solution of potential games arising in the control of cooperative automatic vehicles*. Proceedings of the Conference on Control and its Applications, 38–45, 2019.

[11] Bellman, R. *Dynamic Programming*. Dover Publications, 1957. ISBN 978-0-486-42809-3.

[12] Bertsekas, D. P. *Dynamic Programming and Optimal Control*. Athena Scientific, 2001. ISBN: 978-1-886-52926-7.

[13] Betts, J. T. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Cambridge University Press, New York, NY, USA, 2nd edition, 2009. ISBN 978-0-898-71688-7.

[14] Bertolazzi, E. und Frego, M. *Semianalytical minimum-time solution for the optimal control of a vehicle subject to limited acceleration*. Optimal Control Applications and Methods, 39 (2); 774–791, 2018.

[15] Bauer, C., Frink, A. und Kreckel, R. *Introduction to the GiNaC Framework for Symbolic Computation within the C++ Programming Language*. Journal of Symbolic Computation, 33 (1); 1 – 12, 2002. ISSN 0747-7171.

[16] Britzelmeier, A. und Gerdts, M. *Non-linear Model Predictive Control of Connected, Automatic Cars in a Road Network Using Optimal Control Methods*. IFAC-PapersOnLine, 51 (2); 168 – 173, 2018. ISSN 2405-8963. 9th Vienna International Conference on Mathematical Modelling.

[17] Buyval, A., Gabdulin, A., Mustafin, R. und Shimchik, I. *Deriving overtaking strategy from nonlinear model predictive control for a race car*. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2623–2628. 2017.

[18] Bryson, A. E. J. und Ho, Y.-C. *Applied Optimal Control: Optimization, Estimation, and Control*. Hemisphere Publishing, New York, 1975.

[19] Bock, H. und Plitt, K. *A Multiple Shooting algorithm for direct solution of optimal control problems*. In *Proceedings of the 9th IFAC World Congress*, 242–247. Pergamon Press, Budapest, 1984.

[20] Bazaraa, M., Sherali, H. und Shetty, C. *Nonlinear Programming Theory and Algorithms*. John Wiley, New York, 1993.

[21] Büskens, C. und Wassel, D. *The ESA NLP Solver WORHP*. In *Modeling and Optimization in Space Engineering* (G. Fasano and J. D. Pintér, editors), 73, 85–110. Springer New York, 2013.

[22] Curtis, F., Schenk, O. und Wächter, A. *An interior-point algorithm for large-scale nonlinear optimization with inexact step computations*. SIAM Journal of Scientific Computing, 32 (6); 3447–3475, 2010. ISSN 1064-8275.

[23] Diehl, M., Ferreau, H. J. und Haverbeke, N. *Efficient numerical methods for nonlinear MPC and moving horizon estimation*. In *Nonlinear model predictive control. Towards new challenging applications. Selected papers based on the presentations at the international workshop on assessment and future directions of nonlinear model predictive control (NMPC08), Pavia, Italy, September 5–9, 2008.*, 391–417. Berlin: Springer, 2009. ISBN 978-3-642-01093-4.

[24] Dickmanns, E. D. *4D-Szenenanalyse Mit Integralen Raum-/Zeitlichen Modellen*. In *Mustererkennung 1987*, 257–271. Springer Berlin Heidelberg, Berlin, Heidelberg, 1987. ISBN 978-3-662-22205-8.

[25] Dickmanns, E.-D. *Vehicle Guidance by Computer Vision*. In *High Precision Navigation* (K. Linkwitz and U. Hangleiter, editors), 86–96. Springer Berlin Heidelberg, Berlin, Heidelberg, 1989. ISBN 978-3-642-74585-0.

[26] Dixon, R. *Automobil-Sensorik 2: Systeme, Technologien und Applikationen*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2018. ISBN 978-3-662-56310-6.

[27] Domahidi, A. und Jerez, J. *FORCES Professional*. embotech GmbH (`http://embotech.com/FORCES-Pro`), 2014.

[28] Domahidi, A. *Methods and tools for embedded optimization and control*. Dissertation, ETH Zürich, Diss. ETH No. 21366, 2013.

[29] Duff, I. S. *MA57—A Code for the Solution of Sparse Symmetric Definite and Indefinite Systems*. ACM Trans. Math. Softw., 30 (2); 118–144, 2004. ISSN 0098-3500.

[30] Dickmanns, E. D. und Zapp, A. *A Curvature-based Scheme for Improving Road Vehicle Guidance by Computer Vision*. Mobile Robots, SPIE Proc., Cambridge, Mass., 727, 161–168. 1986.

[31] Ellis, M., Durand, H. und Christofides, P. D. *A tutorial review of economic model predictive control methods*. Journal of Process Control, 24 (8); 1156 – 1178, 2014. ISSN 0959-1524. Economic nonlinear model predictive control.

[32] Farin, G. *Kubische Spline-Interpolation*, 104–121. Vieweg+Teubner Verlag, Wiesbaden, 1994. ISBN 978-3-663-10602-9.

[33] Funke, J., Brown, M., Erlien, S. M. und Gerdes, J. C. *Collision Avoidance and Stabilization for Autonomous Vehicles in Emergency Scenarios*. IEEE Transactions on Control Systems Technology, 25 (4); 1204–1216, 2017.

[34] Ferreau, H. J., Kirches, C., Potschka, A., Bock, H. G. und Diehl, M. *qpOASES: a parametric active-set algorithm for quadratic programming*. Math. Program. Comput., 6 (4); 327–363, 2014. ISSN 1867-2949; 1867-2957/e.

[35] Fletcher, R. *Practical Methods of Optimization; (2Nd Ed.)*. Wiley-Interscience, New York, NY, USA, 1987. ISBN 0-471-91547-5.

[36] Gerdts, M. *User's Guide OCPID-DAE1 (Optimal Control and Parameter Identification with Differential-Algebraic Equations of Index 1)*. Version 1.3, `https://optimal-control.de/`, 2018.

[37] Gershgorin, S. A. *Über die Abgrenzung der Eigenwerte einer Matrix*. Bull. Acad. Sci. URSS, 1931 (6); 749–754, 1931.

[38] Gerdts, M. *Numerische Methoden optimaler Steuerprozesse mit differential-algebraischen Gleichungssystemen höheren Indexes und ihre Anwendungen in der Kraftfahrzeugsimulation und Mechanik*. Dissertation, Bayreuth, 2001.

[39] Gerdts, M. *A moving horizon technique for the simulation of automobile test-drives*. ZAMM, Z. Angew. Math. Mech., 83 (3); 147–162, 2003. ISSN 0044-2267; 1521-4001/e.

[40] Gerdts, M. *Optimal control of ODEs and DAEs*. Berlin: de Gruyter, 2012. ISBN 978-3-11-024995-8.

[41] Geiger, C. und Kanzow, C. *Theorie und Numerik restringierter Optimierungsaufgaben*. Springer-Lehrbuch Masterclass. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-42790-2.

[42] Gerdts, M., Karrenberg, S., Müller-Beßler, B. und Stock, G. *Generating locally optimal trajectories for an automatically driven car*. Optim. Eng., 10 (4); 439–463, 2009. ISSN 1389-4420; 1573-2924/e.

[43] Gerdts, M. und Lempio, F. *Mathematische Optimierungsverfahren des Operations Research*. Berlin: de Gruyter, 2011. ISBN 978-3-11-024994-1.

[44] Gill, P. E., Murray, W. und Saunders, M. A. *SNOPT: An SQP algorithm for large-scale constrained optimization*. SIAM review, 47 (1); 99–131, 2005.

[45] Grüne, L. und Pannek, J. *Nonlinear Model Predictive Control: Theory and Algorithms*. Communications and Control Engineering. Springer International Publishing, 2016. ISBN 978-3-319-46024-6.

[46] Gutjahr, B. *Recheneffiziente Trajektorienoptimierung für automatisierte Fahreingriffe*. Dissertation, Karlsruher Instituts für Technologie, 2019.

[47] Huber, A. und Gerdts, M. *A dynamic programming MPC approach for automatic driving along tracks and its realization with online steering controllers*. IFAC-PapersOnLine, 20th IFAC World Congress, 50, 8686–8691, 2017.

[48] Huber, A., Gerdts, M. und Bertolazzi, E. *Structure Exploitation in an Interior-Point Method for Fully Discretized, State Constrained Optimal Control Problems*. Vietnam Journal of Mathematics. 46, 1089–1113, 2018.

[49] Hibbeler, R. *Technische Mechanik: Dynamik / Übers. aus dem Amerikan.: Georgia Mais ; Frank Langenau. Fachl. Betr. und Erw.:Jörg Wauer ; Wolfgang Seemann*. Number Bd. 3 in Always learning. Pearson, 2012. ISBN 978-3-868-94127-2.

[50] Huber, A. IPBASIC *User's Guide*. E-Mail: `info@ocpbasic.com`, 2020.

[51] Huber, A. OCPBASIC *User's Guide*. E-Mail: `info@ocpbasic.com`, 2020.

[52] Jerez, J. L. *Custom optimization algorithms for efficient hardware implementation*. Dissertation, Department of Electrical and Electronic Engineering, Imperial College London, 2013.

[53] Kritayakirana, K. und Gerdes, J. C. *Autonomous Vehicle Control at the Limits of Handling*. Int. J. Vehicle Autonomous Systems, 10 (4); 271 – 296, 2012. ISSN 1471-0226.

[54] Kapania, N. R., Subosits, J. und Christian Gerdes, J. *A Sequential Two-Step Algorithm for Fast Generation of Vehicle Racing Trajectories*. Journal of Dynamic Systems, Measurement, and Control, 138 (9), 2016. ISSN 0022-0434.

[55] Lot, R. und Biral, F. *A Curvilinear Abscissa Approach for the Lap Time Optimization of Racing Vehicles*. IFAC Proceedings Volumes, 47 (3); 7559 – 7565, 2014. ISSN 1474-6670. 19th IFAC World Congress.

[56] Manz, M. *Modellbasierte visuelle Wahrnehmung zur autonomen Fahrzeugführung*. Dissertation, Universität der Bundeswehr München, Fakultät für Luft- und Raumfahrttechnik, Neubiberg, 2013.

[57] Moder, T. *Optimale Steuerung eines KFZ im fahrdynamischen Grenzbereich*. Diplomarbeit, Technische Universität München, 1994.

[58] Mitschke, M. und Wallentowitz, H. *Dynamik der Kraftfahrzeuge*. VDI-Buch. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-42011-8.

[59] Mattor, N., Williams, T. J. und Hewett, D. W. *Algorithm for solving tridiagonal matrix problems in parallel*. Parallel Computing, 21 (11); 1769 – 1782, 1995. ISSN 0167-8191.

[60] Nocedal, J. und Wright, S. J. *Numerical optimization. 2nd ed*. New York, NY: Springer, 2nd edition, 2006. ISBN 0-387-30303-0/hbk.

[61] Rieck, M., Bittner, M., Grüter, B., Diepolder, J. und Piprek, P. *FALCON.m User Guide*, Institute of Flight System Dynamics, Technical University of Munich (`http://www.falcon-m.com`), 2018.

[62] Rill, G. *Simulation von Kraftfahrzeugen (Grundlagen und Fortschritte der Ingenieurwissenschaften)*. Vieweg+Teubner Verlag, Braunschweig/Wiesbaden, 1994. ISBN 3-528-08931-8.

[63] Roskopf, A. *Towards object-related navigation for mobile robots*. Dissertation, Universität der Bundeswehr München, Fakultät für Luft- und Raumfahrttechnik, Neubiberg, 2019.

[64] Schmid, M. R. *Umgebungserfassung für Fahrerassistenzsysteme mit hierarchischen Belegungskarten*. Dissertation, Universität der Bundeswehr München, Fakultät für Luft- und Raumfahrttechnik, Neubiberg, 2012.

[65] Schmidt, S. *Ein optimales Steuerungs- und Regelungskonzept für autonome Elektrofahrzeuge*. Dissertation, Otto-von-Guericke-Universität Magdeburg, 2013.

[66] Subosits, J. K. und Gerdes, J. C. *From the Racetrack to the Road: Real-Time Trajectory Replanning for Autonomous Driving*. IEEE Transactions on Intelligent Vehicles, 4 (2); 309–320, 2019.

[67] Steinemann, P. *Objektbildung in dreidimensionalen Messdaten für automobile Anwendungen*. Dissertation, Universität der Bundeswehr München, 2019.

[68] Vögel, M. *Fahrbahnmodellierung und Kursregelung für ein echtzeitfähiges Fahrdynamikprogramm*. Diplomarbeit, Technische Universität München, 1997.

[69] Walther, A. *Getting Started with ADOL-C*. In *Combinatorial Scientific Computing* (U. Naumann, O. Schenk, H. D. Simon and S. Toledo, editors), number 09061 in Dagstuhl Seminar Proceedings. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, Dagstuhl, Germany, 2009. ISSN 1862-4405.

[70] Wächter, A. und Biegler, L. T. *Line Search Filter Methods for Nonlinear Programming: Local Convergence*. SIAM Journal on Optimization, 16 (1); 32–48, 2005.

[71] Wächter, A. und Biegler, L. T. *Line Search Filter Methods for Nonlinear Programming: Motivation and Global Convergence*. SIAM Journal on Optimization, 16 (1); 1–31, 2005.

[72] Wächter, A. und Biegler, L. T. *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*. Mathematical programming, 106 (1); 25–57, 2006.

[73] Werling, M. *Ein neues Konzept für die Trajektoriengenerierung und -stabilisierung in zeitkritischen Verkehrsszenarien*. Dissertation, Karlsruher Instituts für Technologie, 2011.

[74] Zanelli, A., Domahidi, A., Jerez, J. und Morari, M. *FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs*. International Journal of Control, 2017.